

| | |
|--|-----------|
| I. OPEN REDIRECT VULNERABILITIES..... | 3 |
| 1. MÔ TẢ..... | 3 |
| 2. VÍ DỤ..... | 3 |
| 3. TÁC HẠI..... | 4 |
| 4. CÁCH PHÁT HIỆN..... | 4 |
| 5. CÁCH NGĂN CHẶN | 4 |
| II. HTTP PARAMETER POLLUTION..... | 4 |
| 1. MÔ TẢ..... | 4 |
| 2. VÍ DỤ..... | 4 |
| 3. TÁC HẠI..... | 5 |
| 4. CÁCH PHÒNG CHỐNG | 5 |
| III. CROSS SITE REQUEST FORGERY..... | 6 |
| 1. MÔ TẢ..... | 6 |
| 2. TÁC HẠI..... | 6 |
| 3. CÁCH PHÁT HIỆN..... | 6 |
| 4. PHÒNG CHỐNG | 6 |
| IV. HTML INJECTION..... | 7 |
| 1. MÔ TẢ..... | 7 |
| 2. TÁC HẠI..... | 7 |
| 3. CÁCH PHÁT HIỆN..... | 7 |
| 4. PHÒNG CHỐNG | 7 |
| V. CRLF INJECTION | 7 |
| 1. MÔ TẢ..... | 7 |
| 2. VÍ DỤ..... | 8 |
| 3. TÁC HẠI..... | 9 |
| 4. CÁCH PHÒNG TRỐNG..... | 9 |
| VI. CROSS-SITE SCRIPTING | 9 |
| 1. MÔ TẢ..... | 9 |
| 2. VÍ DỤ..... | 9 |
| 3. TÁC HẠI..... | 10 |
| 4. CÁCH PHÒNG TRỐNG..... | 10 |
| VII. TEMPLATE INJECTION | 10 |
| 1. MÔ TẢ..... | 10 |
| 2. VÍ DỤ..... | 10 |
| 3. TÁC HẠI..... | 10 |
| 4. CÁCH PHÒNG TRỐNG..... | 10 |
| VIII. SQL INJECTION..... | 11 |
| 1. MÔ TẢ..... | 11 |

| | | |
|-------------|---|-----------|
| 2. | VÍ DỤ..... | 11 |
| 3. | TÁC HẠI..... | 11 |
| 4. | CÁCH PHÒNG TRỐNG..... | 12 |
| IX. | SERVER SIDE REQUEST FORGERY..... | 12 |
| 1. | MÔ TẢ..... | 12 |
| 2. | VÍ DỤ..... | 13 |
| 3. | TÁC HẠI..... | 13 |
| 4. | CÁCH PHÒNG CHỐNG | 13 |
| X. | XML EXTERNAL ENTITY VULNERABILITY..... | 14 |
| 1. | MÔ TẢ..... | 14 |
| 2. | VÍ DỤ..... | 14 |
| 3. | TÁC HẠI..... | 14 |
| 4. | CÁCH PHÒNG CHỐNG | 14 |
| XI. | REMOTE CODE EXECUTION..... | 15 |
| 1. | MÔ TẢ..... | 15 |
| 2. | VÍ DỤ..... | 15 |
| 3. | TÁC HẠI..... | 16 |
| 4. | CÁCH PHÒNG CHỐNG | 16 |
| XII. | MEMORY | 17 |
| 1. | MÔ TẢ..... | 17 |
| 2. | VÍ DỤ..... | 17 |
| 3. | TÁC HẠI..... | 18 |
| 4. | CÁCH PHÒNG CHỐNG | 18 |

I. Open Redirect Vulnerabilities

1. Mô tả

Open Redirect Vulnerabilities cho phép kẻ tấn công điều hướng người dùng thiếu cảnh giác đến các website nguy hiểm.

Kiểu tấn công này sẽ đánh vào lòng tin của nạn nhân. Dẫn nạn nhân đến trang web nguy hiểm của kẻ tấn công.

2. Ví dụ

Chúng ta có một trang web <http://example.com> và trong trang web này có một liên kết như:

```
https://example.com/signup?redirectUrl=https://example.com/login
```

Liên kết này là một trang signup (đăng ký), khi bạn đăng ký, bạn sẽ được chuyển hướng đến <https://example.com/login> được chỉ định trong tham số HTTP GET redirectUrl.

Điều gì sẽ xảy ra nếu chúng ta thay đổi example.com/login thành attacker.com?https://www.example.com/?redirect_to=https://www.sub.example.com

Bằng cách truy cập vào url này, nếu được chuyển hướng đến attacker.com sau khi đăng nhập, điều này có nghĩa trang web này có một lỗ hổng chuyển hướng mở.

Điều này xảy ra do kiểm tra chuyển hướng không kỹ càng trong back-end.

```
<?php
$url_to_redirect = $_GET['redirect_url'];
header('Location: ' . $url_to_redirect);
die();
```

Ở đây, mã php lấy url một cách mù quáng từ `redirect_url` tham số và chuyển hướng đến url đó.

3. Tác hại

Hacker lợi dụng chuyển hướng người dùng đến những trang web xấu thay vì trang gốc.

Những trang web bị tấn công có thể là những trang web nổi tiếng và khi chuyển hướng thì người dùng sẽ hoàn toàn tin tưởng, không nghi ngờ.

Bị mất thông tin khi người dùng không cảnh giác nhập thông tin vào các trang web giả mạo giống như trang web gốc

4. Cách phát hiện

Truy cập vào tất cả các link url trong trang web mục tiêu để tìm ra tham số chuyển hướng trực tiếp.

Tìm thêm nhiều đường dẫn có tham số chuyển hướng bằng cách đọc mã javascript.

Phân tích nơi cần chuyển hướng trong các website mục tiêu

Sử dụng công cụ Burp Suite

5. Cách ngăn chặn

Kiểm tra lại tất cả các url chuyển hướng trong back-end, kiểm tra nếu link chuyển hướng là link domain của mình thì mới chuyển hướng

Chỉ chuyển hướng nếu bạn thật sự muốn.

Cảnh báo chuyển hướng cho người dùng.

II. HTTP parameter pollution

1. Mô tả

HTTP parameter pollution (HPP) là một kỹ thuật tấn công web mà kẻ tấn công sẽ tạo ra các tham số trùng nhau trong HTTP request

Sẽ có 2 loại HPP là Server-Side và Client-Site

Server-Side: hacker gửi các thông tin bất thường cố gắng làm cho máy chủ trả về kết quả không mong muốn

2. Ví dụ

Một URL với các thông tin này chuyển \$5000 đô la từ số tài khoản 12345 sang tài khoản 67890 có thể trông giống như sau

<https://www.bank.com/transfer?from=12345&to=67890&amount=5000>

Hacker có thể thêm tham số để server hiểu sai và làm sai mục đích, như:
<https://www.bank.com/transfer?from=12345&to=67890&amount=5000&from=ABCDEF>

Url này giống như ban đầu nhưng có thêm phần chuyển thêm cho một tài khoản lạ khác

Cả hai lỗ hổng HPP phía máy chủ(Server-Side) và phía máy khách (Client-Site) phụ thuộc vào cách máy chủ xử lý khi nhận nhiều tham số có cùng tên

Client-Side HPP: các lỗ hổng HPP phía máy khách liên quan đến khả năng đưa các tham số vào một URL, sau đó được trả lại trên trang cho người dùng.

Kẻ tấn công thêm một tham số giống với tham số mặc định nhằm để đánh lừa server

3. Tác hại

Truyền tham số bằng HPP có thể vượt mặt ứng dụng WAF của server.

Nếu server không lọc dữ liệu đúng cách thì sẽ gây hại cho người dùng vì liên kết nằm trên chính trang web của họ.

Hacker có thể gửi cả file thông qua đường dẫn, vì vậy có thể tạo các backlink dẫn tới trang web xấu hoặc gửi shell lên server, từ đó chiếm quyền truy cập của server.

4. Cách phòng chống

Mã hóa đầu ra.

Không sử dụng mã hóa kiểu HTML Entities trên server! Thay vào đó hãy mã hóa URL.

Đảm bảo rằng bạn đã mã hóa đầu vào do người dùng cung cấp bất cứ khi nào bạn thực hiện GET / POST.

Xác thực đầu vào từ các biểu mẫu, tiêu đề, cookie...

III. Cross Site Request Forgery

1. Mô tả

CSRF (Cross Site Request Forgery) là kỹ thuật tấn công bằng cách sử dụng quyền chứng thực của người dùng đối với một website. CSRF là kỹ thuật thuật tấn công vào người dùng, dựa vào đó hacker có thể thực thi những thao tác phải yêu cầu chứng thực. Hiểu đơn giản hơn thì đây là kỹ thuật tấn công dựa vào mượn quyền trái phép.

CSRF (Cross Site Request Forgery) là kỹ thuật tấn công bằng cách sử dụng quyền chứng thực của người dùng đối với một website. CSRF là kỹ thuật thuật tấn công vào người dùng, dựa vào đó hacker có thể thực thi những thao tác phải yêu cầu chứng thực. Hiểu đơn giản hơn thì đây là kỹ thuật tấn công dựa vào mượn quyền trái phép.

2. Tác hại

Mất mát dữ liệu của người dùng

Đánh cắp dữ liệu

Sử dụng thông tin cá nhân trái phép

3. Cách phát hiện

Tìm các thẻ: iframe, link, img... xem src của nó xem có phải đường dẫn lạ hay không?

Sử dụng công cụ burp suite

4. Phòng chống

Nên đăng xuất khỏi các website quan trọng: Tài khoản ngân hàng, thanh toán trực tuyến, các mạng xã hội, gmail... khi đã thực hiện xong giao dịch.

Không click vào các đường dẫn lạ trong mail, facebook...

Không lưu các thông tin tài khoản tại trình duyệt của mình.

Sử dụng captcha, các thông báo xác nhận

Kiểm tra IP

IV. Html Injection

1. Mô tả

Html Injection là một kỹ thuật được sử dụng để tận dụng lợi thế của đầu vào không được xác thực để sửa đổi một trang web.

Khi các website không xác thực dữ liệu người dùng, kẻ tấn công có thể gửi văn bản được định dạng HTML để sửa đổi nội dung trang web được hiển thị cho người dùng khác

Hacker chèn đoạn mã HTML vào website, khi website không xác thực dữ liệu đầu vào và dẫn đến việc hacker hiển thị các nội dung quảng cáo hoặc giả mạo, nhằm đánh cắp thông tin người dùng

2. Tác hại

Trang web ban đầu bị thay đổi cấu trúc

Gây mất thông tin người dùng nếu như người dùng click vào các nội dung giả mạo

3. Cách phát hiện

Sử dụng công cụ burp suite

Test ở các input đầu vào

4. Phòng chống

Kiểm tra nội dung người dùng nhập vào.

Mã hóa nội dung người dùng nhập vào.

Sử dụng tường lửa WAF cho website.

V. CRLF Injection

1. Mô tả

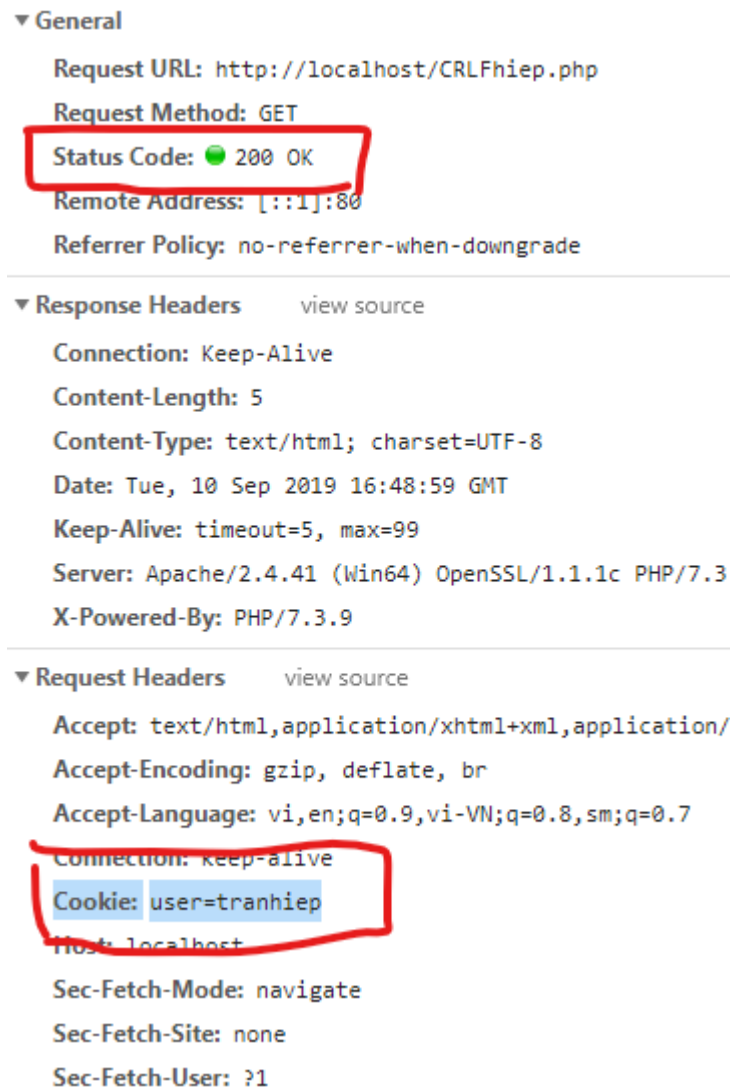
CR và LF là các ký tự điều khiển, được mã hóa tương ứng 0x0D (13 trong hệ thập phân) và 0x0A (10 trong hệ thập phân)

CRLF Injection là một lỗ hổng khi người lập trình không kiểm tra kỹ càng dữ liệu người dùng đẩy lên và cho phép người dùng chèn cả các ký tự CR và LF vào

2. Ví dụ

Kẻ tấn công chèn các kí hiệu CR, LF và các tham số độc hại vào yêu cầu HTTP

Một trang web chuyển hướng người dùng sau khi đăng nhập xong. Phản hồi thông thường



Tuy nhiên tại phần đăng nhập, hacker có thể chèn thêm phần phản hồi giả mạo như sau, phần CRLF được mã hóa là %0d%0a:

```
user: tranhiep
HTTP/1.1 404 Not found
```

Sau đó trang 404 sẽ hiển thị thay vì 200 như bình thường

Giả mạo nhật kí: một ứng dụng web thường có phần nhật kí để truy dấu các yêu cầu HTTP thao tác với nó, giúp lập trình viên có thể tìm và gỡ lỗi dễ dàng hơn. Hacker có thể giả mạo thông tin này nếu thực thi được lỗ hổng CRLF.

3. Tác hại

- Người dùng có thể bị đánh cắp phiên đăng nhập
- Người dùng có thể bị đánh cắp các thông tin nhạy cảm được đưa vào phần Header
- Từ lỗ hổng CRLF có thể khai thác các lỗ hổng khác như XSS

4. Cách phòng tránh

- Lọc và xử lý tất cả dữ liệu người dùng gửi lên, thay thế các kí tự CR và LF bằng các kí tự được mã hóa an toàn
- Sử dụng các ứng dụng WAF

VI. Cross-site Scripting

1. Mô tả

Cross-site scripting (viết tắt: XSS) là một kỹ thuật tấn công bằng cách chèn vào các website động những thẻ HTML hay những đoạn mã script nguy hiểm có thể gây nguy hại cho những nạn nhân sử dụng.

Mã độc được nhúng vào web qua các form không được xử lý chặt chẽ và tạo ra lỗ hổng, hacker dựa vào đó chèn vào các scripts mã độc

2. Ví dụ

Trên trang thông tin cá nhân của bạn, ở ô input nhập liệu bạn nhập đoạn scripts:

```
<script type="text/javascript">  
    window.location="https://github.com/daihieptn97/";  
</script>
```

Những người đó sau khi truy cập vào trang của nhân của bạn nó sẽ chuyển hướng đến trang mà họ muốn

Nó có thể chuyển hướng bạn đến trang web giả mạo giống thật và ăn cắp thông tin tài khoản của bạn

3. Tác hại

- Đánh cắp tài khoản, mật khẩu, ... của người dùng
- Cài các phần mềm hoặc chương trình độc hại lên máy của người dùng mà người dùng không hề hay biết
- Thay đổi nội dung trang web, điều hướng người dùng tới các trang web xấu

4. Cách phòng tránh

- Không nhập thông tin nhạy cảm vào các trang web, nhất là các trang có cảnh báo không bảo mật
- Xử lý, lọc tất cả dữ liệu gửi lên của người dùng
- Bảo mật cookie, không lưu thông tin nhạy cảm của người dùng trong cookie
- Sử dụng các ứng dụng WAF

VII. Template injection

1. Mô tả

- 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.

2. Ví dụ

Ví dụ về template injection trong angular js

```
angular.module('app').controller('AppCtrl', function($scope, injectTemplate) {  
    // sử dụng với jquery  
    injectTemplate(angular.element('#container'), '.child-class', { templateUrl: 'content.tpl.html' }, $scope);  
    // or injectTemplate($('#container'), '.child-class', { templateUrl: 'content.tpl.html' }, $scope);  
});
```

3. Tác hại

- Bị đánh cắp thông tin người dùng
- Tin tặc lợi dụng chúng để tấn công cài đặt mã độc và phá hoại hệ thống

4. Cách phòng tránh

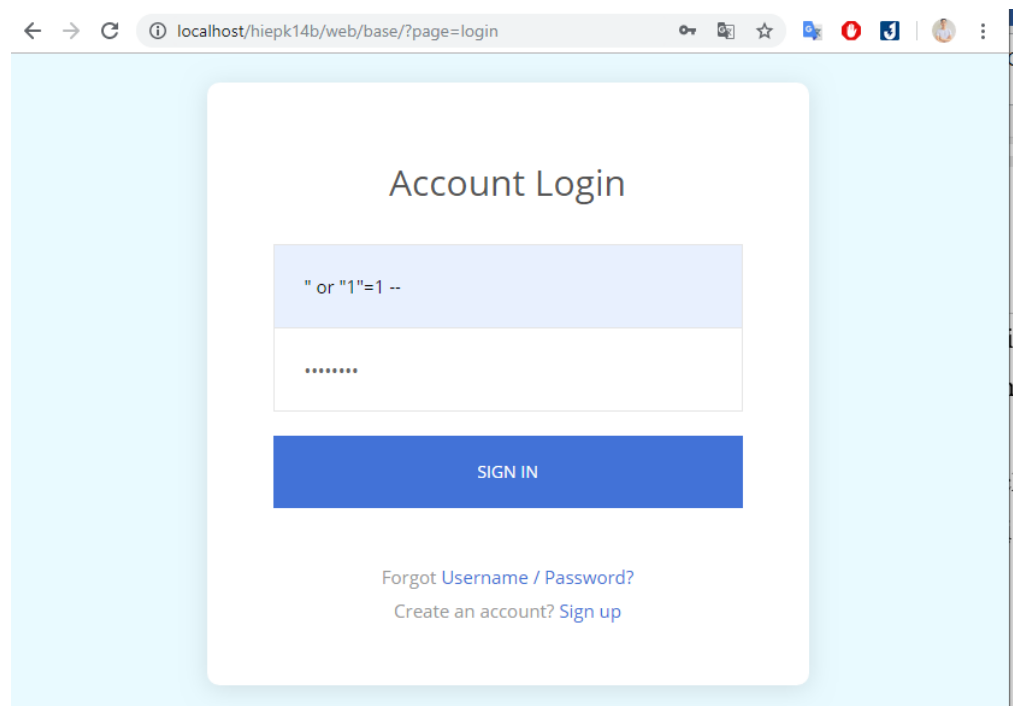
- Kiểm tra các biến nhập vào
- Lọc ra cú pháp biểu thức mẫu từ đầu vào của người dùng

VIII. SQL injection

1. Mô tả

- SQL Injection (còn gọi là SQL Insertion) là một hình thức tấn công trong đó truy vấn SQL của ứng dụng đã bị chèn thêm các tham số đầu vào “không an toàn”.
- SQL injection có thể cho phép những kẻ tấn công thực hiện các thao tác như một người quản trị web, trên cơ sở dữ liệu của ứng dụng.

2. Ví dụ



Và đoạn code xử lý như sau

```
if(isset($_POST['username']) && isset($_POST['password'])){  
    $sql = "SELECT * FROM user WHERE username='".$_POST['username']."' AND password = '".$_POST['password']."'";  
}
```

Và câu truy vấn sẽ trở thành

```
SELECT * FROM `user` WHERE user = "" or "1"=1 -- " AND pass = "220466675e31b9d20c051d5e57974150"
```

Vậy là đã xảy ra lỗi SQL Injection

3. Tác hại

- Bị đánh cắp cơ sở dữ liệu
- Có thể bị xóa toàn bộ các bảng dữ liệu

- Có thể bị tạo các bảng dữ liệu mới
- Bị hacker điều khiển hệ thống

4. Cách phòng tránh

- Luôn ràng buộc kiểu dữ liệu.

Ví dụ ta có đường link lấy bài viết dựa vào id



Giả sử id ở đây dạng số thì ta có thể làm như sau:

```
}

function getType()
{
    $typeBook = (int) $_GET['id'];
    if ($typeBook) {
```

- Sử dụng Regular Expression vd như

```
$reg = "/^(\\s*)select\\s*?.*?\\s*?from(\\s|\\[\\^;]|(\\[\"\\].*\"\\])*?;\\s*?$/i";
```

Hoặc đơn giản là

```
$id = isset($_GET['id']) ? $_GET['id'] : false;
$id = str_replace('/[^0-9]/', '', $id);
```

- Dùng các hàm có sẵn để giảm thiểu lỗi có sẵn trong các ngôn ngữ ví dụ như `mysqli_real_escape_string` với php
- Sử dụng các framework như laravel, codeigniter
- Làm sạch dữ liệu đầu vào
- Xây dựng truy vấn theo mô hình hóa
- Dùng các công cụ kiểm tra để kiểm tra lỗi để khắc phục kịp thời như sqlmap, BSQL Hacker, SQLninja ...

IX. Server Side Request Forgery

1. Mô tả

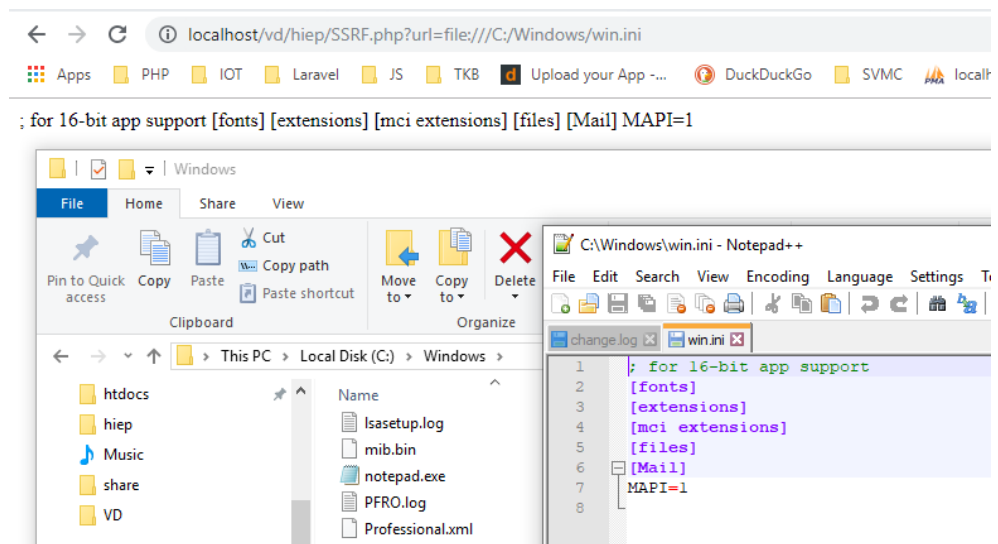
- SSRF (Server Side Request Forgery) hay còn gọi là tấn công yêu cầu giả mạo từ phía máy chủ cho phép kẻ tấn công thay đổi tham số được sử dụng trên ứng dụng web để tạo hoặc kiểm soát các yêu cầu từ máy chủ để bị tấn công.

2. Ví dụ

```
<?php

if (isset($_GET['url'])){
    $url = $_GET['url'];
    $a = fopen($url, 'rb');
    fpassthru($a);
}
```

kẻ tấn công dựa vào đường link url để đọc được các file quan trọng của hệ thống:



3. Tác hại

- Lợi dụng mối quan hệ tin cậy với máy chủ bị ảnh hưởng
- Quét các mạng nội bộ hoặc bên ngoài
- Đọc tệp từ máy chủ

4. Cách phòng chống

- Danh sách trắng và độ phân giải DNS
- Xử lý đáp ứng: Đảm bảo rằng phản hồi nhận được từ máy chủ từ xa thực sự là những gì máy chủ mong đợi là quan trọng để ngăn chặn bất kỳ dữ liệu phản ứng không lường trước được rò rỉ cho kẻ tấn công.

- Tất lược đồ URL không sử dụng: Nếu ứng dụng của bạn chỉ sử dụng HTTP hoặc HTTPS để thực hiện yêu cầu, chỉ cho phép các lược đồ URL đó. Vô hiệu hóa lược đồ URL không sử dụng

X. XML External Entity Vulnerability

1. Mô tả

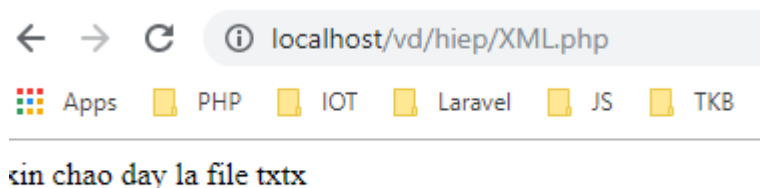
- Kỹ thuật tấn công này dựa vào việc cho phép khai báo External Entity trong phần DTD của dữ liệu XML
- Attacker có thể khai báo một entity để đọc nội dung của file bất kỳ trong hệ thống nếu trình phân tích XML

2. Ví dụ

- Ta có đoạn mã XML như sau:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY>
  <!ENTITY xxe SYSTEM
    "http://localhost/vd/hiep/ssrf.text">
]>
<foo>
  &xxe;
</foo>
```

- Sau khi chạy php và đọc file xml ra ta được nội dung của file trong hệ thống



xin chào đây là file txtx

3. Tác hại

- Thông tin server bị lộ.
- Hacker có thể khai thác lỗ hổng này để tấn công SSRF.
- Mất tính bảo mật của website

4. Cách phòng chống

- Để tránh bị khai thác lỗ hổng này thì cần cấu trúc trình phân tích XML Parser không cho phép sử dụng khai báo external entity trong phần định kiểu tài liệu DTD

XI. Remote Code Execution

1. Mô tả

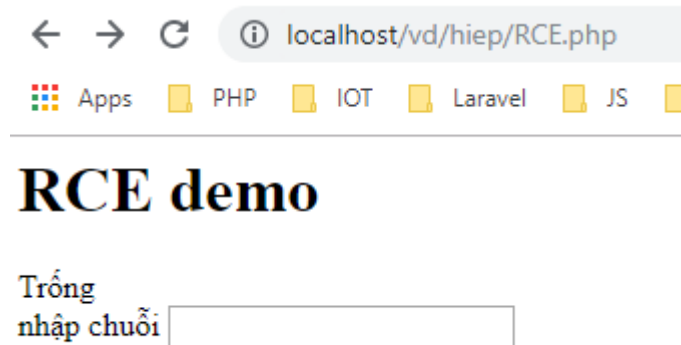
- Tức là bạn có thể thông qua một kỹ thuật nào đó để có thể đạt được quyền điều khiển trên máy nạn nhân, thông qua đó có thể thực thi bash, shell...hoặc code của một vài ngôn ngữ kịch bản như python, perl, javascript...
- Xảy ra Remote Code Execution là do không kiểm soát, lọc dữ liệu đầu vào từ người dùng đúng cách.

2. Ví dụ

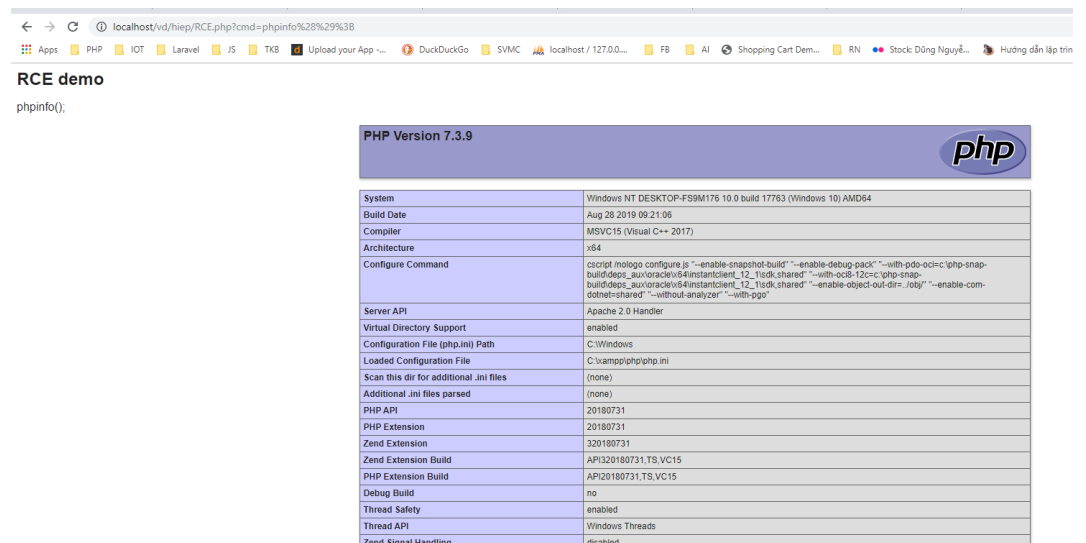
- Có đoạn code như sau:

```
<html>
  <body>
    <h1>RCE demo</h1>
    <?php
      if(isset($_GET['cmd'])){
        $cmd = $_GET['cmd'];
        echo $cmd;
        eval($cmd);
      }else{
        echo 'Trống';
      }
    ?>
    <form>
      nhập chuỗi
      <input name="cmd" />
    </form>
  </body>
</html>
```

- Nhập hàm “phpinfo” vào form:



- Và xảy ra lỗi RCE:



3. Tác hại

- Bị mất bảo mật
- Lộ thông tin quan trọng
- Là tiền đề để tấn công bằng các cuộc tấn công khác
- Mất tính toàn vẹn
- Có thể bị cài phần mềm độc hại bằng web shell

4. Cách phòng chống

- Sử dụng trình quét lỗ hổng Acunetix
- Kiểm soát tốt dữ liệu đầu vào bằng cách kiểm tra đầu vào xem có phải là string không bằng hàm “is_string”, hoặc sử dụng các biểu thức chính quy để kiểm tra


```
<?php
putenv('PATH=/home/rcseservice/jail');

if (isset($_REQUEST['cmd'])) {
    $json = $_REQUEST['cmd'];

    if (is_string($json)) {
        echo "Phát hiện hack<br/><br/>";
    } elseif (preg_match('/^.*(alias|bg|bind|break|builtin|case|cd|command|compgen|complete|continue|declare|dirs|disown|echo|enable|
eval|exec|exit|export|fc|fg|getopts|hash|help|history|if|jobs|kill|let|local|logout|popd|printf|pushd|pwd|read|readonly|return|set
|shift|shopt|source|suspend|test|times|trap|type|typeset|ulimit|umask|unalias|unset|until|wait|while|[\x00-\x1FA-Z0-9!#-~\;-\@\[
-~\x7F]+).*$/',$json)) {
        echo "Hacking attempt detected<br/><br/>";
    } else {
        echo "Cố gắng chạy lệnh:<br/>";
        $cmd = json_decode($json, true)['cmd'];
        if ($cmd !== NULL) {
            system($cmd);
        } else {
            echo "Đầu vào không hợp lệ";
        }
        echo "<br/><br/>";
    }
}
}
```

- Chỉ dùng hàm “eval()” khi thực sự cần thiết, nếu dùng thì cần ề cao việc kiểm soát dữ liệu trong user input validation

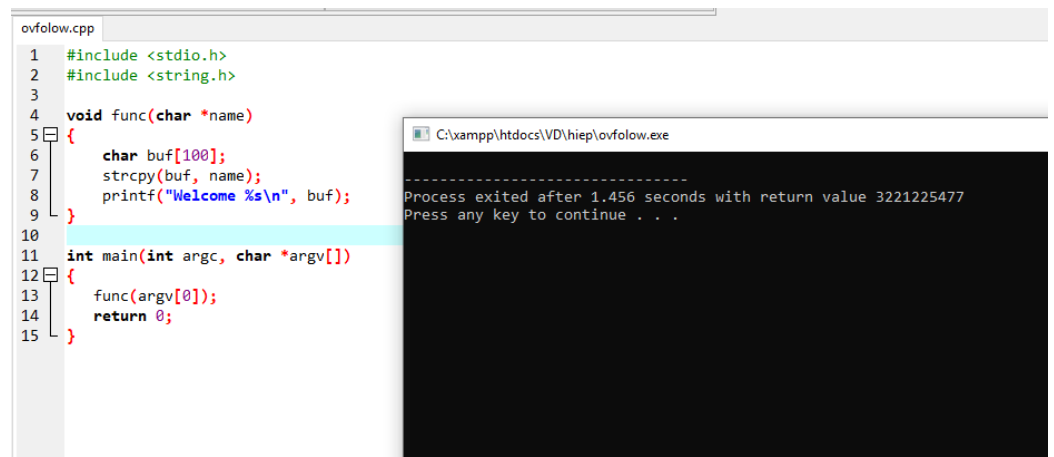
XII. Memory

1. Mô tả

- Buffer Overflow Lỗ hổng tràn bộ đệm trên stack. Lỗ tràn bộ đệm là khi bộ nhớ bị ghi đè nhiều lần trên ngăn xếp. Thông thường nó xảy ra do người dùng gửi một lượng lớn dữ liệu đến server ứng dụng, kết quả là dữ liệu đó sẽ đè lên các vị trí bộ nhớ liên kề.
- Tấn công buffer overflow có nguyên nhân gần giống với tấn SQL Injection, khi người dùng hay hacker cung cấp các biên đầu vào hay dữ liệu vượt quá khả năng xử lý của chương trình làm cho hệ thống bị treo, dẫn tới từ chối dịch vụ hay có khả năng bị các hacker lợi dụng chèn các chỉ thị trái phép nhằm thực hiện các đoạn mã nguy hiểm từ xa

2. Ví dụ

- Tràn bộ nhớ do kích thước quá lớn.
- argv [0] lớn hơn 8 byte, với hệ điều hành 32 bit (4 byte), chúng ta phải lấp đầy bộ nhớ từ kép (32 bit)

The image shows a code editor window titled 'ovfollow.cpp' on the left and a command prompt window titled 'C:\xampp\htdocs\VD\hiiep\ovfollow.exe' on the right. The code in the editor is a C++ program with a function 'func' and a 'main' function. The 'main' function calls 'func' with 'argv[0]'. The command prompt shows the output of the program, which is a series of dashes followed by the message 'Process exited after 1.456 seconds with return value 3221225477' and 'Press any key to continue . . .'.

```
1 #include <stdio.h>
2 #include <string.h>
3
4 void func(char *name)
5 {
6     char buf[100];
7     strcpy(buf, name);
8     printf("Welcome %s\n", buf);
9 }
10
11 int main(int argc, char *argv[])
12 {
13     func(argv[0]);
14     return 0;
15 }
```

Process exited after 1.456 seconds with return value 3221225477
Press any key to continue . . .

- Chương trình đã bị tràn bộ nhớ đệm và không thể vào “welcome”

3. Tác hại

- Làm hệ thống bị treo
- Chèn các chỉ thị trái phép nhằm thực hiện các mã lệnh nguy hiểm
- Làm rò rỉ các bộ nhớ đệm khác
- Có thể bị ghi đè hoặc làm hỏng những gì họ đang dữ

4. Cách phòng chống

- Loại bỏ các ký tự đặc biệt và lọc các chuỗi không chứa kí tự là chữ số hoặc chữ cái.
- Lựa chọn ngôn ngữ lập trình: ngôn ngữ lập trình có một ảnh hưởng lớn đối với sự xuất hiện lỗi tràn bộ đệm
- Sử dụng các thư viện an toàn: các thư viện được viết tốt và đã được kiểm thử dành cho các kiểu dữ liệu trừu tượng mà các thư viện này thực hiện tự động việc quản lý bộ nhớ
- Xác nhận đầu vào trước khi sử lý