

```

0
1  </style>
2
3  <?php
4  $data="";
5  if(isset($_GET['data'])){
6      $data = $_GET['data'];
7  }
8
9  if(isset($_GET['encode'])){
10     if($_GET['encode'] == 'encode'){
11         $data = htmlentities($data);
12     }
13 }
14
15 }

```

Hình 16 Ví dụ sử dụng hàm mã hóa dữ liệu

6.5. Cách phòng chống

- Mã hóa thông tin và dữ liệu người dùng gửi lên
- Sử dụng cách hàm mã hóa dữ liệu như “`htmlentities($str)`”
- Lọc các dữ liệu đầu vào của người dùng

7. Template Injection

7.1. Khái niệm

Template injection xảy ra khi đầu vào của người dùng được nhúng trong một mẫu không an toàn.

Lỗ hổng này thường phát sinh thông qua các nhà phát triển cố ý cho phép người dùng gửi hoặc chỉnh sửa mẫu, một số công cụ mẫu cung cấp chế độ bảo mật cho mục đích rõ ràng.

7.2. Tác hại

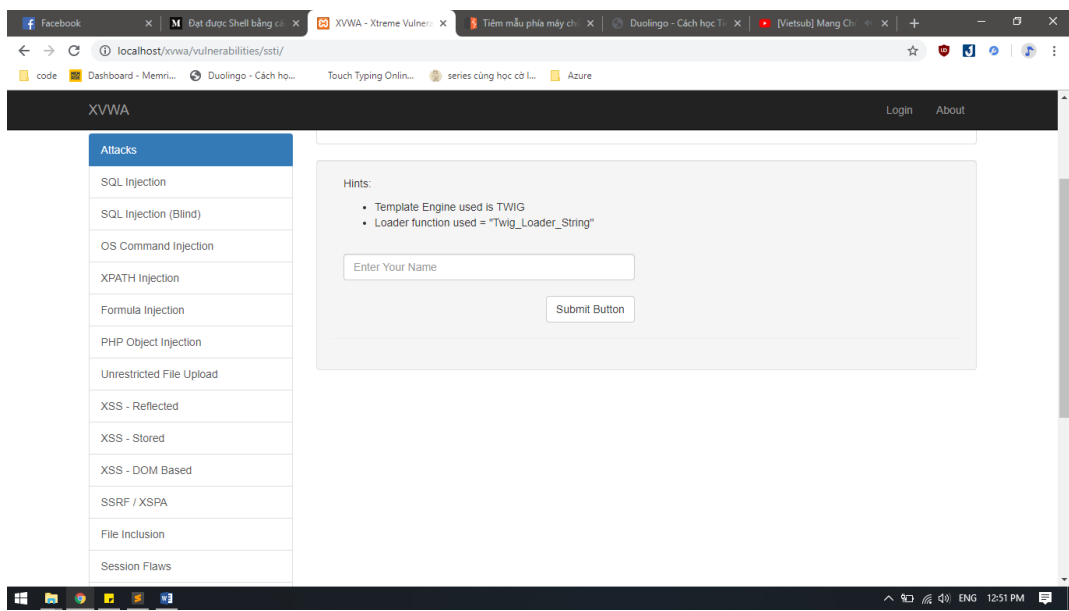
- Thông tin có thể bị đánh cắp hoặc chỉnh sửa.
- Hacker lợi dụng để tấn công hoặc phá hoại hệ thống

7.3. Cách phòng chống

- Kiểm tra, lọc, mã hóa dữ liệu người dùng gửi lên.
- Sử dụng phương thức POST thay cho phương thức GET
- Truyền dữ liệu bằng Ajax...

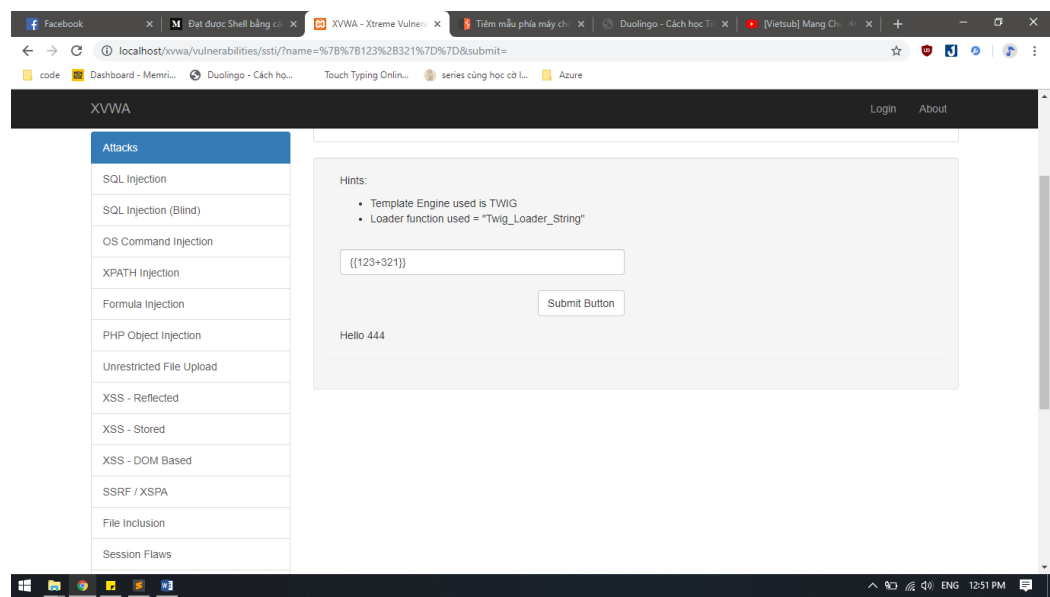
7.4. Ví dụ

Ta có một form nhận tên người dùng và xuất ra tên người dùng như hình 17 .



Hình 17 Form nhập thông tin

Ta nhập dữ liệu “{{123+321}}” kết quả trả về 444 như hình 18. Vậy lỗi Template Injection đã xảy ra ở đây.



Hình 18 Form nhập thông tin

Ta có thể khắc phục lỗi này bằng cách lọc dữ liệu đầu vào của người dùng như hình 19. Kết quả đã như hình 20.

```

if (isset($_GET['submit'])) {
    $name=$_GET['name'];

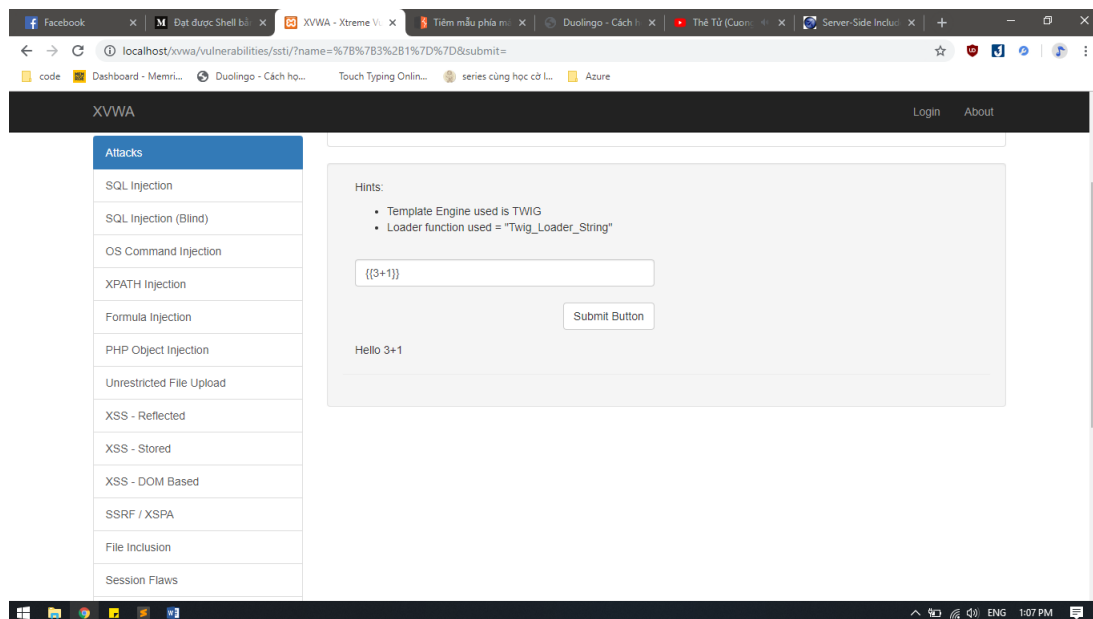
    $name = str_replace("{", "", $name);
    $name = str_replace("}", "", $name);

    // include and register Twig auto-loader
    include 'vendor/twig/twig/lib/Twig/Autoloader.php';
    Twig_Autoloader::register();
    try {
        // specify where to look for templates
        $loader = new Twig_Loader_String();

        // initialize Twig environment
        $twig = new Twig_Environment($loader);
        // set template variables
        // render template
        $result= $twig->render($name);
        echo "Hello $result";
    } catch (Exception $e) {
        die ('ERROR: ' . $e->getMessage());
    }
}

```

Hình 19 Ví dụ lọc dữ liệu của người dùng



Hình 20 Ví dụ đã lọc dữ liệu của người dùng

8. SQL Injection

8.1. Khái niệm

SQL Injection là một hình thức tấn công trong đó truy vấn SQL, do người dùng nhập vào, từ đó mã lệnh được gửi tới máy chủ database để phân tích cú pháp và thực thi.

8.2. Tác hại

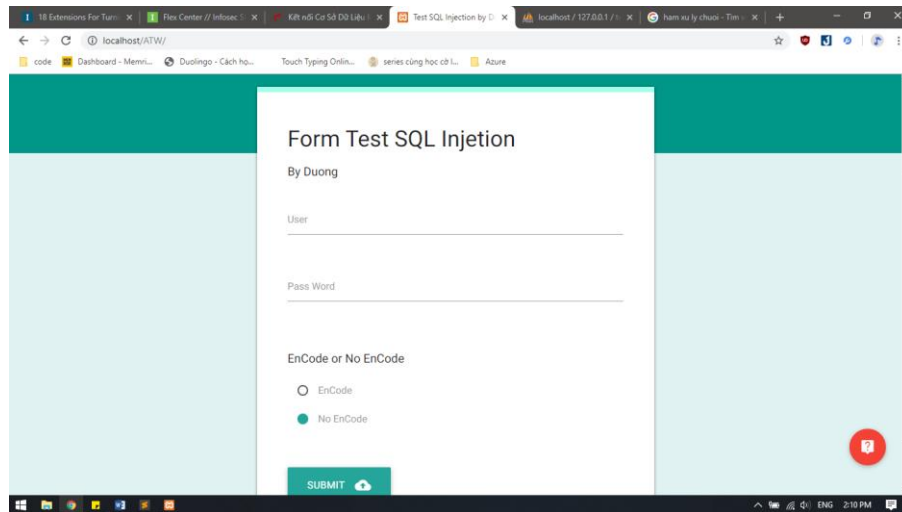
- Lộ cơ sở dữ liệu
- Mất một phần hoặc toàn bộ dữ liệu
- Dữ liệu có thể bị chỉnh sửa

8.3. Cách tấn công

- Tấn công khai thác dữ liệu thông qua toán tử UNION
- Tấn công khai thác thông qua các câu lệnh điều kiện
- Phương thức tấn công nâng cao Blind SQL Injection
- Qua mặt các bộ lọc tham số đầu vào
- Qua mặt bộ lọc của tường lửa web

8.4. Ví dụ

Ta có form đăng nhập. Với dữ liệu chứa được mã hóa.



Hình 21 Form nhập dữ liệu

Ta nhập user và pass word

User

' or 1=1 --

Pass Word

1234

Hình 22 Ví dụ nhập dữ liệu

Kết quả là đăng nhập thành công. Như vậy lỗi SQL Injection đã xảy ra.

Cách khắc phục:

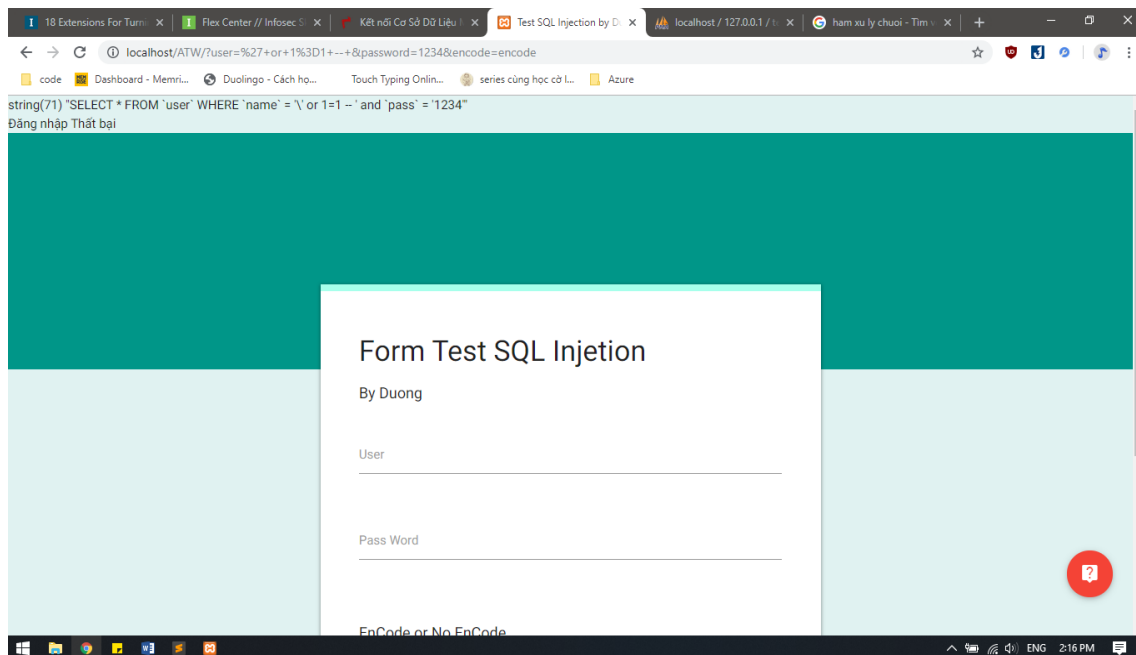
Ta sẽ mã hóa dữ liệu đầu vào của người dùng bằng hàm “addslashes()” trong PHP.

```
$user="";
$password="";
if(isset($_GET['user']) && isset($_GET['password']))){
    $user = $_GET['user'];
    $password = $_GET['password'];
}

if(isset($_GET['encode'])){
    if($_GET['encode'] == 'encode'){
        $user = addslashes($user);
        $password = addslashes($password);
    }
}
```

Hình 23 Ví dụ sử dụng hàm mã hóa

Sau khi mã hóa dữ liệu đầu vào. Việc xảy ra lỗi khi đăng nhập đã không còn.



Hình 24 Form đăng nhập

Chúng ta cũng có thể sử dụng SQLMap để quét các lỗ hổng liên quan đến SQL Injection, như hình 21

9. Server Side Request Forgery

9.1. Khái niệm

SSRF (Server Side Request Forgery) hay còn gọi là tấn công yêu cầu giả mạo từ phía máy chủ cho phép kẻ tấn công thay đổi tham số được sử dụng trên ứng dụng web để tạo hoặc kiểm soát các yêu cầu từ máy chủ bị tấn công.

9.2. Tác hại

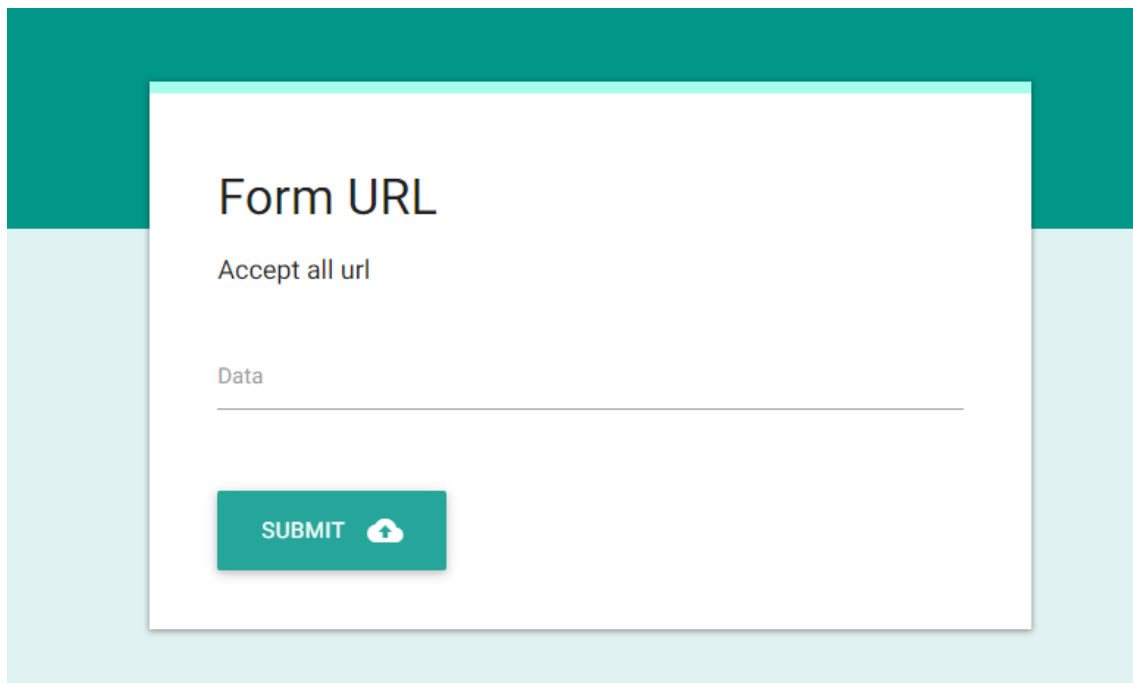
- Đọc và truy cập tệp trái phép
- Thực thi mã từ xa
- Bỏ qua xác thực dự trên máy chủ
- Đánh cắp thông tin phía máy chủ hay người dùng

9.3. Cách phòng chống

- Sử dụng cách danh sách trắng
- Kiểm tra dữ liệu đầu vào một cách cẩn thận
- Lọc và mã hóa dữ liệu gửi lên

9.4. Ví dụ

Ta có một form nhận cách url và hiển thị dữ liệu của url trang web.



The image shows a web form titled "Form URL". Below the title is the text "Accept all url". There is a text input field with the placeholder text "Data". At the bottom of the form is a green button labeled "SUBMIT" with a small cloud upload icon next to it.

Hình 26 Form nhập url

Chúng ta tiến hành thử với [file:///etc](#). Và kết quả nhận được là một dòng lỗi, ghi ra đường dẫn root.

Request

Raw	Params	Headers	Hex
<pre>GET /CRLF/ssrf.php?url=file:///etc HTTP/1.1 Host: 5a345fbd.ngrok.io User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:68.0) Gecko/ Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0 Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3 Accept-Encoding: gzip, deflate Connection: close Referer: http://5a345fbd.ngrok.io/CRLF/ssrf.php Upgrade-Insecure-Requests: 1</pre>			

Hình 27 Ví dụ

Response

Raw	Headers	Hex	HTML	Render
<pre><div class="input-field col m6 s12"> <button type="submit" class="waves-effect waves-light btn-large"><i class="material-icons right">backup</i>Submit</button> </div> </div> </form> <div>
 Warning: fopen(file:///etc): failed to open stream: No such file or directory in D:\PHP\htdocs\CRLF\SSRF.php on line 146

 Warning: fpassthru() expects parameter 1 to be resource, bool given in D:\PHP\htdocs\CRLF\SSRF.php on line 148
 </div> </div> </div> </div> </div></pre>				

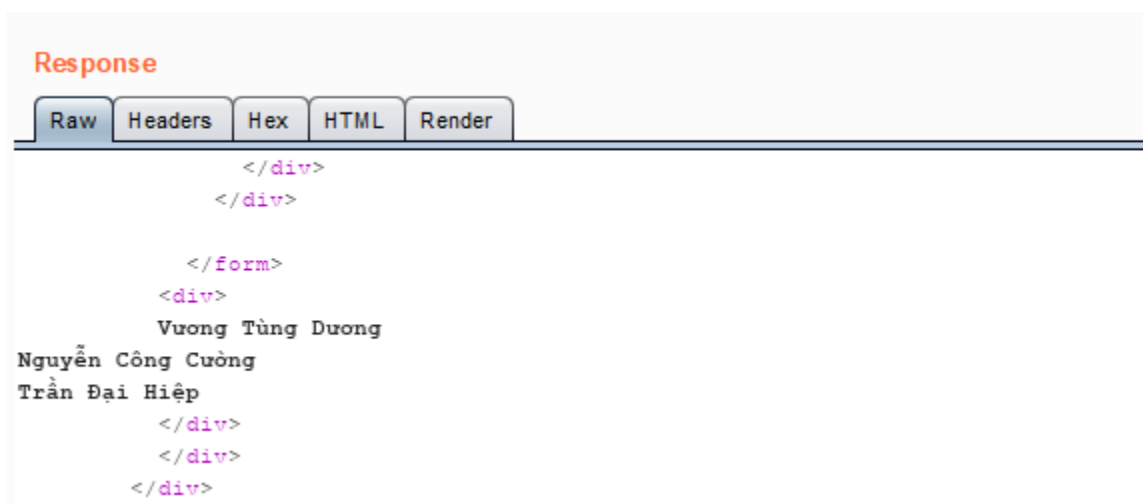
Hình 28 Ví dụ

Như vậy ta có lấy được đường dẫn root. Ta tiến hành quét toàn bộ hệ thống bằng cách nhập thêm đường dẫn gốc.

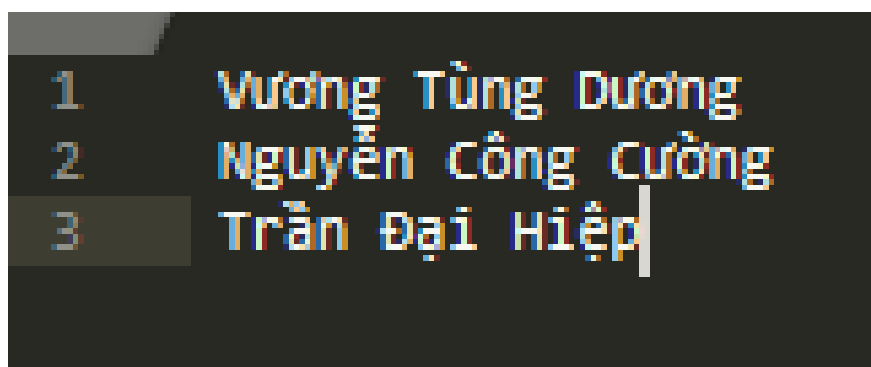
Dưới đây ta có một đường dẫn đã tồn tại và dữ liệu được hiện thị lên “file:///D:/PHP/htdocs/CRLF/etc/pass.txt



Hình 29 Ví dụ gửi dữ liệu



Hình 30 Ví dụ nhận được kết quả



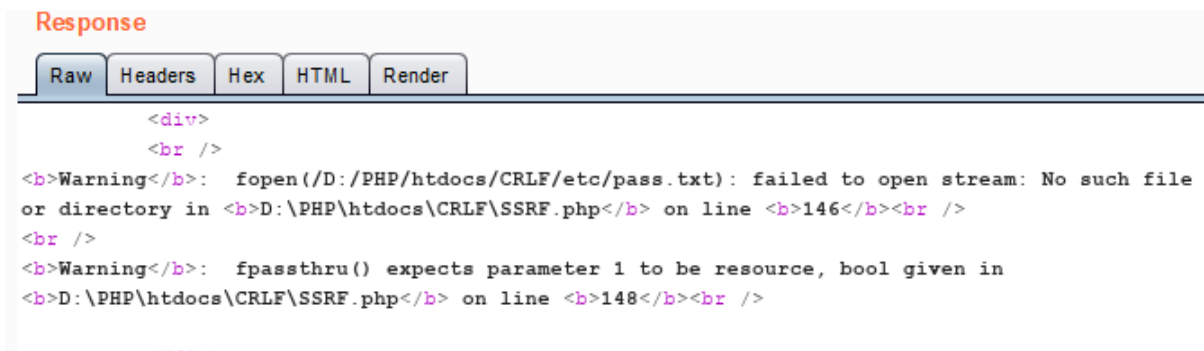
Hình 31 Dữ liệu trong file

Cách khắc phục.

Ta dùng hàm `str_replace` để lọc bỏ “file://” ra khỏi url.

```
<?php if (isset($_GET['url'])) {  
    $url = $_GET['url'];  
    $url = str_replace("file://", "", $url);  
    $data = fopen($url, 'rb');  
    // header("Content-Type: image/png");  
    fpassthru($data);  
} ?>
```

Hình 32 Ví dụ dùng lọc



Hình 33 Kết quả của lọc

10.XML External Entity Vulnerability

10.1. Khái niệm

XML External Entity Vulnerability là một loại tấn công chống lại một ứng dụng phân tích đầu vào XML.

Cuộc tấn công này xảy ra khi đầu vào XML chứa tham chiếu đến một thực thể bên ngoài được xử lý bởi trình phân tích cú pháp XML được cấu hình yếu.

10.2. Tác hại

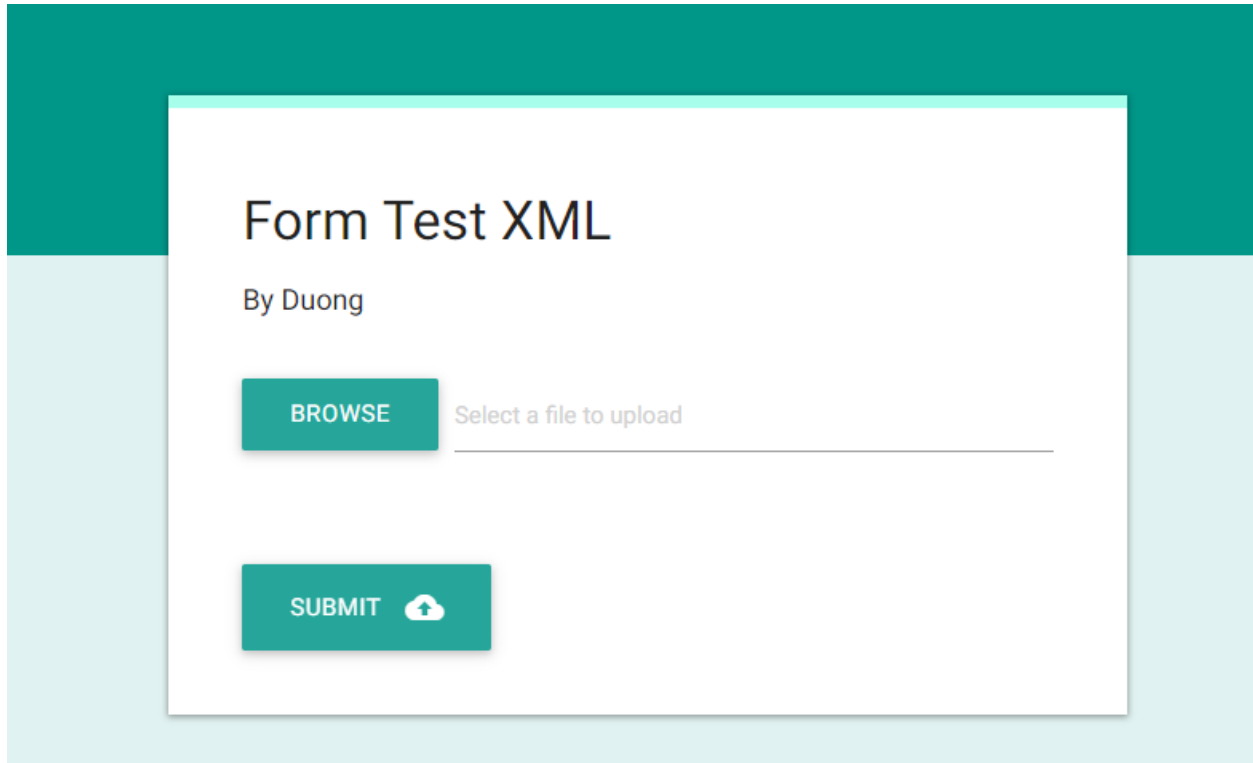
- Thông tin phía sever bị lộ hoặc đánh cắp
- Sever bị ngưng hoạt động
- Giả mạo yêu cầu phía máy chủ

10.3. Cách phòng chống

- Ta ngăn chặn việc tạo và truy cập thực thể từ bên ngoài vào bằng lệnh `libxml_disable_entity_loader(true)`

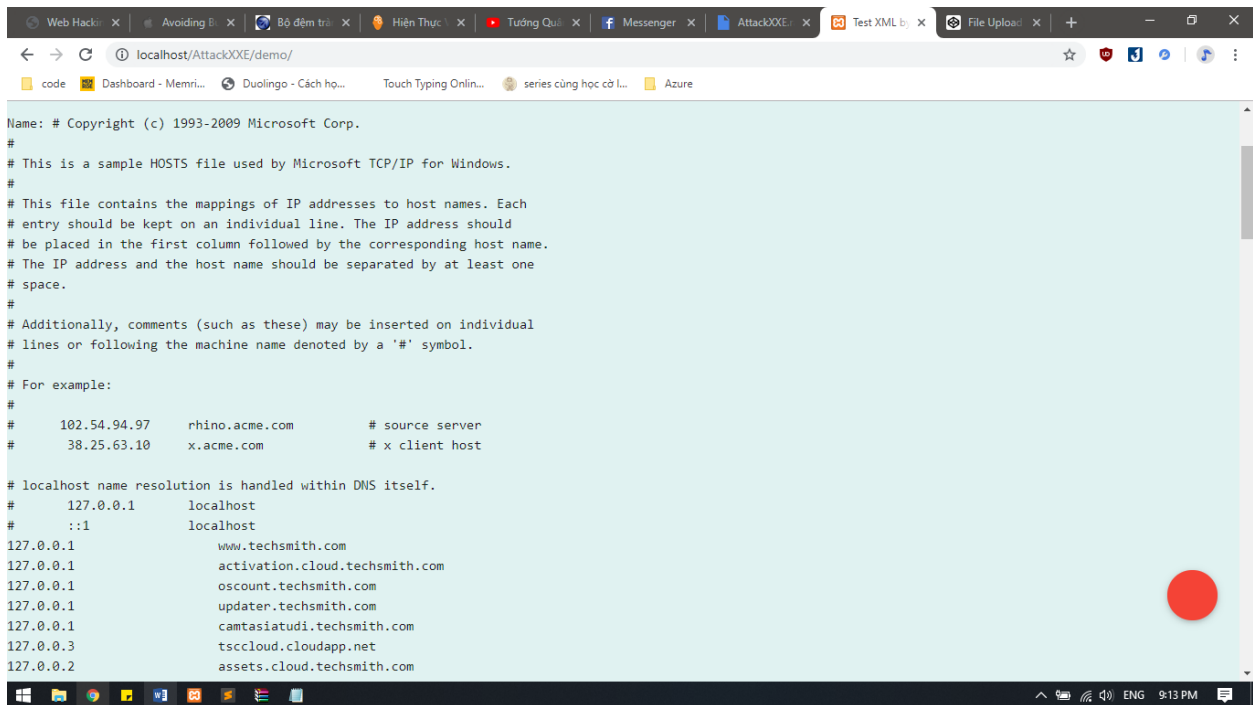
10.4. Ví dụ

Ta có một form upload file và đọc thông tin file xml ra cho người dùng. Không được kiểm tra. Và hacker có thể lợi dụng để chèn các thực thể từ bên ngoài vào.



The image shows a web form titled "Form Test XML" with the text "By Duong" below it. The form is set against a teal background. It contains two main interactive elements: a "BROWSE" button in a teal box next to a text input field with the placeholder "Select a file to upload", and a "SUBMIT" button in a teal box with a white cloud upload icon.

Hình 34 Form upload file xml



Hình 35 Lộ thông tin phía sever

```
File Edit Format View Help
<?php
$xml = <<<XML
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE own [ <!ELEMENT own ANY >
<!ENTITY own SYSTEM "file:/C:/Windows/System32/drivers/etc/hosts" >]>
<information>
    <name>&own;</name>
    <phone>0123456</phone>
    <email>mong.se.nho.ten@gmail.com</email>
    <address>Thai nguyen</address>
</information>
XML;
```

Hình 36 Nội dung trong file xml

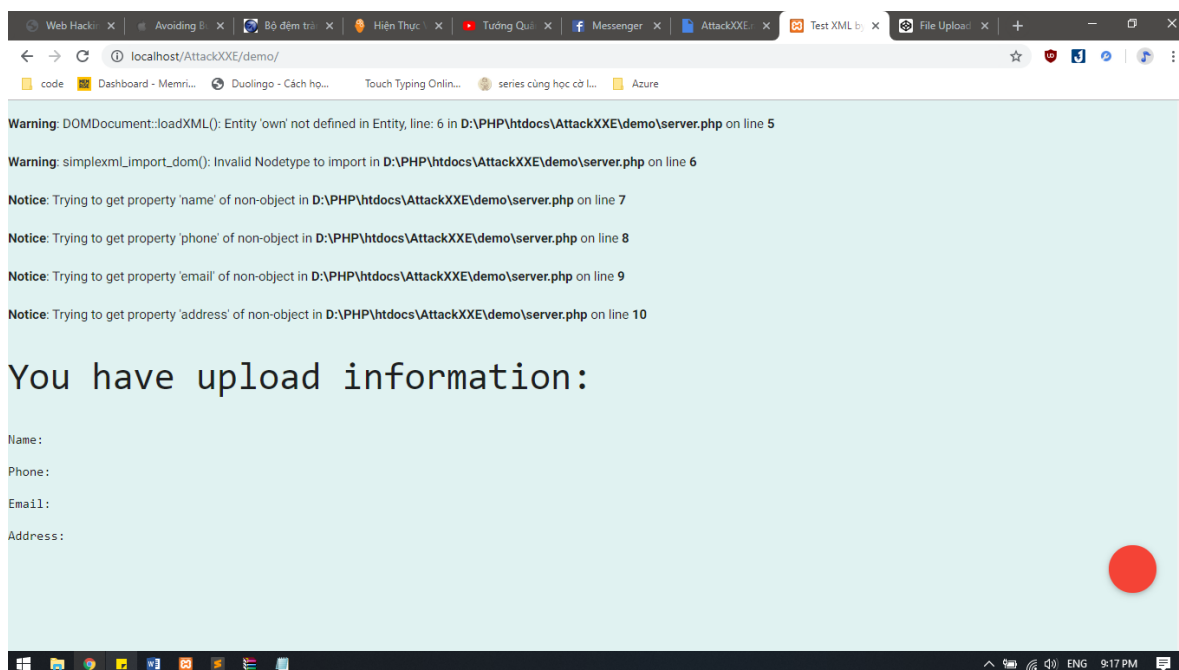
Cách khắc phục. Ta ngăn chặn thực thể bên ngoài nhúng vào file xml bằng hàm “libxml_disable_entity_loader(true);

```

1 <?php
2 libxml_disable_entity_loader(true);
3 $xm = file_get_contents('php://input');
4 $dom = new DOMDocument ();
5 $dom->loadXML ($xm, LIBXML_NOENT | LIBXML_DTDLOAD)
6 $information = simplexml_import_dom( $dom);
7 $name = $information->name; #load name
8 $phone = $information ->phone; #load phone
9 $email = $information ->email; #load email
10 $address = $information ->address; #load address

```

Hình 37 Ví dụ ngăn chặn



Hình 38 Thông báo lỗi vì có thực thể bên ngoài

11. Remote Code Execution

11.1. Khái niệm

Remote Code Execution tên viết tắt là (RCE) : Thực thi mã từ xa. Lỗi xảy trên các hàm eval() của php, khi đầu vào của người dùng được nhúng trong một mẫu dữ liệu không an toàn và không được phía sever kiểm tra kỹ càng.

11.2. Tác hại

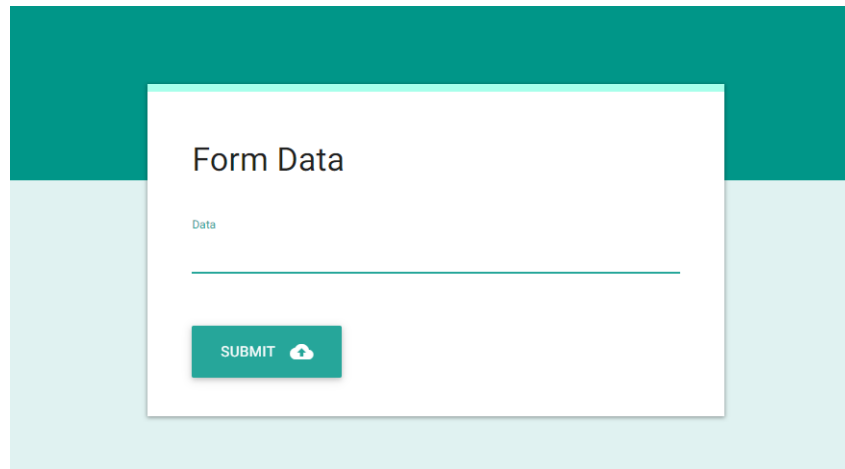
- Dữ liệu có thể bị đánh cắp hoặc chỉnh sửa
- Hệ thống có thể bị thay đổi hoặc chỉnh sửa
- Hệ thống có thể bị ngừng hoạt động

11.3. Cách phòng chống

- Lọc và kiểm tra dữ liệu người dùng một cách cẩn thận và chính xác
- Mã hóa dữ liệu người dùng gửi lên và giải mã
- Nên sử dụng cách phương thức POST thay cho phương thức GET

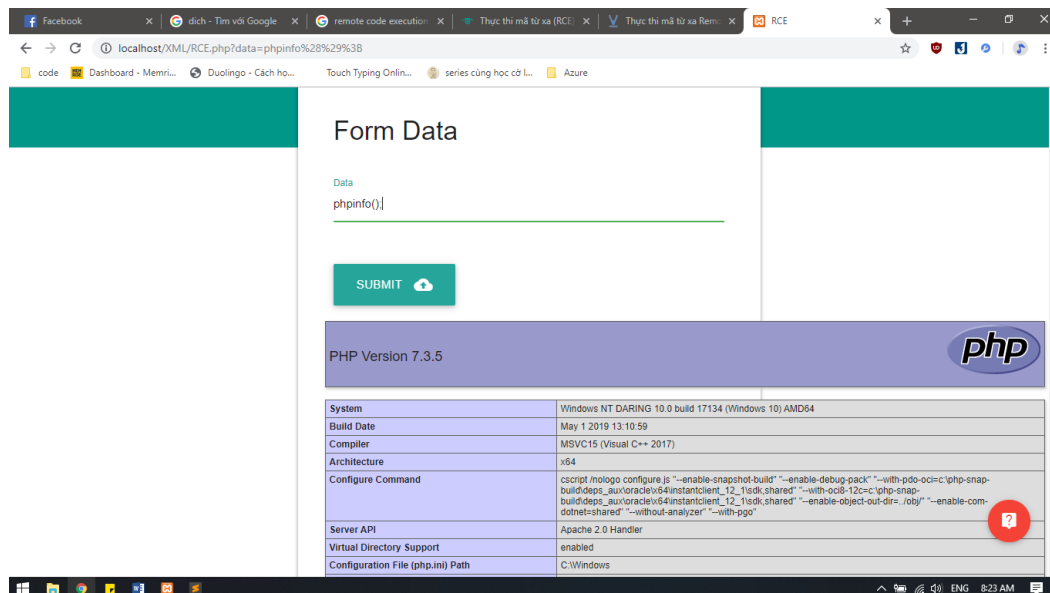
11.4. Ví dụ

Ta có một form nhập dữ liệu và hiển thị dữ liệu ra cho người dùng, nhưng chưa được kiểm tra và lọc dữ liệu một cách cẩn thận.



Hình 39 Form nhập dữ liệu

Khi người dùng nhập “phpinfo()”, lệnh này được chạy và không được kiểm tra. Dẫn đến thông tin phía sever bị lộ.



System	Windows NT DARING 10.0 build 17134 (Windows 10) AMD64
Build Date	May 1 2019 13:10:59
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x64
Configure Command	cmd.exe /c "cd /d C:\php-src & phpize --enable-snapshot-build --enable-debug-pack --with-pdo-oci=C:\php-src\builddeps_aux\oracle\v64\instantclient_12_1\sdk,shared --with-oci8-12c=C:\php-src\builddeps_aux\oracle\v64\instantclient_12_1\sdk,shared --enable-object-out-dir=.obj --enable-com-dotnet=shared --without-analyzer --with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows

Hình 40 Ví dụ lộ thông tin sever

Bây giờ, ta khắc phục lỗi này bằng cách lọc dữ liệu hoặc có thể thay thế hay hạn chế việc sử dụng hàm eval() bằng việc xử lý dữ liệu bằng tay hoặc các hàm khác.

Dưới đây là cách lọc dữ liệu bằng danh sách đen.

```
<?php
$data = "";
if (isset($_REQUEST['data'])) {
    $data = $_REQUEST['data'];

    if (!is_string($data)) {
        echo 'Lỗi chuỗi không hợp lệ<br/><br/>';
    } elseif (preg_match('/^.*(alias|bg|bind|break|builtin|case|cd|command|compgen|
|exit|export|fc|fg|getopts|hash|help|history|if|jobs|kill|let|local|logout|p
source|suspend|test|times|trap|type|typeset|ulimit|umask|unalias|unset|until
7F|+).*$/', $data)) {
        echo 'Bạn không có quyền thực thi cách lệnh này<br/><br/>';
    } else {
        $cmd = json_decode($data, true)['cmd'];
        if ($cmd !== NULL) {
            system($cmd);
        } else {
            echo 'Đầu vào không hợp lệ';
        }
        echo '<br/><br/>';
    }
}
die();
?>
```

Hình 41 Ví dụ lọc dữ liệu

12.Memory

12.1. Khái niệm

Lỗi tràn bộ nhớ đệm thường được cách hacker sử dụng nhằm mục đích làm hỏng bộ nhớ đệm của cách ứng dụng web.

Bằng cách gửi dữ liệu đầu vào được làm cẩn thận đến một ứng dụng web, hacker có thể khiến ứng dụng thực thi mã tùy ý, có thể chiếm quyền điều khiển máy chủ.

12.2. Phương pháp tấn công

- Đọc và ghi dữ liệu vào cách địa chỉ cụ thể
- Các kiểu dữ liệu không được xử lý đúng cách

12.3. Tác hại

- Chiếm quyền điều khiển
- Thực thi mã lệnh không an toàn
- Máy chủ có thể bị lỗi hoặc ngừng hoạt động

12.4. Cách phòng chống

- Kiểm tra dữ liệu đầu ra và đầu vào một cách cẩn thận và chính xác

- Sử dụng strncpy thay vì strcat, strncpy thay vì strcpy,...
- Sử dụng cách framework

12.5. Ví dụ

Ta có một đầu vào không được kiểm soát, việc sao chép dữ liệu và không kiểm tra đến đến việc tràn bộ nhớ. Như ta biến buffer ở đây chỉ có kích thước là 5 và đầu vào đã là hơn 5. Như vậy lỗi tràn bộ nhớ đã xảy ra.

```

Untitled1.cpp
1  #include<iostream>
2  #include <string.h>
3  using namespace std;
4  void f(char* s) {
5      char buffer[5];
6      strcpy(buffer, s);
7      cout<<buffer<<endl;
8  }
9
10 int main() {
11     f("123456789");
12 }
  
```

```

C:\Users\duong\Desktop\Untitled1.exe
123456789
-----
Process exited after 0.8224 seconds with return code 139.
Press any key to continue . . .
  
```

Hình 42 Ví dụ memoy

Ta khắc phục lỗi này bằng việc xử lý và kiểm tra dữ liệu bằng tay. Và việc tràn bộ nhớ đệm đã không còn.

```

Untitled1.cpp
1  #include<iostream>
2  #include <string.h>
3  using namespace std;
4  void f(char* s) {
5      char buffer[5];
6      for(int i = 0 ; i< strlen(s) && i < 4 ; i++){
7          // kiểm tra xem đến cuối chuỗi
8          // kiểm tra xem qua kích thước chuỗi
9          if(i == 3 && i == strlen(s) - 1 ){
10             buffer[i] = s[i];
11             buffer[i+1] = '\0';
12          }else{
13             buffer[i] = s[i];
14          }
15      }
16      cout<<buffer<< " _"<<endl;
17  }
18
19 int main() {
20     f("1234456789");
21 }
  
```

```

C:\Users\duong\Desktop\Untitled1.exe
1234_
-----
Process exited after 0.8344 seconds with return code 139.
Press any key to continue . . .
  
```

Hình 43 Xử lý việc tràn bộ nhớ đệm