

Báo Cáo Cuối Kỳ

Trường Đại Học Sư Phạm TP.HCM

Khoa Công Nghệ Thông Tin



Môn: Bảo Mật An Ninh Mạng

GV hướng dẫn: Thầy Lương Trần Hy Hiến

Thực hiện: Nhóm 13

* Cao Cảnh Linh – 42.01.104.078
* Cao Đức Minh - 42.01.104.082
* Đỗ Phúc Hậu – 42.01.104.043
* Nguyễn Minh Vương – 42.01.104.195
* Nguyễn Thiện Thanh – 42.01.104.160
* Nguyễn Hoàng Anh Tú - 42.01.104.148

XSS, CSRF

CHÈN MÃ SCRIPT

Nội dung

[**I. Giới thiệu chung** 3](#_Toc7783153)

[**II. Cross Site Scripting (XSS)** 3](#_Toc7783154)

[**1. Khái niệm** 3](#_Toc7783155)

[**2. Các dạng tấn công** 4](#_Toc7783156)

[**3. Cách thức và phương pháp tấn công** 7](#_Toc7783157)

[**4. Cách phòng chống và bảo mật** 10](#_Toc7783158)

[**III. Cross-site Request Forgery (CSRF)** 11](#_Toc7783159)

[1. Khái niệm 11](#_Toc7783160)

[**2. Các dạng tấn công** 12](#_Toc7783161)

[**3. Cách thức và phương pháp tấn công** 12](#_Toc7783162)

[**4. Cách phòng chống và bảo mật** 19](#_Toc7783163)

[IV. Tài liệu tham khảo 20](#_Toc7783164)

[1.Tiếng Việt 20](#_Toc7783165)

[2. Tiếng Anh 20](#_Toc7783166)

# **I. Giới thiệu chung**

Website ngày nay rất phức tạp và thường là các web động, nội dung của web được cập nhật thông qua các thành viên tham gia ở khắp mọi nơi trên thế giới. Và hầu hết các website này dùng Cookie để xác thực người dùng. Điều này đồng nghĩa với việc Cookie của ai thì người đó dùng, Nếu lấy được Cookie người dùng nào Hacker sẽ giả mạo được chính người dùng đó(điều này là hết sức nguy hiểm). Vậy làm sao để các hacker có thể lấy cookie của bạn? Có rất nhiều cách để các hacker làm việc đó, ở đây tôi xin trình bày một trong những cách mà hacker thường dùng, đó chính là họ nhờ vào lỗi Cross Site Scripting(XSS). Cross-Site Scripting (XSS) là một trong những kĩ thuật tấn công phổ biến nhất hiện nay, đồng thời nó cũng là một trong những vấn đề bảo mật quan trọng đối với các nhà phát triển web và cả những người sử dụng web. Bất kì một website nào cho phép người sử dụng đăng thông tin mà không có sự kiểm tra chặt chẽ các đoạn mã nguy hiểm thì đều có thể tiềm ẩn các lỗi XSS.

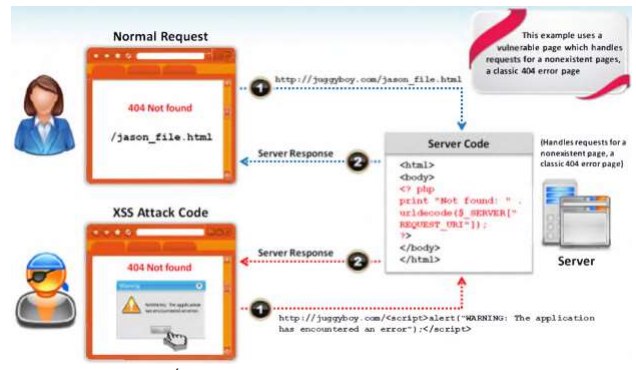
# **II. Cross Site Scripting (XSS)**

## **1. Khái niệm**

Cross-Site Scripting hay còn được gọi tắt là XSS là một kĩ thuật tấn công bằng cách chèn vào các website động (ASP, PHP, CGI, JSP …) những thẻ HTML hay những đoạn mã script nguy hiểm có thể gây nguy hại cho những nạn nhân sử dụng. Trong đó, những đoạn mã nguy hiểm được chèn vào hầu hết được viết bằng các Client-Site Script như JavaScript, Jscript và cũng có thể là các thẻ HTML.

Phụ thuộc vào mục đích của hacker, những đoạn Javascript được chèn vào để lấy những thông tin như:

* **Cookie**: hacker có thể lấy được cookie của người dùng và dùng những thông tin trong cookie để giả mạo phiên truy cập hoặc lấy những thông tin nhạy cảm khác được lưu trong cookie.
* **Keylogging**: hacker có thể ghi lại những thao tác gõ phím của người dùng bằng cách sử dụng sự kiện trong Javascript và gửi tất cả những thao tác gõ phím đó về cho hắn để thực hiện những mục đích như đánh cắp các thông tin nhạy cảm, lấy mật khẩu truy cập website hoặc mã số thẻ tín dụng…
* **Phishing**: hacker có thể thay đổi giao diện của website bằng cách thay đổi cấu trúc HTML trong trang web để đánh lừa người dùng. Hacker có thể tạo ra những form đăng nhập giả nhằm lừa người dùng đăng nhập vào để đánh cắp mật khẩu.



**Hình 1**. Quy trình tấn công XSS đơn giản

## **2. Các dạng tấn công**

Nếu xét theo phương thức xử lý dữ liệu của Website thì có tấn công qua phương thức GET và POST

Nếu xét theo nguyên nhân gây ra lỗi XSS nói chung thì có 2 dạng:

**Server XSS**

Tấn công Server XSS xảy ra khi người sử dụng không đáng tin cậy cung cấp dữ liệu được bao gồm trong một phản hồi HTML được tạo ra bởi các máy chủ. Nguồn dữ liệu này có thể là từ yêu cầu, hoặc từ một địa điểm lưu trữ. Như vậy, bạn có thể có cả hai Server XSS lưu trữ và máy phản hồi Reflected XSS Server.

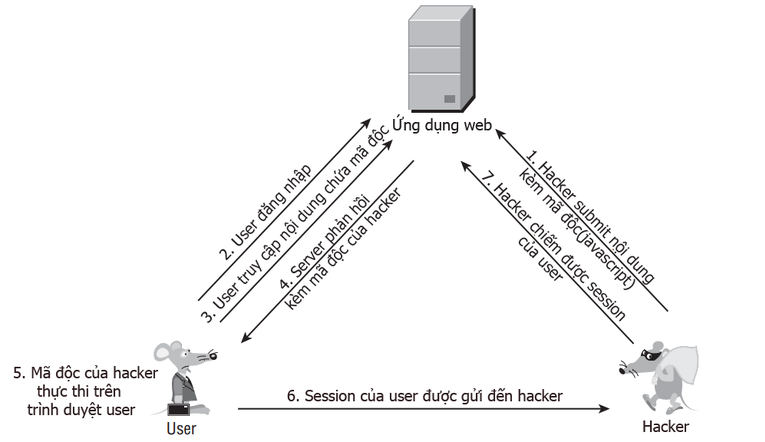
**Client XSS**

Tấn công Client XSS xảy ra khi người sử dụng không đáng tin cậy cung cấp dữ liệu được sử dụng để cập nhật các DOM (Document Object Model) với một truy vấn JavaScript không an toàn. Một truy vấn JavaScript được coi là không an toàn nếu nó có thể được sử dụng để giới thiệu các đoạn mã JavaScript hợp lệ vào trong DOM. Nguồn này các dữ liệu này có thể là từ DOM, hoặc nó có thể đã được gửi đi bởi các máy chủ (thông qua một cuộc gọi AJAX, hoặc tải trang). Nguồn cuối cùng của dữ liệu có thể có được từ một yêu cầu, hoặc từ một vị trí lưu trữ trên máy khách hoặc máy chủ. Như vậy, bạn có thể có cả hai máy phản hồi XSS và máy lưu trữ Client XSS.

**Chi tiết hơn của 2 dạng trên là:**

**Stored XSS** thường xảy ra khi người dùng nhập dữ liệu vào được lưu trữ trên máy chủ mục, và sau đó là một nạn nhân có thể lấy dữ liệu được lưu trữ từ các ứng dụng web mà không cần qua phương pháp an toàn nào khi biểu diễn trên trình duyệt. Cụ thể -Stored XSS là lỗi XSS mà đoạn mã chèn thêm vào được lưu trữ trên server, như trong CSDL dưới dạng các comment trong blog, message trong forum hoặc các visitor log. Khi người dùng vào trang này, thì lập tức các script độc được thực thi, ở đây là sẽ tiến hành lấy trộm cookie của người dùng, rồi lưu nó vào một file txt. Stored XSS phát sinh do chúng ta không lọc dữ liệu do thành viên gởi lên một cách đúng đắn, khiến cho mã độc được lưu vào Database của website

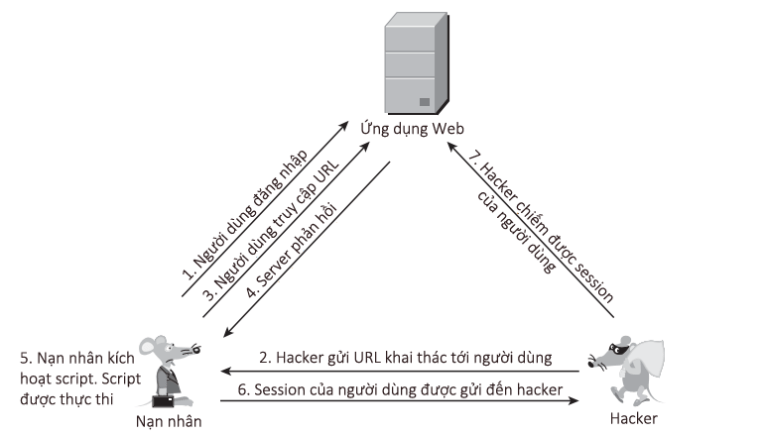
Kịch bản khai thác được mô tả như hình sau:

 **Hình 2:** Quy trình khai thác **Stored XSS**

**Reflected XSS** xảy ra khi người dùng nhập vào và ngay lập tức quay trở lại bởi một ứng dụng web trong đó có các thông báo lỗi, form kết quả tìm kiếm, form tìm kiếm, hoặc bất kỳ hồi đáp nào khác bao gồm một số hoặc tất cả các đầu vào được cung cấp bởi người sử dụng như một phần của yêu cầu, mà không dữ liệu không cần qua phương pháp an toàn nào khi biểu diễn trên trình duyệt. Cụ thể Kiểu tấn công này áp dụng đối với những form để nhập dữ liệu đầu vào, đặc biệt là các form search. Từ khóa tìm kiếm mà ta nhập vào ô textbox của các Website được hiển thị lại trên trình duyệt. Lợi dụng việc không kiểm soát giá trị này, Hacker thể chèn thêm đoạn mã gây hại vào.

**DOM XSS** là kiểu khai thác XSS khác đi ngược lại với đặc điểm chung là các đoạn mã nguy hiểm sau khi được chèn vào sẽ được thực thi sau respond của server, có nghĩa là lỗi nằm về phía server còn ở đây mã độc được thực thi ngay khi xử lý phía client mà không thông qua server DOM viết tắt của Document Object Model là 1 dạng chuẩn của W3C đưa ra nhằm để truy xuất và thao tác dữ liệu của tài liệu có cấu trúc như HTML, XML. Mô hình này thể hiện tài liệu dưới dạng cấu trúc cây phân cấp. Tất cả các thành phần trong HTML, XML đều được xem như một node. DOM Based XSS là kỹ thuật khai thác XSS dựa trên việc thay đổi cấu trúc DOM của tài liệu, cụ thể là HTML và sự nguy hiểm của DOMBase XSS chính là với JavaScript.

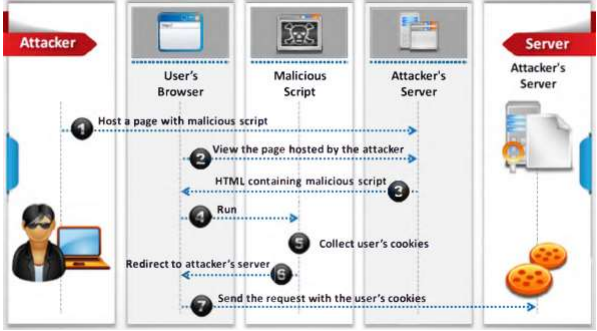
Kịch bản khai thác được mô tả như hình sau:



**Hình 3:** Quy trình khai thác **Stored XSS**

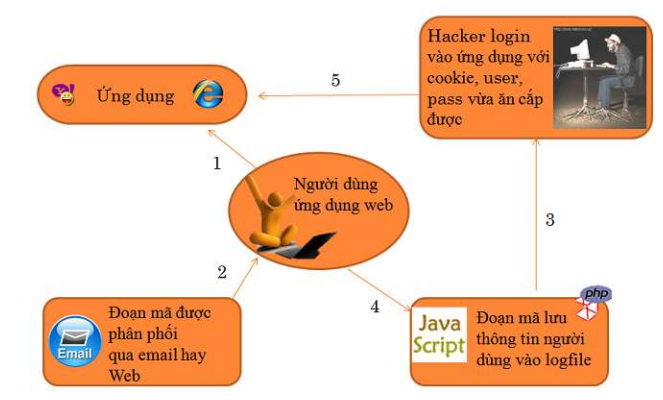


**Hình 4.** Ví dụ tấn công XSS thông qua Email



**Hình 5.** Ví dụ tấn công XSS bằng ăn cắp Cookies

## **3. Cách thức và phương pháp tấn công**



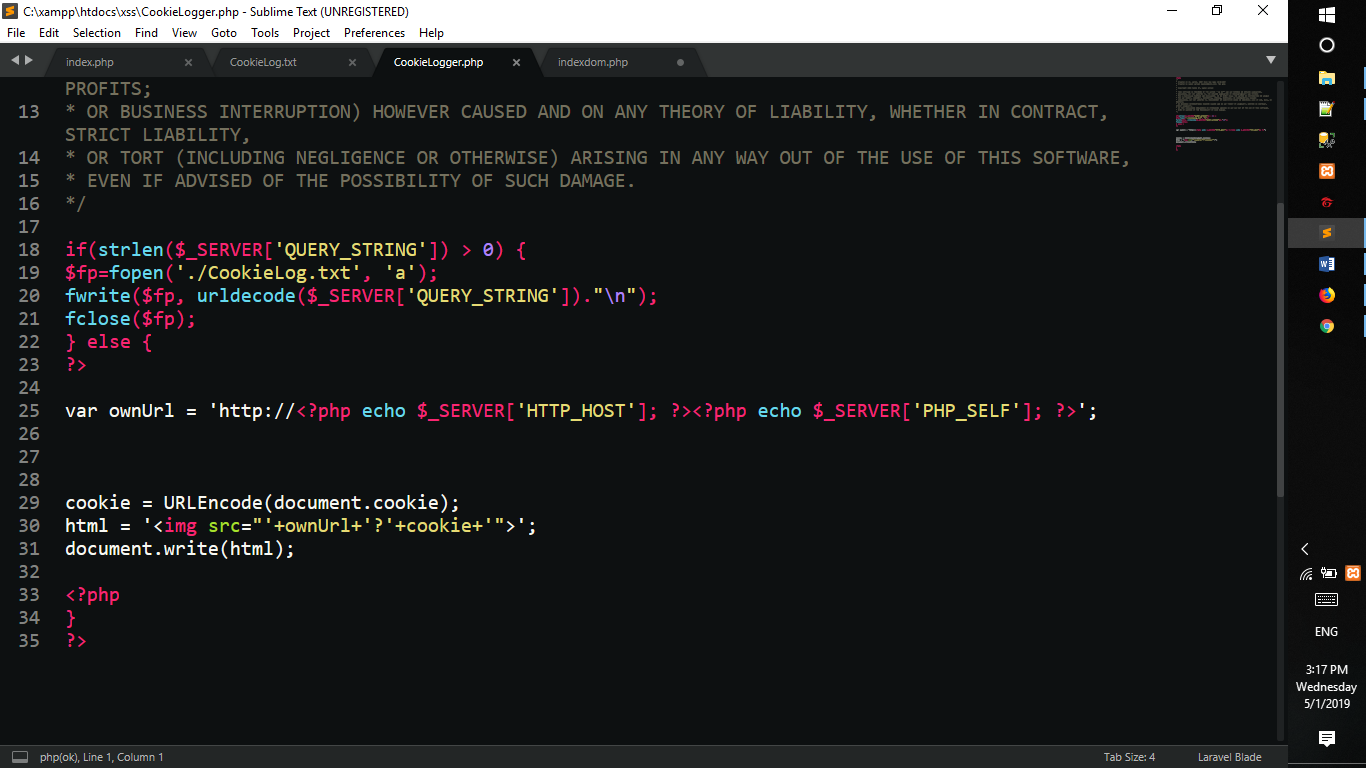
**Hình 6 .** Quy trình của tấn công XSS

**Giải thích quy trình tấn công XSS**

Hacker biết được người dùng đang sử dụng một ứng dụng Web có lỗ hổng XSS. Người dùng nhận được 1 liên kết thông qua email hay trên chính trang Web (như trên guestbook, banner dễ dàng thêm 1 liên kết do chính hacker tạo ra…). Thông thường hacker khiến người dùng chú ý bằng những câu kích thích sự tò mò của người dùng như “ Kiểm tra tài khoản”, “Một phần thưởng hấp dẫn đang chờ bạn”… Chuyển nội dung thông tin (cookie, tên, mật khẩu…) về máy chủ của hacker. Hacker tạo một chương trình hoặc một trang Web để ghi nhận những thông tin đã đánh cắp vào 1 tập tin. Sau khi nhận được thông tin cần thiết, hacker có thể sử dụng để thâm nhập vào tài khoản của người dùng.

**Kỹ thuật lấy Cookie**

**Bước 1:** Tạo file đánh cắp cookie có tên là CookieLogger.php với nội dung

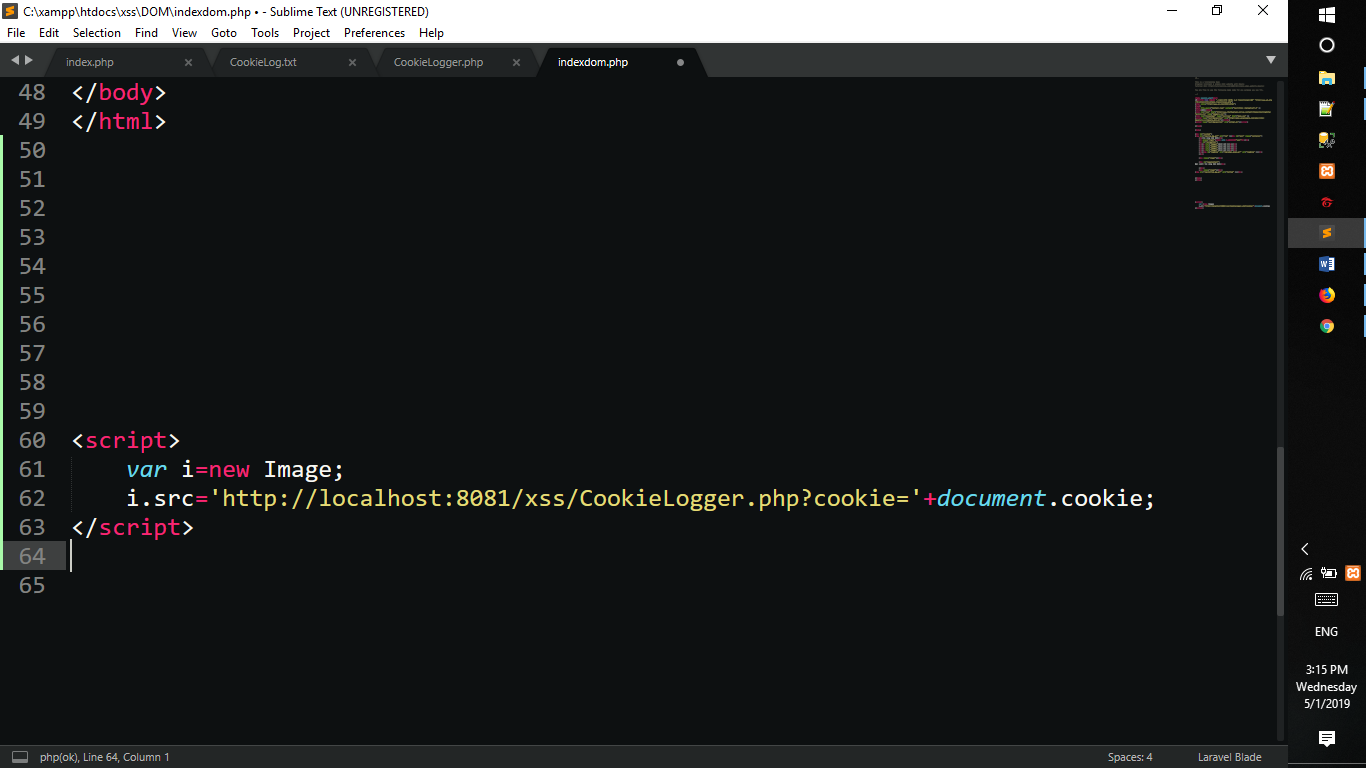
\

File này có nhiệm vụ đánh cắp cookie của victim và ghi thông tin vào file CookieLog.txt

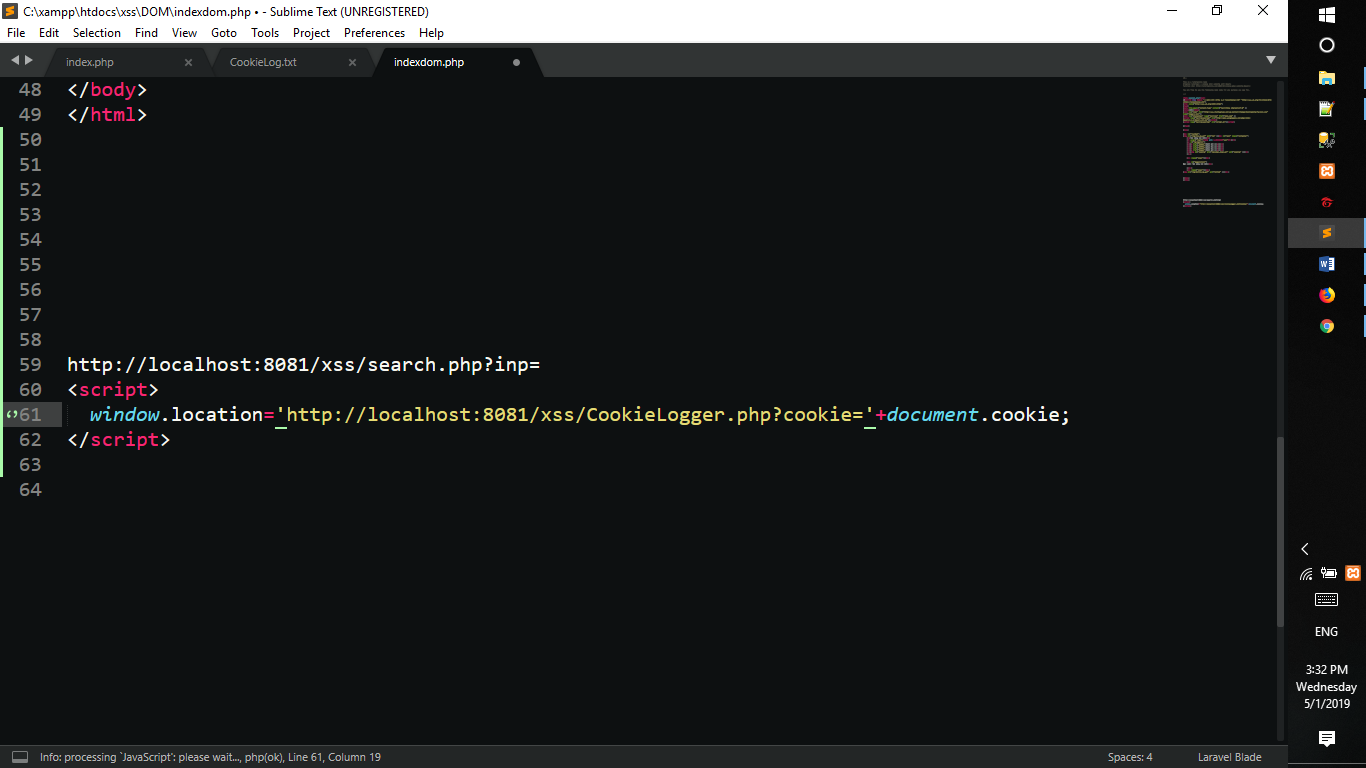
**Bước 2:** Up các file lên host của bạn

UP lên host 2 file CookieLogger.php và CookieLog.txt. Trong đó file CookieLogger.php có nội dung như trên và file CookieLog.txt là file rỗng để lưu trữ toàn bộ thông tin của http://localhost:8081/xss được gửi về thông qua mệnh lệnh được đưa ra từ file CookieLogger.php. Lưu ý, phải chmod file log.txt về 777

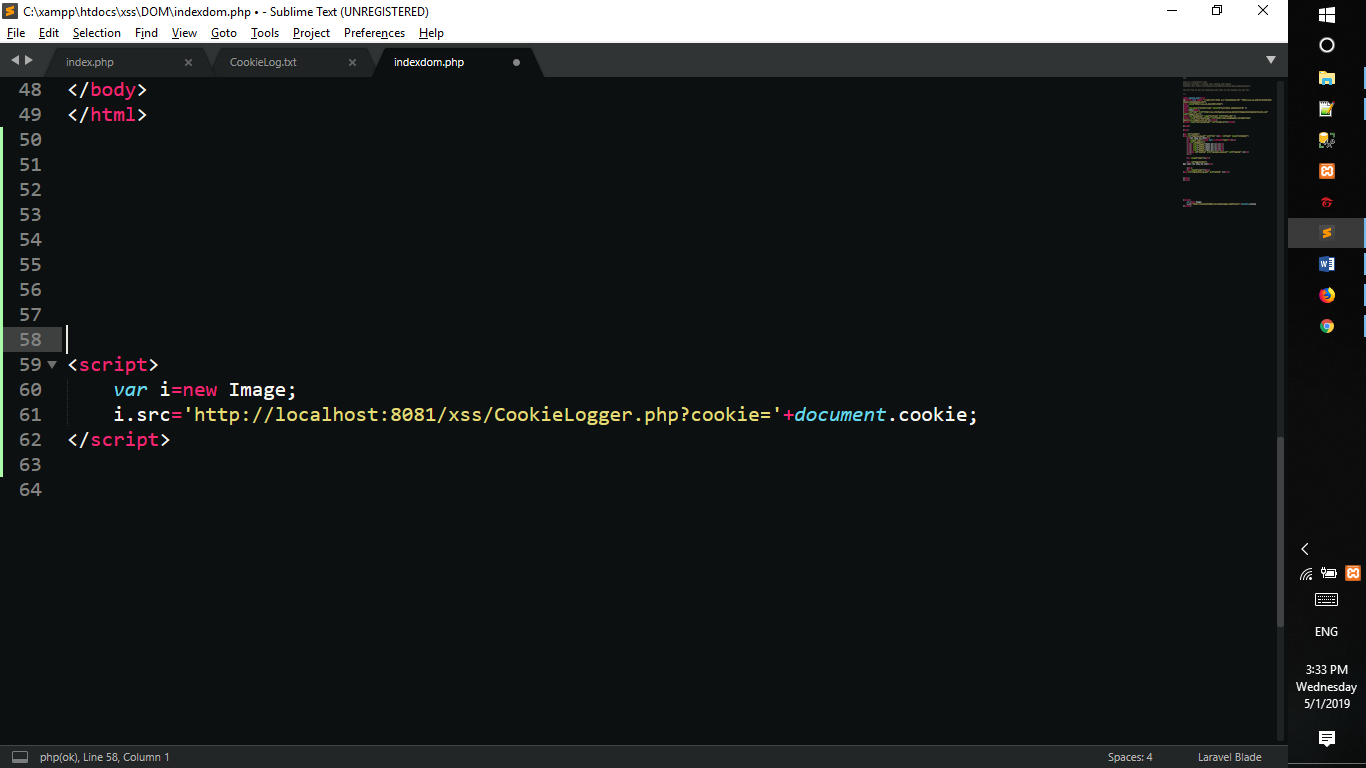
**Bước 3:** Khai thác lỗ hổng XSS - Giả sử bạn up 2 file của bạn lên host của site http://localhost:8081/xss thì đoạn mã script ăn cắp cookies có dạng như sau



- Chèn đọa mã vào site dính lỗi XSS là http://localhost:8081/xss ta được link tương tự như sau

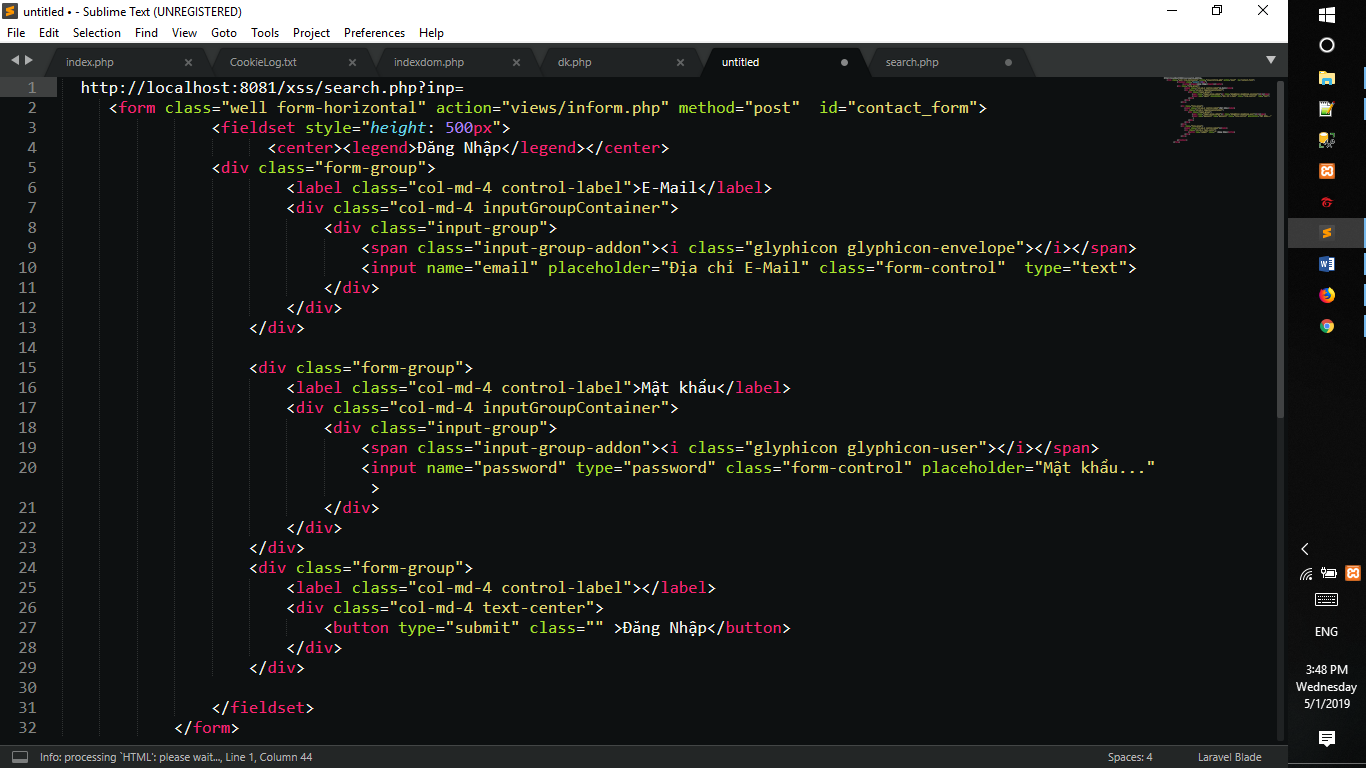


Trong nội dung comment ta sẽ viết một đoạn script và nên đặt cuối comment để tránh bị phát hiện. Ở đây ta sử dụng tạo mới một đối tương image, src của nó dẫn tới trang web của Hacker, trang web này có tác dụng lấy cookie của người dùng nếu người dùng đã đăng nhập



**Kỹ thuật đánh cắp Anccount**

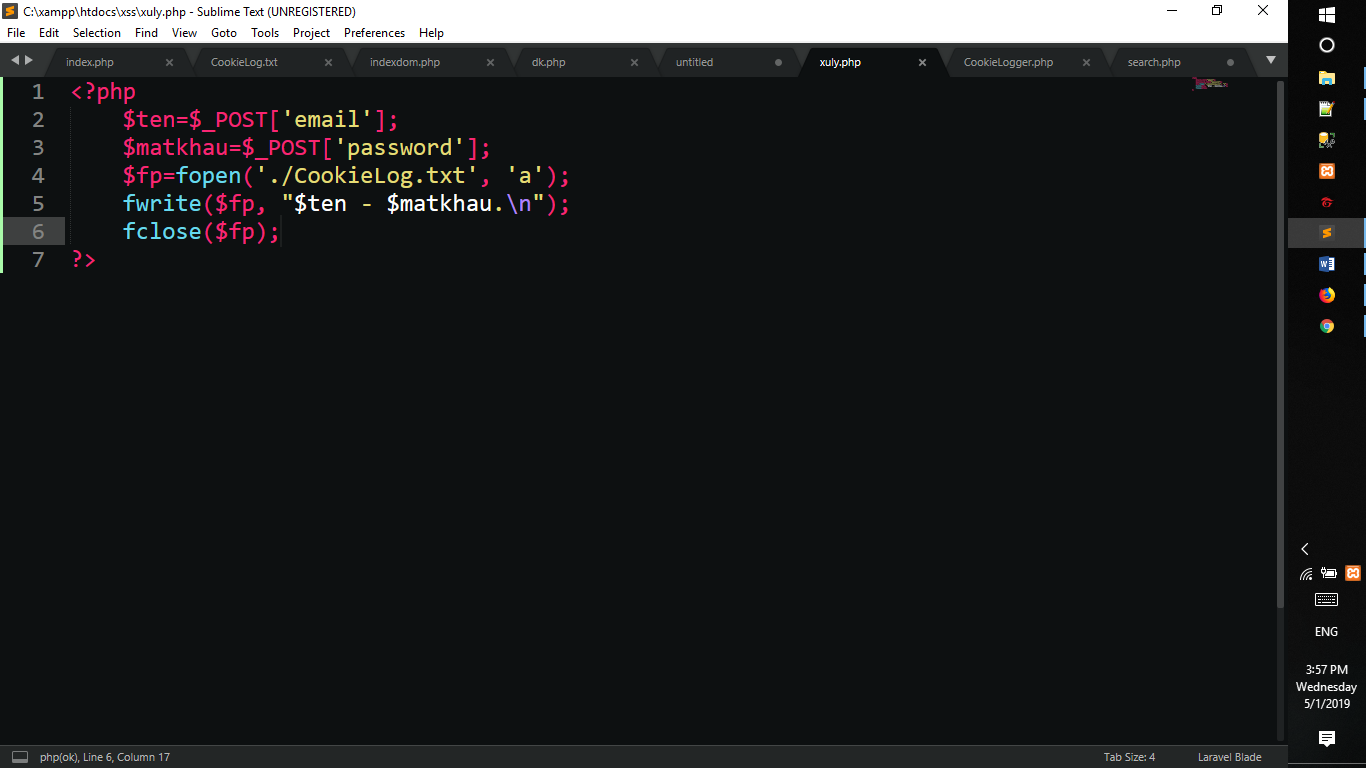
**Bước 1:** Giả mạo trang đăng nhập để lấy thông tin tài khoản



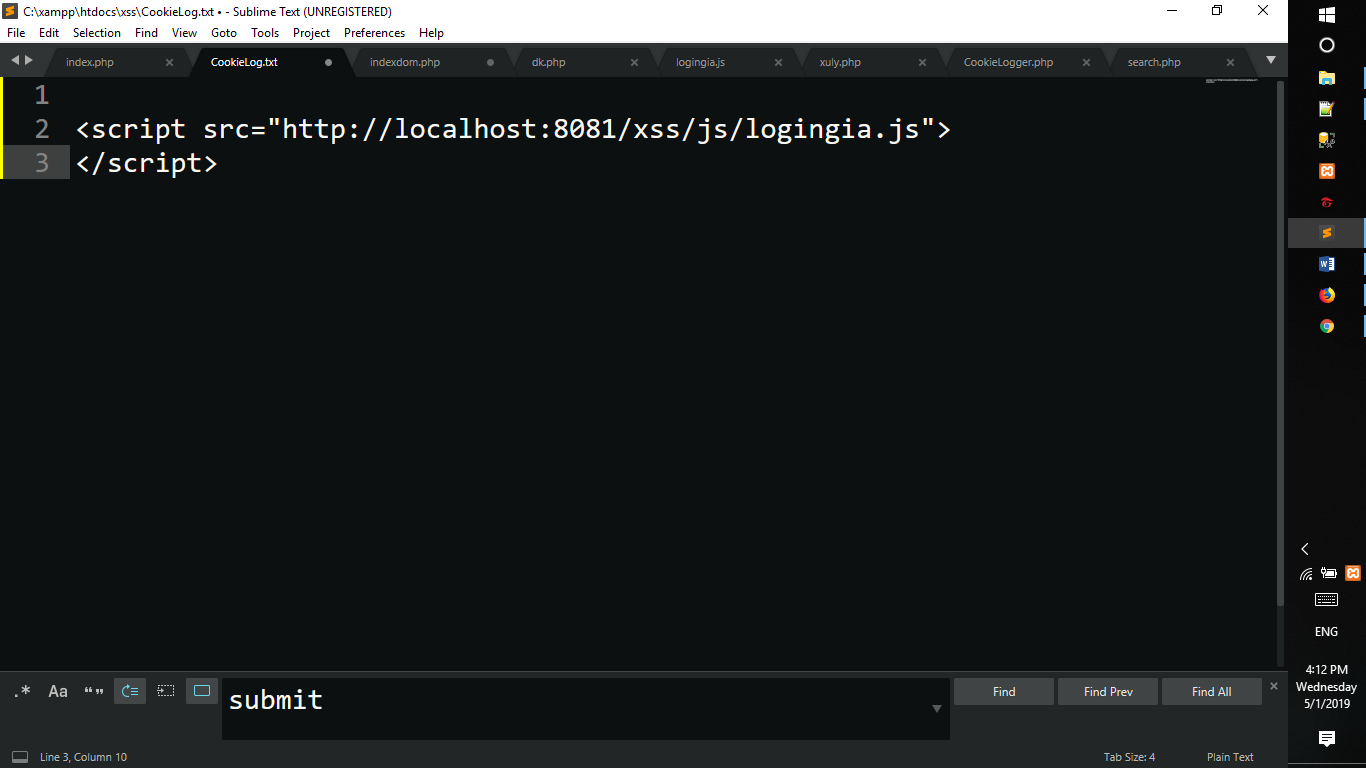
Dưới cùng của Biểu mẫu

Hacker sẽ chèn vào một form đăng nhập , action sẽ chuyển về trang web xử lý của hacker. Khi nạn nhân thực hiện việc đăng nhập thì sẽ bị lấy thông tin tài khoản

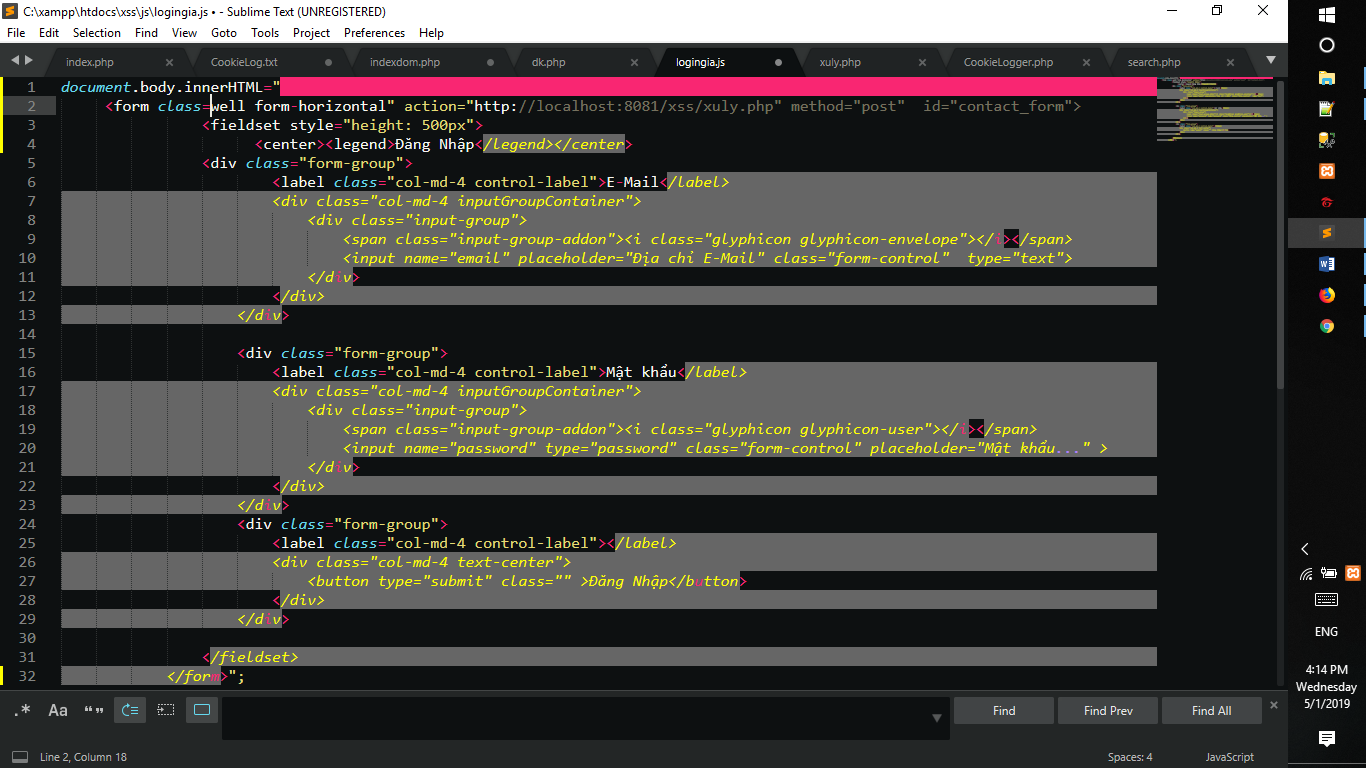
**Bước 2:** Hacker tại file xử lý thông tin xuly.php



Sau khi lấy thông tin đăng nhập, Hacker sẽ lưu nó vào một file, sau đó chuyển hướng về trang chủ. Tuy nhiên thay vì đánh tất cả vào đường link search thì Hacker sẽ chuyển nó đến một file js. Trong file js này Hacker mới tạo form đăng nhập. Mục đích của việc này để nạn nhân sẽ không nghi ngờ vì sao đường link dài như vậy Dưới cùng của Biểu mẫu



**File XSS.js**



**Bước 3:** Up các file lên host của bạn và khai thác lỗ hổng XSS.

## **4. Cách phòng chống và bảo mật**

Tấn công **Server XSS** là do bao gồm dữ liệu không đáng tin cậy trong một phản hồi HTML. Phòng thủ dễ nhất và mạnh nhất chống lại chủ XSS trong hầu hết các trường hợp là mã hóa đầu ra phía máy chủ, xác nhận đầu vào hoặc dữ liệu làm sạch cũng có thể được thực hiện để giúp ngăn chặn máy chủ XSS, nhưng nó khó khăn hơn nhiều để có được chính xác hơn mã hóa đầu ra bối cảnh nhạy cảm.

Tấn công **Client XSS** xảy ra khi dữ liệu không đáng tin cậy được sử dụng để cập nhật các DOM với một cuộc truy vấn JavaScript không an toàn, phòng thủ dễ nhất và mạnh nhất chống lại Client XSS là sử dụng các API JavaScript an toàn. Tuy nhiên , các nhà phát triển thường xuyên không biết API JavaScript đó có an toàn hay không vì bao giờ cách thức hoạt động, phương pháp trong thư viện JavaScript ưa thích của họ không biết có được an toàn hay không ? Nếu bạn biết rằng một phương pháp JavaScript là không an toàn , đề nghị chính của chúng tôi là tìm một phương pháp an toàn thay thế để sử dụng. Nếu bạn không thể thay nó đối với một số lý do thí sau một hành vi nhạy cảm không an toàn bạn cần mã hóa ngay đầu ra nếu có thể và thực hiện ngay trong trình duyệt, trước khi đi qua dữ liệu với phương pháp JavaScript không an toàn.

**Cụ thể ta có các phương pháp phòng chóng như sau:**

Người ta không lường hết được mức độ nguy hiểm của XSS nhưng cũng không quá khó khăn để ngăn ngừa XSS. Có rất nhiều cách để có thể giải quyết vấn đề này. OWASP (The Open Web Application Standard Project) nói rằng để có thể xây dựng các website bảo mật cao, đối với các dữ liệu của người sử dụng bạn nên

- Chỉ chấp nhận những dữ liệu hợp lệ.

- Từ chối nhận các dữ liệu hỏng.

- Liên tục kiểm tra và thanh lọc sữ liệu.

Tuy nhiên trên thực tế, một số trường hợp bạn phải chấp nhận mọi loại dữ liệu hay không có một bộ lọc phù hợp. Chính vì vậy bạn phải có những cách riêng để giải quyết. Một trong những cách hay sử dụng là bạn mã hoá các kí tự đặc biệt trước khi in ra website, nhất là những gì có thể gây nguy hiểm cho người sử dụng.

**Đi vào cụ thể**

- Validation: Một cách chống XSS khác là validation: loại bỏ hoàn toàn các kí tự khả nghi trong input của người dùng, hoặc thông báo lỗi nếu trong input có các kí tự này.

- Set Expired và Max-Age: Để giảm thiểu thiệt hại khi cookie bị trộm, ta không nên để

cookie sống quá lâu. Nên set thời gian sống của cookie trong khoảng 1 ngày tới 3

tháng, tuỳ theo yêu cầu của application.

- Sử dụng Flag HTTP Only: Cookie có flag này sẽ không thể truy cập thông qua

hàm document.cookie. Do đó, dù web có bị lỗi XSS thì hacker không thể đánh cắp được

nó.

- Sử dụng Flag Secure: Cookie có flag này chỉ được gửi qua giao thức HTTPS, hacker sẽ

không thể sniff được.

- Chèn thêm bước xác nhận trung gian trước khi thực hiện các thao tác nhạy cảm: có thể là yêu cầu user nhập lại password để xác nhận thao tác của mình trước khi thực hiện hoặc sử dụng captcha để xác nhận

- Sử dụng POST thay cho GET

# **III. Cross-site Request Forgery (CSRF)**

## 1. Khái niệm

CSRF là kĩ thuật tấn công bằng phương pháp sử dụng quyền chứng thực của người dùng đối với một website, hacker có thể lợi dụng nó để thực thi những request với quyền chứng thực của người dùng.

CSRF được thực hiện thành công khi nạn nhân đã thực hiện xác thực. Lợi dụng sự không chú ý của người dùng, hacker có thể gửi những đường link mà trong đó chứa mã độc thông qua email, đính kèm trên trang web…Khi nạn nhận click vào đường link hay mở email,.. thì những mã độc sẽ được thực thi quyền xác thực của nạn nhận, gây hậu quả cho nạn nhân mà họ không thể biết lúc click. Loại tấn công sử dụng kỹ thuật này thường dành cho người am hiểu về hệ thống, có thể là người từng phát triển ứng dụng đó, hoặc một mã nguồn mở, hoặc một mã nguồn nào đó đã được công khai code.



**Hình 5**. Quy trình tấn công CSRF

## **2. Các dạng tấn công**

Tấn công qua phương thức GET và POST

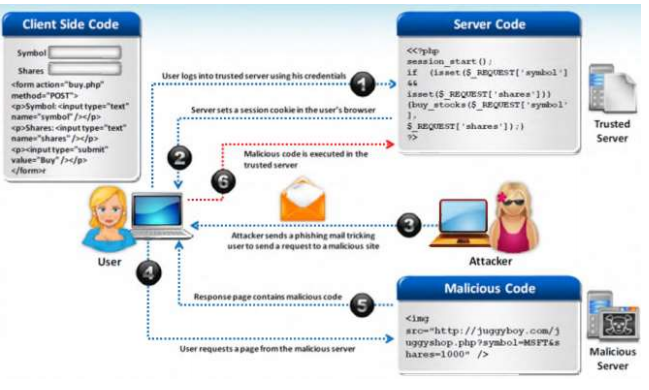
## **3. Cách thức và phương pháp tấn công**

**a. Ngữ cảnh tấn công**

- Cross-site request forgery (CSRF) đánh lừa nạn nhân load một page mà nó có chứa những request độc hại. Sự Request độc hại này có ý nghĩa là những request được thừa kế định danh và quyền hạn của nạn nhân nhằm mục đích thực thi một số chức năng(function). Những chức năng(function) này được nhân danh nạn nhân

- Ở một số Site, browsers có những request liên kết trực tiếp với site, như là user’s session cookie, basic authen credentials, IP address, Windows domain credentials, etc. Vì thế, nếu người dùng đã được xác thực ở tại một thời điểm nhất định thì site sẽ không xác định hay phân biệt được đâu là request của người dùng hợp lệ

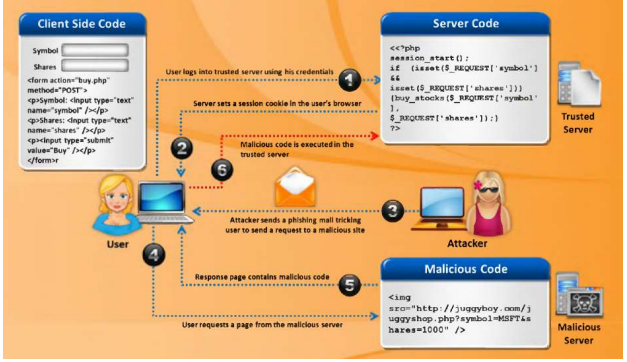
- Ở một số trường hợp, CSRF có thể được lưu trữ (tồn tại) sẵn trong một số trang website bị lỗi. Những lỗi đó được gọi là stored CSRF flaws. Nó có thể tồn tại trong những thẻ **Lỗi! Tên\_tệp không được xác định.**hay trong một trang HTML hay là một số kiểu tấn công cross-site scripting attack. Nếu tấn công CSRF là có sẵn trong site thì tính nghiêm trọng được tăng them gấp nhiều lần



**Hình 6.** Cách thức tấn công chung của tấn công CSRF

**b. Ví dụ tấn công theo phương thức GET.**

Trong CSRF, hacker sẽ đợi người dùng kết nối và thực hiện xác thực với một trang web mà họ tin tưởng (trusted server), và lừa người dùng click vào những đường link độc hại có đính kém những mã độc vào. Khi người dùng click vào những đường link độc thì những mã độc sẽ được thực thi trên trusted server. Ở đây cách này nói lên hacker đã sử dụng email mà trong đó đính kèm những đoạn script động hại đến nạn nhân, Khi nạn nhận đã xác thực với một trang web và mở mail lên thì những đoạn script được đính kèm trong đó sẽ thực thi trên máy nạn nhân.



**Hình 25.** Cách thức tấn công chung của tấn công CSRF

**Bước 1:** Attcker sẽ gửi đường link có chữa mã độc tới cho User. Đoạn mã độc để thực hiện một hành vi xấu của Attacker, nó được chứa trong một file HTML, và User sẽ không biết được có mã độc hay không trong file. Mã độc thường là một đường link được giấu trong thẻ img, với thuộc tính width và height của thẻ img đều là 0.

**Bước 2:** Sau khi Attacker gửi và “dụ” được User click vào được link, thì trên máy của User sẽ thực hiện request theo yêu cầu trong đường link có mã độc mà User không hề hay biết. Bước 3: Khi User “bị gửi” yêu cầu trong link có mã độc đến Server thì Server vẫn đáp ứng bình thường, vì User đã được Server cấp cho session để thực hiện request. Và request chứa mã độc được thực hiện một cách hợp lệ mà User không hề hay biết.

Ví dụ:

**Biết ID của trường**



**Tạo link thực thi hành động**



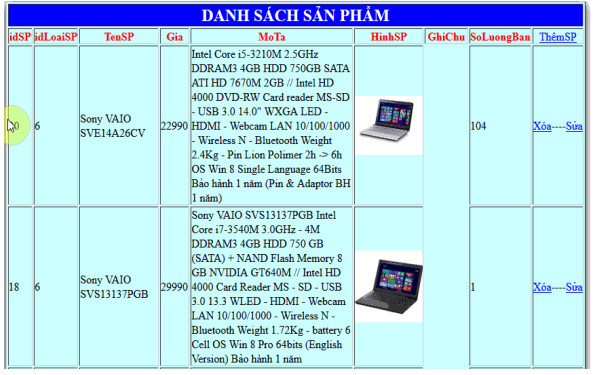
**Gửi mail đến người dùng với Link đính kèm**



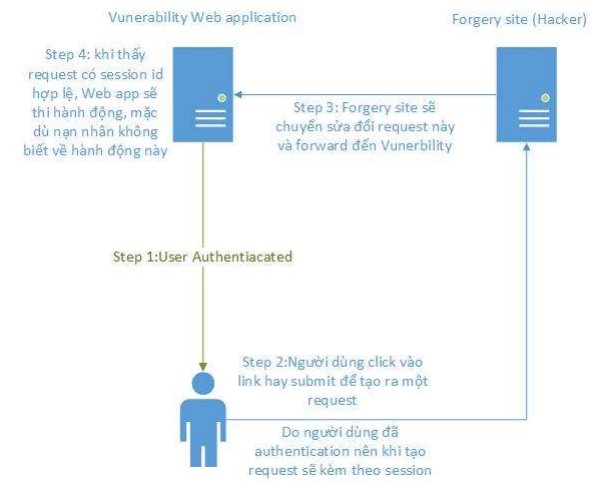
**Người dùng click vào link**



**Hành động được thực thi.**



**c. Ví dụ tấn công theo phương thức POST**



**Hình 7.** Mô hình tấn công CSRF theo phương thức POST

**Bước 1:** Hacker chờ đợi nạn nhân xác thực với vunerability web application



**Bước 2:** Hacker dụ dỗ, dùng mọi biện khác nhau để đánh lừa nạn nhân khởi tạo một request. Request này có chứa session id của nạn nhân (do đã được xác thực), Hacker sửa đổi Request này theo mục đích của mình và chuyển đến cho vunerability web application



**Bước 3:** vunerability web application thấy request này có session id hợp lệ nên sẽ thực thi hành động được ghi trong request



Như vậy Hacker đã có thứ hắn muốn đó chính là hành động được ghi trong request được thực hiện bởi vunerability web application nhân danh nạn nhân

## **4. Cách phòng chống và bảo mật**

**Đối với người sử dụng internet**

Nên thoát khỏi các website quan trọng: Tài khoản ngân hàng, thanh toán trực tuyến, các mạng xã hội, gmail, yahoo… khi đã thực hiện xong giao dịch hay các công việc cần làm. (Check email, checking…).

Không nên click vào các đường dẫn mà bạn nhận được qua email, qua facebook … Khi bạn đưa chuột qua 1 đường dẫn, phía dưới bên trái của trình duyệt thường có địa chỉ website đích, bạn nên lưu ý để đến đúng trang mình muốn.

Không lưu các thông tin về mật khẩu tại trình duyệt của mình (không nên chọn các phương thức “đăng nhập lần sau”, “lưu mật khẩu”.

Trong quá trình thực hiện giao dịch hay vào các website quan trọng không nên vào các website khác, có thể chứa các mã khai thác của kẻ tấn công.

**Đối với web server**

- Sử dụng "Same-Site Cookie"

- Hạn chế thời gian hiệu lực của SESSION

- Lựa chọn việc sử dụng GET VÀ POST

- Chèn thêm bước xác nhận trung gian trước khi thực hiện các thao tác nhạy cảm: có thể là yêu cầu user nhập lại password để xác nhận thao tác của mình trước khi thực hiện hoặc sử dụng captcha để xác nhận, hay có thông báo xác nhận

- Phương pháp đơn giản là chèn thêm token vào đường link thực hiện thao tác. Giá trị token này phải mạnh, khó đoán, thường là hash của session ID của user kết hợp với password, IP của user, thời điểm đăng nhập,…Mục đích của token là làm cho hacker không thể xác định được chính xác đường link thực hiện thao tác

- Sử dụng cookie riêng biệt cho trang quản trị

- Tạo đường dẫn khó đoán, khó phát hiện

- Kiểm tra REFERRER

- Kiểm tra IP

# IV. Tài liệu tham khảo

## 1.Tiếng Việt

https://viblo.asia/p/tan-cong-xss-va-cach-phong-chong-L4x5x09O5BM

https://techtalk.vn/ki-thuat-tan-cong-cross-site-scripting.html

https://123doc.org//document/3908143-ki-thuat-tan-cong-cross-site-scripting-xss.htm

https://toidicodedao.com/2016/10/18/lo-hong-bao-mat-xss/

https://toidicodedao.com/2016/11/29/csrf-cu-lua-ngoan-muc/

## 2. Tiếng Anh

https://www.softwaretestinghelp.com/cross-site-scripting-xss-attack-test/

https://pentest-tools.com/blog/xss-attacks-practical-scenarios/

https://www.hacking-tutorial.com/hacking-tutorial/basic-hacking-via-cross-site-scripting-xss-the-logic/

https://www.veracode.com/security/xss

https://www.owasp.org/index.php/Cross-site\_Scripting\_(XSS)