**File UpLoad Cyber Jutsu**

**CopyRight: TimoMangCut**

**Trước khi viết về write up các level, tôi sẽ nói một vài điều trước khi bắt đầu vào challenges**

**I/Introduction**

1. **Về Untrust Data :**

Untrust Data có nghĩa là gì ?

* Untrust Data nghĩa là dữ liệu không tin cậy, là các nơi mà hacker có thể sửa đổi dữ liệu trên một gói http request

1. **Về File Upload :**

* Có những Untrust Data nào đối với một file upload ?

Hãy xem trong một gói tin http POST request upload ảnh dưới đây

A screenshot of a computer

Description automatically generated

* Những nơi được đánh dấu mũi tên là Untrust Data đối với File Upload vulnerable

Ở Write-up này tôi sẽ không phân tích code một cách sâu xa và chi tiết, chỉ quan tâm đến untrust data và các method do anh developer tạo ra để defend

**II/ Review Code & Exploit each Level**

**1/ Level 1**

Source Code level 1

A computer screen shot of text

Description automatically generated

* Ở đoạn code trên không hề có một method nào để validate file được upload lên cả
* Vì vậy ta có thể upload tuỳ ý các file
* Tuy nhiên, làm thế nào để gửi cho đúng file nếu như không có source code ?
* Ta có thể dựa vào HTTP response khi gửi một gói tin bất kì đến server nằm trong header `Server`

A screenshot of a computer

Description automatically generated

Trước khi gửi một file PHP lên server để RCE thì ta phải xác định được untrust data nằm ở đâu.

Nhìn vào source code rõ ràng ta có thể thấy được rằng biến $\_FILES là untrust data, tiếp dẫn đến thư mục /var/www/html/upload

Giờ tôi sẽ tiến hành upload một file PHP với nội dung là : `<?php phpinfo(); ?>`

Để kiểm tra xem liệu server có nhận xử lí file PHP hay không



Okay, vậy là nó đã được xử lí thông qua mod-php

Giờ tôi sẽ sửa đổi một chút file php trên, tôi sẽ đổi thành : `<?php system($\_GET[x]); ?>`

Giải thích một chút : hàm system là một os command với parameter x trên GET http request

Ok sau đó tôi sẽ sử dụng lệnh cat với tham số x để get flag



**2/ Level 2**

Source Code cơ bản của level 2 so với level 1 có lẽ là anh developer đã sợ sợ và có thay đổi một chút về validate

A computer code on a black background

Description automatically generated

Giải thích về code : Ở đây method explode sẽ tách dấu chấm trong $filename ra và sẽ lấy string đầu tiên sau dấu chấm để xác định extension của file upload

Vậy sau khi validate bởi method explode thì liệu rằng $filename có còn là untrust data không?

Đáp án là vẫn còn, tôi sẽ thử đổi tên file name từ `test.php` thành `test.png.php` xem sao

A close-up of a number

Description automatically generated

Vậy là cách này có thể bypass được, tôi sẽ xem nó có thực thi code được không và cat flag

A red and black text

Description automatically generated

Mẹo : mẹo này sẽ suyên suốt quá trình học CBJS của tôi, đó chính là :

Unsafe Method + Untrust Data = Vỡ mồm 😊)))

**3/ Level 3**

Lần này thì anh developer có lẽ đã nhận ra được lỗ hổng trong cách phòng chống của mình và bắt đầu sửa lại

A computer code on a black background

Description automatically generated

Giải thích code : Lần này anh developer đã sử dụng hàm `end()` để chọn string cuối cùng của $filename sau khi được tách ra bởi dấu chấm để xác định extension file PHP là sẽ bị chửi

Suy nghĩ 5 giây

Vậy liệu có đuôi file nào khác ngoài PHP nhưng httpd vẫn có thể nhận dạng và đưa nó cho mod-php thực thi như một file PHP hay không ?

Thử research thì tôi thấy rằng có các đuôi file sau có thể thay thế PHP : `phar, phtml, và một vài phiên bản php thấp hơn như php5`

OK thử luôn nhé

A red silhouette of an object

Description automatically generated

Rồi xong, tới công chuyện

**4/ Level 4**

Lần này anh developer lại rút kinh nghiệm nữa, ảnh block hết file extension của php rùi, nếu có thì ảnh chửi

A computer code with text

Description automatically generated with medium confidence

Đưa blacklist này cho PHP thì quá bí luôn

Nhưng có một câu nói của anh Nuật rất hay, đó chính là :

“Khi bạn đang bí thì cho dù có suy nghĩ tiếp nó vẫn sẽ bí, thay vào đó hãy đọc lại tất cả mọi thứ từng dòng và suy ngẫm xem, có khi nào mình đã bỏ qua dữ kiện gì không?”

Trong source code PHP của level 4 này không có gì lạ cả.

Tôi đã thử đọc đến những file docker và các file config

Và sau đó tôi đã thấy một điều kì bí

A screen shot of a computer

Description automatically generated

Những dòng config này có ý nghĩa là được phép ghi đè lên file ht.access

(Lưu ý, việc ghi đè lên file .htaccess là từ phiên bản 3.8 trở về trước của PHP, từ phiên bản 3.9 trở đi, server sẽ tự động ẩn đi file .htaccess)

Muốn tìm hiểu về biểu thức chính quy để viết file config thì các bạn hãy vào trang `regrex101.com` để tìm hiểu. sau đó viết lại một file .htaccess cho phép file extension bất kì được upload sẽ được xử lí như PHP

Với dòng config `Set Handler`

A screen shot of a computer code

Description automatically generated



Ok kiểm tra xem liệu giờ có upload được file html lên và httpd có nhận biết nó như một file php không

A close-up of a number

Description automatically generated

A red and black text

Description automatically generated

Okey đã thực thi được file .html như một file php

Vậy giả sử/ thực tế, ta kiểm soát được .htaccess thì liệu có dừng lại ở lỗi File Upload hay không? Còn thứ gì có thể chèn vào file được nữa?

Suy nghĩ 5 giây nhé. Thử Research nào !

OK, thường thì PHP sẽ đi kèm với một số tag html, vậy liệu ta có thể chèn html được ko?

Thử nhé !

A white background with black text

Description automatically generated

OK ở đây tui đã chèn một tag <h1> với value là “Phuc” ! Chứng minh chèn thành công

Ta có thể làm gì với html bây giờ ?

Thử nghĩ đến các lỗ hổng phía front-end thay vì back-end nào! Đó là XSS ! Liệu ta có thể làm gì với file upload dính XSS này ?

Ý tưởng : Ta có thể dụ dỗ nạn nhân truy cập vào file này, đánh cắp IP, Cookie của họ

Có 2 cách để ta có thể làm ý tưởng trên :

1. Mua/Thuê một Hosting để nhận được response từ nạn nhân và đánh cắp nó khi nạn nhân truy cập vào file chứa xss
2. Có một trang web giả lập về việc hosting đó chính là `webhook.site`. Nó cho phép bạn như một người quản lí truy cập của website, cho phép người dung truy cập vào đường link và gửi các gói tin đến server, và đương nhiên. Bạn cũng sẽ nhận được HTTP Request

A computer code with red text

Description automatically generated

Tôi đã tạo một server random và quản lí các gói tin đến server. Sau đó tôi chèn đoạn html độc hại vào file để tạo một đường dẫn đến site của tôi. Khi truy cập file, nó sẽ tự động gửi gói tin đến site, và cung cấp cookie cũng như IP public khi truy cập

A screenshot of a computer

Description automatically generated

Khi truy cập file ảnh, lập tức có một http request đến server với header cookie = phpsessid của trang web chứa file đó !

**5/ Level 5**

Oh, anh developer chán nản với việc dùng blacklist để defend nữa rồi, nên giờ anh ấy sẽ chuyển sang dùng white list

A black background with text

Description automatically generated

Lần này, anh ấy không còn dựa vào đuôi file để kiểm tra nữa, mà dựa vào mime\_type

Cho các bạn không biết thì hàm này sẽ đọc HTTP request để kiểm tra header Content-Type là gì để gán cho Mime\_type và validate nó

Quay lại phía trên Untrust Data, các bạn có thể thấy rằng tôi đã note nơi header Content-Type là một Untrust Data đối với File Upload vì nó có thể được thay đổi trước khi gửi bởi hacker

Lưu ý rằng đoạn code level 5 này không hề validate file extension

A screenshot of a computer code

Description automatically generated

Ta có thể up một file .php và đổi Content-Type trong http request

A screenshot of a computer

Description automatically generated

A red and black text

Description automatically generated

**6/ Level 6**

Lần này anh ấy lại quyết định không tin vào mime\_type được nữa. Anh ấy sử dụng đến Finfo\_open để mở thư mục ra và kiểm tra xem liệu đó có phải là file ảnh hay không?

Vậy hàm Finfo\_open này kiểm tra bằng cách nào ?

Searching nhé !

Nó sẽ kiểm tra bằng cách mở file ra và đọc các bytes đầu để xác định file signature và kết luận nó.

Liệu file signature có thay đổi được không? Ta thử nhé . Lưu ý rằng cả đoạn code trên vẫn không kiểm tra file extension, vì vậy ta có thể up một file .php lên

A computer code with text

Description automatically generated

Sử dụng file signature là GIF89a để hàm finfo\_open xác định nó là một file GIF



Ta đã có thể upload thành công

Cat flag thui nào

A red mark on a white background

Description automatically generated

* Tuy là GIF89a được finfo\_open nhận định là file signature của file GIF nhưng đối với PHP nó chỉ là cái tên….

**7/ Level 7**

Ở level này ta sẽ combine với 1 lỗi nữa

Xem config thì không cho ghi đè .htaccess & không xử lí các loại file up vào thư mục trên

A screen shot of a computer

Description automatically generated

Và đoạn code này xử lí album hay còn gọi là Untrust Data

A computer screen with text

Description automatically generated

Review Code : Ở đây nó sẽ tạo thư mục dựa trên sessionID của user sau đó cho phép upload file vào thư mục của user, giống như upload lên trang cá nhân vậy

Ở đây thì không có validate gì. Nhưng file up lên không được xử lí thì phải làm sao ?

Suy nghĩ thử xem nào ?

Nếu như không thể xử lí file ở thư mục upload trở đi đến users

Vậy liệu rằng ở thư mục gốc `var/www/html` có xử lí được không? Thử nghĩ tiếp xem, làm cách nào để vào đến thư mục gốc ?

Thử kết hợp path traversal với album xem sao

A screen shot of a computer code

Description automatically generated

Nhấn vào một thư mục bất kì để xem đường dẫn



Khi ta dùng `../..` thì album mà ta up sẽ ở cùng thư mục gốc với images, vì vậy ta sẽ thay đổi thư mục và truy cập vào file test.php. OK vậy là nó đã được upload và xử lí thành công

A close-up of a purple and black rectangle

Description automatically generated

Giờ ta sẽ sửa shell và sử dụng tham số x để đọc flag

A red hand with black text

Description automatically generated

**8/ Level 8**

Nếu nhìn sơ code như thế này thì cách hack sẽ bằng 0

A screen shot of a computer code

Description automatically generated

A computer screen shot of a black background

Description automatically generated

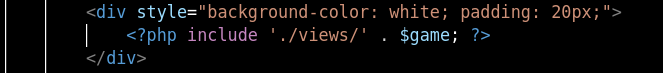
Review code : Hình ảnh code ban đầu đang chọn game để chơi, tính điểm và nếu như chưa có session thì sẽ quay lại register.php. Ở đây tôi không đề cập đến register bởi vì nó đã được validate whitelist quá kĩ không thể hack hehe.

Hình ảnh code phía dưới thì về profile upload, dù có upload như thế nào, và sử dụng lại cách trên thì file shell sẽ tự động bị đổi tên thành avatar.jpg -> không thể thực thi theo ý muốn.

Nhưng tôi sẽ nhắc lại câu nói của anh Nuật, `Nếu như đang bí thì có cố giải nó cũng sẽ bí, cho nên đọc lại từ đầu đến cuối một cách cẩn thận nhất để xem liệu rằng mình đã bỏ qua thứ gì đó không?`

Có lẽ bạn không có source code, nhưng tôi có. Tôi đã phát hiện một unsafe method

Method đó là `include`. Giải thích sơ qua thì include là 1 hàm trong php gọi 1 file php khác và `thực thi` nó. Liệu rằng nó có thực thi được file .jpg không ?



Đây, vấn đề là anh developer không include từng file một mà ảnh include nguyên cả cái thư mục views và ảnh gán biến game vô luôn. Vậy nếu ta sử dụng thư mục views để đẩy method include qua một thư mục khác thì sao?

A group of colors in a row

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generatedVậy là có thể. Ta sẽ sửa shell và đọc flag ở `/`

A screenshot of a computer code

Description automatically generated

Trước tiên ta sẽ viết một đoạn shell để list ra các thư mục ở root directory và tìm thấy được random file flag

Sau đó sửa shell tiếp để cat flag

A screen shot of a computer

Description automatically generated

Vậy nếu như trong trường hợp server không cho upload gì lên server thì sao?

Vậy ta phải nghĩ đến trường hợp khác, đó chính là : Liệu rằng có file gì đó mà mình có thể chèn code vào và nó chạy theo hàm include ko? Có cách nào mà ta có thể làm cho untrust data rơi vào một file có sẵn trên server không?

Research Time Start and End after 5minutes

Tính năng mặc định của Apache Server là tự động lưu log của users vào /var/log/apache2 với các file log như access.log và error.log dựa trên file config của challenges

A computer screen with white text

Description automatically generated

Có nhiều trường trong gói tin HTTP ta có thể thay đổi và chèn code PHP vào đó như : Referer, User-Agent, Request String hay Request String in Body,…

Ta sẽ sử dụng User-Agent để chèn code PHP vào xem sao. Lần này tôi sẽ bỏ qua bước test, và vào luôn 1 con shell.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Vì sao tôi lại không test phpinfo() trước khi viết một con shell ngon vứt vào, vì nó sẽ display full cái màn hình và khó khăn hơn cho việc tìm kiếm, nói chung là `bạn không hiểu, tôi hiểu là được.`

**9/ Level 9**

Bài này review code sẽ hơi mệt hehe

A screen shot of a computer program

Description automatically generated

Review code : Tóm lại trong file index này sẽ cho phép upload file .zip nếu muốn 1 lần upload nhiều file và sẽ unzip nếu đó là file .zip, còn nếu ko phải .zip thì nó sẽ được đặt trong folder đã được tạo trước đó. Có Untrust Data duy nhất đó là file\_name. Vậy liệu rằng ta có thể sử dụng cách cũ để hack không? Tôi sẽ thử viết shell và đặt tên là `../../test.php` để thoát ra ngoài thư mục upload (Vì thư mục upload không cho thực thi bất cứ file gì).

A screenshot of a computer

Description automatically generated

Không truy cập được? Tại sao nhỉ? Debug một chút, ta sẽ var\_dump Untrust Data để xem nó được xử lí như thế nào.

A screenshot of a computer

Description automatically generated  
Vậy bằng một cách nào đó mà Server đã lọc hết các kí tự trong biến File\_name để lại thằng test.php bơ vơ. Vậy là không thể bypass bằng cách này được

Ok, không sao có lẽ do ta quá vội vàng, tiếp tục đọc code nào.

A computer screen shot of text

Description automatically generated

Review Code : Ta sẽ đến với trường hợp thứ 2 đó chính là file .zip, đọc function xử lí file zip xem sao

Ở dòng 44, biến này lấy thông tin của file được nén từ zip

Ở dòng 47, biến này lấy nội dung của file được nén từ zip

Cả 2 Untrust Data này không được validate, vì thế nó rơi vào dòng 52, đọc file.

Exploit : Ta sẽ thử nén một file php và sau đó đổi tên trong burpsuite thử xem sao, nhưng giả thuyết này có vẻ khó thực thi vì nếu sai 1 ly thì server sẽ chửi ngay. Nên tôi đã research google xem có cách nào đổi tên file trong một file zip không? Thì có một con tool hỗ trợ việc đó, là `evilarc`, tool này hỗ trợ bypass path traversal.

A screen shot of a computer program

Description automatically generated

Viết một shell sẵn, sau đó sử dụng `evilarc` để nén nó và đổi tên

A screen shot of a computer

Description automatically generated

-d : 2 lần `../`

-o : hệ điều hành

-f : file

Ok, ta sẽ thử upload file này lên server xem sao

A screenshot of a computer

Description automatically generated

Vậy có nghĩa là đã giải nén thành công tên file và hack theo lỗ hổng path traversal thành công. Giờ ta sẽ gán giá trị cho tham số và cat flag

A red mark on a white background

Description automatically generated

**III/ Conclusion**

* Kết luận : Với những bài file upload `cơ bản` này, đã giúp tôi củng cố hơn về định luật sau : Unsafe Method + Untrust Data = Vỡ mồm 😊))
* Khi tiếp cận các bài lab hoặc website thực tế khi pentest, các bạn hãy để ý thật kĩ Untrust Data nằm ở đâu, và đi theo đường nào, liệu nó có dính dáng đến một method nào không? Và hãy tìm hiểu các dangerous method của các ngôn ngữ lập trình khác nhau khi pentest nhé ! Và hãy nhớ kết hợp lỗi khi có thể