I. OPEN REDIRECT VULNERABILITIES 2
1. MÔ TẢ 2
2. VÍ DỤ 2
3. TÁC HẠI 3
4. CÁCH PHÁT HIỆN 3
5. CÁCH NGĂN CHẶN 3

II. HTTP PARAMETER POLLUTION 3
1. MÔ TẢ 3
2. VÍ DỤ 3
3. TÁC HẠI 4
4. CÁCH PHÒNG CHỐNG 4

III. CROSS SITE REQUEST FORGERY 5
1. MÔ TẢ 5
2. TÁC HẠI 5
3. CÁCH PHÁT HIỆN 5
4. PHÒNG CHỐNG 5

IV. HTML INJECTION 6
1. MÔ TẢ 6
2. TÁC HẠI 6
3. CÁCH PHÁT HIỆN 6
4. PHÒNG CHỐNG 6

V. CRLF INJECTION 6
1. MÔ TẢ 6
2. VÍ DỤ 7
3. TÁC HẠI 8
4. CÁCH PHÒNG TRỐNG 8

VI. CROSS-SITE SCRIPTING 8
1. MÔ TẢ 8
2. VÍ DỤ 8
3. TÁC HẠI 9
4. CÁCH PHÒNG TRỐNG 9

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 nhận. 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 kĩ càng trong back-end.

-
```php
<?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 khách

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 HTMLEntities 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 hiên 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 kĩ thuật được sử dụng để tận dung 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 webstie, khi website không xác thực dữ liệu đầu vào và dẫ 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 WAP cho website.

V. CRLF Injection

1. Mô tả

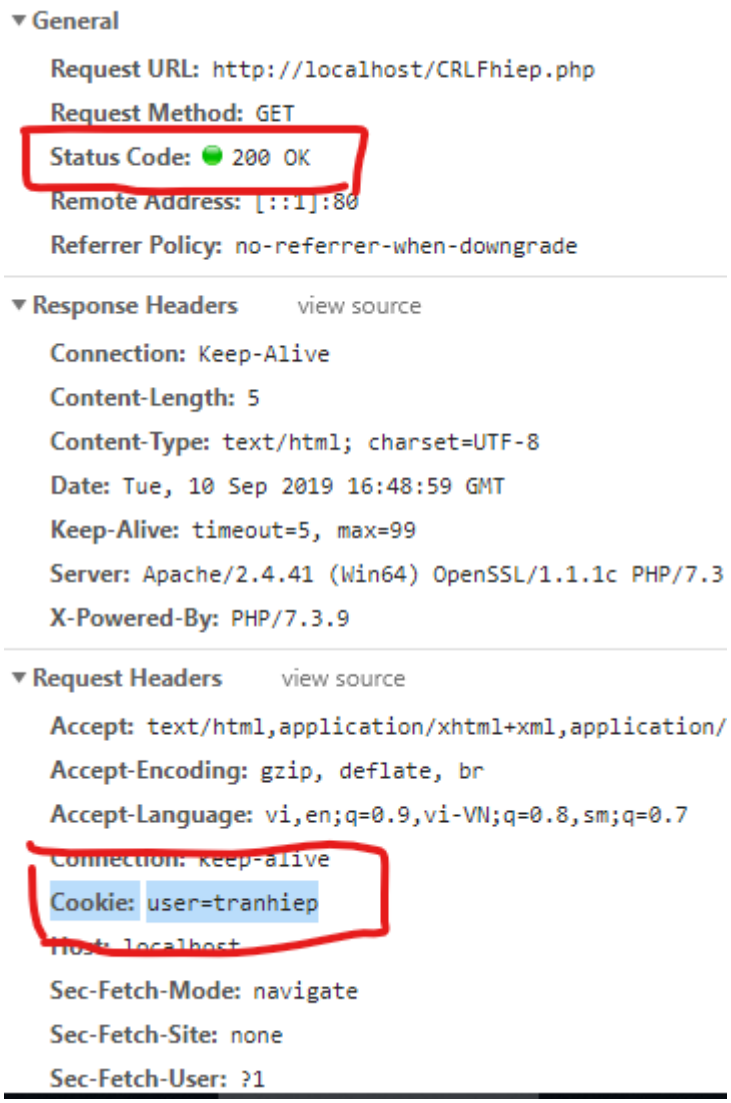
CR và LF là các ký tự điều khiển, được mã hóa tương ứng 0x0D (13 trong hệ thập phân) và 0x0A (10 trong hệ thập phân)

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

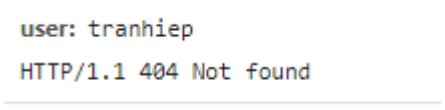
2. Ví dụ

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:



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 ngạy cảm được đưa vào phần Header
- Từ lỗ hổng CRLF có thể khai thác các lỗ hổng khác như XSS

4. Cách phòng trống

- Lọc và xử lí 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 kĩ 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 sử 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 trống

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