Lê Đức Quý

**Báo cáo**

1. **SQL Injection là gì?**
2. 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,… trên cơ sở dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy, lỗi này thường xảy ra 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...
3. Các kiểu tấn công thường gặp

* Tấn công bằng mệnh đề luôn đúng

Trường hợp user đăng nhập: username = admin, pass = admin thì câu truy vấn sẽ là:

|  |
| --- |
| SQL: select \* from user where username = 'admin' and password = 'admin' |

Nếu biến "userName & password" được nhập chính xác theo một cách nào đó bởi người dùng ác ý, nó có thể trở thành một câu truy vấn SQL với mục đích khác .

Ví dụ, ta nhập vào giá trị của biến userName & pass như sau:

|  |
| --- |
| a' or '1'='1 |

Thì khi truy vấn, câu lệnh của chúng ta sẽ trở thành mệnh đề luôn đúng như dưới đây:

|  |
| --- |
| SQL: select \* from user where username = 'a' or '1'='1' and password = 'a' or '1'='1' |

Dòng query trên sẽ select mọi thứ trong table user bất chấp username & password có bằng 'a' hay không. Bởi vì '1'='1' luôn luôn true, đó là lý do vì sao chúng ta có thể vượt qua màn hình đăng nhập một cách dễ dàng.

* **Tấn công phá hoại dữ liệu**

Lỗi SQL injection dạng này thường xảy ra do lập trình viên hay người dùng định nghĩa đầu vào dữ liệu không rõ ràng hoặc thiếu bước kiểm tra dữ liệu đầu vào để xác minh kiểu của dữ liệu mà người dùng nhập vào có phải là số hay không.

|  |
| --- |
| sql = "select \* from user where id = '$id' "; |

Ta có thể nhận thấy một cách rõ ràng id - trường số (kiểu int). Tuy nhiên, người dùng , thay vì nhập vào một số, họ có thể nhập vào một chuỗi ký tự, và do vậy có thể trở thành một câu truy vấn SQL hoàn chỉnh mới. Ví dụ, ta thiết lập giá trị của biến id là:

|  |
| --- |
| 1;DROP TABLE users |

Câu sql có thể được hiểu như sau:

|  |
| --- |
| SELECT \* FROM user WHERE id=1;DROP TABLE user; |

Dấu ( ; ) dùng để phân tách câu lệnh trong SQL, điều này đồng nghĩa với việc hacker có thể hành động thêm bất kì lệnh gì (như update,select,insert,delete..) vào trong câu lệnh trên, Như vậy sẽ rất nguy hiểm.

1. Phòng chống SQL Injection

Vậy để ngăn chặn hình thức tấn công SQL Injection chúng ta phải làm thế nào? Như các bạn đã thấy ở trên, Hacker lợi dụng sự lỏng lẻo về thiết kế của chương trình để tấn công, chúng ta cần có các biện pháp ngăn chặn:

* Dữ liệu nhập vào từ form giao diện người dùng phải được “làm sạch” trước khi tạo ra câu lệnh truy vấn.
* Đảm bảo dữ liệu đầu vào được lọc mà không ảnh hưởng đến cấu trúc của câu lệnh truy vấn.
* Phân quyền truy cập vào Database server với từng mức độ theo yêu cầu của từng chức năng.
* Ngoài ra để tránh các nguy cơ từ tấn công SQL Injection, nên chú ý loại bỏ bất kì thông tin kĩ thuật nào chứa trong thông điệp chuyển tới cho người dùng khi ứng dụng có lỗi. Các thông báo lỗi thông thường tiết lộ các chi tiết kĩ thuật có thể cho phép kẻ tấn công biết được điểm yếu của hệ thống.

1. **XSS**

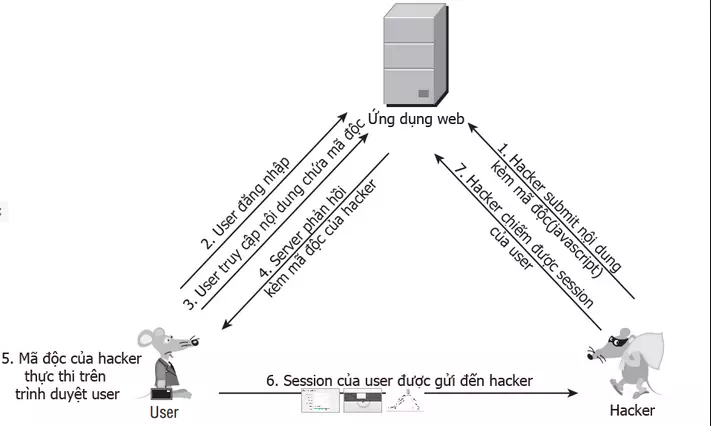
Cross Site Scripting (XSS) là một trong những tấn công phổ biến và dễ bị tấn công nhất mà tất cả các Tester có kinh nghiệm đều biết đến. Nó được coi là một trong những tấn công nguy hiểm nhất đối với các ứng dụng web và có thể mang lại những hậu quả nghiêm trọng. Giới thiệu về tấn công XSS Tấn công XSS 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.

1. **CSRF là gì?**
2. **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.
3. **Nguyên tắc hoạt động:**

Nguyên tắc hoạt động của CRSF rất đơn giản. Chúng ta biết rằng server sẽ lưu trữ cookie ở phía người dùng để phân biệt người dùng. Mỗi khi người dùng gửi một request tới một domain nào đó, cookie sẽ được gửi kèm theo.



* Đầu tiên, người dùng phải đăng nhập vào trang mình cần (Tạm gọi là trang A).
* Để dụ dỗ người dùng, hacker sẽ tạo ra một trang web độc.
* Khi người dùng truy cập vào web độc này, một request sẽ được gửi đến trang A mà hacker muốn tấn công (thông qua form, img, …).
* Do trong request này có đính kèm cookie của người dùng, trang web A đích sẽ nhầm rằng đây là request do người dùng thực hiện.
* Hacker có thể mạo danh người dùng để làm các hành động như đổi mật khẩu, chuyển tiền, ….

**3. Các kiểu tấn công thường gặp**

a. Kiểu 1. Dùng form

Với 1 phương pháp đơn gianr là : 1 trang web có form đổi mật khẩu có 2 field là password và password\_confirm , và 1 button Submit. Hacker sẽ tạo ra 1 web giả với 1 form ẩn với các gía trị tương tự form trên và gửi cho user. Khi user vào link và bấm button submit, thì 1 request đổi password được gửi đến trang web đó, kèm thoe cookie của account user. Cuối cùng hacker chỉ cần dùng email và mật khẩu mới để đăng nhập vào acc của user.

**b. Kiểu 2. Dùng thẻ img**

Ví dụ khi user A muốn chuyển 1 khoản tiền là 1000 cho user B, thì trang web ngân hàng sẽ tạo ra 1 URL, ví dụ URL đó như sau: http://jav.bank?from=Person1&to=Person2&amount=1000. Hacker sẽ bỏ URL này vào 1 thẻ img. Khi user truy cập vào trang, trình duyệt sẽ tự gọi GET request, gắn kèm với cookie của user. Thông qua cookie, ngân hàng xác nhận đó là user, rồi chuyển tiền cho Hacker.

**3. Cách phòng chống tấn công CSRF**

**a. 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

**b. Phía server**

* Cấu hình Web.config đơn giản giúp website an toàn
* 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.
* 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.
* 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