**BÁO CÁO BASIC WEB ATTACK**

1. **SQL INJECTION**

**SQL Injection** là một kỹ thuật lợi dụng những lỗ hổng về câu truy vấn của các ứng dụng. Được thực hiện bằng cách chèn thêm một đoạn [SQL](https://topdev.vn/blog/sql-la-gi/) để làm sai lệnh đi câu truy vấn ban đầu, từ đó có thể khai thác dữ liệu từ database. **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.

Ví dụ, trong form đăng nhập, người dùng nhập dữ liệu, trong trường tìm kiếm người dùng nhập văn bản tìm kiếm, trong biểu mẫu lưu dữ liệu, người dùng nhập dữ liệu cần lưu. Tất cả các dữ liệu được chỉ định này đều đi vào cơ sở dữ liệu.

Thay vì nhập dữ liệu đúng, kẻ tấn công lợi dụng lỗ hổng để insert và thực thi các câu lệnh SQL bất hợp pháp để lấy dữ liệu của người dùng… SQL Injection được thực hiện với ngôn ngữ lập trình SQL. SQL (Structured Query Language) được sử dụng để quản lý dữ liệu được lưu trữ trong toàn bộ cơ sở dữ liệu.

**Tuy nhiên** ngày nay chứng ta thường làm việc trên những [framework](https://topdev.vn/blog/framework-la-gi/) hiện đại. Các framework đều đã được test cẩn thận để phòng tránh các lỗi, trong đó có SQL Injection.

Sự nguy hiểm của SQL Injection

* Hack tài khoản cá nhân.
* Ăn cắp hoặc sao chép dữ liệu của trang web hoặc hệ thống.
* Thay đổi dữ liệu nhạy cảm của hệ thống.
* Xóa dữ liệu nhạy cảm và quan trọng của hệ thống.
* Người dùng có thể đăng nhập vào ứng dụng với tư cách người dùng khác, ngay cả với tư cách quản trị viên.
* Người dùng có thể xem thông tin cá nhân thuộc về những người dùng khác, ví dụ chi tiết hồ sơ của người dùng khác, chi tiết giao dịch của họ,…
* Người dùng có thể sửa đổi cấu trúc của cơ sở dữ liệu, thậm chí xóa các bảng trong cơ sở dữ liệu ứng dụng.
* Người dùng có thể kiểm soát máy chủ cơ sở dữ liệu và thực thi lệnh theo ý muốn.

Việc kiểm tra lỗ hổng này có thể được thực hiện rất dễ dàng. Đôi khi ta chỉ cần nhập ký hiệu ' hoặc " vào các trường được kiểm tra. Nếu nó trả về bất kỳ thông báo bất ngờ hoặc bất thường, thì ta có thể chắc chắn rằng SQL Injection khả thi cho trường đó.

Nếu như người dùng không nhập bình thường nữa mà chẳng hạn như họ có thêm một dấu nháy ' hoặc " vào thì dòng code của bạn sẽ bị lỗi ngay. Hoặc họ có thể sửa thành một câu truy vấn luôn luôn đúng như sau.

SELECT \* FROM tbl\_user WHERE username = '' OR '1' = '1' and password = '' OR '1' = '1'

Hoặc chèn thêm một câu lệnh truy vấn phía sau:

**VD**:

SELECT \* FROM tbl\_user WHERE username = 'admin' and password = 'admin'; Drop table users;

## Các phần dễ bị tấn công

Các phần dễ bị tấn công bao gồm:

* Form đăng nhập
* Form tìm kiếm
* Form nhận xét
* Bất kì trường lưu hoặc trường đầu vào của dữ liệu
* Liên kết của website

Cần lưu ý là trong khi thử nghiệm chống lại tấn công này là không thể chỉ kiểm tra một hoặc một vài trường bởi vì một trường có thể được bảo vệ chống lại SQL Injection, nhưng một trường khác thì không. Do đó, điều quan trọng là đừng quên kiểm tra tất cả các trường của trang web.

## Cách giảm thiểu và phòng ngừa SQL Injection

Luôn kiểm tra kỹ các trường nhập dữ liệu và các bạn cần ràng buộc thật kỹ dữ liệu người dùng nhập vào.

Ví dụ:

//Thông thường

$id = $\_GET['id'];

//Ràng buộc

$id = isset($\_GET['id'])?(string)(int)$\_GET['id']:false;

Dùng [Regular Expression](https://topdev.vn/blog/regular-expression-la-gi/) để loại bỏ đi các ký tự lạ hoặc các ký tự không phải là số.

$id = isset($\_GET['id']) ? $\_GET['id'] : false;

$id = str\_replace('/[^0-9]/', '', $id);

Hoặc dùng các hàm có sẵn để giảm thiểu lỗi. Mỗi khi truy vấn thì mọi người nên sử dụng thêm hàm mysqli\_real\_escape\_string để chuyển đổi một chuỗi thành một query an toàn.

$id = isset($\_GET['id'])?(string)(int)$\_GET['id']:false;

$sql= 'SELECT \* FROM tbl\_user WHERE id= ' . mysqli\_real\_escape\_string($id);

Và lời khuyên cuối cùng là chúng ta nên dùng các Framework và hạn chế dùng code thuần tối đa nếu có thể. Framework luôn có cộng đồng hoặc các chuyên gia bảo mật giúp tìm lỗi và update liên tục, từ đó chúng ta có thể giảm bớt thời gian xử lý lỗi để tăng thời gian làm sản phẩm cũng là một điều hay.

1. **Kỹ thuật tấn công XSS**

# XSS là gì?

Cross Site Scripting (XSS) là một trong những tấn công phổ biến và dễ bị tấn công nhất mà tất cả các Tester có kinh nghiệm đều biết đến. Nó được coi là một trong những tấn công nguy hiểm nhất đối với các ứng dụng web và có thể mang lại những hậu quả nghiêm trọng. Giới thiệu về tấn công XSS Tấn công XSS là một đoạn mã độc, để khái thác một lỗ hổng XSS, hacker sẽ chèn mã độc thông qua các đoạn script để thực thi chúng ở phía Client. Thông thường, các cuộc tấn công XSS được sử dụng để vượt qua truy cập và mạo danh người dùng.

Mục đích chính của cuộc tấn công này là ăn cắp dữ liệu nhận dạng của người dùng như: cookies, session tokens và các thông tin khác. Trong hầu hết các trường hợp, cuộc tấn công này đang được sử dụng để ăn cắp cookie của người khác. Như chúng ta biết, cookie giúp chúng tôi đăng nhập tự động. Do đó với cookie bị đánh cắp, chúng tôi có thể đăng nhập bằng các thông tin nhận dạng khác. Và đây là một trong những lý do, tại sao cuộc tấn công này được coi là một trong những cuộc tấn công nguy hiểm nhất.

Tấn công XSS đang được thực hiện ở phía client. Nó có thể được thực hiện với các ngôn ngữ lập trình phía client khác nhau. Tuy nhiên, thường xuyên nhất cuộc tấn công này được thực hiện với Javascript và HTML.

# Tấn công XSS thực hiện như thế nào?

Tấn công Cross Site Scripting nghĩa là gửi và chèn lệnh và script độc hại, những mã độc này thường được viết với ngôn ngữ lập trình phía client như Javascript, HTML, VBScript, Flash… Tuy nhiên, cách tấn công này thông thường sử dụng Javascript và HTML. Cách tấn công này có thể được thực hiện theo nhiều cách khác nhau, phụ thuộc vào loại tấn công XSS, những mã độc có thể được phản chiếu trên trình duyệt của nạn nhân hoặc được lưu trữ trong cơ sở dữ liệu và được chạy mỗi khi người dùng gọi chức năng thích hợp. Nguyên nhân chính của loại tấn công này là xác thực đầu vào dữ liệu người dùng không phù hợp, dữ liệu độc hại từ đầu vào có thể xâm nhập vào dữ liệu đầu ra. Mã độc có thể nhập một script và được chèn vào mã nguồn của website. Khi đó trình duyệt không thể biết mã thực thi có phải độc hại hay không. Do đó mã độc hại có thể đang được thực thi trên trình duyệt của nạn nhận hoặc bất kỳ hình thức giả nào đang được hiển thị cho người sử dụng. Có một số hình thức tấn công XSS có thể xảy ra. Bên dưới là những hình thức tấn công chính của Cross Site Scripting:

* Cross Site Scripting có thể xảy ra trên tập lệnh độc hại được thực hiện ở phía client.
* Trang web hoặc form giả mạo được hiển thị cho người dùng (nơi nạn nhân nhập thông tin đăng nhập hoặc nhấp vào liên kết độc hại).
* Trên các trang web có quảng cáo được hiển thị.
* Email độc hại được gửi đến nạn nhân. Tấn công xảy ra khi tin tặc tìm kiếm những lỗ hổng trên website và gửi nó làm đầu vào độc hại. Tập lệnh độc hại được tiêm vào mã lệnh và sau đó được gửi dưới dạng đầu ra cho người dùng cuối cùng.

Chúng ta hãy phân tích một ví dụ đơn giản sau đây: Tưởng tượng chúng ta có 1 trang web với trường Search.

Nếu trường Search là trường có lỗ hổng, khi người dùng nhập bất kỳ đoạn script thì nó sẽ được thực thi.

**Ví dụ 1**: Hãy xem xét rằng trong trường review nếu hacker nhập đoạn code sau:

**<script>destroyWebsite();</script>**

Sau đó, hàm destroyWebsite() sẽ được gọi và nó sẽ thực hiện các hành động có hại của nó. Như hầu hết chúng ta biết, cuộc tấn công này chủ yếu được sử dụng để thu thập cookie của người khác, có thể được sử dụng để đăng nhập bằng các danh tính khác. Hãy để chúng tôi phân tích một ví dụ khác về kịch bản XSS có thể có với hành vi trộm cắp cookie có thể xảy ra.

**Ví dụ 2**: thông qua lỗ hổng của website, tin tặc sẽ tiêm mã thích hợp.

\*\*<script type=”text/javascript”>

Var test=’../example.php?cookie\_data=’+escape(docuent.cookie);

</script>\*\*

Như đã thấy trong Ví dụ trên, cookie bị mất và được gửi tới biến ‘cookie\_data’ của tập lệnh mẫu example.php. Nếu hacker sẽ chèn tập lệnh này vào mã của trang web, thì mã sẽ được thực thi trong trình duyệt của người dùng và cookie sẽ được gửi tới hacker.

# Các cách để ngăn chặn XSS

Mặc dù loại tấn công này được coi là một trong những loại nguy hiểm và rủi ro nhất, nhưng vẫn nên chuẩn bị một kế hoạch ngăn ngừa. Bởi vì sự phổ biến của cuộc tấn công này, có khá nhiều cách để ngăn chặn nó.

Các phương pháp phòng ngừa chính được sử dụng phổ biến bao gồm:

* Data validation
* Filtering
* Escaping

Bước đầu tiên trong công tác phòng chống tấn công này là Xác thực đầu vào. Mọi thứ, được nhập bởi người dùng phải được xác thực chính xác, bởi vì đầu vào của người dùng có thể tìm đường đến đầu ra. Xác thực dữ liệu có thể được đặt tên làm cơ sở để đảm bảo tính bảo mật của hệ thống. Tôi sẽ nhắc nhở rằng ý tưởng xác thực không cho phép đầu vào không phù hợp. Vì vậy nó chỉ giúp giảm thiểu rủi ro, nhưng có thể không đủ để ngăn chặn lỗ hổng XSS có thể xảy ra.

Một phương pháp ngăn chặn tốt khác là lọc đầu vào của người dùng. Ý tưởng lọc là tìm kiếm các từ khóa nguy hiểm trong mục nhập của người dùng và xóa chúng hoặc thay thế chúng bằng các chuỗi trống. Những từ khóa đó có thể là:

Thẻ <script> </ script>

Lệnh Javascript

Đánh dấu HTML

Lọc đầu vào khá dễ thực hành. Nó có thể được thực hiện theo nhiều cách khác nhau. Như:

Bởi các developers đã viết mã phía server.

Thư viện ngôn ngữ lập trình thích hợp đang được sử dụng.

Trong trường hợp này, một số developer viết mã riêng của họ để tìm kiếm các từ khóa thích hợp và xóa chúng. Tuy nhiên, cách dễ dàng hơn là chọn thư viện ngôn ngữ lập trình thích hợp để lọc đầu vào của người dùng. Tôi muốn lưu ý rằng việc sử dụng thư viện là một cách đáng tin cậy hơn, vì các thư viện đó đã được nhiều nhà phát triển sử dụng và thử nghiệm.

Một phương pháp phòng ngừa khác có thể là ký tự Escape. Trong thực tế này, các ký tự thích hợp đang được thay đổi bằng các mã đặc biệt.

Ví dụ: <ký tự Escape có thể giống như & # 60. Điều quan trọng cần biết là chúng ta có thể tìm thấy các thư viện thích hợp với ký tự escape.

Trong khi đó, việc kiểm thử tốt cũng không nên quên điều đó. Chúng ta cần những kiểm thử phần mềm có kiến thức tốt và những công cụ kiểm thử phần mềm đáng tin cậy. Bằng cách này, chất lượng phần mềm sẽ được bảo đảm tốt hơn.

1. **Kỹ thuật tấn công CSRF**

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 tấn công vào người dùng, dựa vào đó hacker có thể thực thi những thao tác phải yêu cầu sự chứng thực. Hiểu một cách nôm na, đây là kỹ thuật tấn công dựa vào mượn quyền trái phép.

CSRF còn được gọi là "session riding", "XSRF"

## Kịch bản tấn công CSRF

Các ứng dụng web hoạt động theo cơ chế nhận các câu lệnh HTTP từ người sử dụng, sau đó thực thi các câu lệnh này. Hacker sử dụng phương pháp CSRF để lừa trình duyệt của người dùng gửi đi các câu lệnh http đến các ứng dụng web. Điều đó có thể thực hiện bằng cách chèn mã độc hay link đến trang web mà người dùng đã được chứng thực. Trong trường hợp phiên làm việc của người dùng chưa hết hiệu lực thì các câu lệnh trên sẽ được thực hiện với quyền chứng thực của người sử dụng. Ta có thể xét ví dụ sau:

* Người dùng Alie truy cập 1 diễn đàn yêu thích của mình như thường lệ. Một người dùng khác, Bob đăng tải 1 thông điệp lên diễn đàn. Giả sử rằng Bob có ý đồ không tốt và anh ta muốn xóa đi một dự án quan trọng nào đó mà Alice đang làm.
* Bob sẽ tạo 1 bài viết, trong đó có chèn thêm 1 đoạn code như sau:

<img height="0" width="0" src="http://www.webapp.com/project/1/destroy">

Để tăng hiệu quả che dấu, đoạn mã trên có thể được thêm các thông điệp bình thường để người dùng không chú ý. Thêm vào đó thẻ img sử dụng trong trường hợp này có kích thước 0x0 pixel và người dùng sẽ không thể thấy được.

* Giả sử Alie đang truy cập vào tài khoản của mình ở www.webapp.com và chưa thực hiện logout để kết thúc. Bằng việc xem bài post, trình duyệt của Alice sẽ đọc thẻ img và cố gắng load ảnh từ www.webapp.com, do đó sẽ gửi câu lệnh xóa đến địa chỉ này.
* Ứng dụng web ở www.webapp.com sẽ chứng thực Alice và sẽ xóa project với ID là 1. Nó sẽ trả về trang kết quả mà không phải là ảnh, do đó trình duyệt sẽ không hiển thị ảnh.

Ngoài thẻ img, các thẻ html có thể sử dụng kĩ thuật trên có thể là:

<iframe height="0" width="0" src="http://www.webapp.com/project/1/destroy">

<link ref="stylesheet" href="http://www.webapp.com/project/1/destroy" type="text/css"/>

<bgsound src="http://www.webapp.com/project/1/destroy"/>

<background src="http://www.webapp.com/project/1/destroy"/>

<script type="text/javascript" src="http://www.webapp.com/project/1/destroy"/>

Các kĩ thuật CSRF rất đa dạng, lừa người dùng click vào link, gửi email chứa các đoạn mã độc đến người dùng… Hacker còn có thể che giấu các link ở trên rất khéo léo. Ví dụ trong trường hợp thẻ img, người dùng có thể nhận ra nếu vào đường link chứa trong

<ing src="http://www.webapp.com/project/1/destroy"/>

Tuy nhiên, người dùng sẽ rất có phát hiện nếu hacker dùng đường link như sau:

<img height="0" width="0" src="http://www.ahackersite.com/abc.jpg"/>

và cấu hình lại máy chủ:

Redirect 302/abc.jpg http://www.webapp.com/project/1/destroy"/>

Như vậy người dùng sẽ rất khó để có thể phát hiện, vấn đề trách nhiệm phần lớn thuộc về các website của các nhà cung cấp.

## 2. Cách phòng chống tấn công CSRF

Dựa trên nguyên tắc của CSRF "lừa trình duyệt của người dùng (hoặc người dùng) gửi các câu lệnh HTTP", các kĩ thuật phòng tránh sẽ tập trung vào việc tìm cách phân biệt và hạn chế các câu lệnh giả mạo.

### 2.1 Phía user

Để phòng tránh trở thành nạn nhân của các cuộc tấn công CSRF, người dùng internet nên thực hiện một số lưu ý sau:

* 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, checkin…)
* 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.

### 2.2 Phía server

Có nhiều lời khuyến cáo được đưa ra, tuy nhiên cho đến nay vẫn chưa có biện pháp nào có thể phòng chống triệt để CSRF. Sau đây là một vài kĩ thuật sử dụng.

#### 2.2.1 Lựa chọn việc sử dụng GET VÀ POST

Sử dụng GET và POST đúng cách. Dùng GET nếu thao tác là truy vấn dữ liệu. Dùng POST nếu các thao tác tạo ra sự thay đổi hệ thống (theo khuyến cáo của W3C tổ chức tạo ra chuẩn http) Nếu ứng dụng của bạn theo chuẩn RESTful, bạn có thể dùng thêm các HTTP verbs, như PATCH, PUThay DELETE

#### 2.2.2 Sử dụng captcha, các thông báo xác nhận

Captcha được sử dụng để nhận biết đối tượng đang thao tác với hệ thống là con người hay không? Các thao tác quan trọng như "đăng nhập" hay là "chuyển khoản" ,"thanh toán" thường là hay sử dụng captcha. Tuy nhiên, việc sử dụng captcha có thể gây khó khăn cho một vài đối tượng người dùng và làm họ khó chịu. Các thông báo xác nhận cũng thường được sử dụng, ví dụ như việc hiển thị một thông báo xác nhận "bạn có muốn xóa hay k" cũng làm hạn chế các kĩ thuật Cả hai cách trên vẫn có thể bị vượt qua nếu kẻ tấn công có một kịch bản hoàn hảo và kết hợp với lỗi XSS.

#### 2.2.3 Sử dụng token

Tạo ra một token tương ứng với mỗi form, token này sẽ là duy nhất đối với mỗi form và thường thì hàm tạo ra token này sẽ nhận đối số là"SESSION" hoặc được lưu thông tin trong SESSION. Khi nhận lệnh HTTP POST về, hệ thống sẽ thực hiên so khớp giá trị token này để quyết định có thực hiện hay không. Mặc định trong Rails, khi tạo ứng dụng mới:

class ApplicationController < ActionController::Base

protect\_from\_forgery with: :exception

end

Khi đó tất cả các form và Ajax request được tự động thêm sercurity token generate bởi Rails. Nếu security token không khớp, exception sẽ được ném ra.

#### 2.2.4 Sử dụng cookie riêng biệt cho trang quản trị

Một cookie không thể dùng chung cho các domain khác nhau,chính vì vậy việc sử dụng "[admin.site.com](http://admin.site.com/)" thay vì sử dụng "[site.com/admin](http://site.com/admin)" là an toàn hơn.

#### 2.2.5 Kiểm tra REFERRER

Kiểm tra xem các câu lệnh http gửi đến hệ thống xuất phát từ đâu. Một ứng dụng web có thể hạn chế chỉ thực hiện các lệnh http gửi đến từ các trang đã được chứng thực. Tuy nhiên cách làm này có nhiều hạn chế và không thật sự hiệu quả.

#### 2.2.6 Kiểm tra IP

Một số hệ thống quan trọng chỉ cho truy cập từ những IP được thiết lập sẵn