**BỘ GIÁO DỤC ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN: BẢO MẬT THÔNG TIN**



***Giáo viên hướng dẫn:* Thầy NGUYỄN THANH PHONG**

***Sinh viên thực hiện:* LỚP 17DTHC6**

1. **Nguyễn Ngọc Thuỷ Tiên**
2. **Châu Văn Hưng**
3. **Cao Hữu Duy**
4. **Lê Đồng Phương**

Ngày 10 tháng 12 năm 2019

**Chương 1: Kỹ thuật tấn công CSRF và cách phòng chống**

## **1. CSRF là gì?**

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"

## **2. Lịch sử về tấn công CSRF**

Các kiểu tấn công CSRF xuất hiện từ những năm 1990, tuy nhiên các cuộc tấn công này xuất phát từ chính IP của người sử dụng nên log file của các website không cho thấy các dấu hiệu của CSRF. Các cuộc tấn công theo kĩ thuật CSRF không được báo cáo đầy đủ, đến năm 2007 mới có một vài tài liệu miêu tả chi tiết về các trường hợp tấn công CSRF.

Năm 2008 người ta phát hiện ra có khoảng 18 triệu người sử dụng eBay ở Hàn Quốc mất các thông tin cá nhân của mình. Cũng trong năm 2008, một số khách hàng tại ngân hàng Mexico bị mất tài khoản cá nhân của mình.Trong 2 trường hợp kể trên hacker đều sử dụng kĩ thuật tấn công CSRF.

## **3. 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

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

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

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

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.

## **4. 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.

### 4.1 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.

### 4.2 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.

#### **4.2.1 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, PUT hay DELETE

#### **4.2.2 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 không?" 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.

#### **4.2.3 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.

#### **4.2.4 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.

#### **4.2.5 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. **Kiểm tra giá trị Referer và Origin trong header:**Origin cho ta biết trang web gọi request này. Giá trị này được đính kèm trong mỗi request, hacker không chỉnh sửa được. Kiểm tra giá trị này, nếu nó là trang lạ thì không xử lý request.

**Cần cẩn thận đề phòng lỗi XSS:** Với XSS, hacker có thể cài mã độc trên chính trang web cần tấn công. Lúc này, **mọi phương pháp phòng chống CSRF** như token, referrer **đều bị vô hiệu hoá.** Bản thân bác juno\_okyo từng áp dụng lỗi CSS kết hợp CSRF để tấn công sinhvienit.net

#### **4.2.6 Kiểm tra IP**

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

**Chương 2: Kỹ thuật tấn công XSS và cách phòng chống**

Tính đến thời điểm tháng 3/2018 trên toàn thế giới đã có khoảng 1,8 tỷ trang web. Những trang web thuộc nhiều đối tượng, trong đó có các tổ chức chính quyền, các tập đoàn kinh tế lớn, các cá nhân có ảnh hưởng, …Khi bị tấn công vào bảo mật thì một website sẽ có nguy cơ sụp đổ, kéo theo đó là ảnh hưởng rất lớn đến tổ chức hay cá nhân sở hữu website này. XSS là loại tấn công phổ biến nhất và đang thực sự đe dọa tới rất nhiều người dùng web hiện nay.

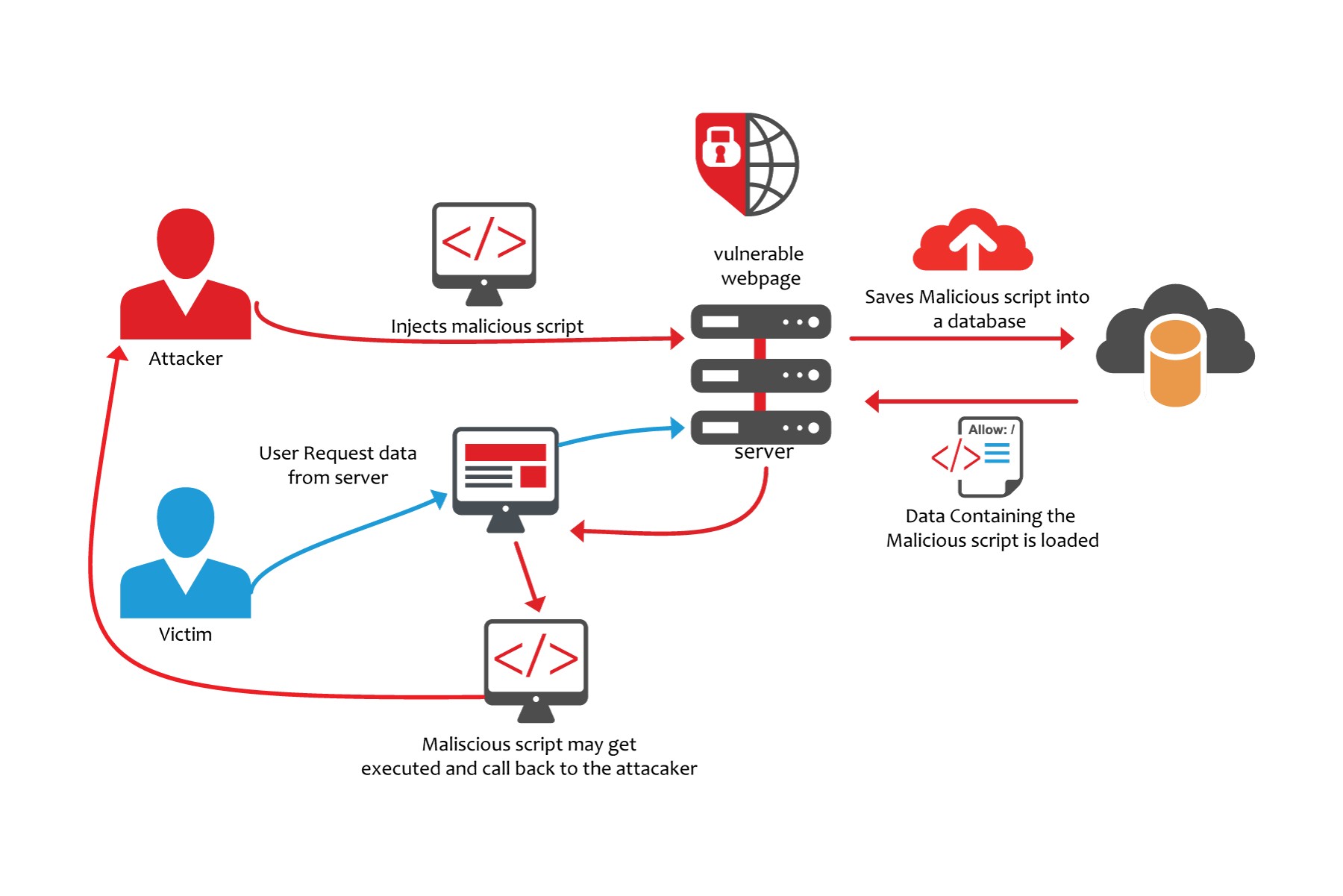
## **1. Khái niệm**

XSS (Cross Site Scripting) là một lỗi [bảo mật](https://toidicodedao.com/2016/09/13/bao-mat-co-ban-phan-1/) **cho phép hacker nhúng mã độc (javascript) vào một trang web khác**. Hacker có thể lợi dụng mã độc này để deface trang web, cài keylog, chiếm quyền điều khiển của người dùng, dụ dỗ người dùng tải virus về máy.

Đây là một trong những lỗi bảo mật thường gặp nhất trên các trang Web. Các hệ thống từ lớn đến nhỏ như Facebook, Twitter, [một số forum Việt Nam](http://kipalog.com/posts/Toi-da-hack-VietDesigner-net-va-Kenhsinhvien-vn-nhu-the-nao), … **đều từng dính phải lỗi** này. Do mức độ phổ biến và độ nguy hiểm của nó, XSS luôn được vinh dự được nằm trong top 10 lỗi bảo mật nghiêm trọng nhất trên OWASP ([Open Web Application Security Project](https://www.owasp.org/index.php/Top_10_2013-A3-Cross-Site_Scripting_(XSS)))

Ví dụ: http://www.xxx.vn//index.php?pg=news&cat=<script>alert(“LỗiXSS”)</script>

Theo đó hacker đã sử dụng lỗ hổng XSS để chèn thêm đoạn mã Javascript nhằm phục vụ mục đích riêng.

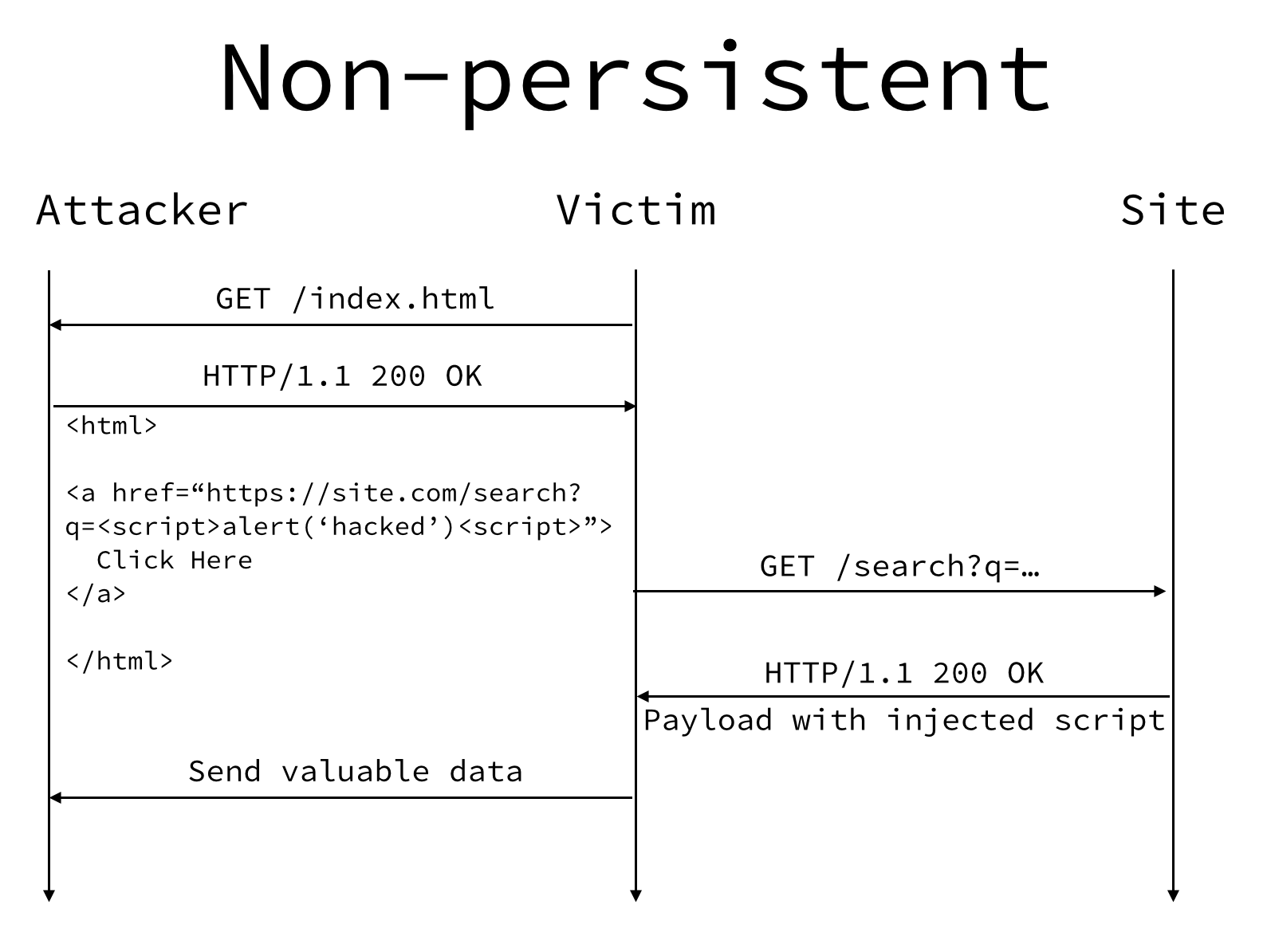


## **2. Phân loại XSS**

### 2.1. Non-persistent (hay Reflected XSS)

Non-persistent (hay reflected) XSS là một loại XSS phổ biến nhất. Loại này xuất hiện khi dữ liệu được cung cấp từ một web client nào đó. Hacker khi muốn tấn công thì điều đầu tiên là sẽ phải tìm ra lỗ hỗng bảo mật trên website bằng cách gắn một đoạn mã test vào web client để web client gửi đến web server và chờ phản hồi của web server để tìm ra lỗ hổng bảo mật. Hacker tấn công dựa vào sự thiếu chú ý về việc lọc dữ liệu vào từ URL của webiste.

Với cách tấn công này, [hacker](https://toidicodedao.com/2016/08/30/hack-lotte-cinema/) chèn mã độc vào URL dưới dạng query string. Khi người dùng không biết nhấp vào URL này, trang web sẽ đọc query string, **render mã độc vào HTML** và người dùng “dính bẫy”.

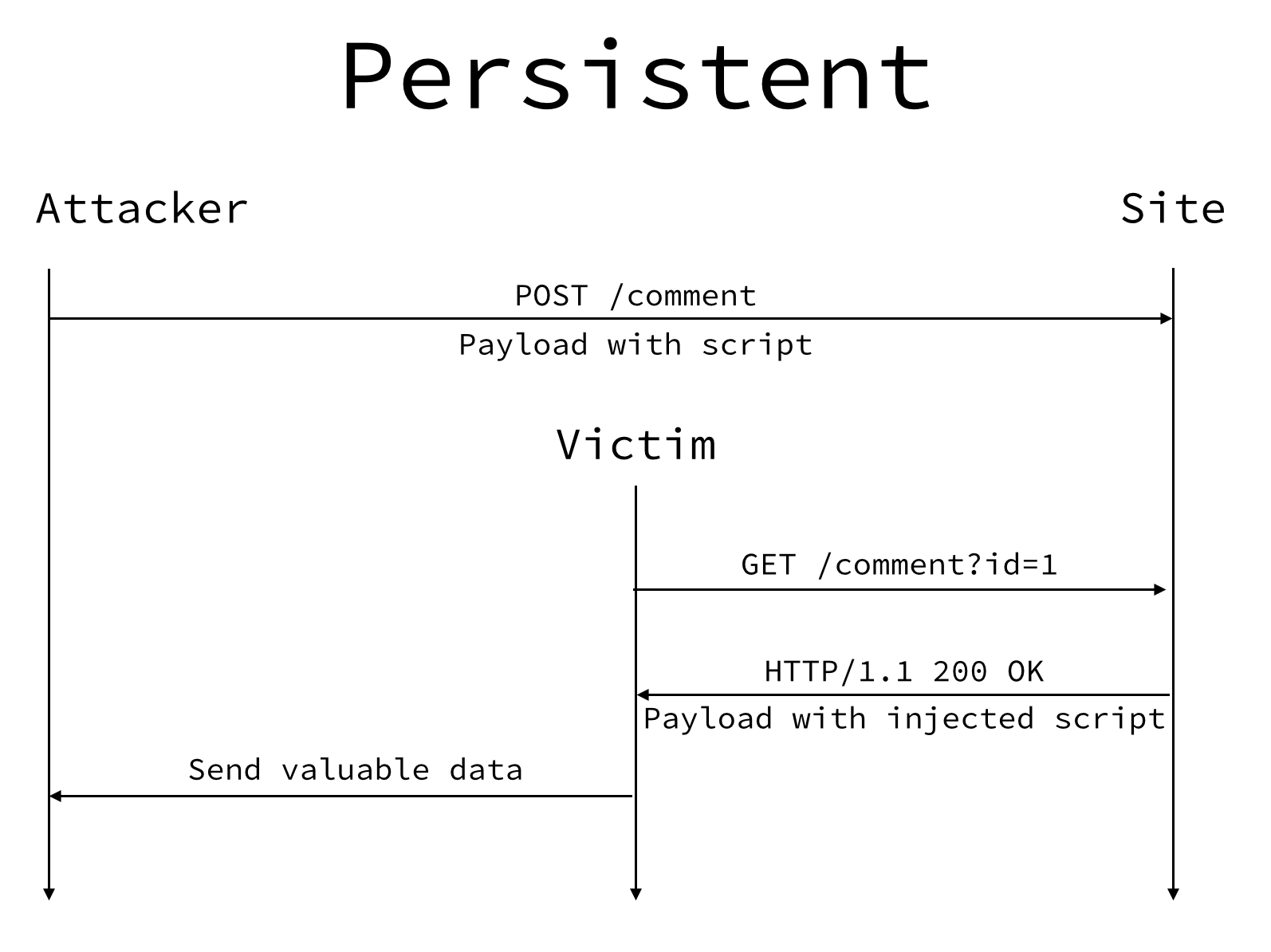


### 2.2. Persistent XSS

Loại XSS này xảy ra khi dữ liệu do các hacker cung cấp được lưu trữ trên các máy chủ thông qua một số chức năng trên website và từ đó về sau thì các dữ liệu này hiển nhiên được hiển thị một cách bình thường trên các trình duyệt của người dùng mà không cần tới HTML riêng nữa. Khi người dùng click vào những phần bị gắn mã độc thì đã bị dính XSS.

Trong kiểu tấn công này, mã độc đựợc lưu trong database trên server, hiển thị ra với toàn bộ người dùng, do đó ta gọi nó là Persistance XSS. Hacker chèn một đoạn mã script và POST lên trên server bằng cách viết trong thanh tìm kiếm hoặc commend, chat..v…v..

Bất kì ai thấy comment này đều bị dính mã độc này, do đó kiểu tấn công này có tầm ảnh hưởng lớ, khá nguy hiểm.



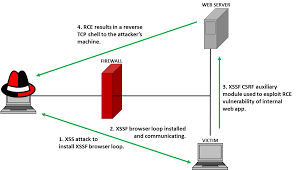
## **3. Các phương thức tấn công**

### 3.1. Đánh cắp Cookies người dùng:

Cookie là một bộ nhắc nhở mà website lưu trữ ở trên máy tính của bạn có thể định danh cho bạn. Nếu không có cookie bạn sẽ phải nhập lại thông tin của mình trên mỗi màn hình web. Thông tin duy nhất mà cookie lưu trữ là thông tin mà bản thân bạn chia sẻ với website tạo ra cookie. Cookie có thể chứa rất nhiều thông tin quan trọng như phiên làm việc của bạn. Nếu hacker có đoạn cookie chưa phiên làm việc của bạn, rất có thể hacker sẽ có khả năng đăng nhập vào website dưới tư cách của bạn mà không cần biết mật khẩu.

## **3.2. Tấn công qua mạng Intranet:**

Hầu hết chúng ta tin rằng trong khi lướt Web mình đã được bảo vệ bởi tường lửa, cách ly thông qua lớp địa chỉ IP riêng. Với sự hiểu biết này, giả sử các phần mềm bảo mật của những trang Web mạng nội bộ và giao diện Web dựa trên các thiết bị định tuyến router, hệ thống tường lửa, IP Phone…. thì ngay cả khi các bản vá lỗi chưa được cập nhật chúng ta vẫn an toàn trong khu vực được bảo vệ bởi các phần mềm bảo mật trên, điều này có vẻ không khả thi lắm. Trình duyệt Web hoàn toàn có thể được kiểm soát bởi bất kỳ trang web nào, cho phép người dùng trở thành tâm điểm cho các cuộc tấn công mạng nội bộ. Khi truy cập vào một Website có chứa phần mềm độc hại với các đoạn mã JavaScript, nó có thể cấu hình lại một cách tự động router hay tường lửa từ đó tạo thành một đường hầm thông ra thế giới mạng bên ngoài.



Các bước khai thác:

* Bước 1: Một nạn nhân truy cập vào một trang Web độc hại hoặc nhấn vào một liên kết không rõ ràng, sẽ bị nhúng mã JavaScript chứa phần mềm độc hại, sau đó sẽ kiểm soát trình duyệt của họ.
* Bước 2: Mã độc JavaScript Malware sẽ tải một ứng dụng trên nền Java Applet và làm lộ ra địa chỉ IP của nạn nhân thông qua NAT IP.
* Bước 3: Sau đó sử dụng trình duyệt của nạn nhân như một nền tảng để tấn công, mã độc JavaScript sẽ xác định máy chủ Web trên mạng nội bộ.
* Bước 4: Phát động tấn công chống lại các Web nội bộ hoặc Web bên ngoài, thu thập thông tin đánh cắp được và gửi ra mạng bên ngoài.

### 3.3. XSS Defacements:

Có hai loại XSS Defacement: liên tục và không liên tục

Mức độ nghiêm trọng của XSS Defacement liên tục là cao hơn so với XSS Defacement không liên tục vì những kẻ tấn công có thể sẽ thay đổi vĩnh viễn thông tin các trang bị tấn công như sửa đổi nội dung, đánh cắp một số thông tin cá nhân của người dùng. Mặc dù kẻ tấn công không có quyền truy cập trực tiếp vào hệ thống tập tin tại nơi trang Web bị lỗi XSS. XSS Defacement không liên tục thường dễ dàng tìm kiếm và thực thi nhưng để nó làm việc attacker sẽ đánh lừa người dùng qua một URL cụ thể. Khái niệm XSS Defacement về cơ bản cũng tương tự như các loại hình tấn công XSS khác. Tuy nhiên thay vì tiêm những đoạn mã JavaScript để thực thi và chuyển thành dữ liệu cookie hoặc chiếm đoạt quyền kiểm soát trình duyệt, attacker sẽ tiêm những đoạn mã làm thay đổi cấu trúc, nội dung ban đầu của Website.

## **4. NGĂN CHẶN XSS**

Vì XSS là một dạng tấn công hay gặp, dễ gây hậu quả cao nên hầu như các Web Framework nổi tiếng (Spring, Django, ASP.NET MVC) đều tích hợp sẵn cách phòng chống. Chỉ cần sử dụng framework bản mới nhất là đã đề phòng được kha khá lỗi.

### 4.1. Lọc

Có hai khái niệm cơ bản về quá trình lọc (filter) XSS: lọc đầu vào (input filtering) và lọc đầu ra (output filtering). Cách sử dụng phổ biến nhất là lọc đầu vào. Input Filtering được xem là chính xác hơn so với Output Filtering, đặc biệt trong trường hợp XSS Reflected. Tuy nhiên có một sự khác biệt nhỏ, quá trình lọc đầu vào áp dụng cho tất cả các loại dữ liệu, loại bỏ những nội dung không hợp lệ trong khi lọc đầu ra chỉ mang tính áp dụng lại, mục đích bài trừ các loại mã độc còn xót lại.

Có hai loại thanh lọc dữ liệu đầu vào và đẩu ra: ***White-List Filtering*** và ***Black-List Filtering***

**Black-List Filtering**

Lọc dữ liệu được định nghĩa sẵn trong 1 danh sách cho trước, khi gặp 1 yêu cầu không hợp lệ sẽ hủy, không thực hiện yêu cầu. Ưu điểm là dễ cấu hình, triển khai nhưng nhược điểm là khi xuất hiện một cuộc tấn công kiểu mới (chưa được định nghĩa trong black-list) thì không thể phát hiện và ngăn chặn cuộc tấn công.

**White-List Filtering**

Cho phép quy định sẵn trước 1 danh sách hợp lệ, chỉ có những yêu cầu thuộc danh sách này mới được thực hiện. Vì thế ngăn chặn được các kiểu tấn công mới, nhược điểm là khi có một ứng dụng mới được phát triển thì cũng phải được cập nhật trong White-List. Tuy nhiên White-List Filtering bảo mật hơn so với Black-List Filtering.

### 4.2. Input Encoding

Mã hóa đầu vào có thể trở thành một vị trí trung tâm cho tất cả các bộ lọc, đảm bảo chỉ có một điểm duy nhất cho tất cả các bộ lọc.

Mã hóa phía máy chủ là một tiến trình mà tất cả nội dung phát sinh động sẽ đi qua một hàm mã hóa nơi mà các thẻ script sẽ được thay thể bởi mã của nó. Nói chung, việc mã hóa (encoding) được khuyến khích sử dụng vì nó không yêu cầu bạn phải đưa ra quyết định những kí tự nào là hợp lệ hoặc không hợp lệ.Tuy nhiên việc mã hóa tất cả dữ liệu không đáng tin cậy có thể tốn tài nguyên và ảnh hưởng đến khả năng thực thi của một số máy chủ.

Không được tin tưởng bất kì thứ gì người dùng nhập vào!! Hãy sử dụng hàm encode có sẵn trong [ngôn ngữ](https://toidicodedao.com/2015/07/30/c-la-ngon-ngu-tuyet-voi-nhat-java-php-c-c-ruby-chi-toan-la-thu-re-tien/)[/framework](https://toidicodedao.com/2015/06/23/cach-tiep-can-1-ngon-ngucong-nghe-moi-phan-1/) để chuyển các kí tự < > thành &lt; %gt;.

### 4.3. ****Validation/Sanitize****

### 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.

Ngoài ra, nếu muốn cho phép người dùng nhập vào HTML, hãy sử dụng các [thư viện](https://toidicodedao.com/2015/04/16/tang-suc-manh-cho-javascript-voi-lodash/) sanitize. Các thư viện này sẽ lọc các thẻ HTML, CSS, JS nguy hiểm để chống XSS. Người dùng vẫn có thể sử dụng các thẻ <p>, <span>, <ul> để trình bày văn bản.

### 4.4. ****CSP (Content Security Policy)****

### Hiện tại, ta có thể dùng chuẩn CSP  để chống XSS. Với CSP, [trình duyệt](https://toidicodedao.com/2015/09/17/tai-sao-gioi-lap-trinh-thu-ghet-internet-explorer/) chỉ chạy JavaScript từ những domain được chỉ định. Giả sử thiendia.com có sử dụng CSP, chỉ chạy JavaScript có nguồn gốc thiendia.com.  Vì Khoa để mã độc trên khoatran.com nên đoạn JavaScipt sau sẽ không được thực thi.

[screen-shot-2016-10-01-at-10-23-06-pm](https://toidicodedao.files.wordpress.com/2016/10/screen-shot-2016-10-01-at-10-23-06-pm.png)

Để sử dụng CSP, server chỉ cần thêm header **Content-Security-Policy** vào mỗi response. Nội dung header chứa những  domain mà ta tin tưởng.

[screen-shot-2016-10-01-at-10-24-35-pm](https://toidicodedao.files.wordpress.com/2016/10/screen-shot-2016-10-01-at-10-24-35-pm.png)

**Chương 3: Kỹ thuật tấn công SQL Injection**

## **1. Tại sao SQL Injection lại “Nguy hiểm”?**

**SQL Injection** là một kỹ thuật lợi dụng những lỗ hổng về câu truy vấn lấy dữ liệu của những website không an toàn trên web, đây là một kỹ thuật tấn công rất phổ biến và sự thành công của nó cũng tương đối cao.

Thiệt hại như thế nào?:

* Có thể gây ra những thiệt hại khổng lồ. Với SQL Injection, hacker có thể truy cập một phần hoặc toàn bộ dữ liệu trong hệ thống.
* Lỗ hổng này rất nổi tiếng, từ developer đến hacker gần như ai cũng biết. Ngoài ra, còn có 1 số tool tấn công SQL Injection cho dân “ngoại đạo”, những người không biết gì về lập trình.
* Rất nhiều ông lớn từng bị dính – Sony, Microsoft UK, VNG..v..v... Mọi vụ lùm xùm liên quan tới “lộ dữ liệu người dùng” ít nhiều đều dính dáng tới SQL Injection.

**Hậu quả của SQL Injection**

* Hậu quả lớn nhất mà SQL Injection gây ra là: Làm lộ dữ liệu trong database. Tuỳ vào tầm quan trọng của dữ liệu mà hậu quả dao động ở mức nhẹ cho đến vô cùng nghiêm trọng.
* Lộ dữ liệu khách hàng có thể ảnh hưởng rất nghiêm trọng đến công ty. Hình ảnh công ty có thể bị ảnh hưởng, khách hàng chuyển qua sử dụng dịch vụ khác, dẫn đến phá sản v…v...
* Lỗ hỗng này cũng ảnh hưởng lớn đến khách hàng. Do họ thường dùng chung một mật khẩu cho nhiều tài khoản, chỉ cần lộ mật khẩu một tài khoản thì các tài khoản khác cũng lộ theo.

Trong nhiều trường hợp, hacker không chỉ đọc được dữ liệu mà còn có thể chỉnh sửa dữ liệu. Lúc này hacker có thể đăng nhập dưới vai trò admin, lợi dụng hệ thống, hoặc xoá toàn bộ dữ liệu để hệ thống ngừng hoạt động.

## **2. Tấn công SQL Injection như thế nào?**

Cơ chế SQL Injection vô cùng đơn giản. Ta thường sử dụng câu lệnh SQL để truy cập dữ liệu. Giả sử, muốn tìm đăng nhập user, ta thường viết code như sau:

var username = request.username; // Phuonghuy2205

var password = request.password; // 12345678

var sql = "SELECT \* FROM Users WHERE Username = '" + username + "' AND Password = '" + password + "'";

// SELECT \* FROM Users WHERE Username = 'GiangLeLe' AND Password = '12345678'

Đoạn code trên đọc thông tin nhập vào từ user và cộng chuỗi để thành câu lệnh SQL. Để thực hiện tấn công, Hacker có thể thay đổi thông tin nhập vào, từ đó thay đổi câu lệnh SQL.

var password = request.password; //"or 1=1--";

var sql = "SELECT \* FROM Users WHERE Username = '" + username + "' AND Password = '" + password + "'";

// SELECT \* FROM Users WHERE Username = 'GiangLeLe' AND Password = '' OR '' = ''

// Câu SQL này luôn cho kết quả true

Hacker có thể thông qua SQL Injection để dò tìm cấu trúc dữ liệu (Gồm những table nào, có những column gì), sau đó bắt đầu khai thác dữ liệu bằng cách sử dụng các câu lệnh như UNION, SELECT TOP 1…

## **3. Cách phòng chống**

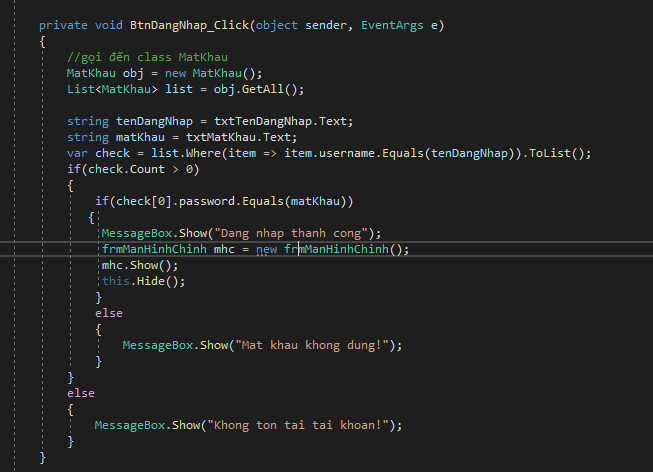
Gần đây, hầu như chúng ta ít viết SQL thuần mà toàn sử dụng ORM (Object-Relational Mapping) framework. Các framework web này sẽ tự tạo câu lệnh SQL nên hacker cũng khó tấn công hơn.

Tuy nhiên, có rất nhiều site vẫn sử dụng SQL thuần để truy cập dữ liệu. Đây chính là mồi ngon cho hacker. Để bảo vệ bản thân trước SQL Injection, ta có thể thực hiện các biện pháp sau.

* 1. **Lọc dữ liệu từ người dùng (Escaping Inputs):**

Cách phòng chống này tương tự như XSS. Ta sử dụng filter để lọc các kí tự đặc biệt (; ” ‘) hoặc các từ khoá (SELECT, UNION) do người dùng nhập vào. Nên sử dụng thư viện/function được cung cấp bởi framework. Các ngôn ngữ lập trình đều đã có thư viện hổ trợ.

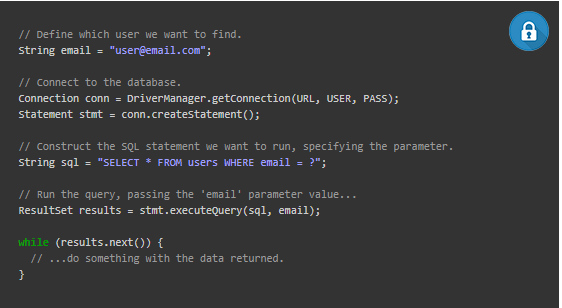
*Dùng thư viện LinQ trong C#*



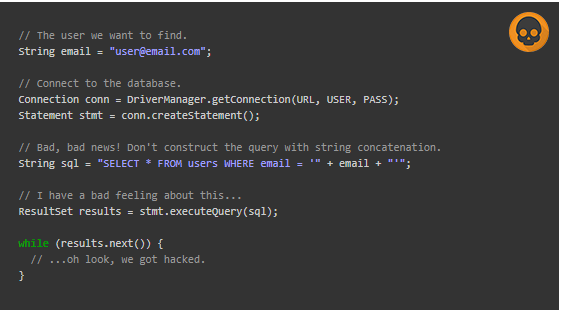
* 1. **Parameterized Statements** (**Không cộng chuỗi để tạo SQL**)**:**

Sử dụng parameter statement thay vì cộng chuỗi. Parameter Statement sẽ validade chuỗi người dùng nhập vào có hợp lệ hay không. Nếu dữ liệu truyền vào không hợp pháp, SQL Engine sẽ tự động báo lỗi, ta không cần dùng code để check.

*Đây là code an toàn*



*Đây là code cộng chuỗi không an toàn, dễ bị tấn công*



* 1. **Không hiển thị exception, message lỗi:**

 Hacker dựa vào message lỗi để tìm ra cấu trúc database. Khi có lỗi, ta chỉ hiện thông báo lỗi chứ đừng hiển thị đầy đủ thông tin về lỗi, tránh hacker lợi dụng.

* 1. **Principle of Least Privilege** (**Phân quyền rõ ràng trong DB):**

 Nếu chỉ truy cập dữ liệu từ một số bảng, hãy tạo một account trong DB, gán quyền truy cập cho account đó chứ đừng dùng account root hay sa. Lúc này, dù hacker có inject được sql cũng không thể đọc dữ liệu từ các bảng chính, sửa hay xoá dữ liệu.

Phân quyền hạn chế thấp nhất cho các user, ví dụ trang nào người dùng chỉ có thể đọc, trang nào người dùng có thể thao tác…v..v..

* 1. **Backup dữ liệu thường xuyên:**

 Dữ liệu phải thường xuyên được backup để nếu có bị hacker xoá thì ta vẫn có thể khôi phục được dữ liệu.

* 1. **Password Hashing (mã hóa dữ liệu lưu trữ):**

Ví dụ chúng ta có một website, và ta cần lưu thông tin về người dùng (user) trong database. Database sẽ lưu giữ thông tin về “username”, “password”, và một số thông tin quan trọng khác của user. Khi đó, chúng ta phải đổi mặt với một mối đe doạ về bảo mật thông tin ở đây. Database lưu lại password của user để khi user đăng nhập vào hệ thống chúng ta có thể kiểm tra được password mà user nhập vào có đúng với password mà user đã đăng ký hay không. Nếu bằng cách nào đó database của chúng ta bị xâm phạm (bởi các lực lượng bên ngoài hoặc bên trong, bên trong ở đây có thể là chính người quản trị cơ sở dữ liệu, bên ngoài ở đây có thể là các hacker, kẻ phá hoại,…) thì tất cả các thông tin về user trong đó có password sẽ bị lộ. Đây chính là lý do mà chúng ta cần đến kỹ thuật gọi là “password hashing” – mã hoá password bằng hash function.

Thay về lưu password dưới dạng plaintext (dạng text bình thường mà con người có thể đọc hiểu được) thì password sẽ được đưa vào một hash function, hash value sinh ra từ hash function sẽ được dùng để lưu và database đại diện cho password. Tức là ta sẽ lưu vào database hash value của password chứ không lưu trực tiếp password. Khi user đăng nhập thì ta cũng làm tương tự, password mà user nhập vào sẽ được đưa vào hash function lúc đầu để tính ra hash value, hash value này sẽ được dùng để so sánh với hash value lưu trong database, nêu giống nhau thì có nghĩa là user đã nhâp đúng password và ngược lại. Bằng cách này thì nếu chẳng may database bị lấy trộm hoặc bị lộ ra ngoài thì kẻ xấu cũng chỉ biết hash value của password chứ không biết password thật là gì. Sẽ mất rất nhiều thời gian để tìm ra hoặc thậm chí không thể tìm ra được password thật của một user nào đó, nhờ đó giảm thiếu rủi ro cho hệ thống.

Một số thuật toán mã hóa trên web thường gặp:

* MD-5
* SHA-1
* SHA-2
* SHA-3

Hiện nay có ba thuật toán có thể sử dụng được một cách an toàn:

* PBKDF2
* bcrypt
* scrypt

ví dụ câu lệnh hash mật khẩu theo ID5 và SHA trong c#:

var md5 = new MD5CryptoServiceProvider();

var md5data = md5.ComputeHash(data);

or

var sha1 = new SHA1CryptoServiceProvider();

var sha1data = sha1.ComputeHash(data);