# **Báo cáo kiến thức tìm hiểu được**

## **SQL Injection**

1. **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 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.

1. **Ví dụ thực tiễn SQL Injection**

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.

1. **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.

1. **Ví dụ tấn công**

Trong form xử lý đăng nhập của bạn có đoạn code server xử lý:

String sql = "SELECT \* FROM tbl\_user WHERE username='"+ username +"' AND password = '"+ password +"'";

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;

1. **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.

1. **Cách phòng chống SQL Injection**

May thay, mặc dù SQL rất nguy hại nhưng cũng dễ 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.

* **Lọc dữ liệu từ người dùng:** 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. Viết lại từ đầu vừa tốn thời gian vừa dễ sơ sót.
* **Không cộng chuỗi để tạo SQL:** Sử dụng parameter thay vì cộng chuỗi. Nếu dữ liệu truyền vào không hợp pháp, SQL Engine sẽ tự động báo lỗi, ta không cần dùng code để check.
* **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.
* **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.
* **Backup dữ liệu thường xuyên:** Các cụ có câu “cẩn tắc vô áy náy”. Dữ liệu phải thường xuyên được backup để nếu có bị hacker xoá thì ta vẫn có thể khôi phục được. Còn nếu cả dữ liệu backup cũng bị xoá luôn thì … chúc mừng bạn, update CV rồi tìm cách chuyển công ty thôi!

## **Cross - Site Scripting (XSS)**

1. **XSS**

**Cross - Site Scripting hay còn được viết tắt là XSS** là một kỹ thuật tấn công bằng cách chèn vào những website động (ASP, PHP, CGI, …) những thẻ HTML hay những đoạn mã script nguy hiểm có thể gây hại cho những người sử dụng khác. Trong đó những đoạn mà nguy hiểm thường được viết bằng các Client Site Script như: JavaScript, Jscript, DHTML và cũng có thể là các thẻ HTML.

Ví dụ:

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

1. **Phân loại XSS**
2. **Non-persistent**

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.

1. **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.

1. **Các phương thức tấn công**
2. **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.

1. **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.

1. **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.

1. **NGĂN CHẶN XSS**
2. **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.

1. **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.

1. **Output Encoding**

Mục đích của việc mã hóa đầu ra (vì nó liên quan Cross Site Scripting) là chuyển đổi đầu vào không tin cậy vào một hình thức an toàn, nơi đầu vào sẽ được hiển thị như dữ liệu cho người sử dụng mà không thực hiện được như đang trong trình duyệt.

1. **Sử dụng thư viện**

Hiện nay có rất nhiều thư viện giúp ta ngăn ngừa XSS, chúng giúp ta thực hiện các bước ngăn chặn XSS như đã liệt kê ở trên. **Thậm chí là các framework để làm web cũng đã tích hợp sẵn rất nhiều các công nghệ chống loại hình tấn công này, tuy nhiên tất cả là không đủ nếu chung ta không có sự hiểu biết.**

## **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"

1. **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.

1. **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.

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.

1. **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.

**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

**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.

**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.

**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" thay vì sử dụng "site.com/admin" là an toàn hơn.

**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.

**Kiểm tra IP**

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