

# BÀI THỰC HÀNH SỐ 8

## KIỂM THỬ LỖ HỔNG SQL INJECTION

**Tự học lập trình với PHP, MySQL và HTML cơ bản để có thể hoàn thành bài thực hành này**

### 1 Giới thiệu chung

#### 1.1 Mục đích

- Hiểu về lỗ hổng SQL Injection
- Biết cách kiểm thử lỗ hổng SQL Injection
- Biết cách phòng ngừa lỗ hổng SQL Injection
- Làm quen với các công cụ hỗ trợ kiểm thử

#### 1.2 Yêu cầu kiến thức

- Đã biết lập trình Web cơ bản: PHP, HTML, Javascript
- Đã biết sử dụng hệ cơ sở dữ liệu quan hệ MySQL
- Có kiến thức về giao thức HTTP (<https://www.ietf.org/rfc/rfc2616.txt>)

#### 1.3 Chuẩn bị thực hành

- Môi trường thực hành: Linux/Windows cài đặt các công cụ Burpsuite, sqlmap
- Sử dụng cơ bản các ngôn ngữ lập trình Web: PHP, MySQL, HTML
- Đọc tài liệu về kiểm thử lỗ hổng SQL Injection: OWASP Web Security Testing Guide

#### 1.4 Cài đặt môi trường luyện tập

##### 1.4.1. Cài đặt và cấu hình Virtualbox

- **Bước 1:** Download phần mềm Virtualbox tại địa chỉ sau và cài đặt như một phần mềm thông thường trên Windows:

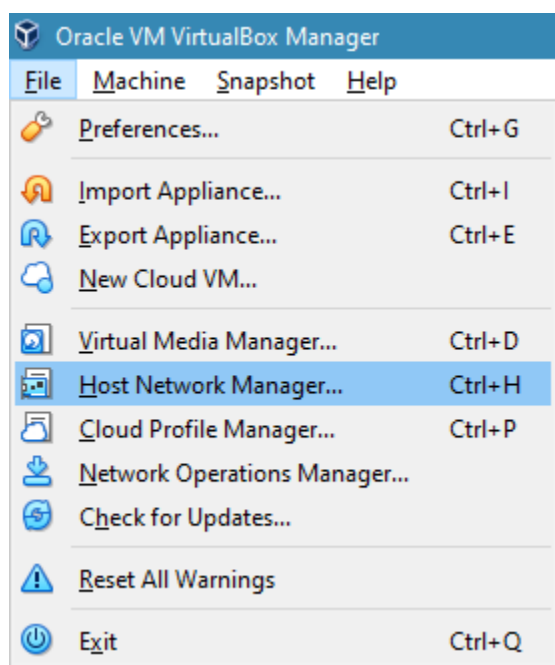
<https://download.virtualbox.org/virtualbox/6.1.12/VirtualBox-6.1.12-139181-Win.exe>

- **Bước 2:** Download gói mở rộng cho Virtualbox từ địa chỉ sau:

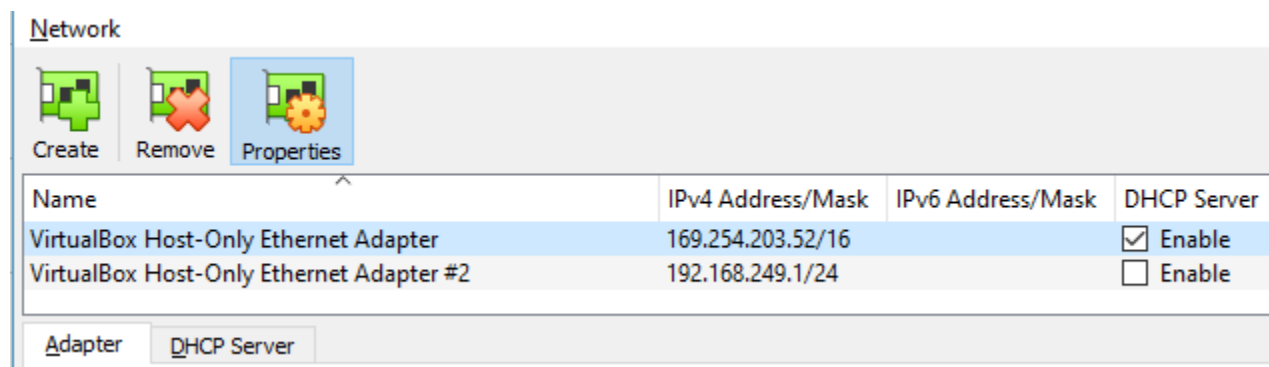
[https://download.virtualbox.org/virtualbox/6.1.12/Oracle\\_VM\\_VirtualBox\\_Extension\\_Pack-6.1.12.vbox-extpack](https://download.virtualbox.org/virtualbox/6.1.12/Oracle_VM_VirtualBox_Extension_Pack-6.1.12.vbox-extpack)

Sau khi download xong, nhấp đúp chuột vào file để cài đặt.

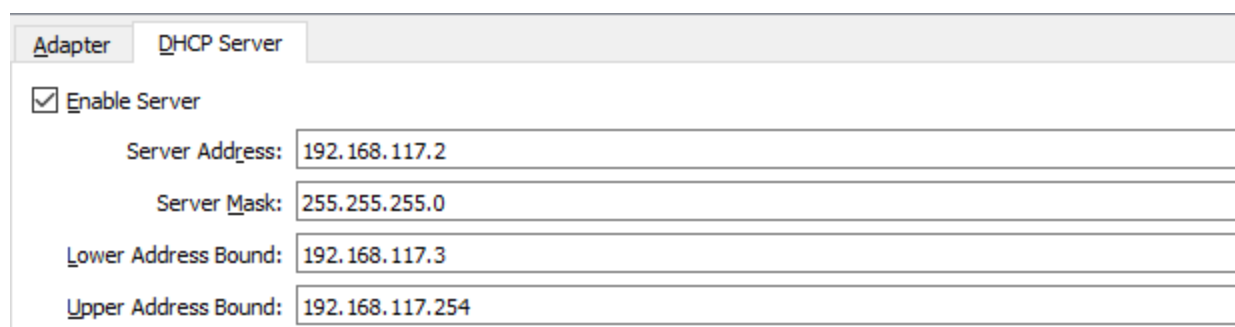
- **Bước 3:** Khởi động phần mềm Virtualbox
- **Bước 4:** Trên giao diện của Virtualbox, chọn File → Host Network Manager...



- **Bước 5:** Chọn các mạng ảo VirtualBox Host-Only Ethernet Adapter. Chọn



- **Bước 6:** Chọn thẻ Adapter và lựa chọn Configure Adapter Automatically.
- **Bước 7:** Chọn thẻ DHCP Server và thiết lập các thông số như hình ảnh sau:



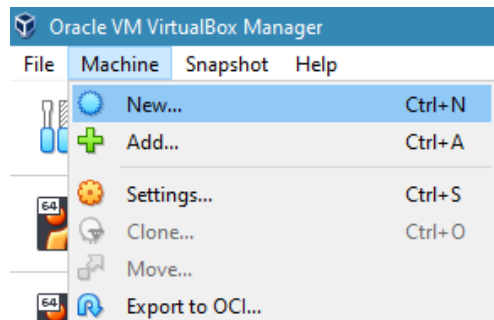
- **Bước 8:** Nhấp nút Apply và Close để hoàn tất.

#### 1.4.2. Triển khai máy ảo Web Server

- **Bước 1:** Download máy ảo từ địa chỉ sau và giải nén

[https://drive.google.com/file/d/1LSK\\_CZoha8LIKqr-8KkfyztZ6MM-eEF2/view?usp=sharing](https://drive.google.com/file/d/1LSK_CZoha8LIKqr-8KkfyztZ6MM-eEF2/view?usp=sharing)

- **Bước 2:** Trên cửa sổ chính của Virtualbox, chọn Machine → New...



- **Bước 3:** Trên cửa sổ tạo máy ảo, đặt các thông số như sau. Sau đó nhấn Next.
  - **Name:** Tên máy ảo
  - **Machine Folder:** Thư mục chứa máy ảo
  - **Type:** Linux
  - **Version:** Ubuntu (32-bit)

### Name and operating system

Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:

Machine Folder:

Type:

Version:

- **Bước 4:** Chọn dung lượng bộ nhớ RAM cho máy ảo là 2048 MB. Nhấn Next để tiếp tục.

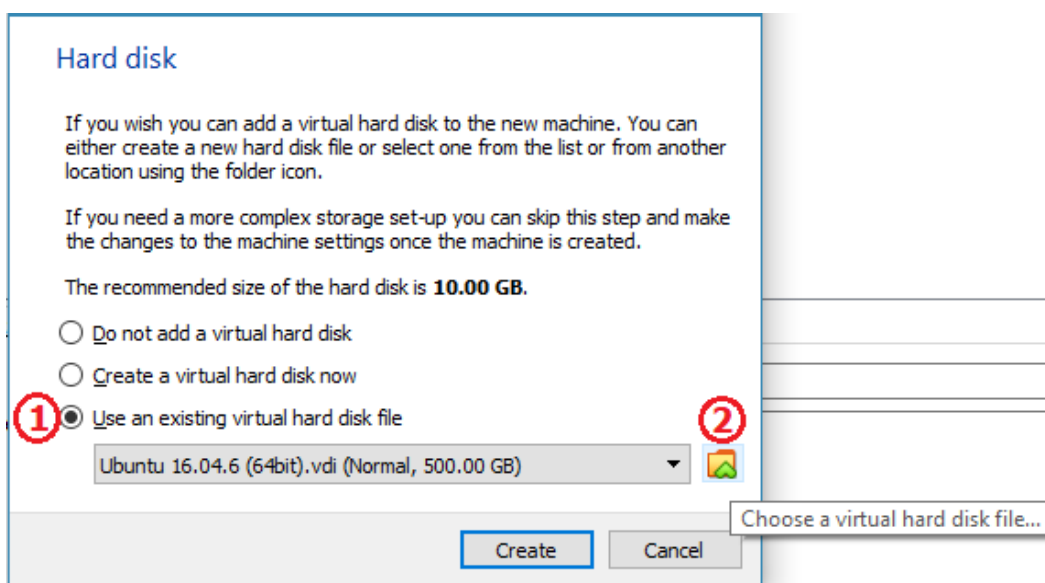
### Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

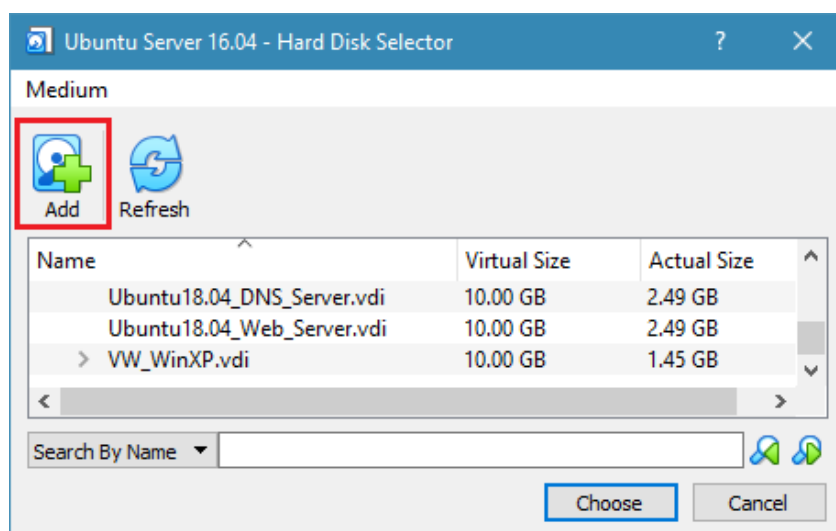
The recommended memory size is **1024 MB**.



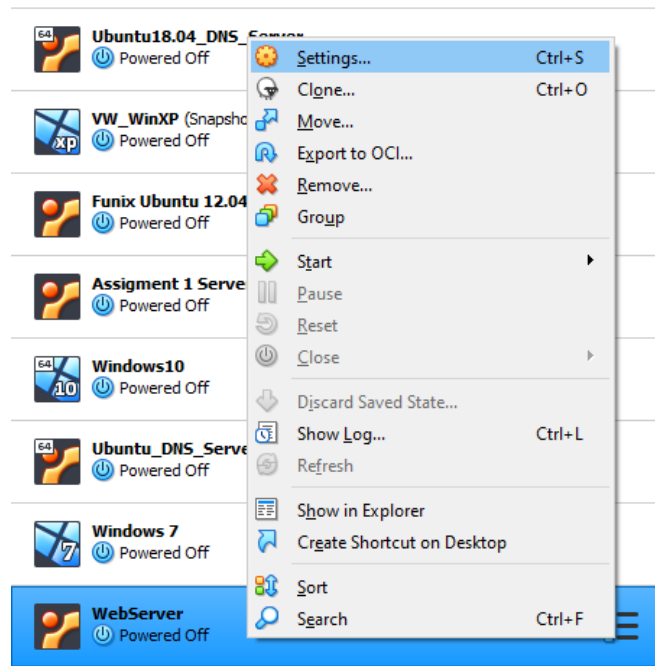
- **Bước 5:** Trong cửa sổ Hard disk tạo ổ cứng máy ảo, chọn mục **Use an existing virtual hard disk file**. Sau đó bấm nút **Choose a virtual hard disk file...**



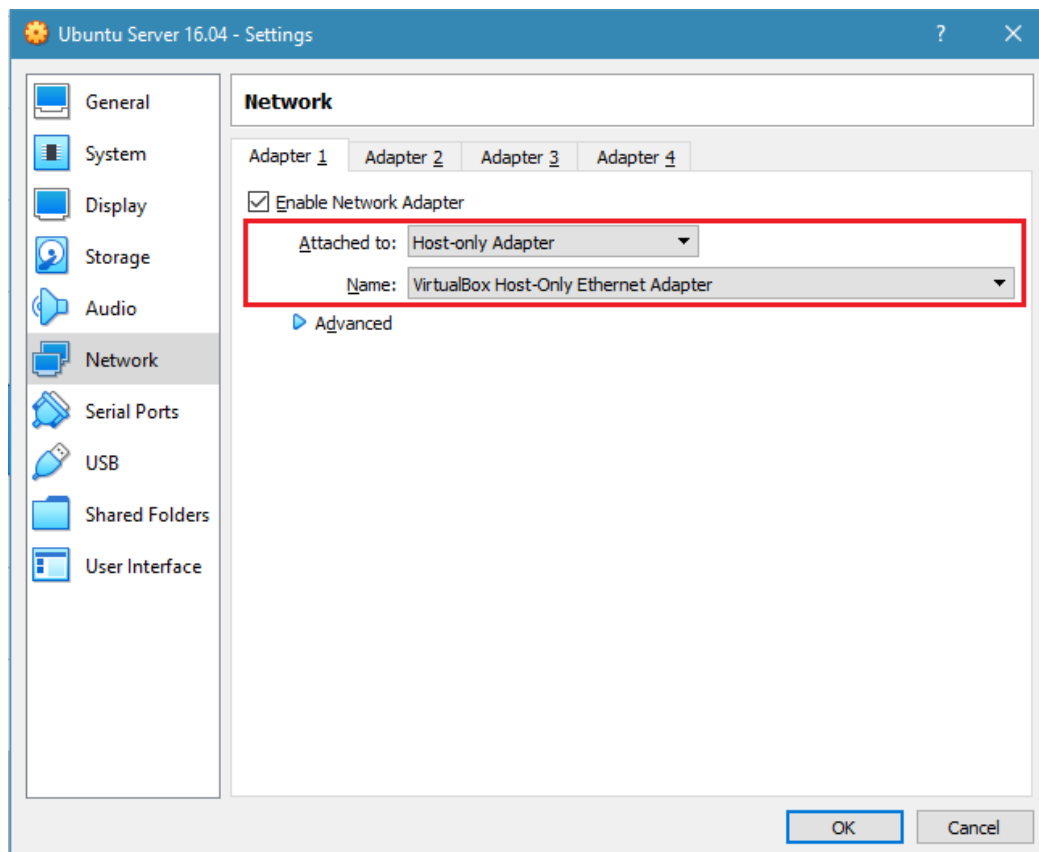
- **Bước 6:** Trên cửa sổ Hard Disk Selector, nhấn Add và chọn file Server.vdi đã download ở bước 1 để thêm vào danh sách



- **Bước 7:** Chọn file Server.vdi vừa được thêm vào trong danh sách ổ cứng ảo. Nhấn Choose để lựa chọn và đóng cửa sổ.
- **Bước 8:** Trên cửa sổ Hard disk, nhấn Create để tạo ổ cứng ảo.
- **Bước 9:** Trên cửa sổ chính của Virtualbox, chọn máy ảo vừa tạo và nhấp chuột phải. Chọn Settings...



- **Bước 10:** Chọn Network → Adapter 1. Thiết lập các thông số như sau:
  - **Attached to:** Host-only Adapter
  - **Name:** VirtualBox Host-Only Ethernet Adapter (hoặc còn gọi là VirtualBox Host-Only Network)



Sau khi máy ảo khởi động xong, đăng nhập bằng tài khoản sau:

- Username: bkcs
- Password: bkcs

Khi thực hiện các nội dung luyện tập trong tài liệu hướng dẫn bài thực hành số 5 và số 6, địa chỉ localhost khi truy cập vào các trang Web được thay thế bằng địa chỉ IP của Web Server. Trên Web Server mở cửa sổ Terminal và thực hiện lệnh ifconfig. Trong hình ảnh minh họa sau, địa chỉ của Web Server là 192.168.117.24

```
bkcs@ubuntu:~$ ifconfig
eth14      Link encap:Ethernet  HWaddr 08:00:27:28:d0:98
           inet addr:192.168.117.24 Bcast:192.168.117.255 Mask
           inet6 addr: fe80::a00:27ff:fe28:d098/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:1 errors:0 dropped:0 overruns:0 frame:0
           TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:590 (590.0 B)  TX bytes:9115 (9.1 KB)
```

## 2 Lỗ hổng SQL Injection

### 2.1 Giới thiệu lỗ hổng SQL Injection

Lỗ hổng SQL Injection là lỗ hổng mà ứng dụng Web không phát hiện giá trị đầu vào cần xử lý có chứa các ký tự đặc biệt, từ khóa, câu lệnh truy vấn bằng ngôn ngữ SQL. Lỗ hổng này ảnh hưởng tới các cơ sở dữ liệu quan hệ sử dụng ngôn ngữ truy vấn SQL như MySQL, MariaDB, MS SQL, Oracle... SQL Injection là một trong những lỗ hổng bảo mật nguy hiểm nhất trên Website. Các giá trị khai thác lỗ hổng chèn vào câu truy vấn gốc và khiến câu truy vấn thực thi theo cách nằm ngoài mong đợi. Qua đó, kẻ tấn công có thể khai thác lỗ hổng này để vượt qua bước kiểm tra xác thực, đánh cắp dữ liệu quan trọng, sửa đổi giá trị dữ liệu và cấu trúc của cơ sở dữ liệu. Thậm chí, trong một số trường hợp, kẻ tấn công có thể thực thi các lệnh shell của hệ thống, tạo backdoor để xâm nhập vào hệ thống.

### 2.2 Một số kỹ thuật khai thác

- Thu thập thông tin: Để thực hiện khai thác lỗ hổng SQL Injection, trước tiên kẻ tấn công cần phải xác định các thông tin về phần mềm quản trị CSDL. Một số cách thức có thể thực hiện bao gồm:
  - Sử dụng các ký tự đặc biệt của ngôn ngữ SQL và quan sát thông báo lỗi trả về. Một số ký tự có thể sử dụng như dấu ' hay ;
  - Sử dụng hàm version()
  - Sử dụng một số toán tử đặc trưng của các hệ quản trị cơ sở dữ liệu, điển hình là toán tử nối chuỗi. Nếu sử dụng không đúng toán tử, ứng dụng có thể trả về kết quả báo lỗi:

MySQL: 'test' + 'ing'

SQL Server: 'test' 'ing'

Oracle: 'test' || 'ing'

PostgreSQL: 'test' || 'ing'

- Tautology-based: Thay đổi ý nghĩa của biểu thức kiểm tra điều kiện trong mệnh đề WHERE.

- Batched query (Stacked query): Đây là phương pháp áp dụng khả năng thực thi cùng lúc nhiều câu lệnh SQL của một số hệ quản trị cơ sở dữ liệu và khả năng hỗ trợ của ngôn ngữ lập trình. Phương pháp này rất mạnh mẽ và gây nguy hiểm ngay với hệ thống. Bằng cách thêm vào một dòng lệnh Update, Insert hay Delete, dữ liệu trong cơ sở dữ liệu của ứng dụng web không còn toàn vẹn nữa.

<i>Support</i>	<i>ASP</i>	<i>ASP.NET</i>	<i>PHP</i>
<i>MySQL</i>	<i>No</i>	<i>Yes</i>	<i>No</i>
<i>PostgreSQL</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>
<i>MS SQL</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>

*Bảng 1: Khả năng hỗ trợ batched query trong các ngôn ngữ và cơ sở dữ liệu*

- Sử dụng mệnh đề UNION:
  - Thăm dò cấu trúc của CSDL
  - Đánh cắp dữ liệu bằng cách gộp kết quả truy vấn khai thác vào kết quả truy vấn gốc.
- Sử dụng mệnh đề ORDER BY để thăm dò cấu trúc của CSDL
- Blind SQL Injection: Đây là kỹ thuật được sử dụng trong trường hợp không có dữ liệu thực sự nào được trả về khi thực hiện các kỹ thuật khai thác ở trên. Có 2 kỹ thuật khai thác chính:
  - Boolean-based: Suy luận dựa trên kết quả trả về khi chèn các lệnh kiểm tra điều kiện trong giá trị đầu vào
  - Time-based: Suy luận dựa trên thời gian thực hiện truy vấn
- **Second-order Injection:** Đây là kỹ thuật ít được sử dụng vì rất khó để nhận biết một ứng dụng web có bị mắc lỗi này hay không. Kỹ thuật này được mô tả như sau : Trước hết, hacker “inject” vào cơ sở dữ liệu một đoạn mã. Đoạn mã này chưa hề gây nguy hiểm cho hệ thống nhưng nó sẽ được sử dụng làm bàn đạp cho lần inject tiếp theo của hacker. Chúng ta hãy xem một ví dụ cụ thể để hiểu hơn về kỹ thuật này.

Một hacker truy cập vào một ứng dụng web và cố gắng đăng ký một tài khoản có username là “**administrator**’ --”. Sau đó hacker này thực hiện thao tác thay đổi mật khẩu. Thao tác thay đổi mật khẩu được ứng dụng web xử lý như sau :

update Account set pwd = "" + newpwd + "" where username = "" + username + "";

Với *username* đã đăng ký ở trên, câu truy vấn trên trở thành :

update Account set pwd = "" + newpwd + "" where username = ‘administrator’--’;

Như vậy, *hacker* đã thay đổi được *password* của tài khoản *administrator* và hoàn toàn có thể đăng nhập dưới tên tài khoản *administrator*. Ý đồ của hắn đã thành công.

## 2.3 Một số kỹ thuật vòng tránh kiểm tra đầu vào

- Bỏ qua hoặc thêm ký tự dấu cách. Ví dụ:

```
or 'a'='a'
or 'a' = 'a'
```

- Thêm vào ký tự ký tự tab, ký tự xuống dòng. Ví dụ:

```
or  
'a'=  
      'a'
```

- Sử dụng null byte: %00. Ví dụ:

```
%00' UNION SELECT password FROM Users WHERE username='admin'--
```

- Sử dụng ký tự chú thích: /\*\*/. Ví dụ:

```
'/**/UNION/**/SELECT/**/password/**/FROM/**/Users/**/WHERE/**/name/**/LIKE/**/'admin'
```

- Sử dụng URL Encoding. Ví dụ:

```
%27%20UNION%20SELECT%20password%20FROM%20Users%20WHERE%20name%3D%27admin%27--
```

- Sử dụng hàm char() để sử dụng mã ASCII. Ví dụ:

```
' UNION SELECT password FROM Users WHERE name=char(114,111,111,116)--
```

- Sử dụng Hexa Encoding. Ví dụ:

```
Select user from users where name = unhex('726F6F74')
```

- Sử dụng nối chuỗi. Ví dụ trong MS SQL:

```
EXEC('SEL' + 'ECT 1')
```

- Khai báo biến. Ví dụ trong MS SQL:

```
; declare @SQLIvar nvarchar(80); set @myvar = N'UNI' + N'ON' + N' SELECT' + N'password';  
EXEC(@SQLIvar)
```

### 3 Kiểm thử lỗ hổng SQL Injection

#### 3.1 Phương pháp chung

Việc kiểm thử lỗ hổng SQL Injection thường được thực hiện bằng các kỹ thuật kiểm thử xâm nhập, trong đó thực hiện các cách thức khai thác mà kẻ tấn công thường sử dụng ở trên. Có 3 phương pháp kiểm thử lỗ hổng SQL Injection:

- Kiểm thử hộp đen: Người kiểm thử chỉ biết điểm vào(input) của website mà không có thêm thông tin về website như cấu trúc chương trình, luồng xử lý dữ liệu, cơ sở dữ liệu. Trong trường hợp này, người kiểm thử thường phải thực hiện thêm công đoạn thu thập thông tin về website trước khi sử dụng các kỹ thuật kiểm thử. Mặt khác, người kiểm thử có thể cần sử dụng nhiều dạng giá trị đầu vào khác nhau và dựa vào kết quả trả về để thiết kế chuỗi đầu vào phù hợp cho kiểm thử.
- Kiểm thử hộp trắng: Người kiểm thử có đầy đủ thông tin về website. Do đó, bên cạnh các kỹ thuật kiểm thử qua tham số đầu vào, người kiểm thử có thể phân tích mã nguồn của website để phát hiện lỗ hổng. Tuy nhiên, việc phân tích mã nguồn không phải lúc nào cũng dễ dàng thực hiện nếu website của cấu trúc và luồng xử lý dữ liệu phức tạp.
- Kiểm thử hộp xám: Người kiểm thử có một phần thông tin về website. Phương pháp kiểm thử này kết hợp các kỹ thuật kiểm thử hộp xám và kiểm thử hộp đen. Kiểm thử hộp xám thường thực hiện đối với các website phát triển trên nền tảng một framework hoặc sử dụng các thư viện lập trình của bên thứ ba. Các bạn có thể đọc thêm về các bài viết sau về phát hiện lỗ hổng SQL Injection trong những tình huống này:



<https://freek.dev/1317-an-important-security-release-for-laravel-query-builder>

<https://snyk.io/blog/sequelize-orm-npm-library-found-vulnerable-to-sql-injection-attacks/>

### 3.2. Kỹ thuật kiểm thử hộp đen

Chúng ta sử dụng cách thức kiểm thử hộp đen khi không biết được cách thức xử lý dữ liệu đầu vào của website mục tiêu. Các bước thực hiện kiểm thử bao gồm:

- (1) Phát hiện tham số đầu vào và cách thức chúng được truyền tới server:
- (2) Xác định kiểu truy vấn
- (3) Phân tích tham số đầu vào
- (4) Phát hiện lỗ hổng
- (5) Kiểm thử mức độ ảnh hưởng

### 3.2 Xác định đầu vào

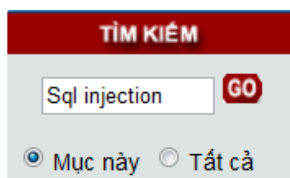
Các tham số đầu vào cho một Website có thể bao gồm:

- Các tham số được người dùng cung cấp truyền qua phương thức GET, POST
- Các giá trị của trường ẩn trong Web form
- Các giá trị được định nghĩa trước cho các lựa chọn trong Web form
- Các giá trị trong tiêu đề HTTP Request, điển hình như Cookie.

Để xác định được hết các giá trị đầu vào, chúng ta có thể sử dụng các công cụ như Burp Suite.

#### 3.2.1 Tham số đầu vào trong HTTP Request

Giá trị đầu vào điển hình thường đến từ các tham số trong đường dẫn URL, form nhập dữ liệu. Những dữ liệu này được trình duyệt Web gửi đến Server thông qua phương thức *HTTP GET* hay *POST* và trở thành các tham số cho ứng dụng web truy cập tới cơ sở dữ liệu. Ví dụ như trong một *form tìm kiếm*, khi người dùng điền vào “*Sql Injection*”, đơn giản ứng dụng web sẽ truy cập cơ sở dữ liệu và tìm ra các bản ghi mà nội dung của nó chứa từ khóa “*Sql Injection*” để trả lại kết quả cho người dùng.



Hình 1. Minh họa form tìm kiếm

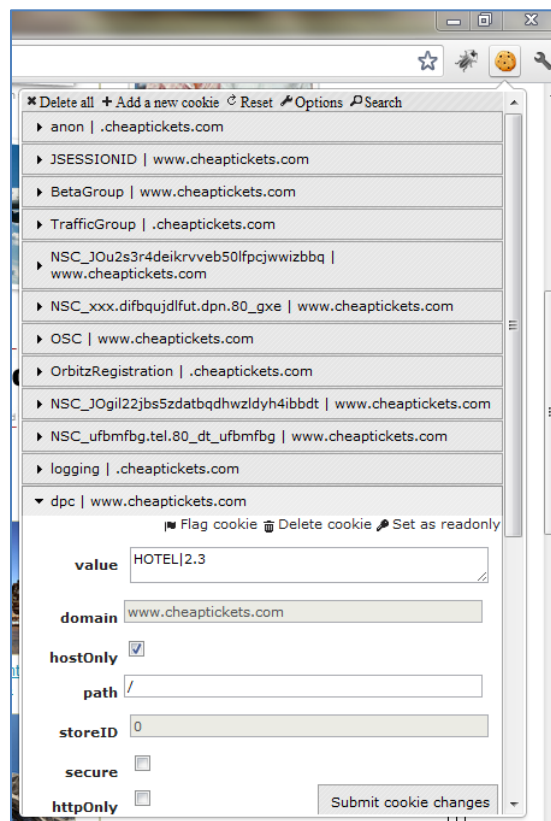
Một trường hợp phổ biến khác trong kỹ thuật tấn công Sql Injection, khi người dùng request một tài liệu mà các tham số của nó được truyền qua URL (như ở ảnh minh họa bên dưới, tham số *id* được truyền qua url theo phương thức *HTTP GET*). Khi nhận được request, ứng dụng web tìm trong cơ sở dữ liệu và trả về cho người dùng bài viết có *id=31*.



Hình 2. Minh họa tham số trong địa chỉ URL

Một tình huống ít phổ biến hơn là các giá trị tham số nằm trong trường ẩn của form HTML.

### 3.2.2 Tham số đầu vào trong Cookie



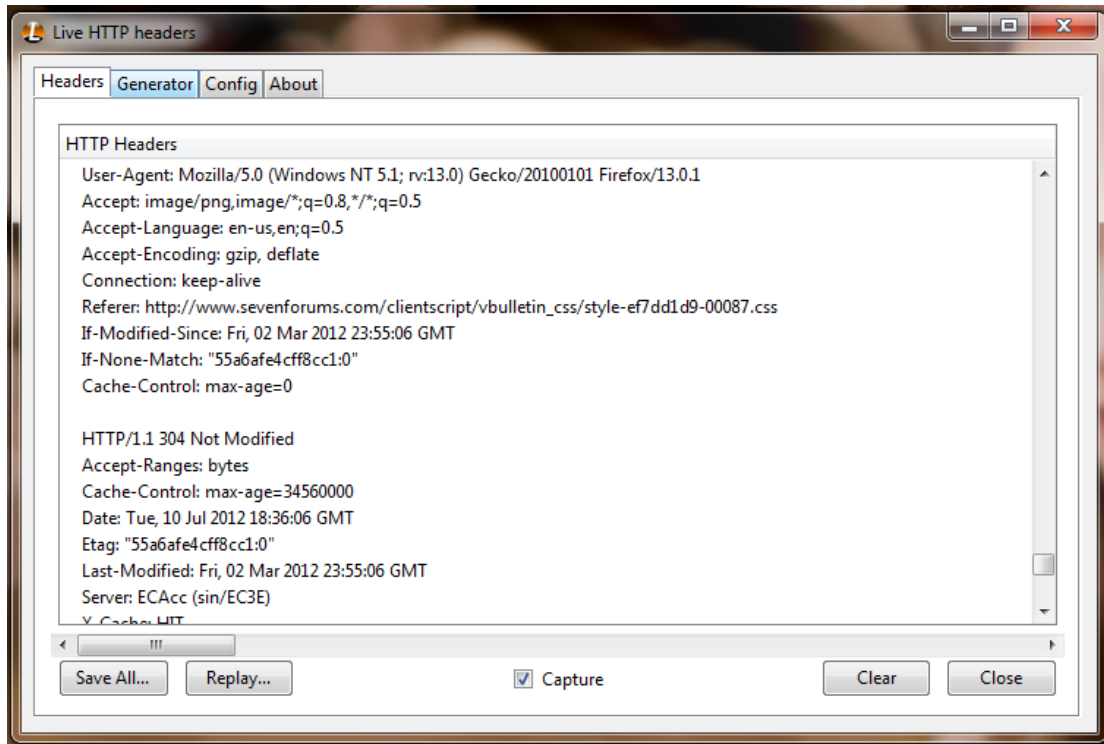
Hình 3. Dữ liệu cookie

Cookies lưu trữ thông tin trạng thái của người dùng khi truy cập các ứng dụng web. Những thông tin này do người lập trình quyết định, được tạo ra ở server và lưu trữ tại client. Khi người dùng truy cập lại ứng dụng web, cookies được trình duyệt gửi lên server giúp phục hồi lại những trạng thái của người dùng trong lần truy cập trước đó. Ở một số ứng dụng website, cookies còn được lưu trữ những sở thích của người dùng, khi đó ứng dụng sẽ sử dụng cookies để đưa ra những gợi ý tốt nhất cho người dùng khi mua sản phẩm. Do được lưu trữ ở client nên người dùng có thể chỉnh sửa tùy ý, vì vậy nếu ứng dụng web sử

dùng những thông tin lưu trong cookies mà không kiểm tra giá trị để xây dựng các truy vấn tới cơ sở dữ liệu thì hacker hoàn toàn có thể thực hiện một cuộc tấn công SQL Injection.

### 3.2.3 Thông qua các HTTP Header

Mặc dù không phổ biến lắm nhưng các giá trị được lưu trong HTTP Header có thể được ứng dụng web sử dụng như trong việc ghi nhật ký truy cập hay thống kê truy cập theo User Agent. Những công việc này đều có sự tương tác với cơ sở dữ liệu nên các hacker hoàn toàn có thể sử dụng giá trị của HTTP Header trong việc khai thác SQL Injection.



Hình 4. Header của một thông điệp HTTP Request

### 3.3. Xác định kiểu truy vấn

Xác định kiểu truy vấn là một bước quan trọng để chúng ta phán đoán cách thức giá trị đầu vào được sử dụng như thế nào trong câu truy vấn. Trên lý thuyết, giá trị tham số có thể xuất hiện ở mọi mệnh đề trong câu truy vấn. Trước tiên, hãy thử với các giá trị thông thường; dựa trên chức năng đang được kiểm thử và kết quả trả về để phán đoán kiểu truy vấn đang được sử dụng.

Các kiểu truy vấn điển hình bao gồm:

- SELECT: Tìm kiếm thông tin.
- INSERT INTO: Thêm mới thông tin.
- UPDATE: Sửa thông tin.
- DELETE: Xóa bản ghi dữ liệu.
- ALTER TABLE: Thay đổi cấu trúc bảng
- CREATE: Tạo CSDL hoặc bảng
- DROP: Xóa cơ sở dữ liệu hoặc bảng

### 3.4. Phân tích tham số đầu vào

Sau khi xác định được kiểu truy vấn, chúng ta thực hiện phán đoán các tham số đầu vào được sử dụng ở vị trí nào trong câu truy vấn. Tham số đầu vào thường được sử dụng như sau:

Các kiểu truy vấn điển hình bao gồm:

- SELECT: Tham số thường xuất hiện trong mệnh đề sau:
  - WHERE: Điều kiện tìm kiếm
  - HAVING: Điều kiện nhóm

Mặc dù hiếm khi xảy ra nhưng tham số cũng có thể xuất hiện trong những mệnh đề sau

- GROUP BY: Tên cột để nhóm kết quả
  - ORDER BY: Tên cột để sắp xếp kết quả
- INSERT INTO: Tham số thường xuất hiện trong mệnh đề VALUES đóng vai trò là dữ liệu cần thêm mới.
- UPDATE: Tham số thường xuất hiện trong mệnh đề sau:
  - SET: Thay đổi giá trị
  - WHERE: Điều kiện thay đổi
- DELETE: Tham số thường xuất hiện trong mệnh đề WHERE xác định điều kiện xóa

Lưu ý rằng, SQL cho phép lồng các câu truy vấn với nhau theo nhiều cách thức. Vì vậy, việc xác định đúng vị trí xuất hiện của mỗi tham số trong câu truy vấn lồng nhau là cần thiết để kiểm thử được mức độ ảnh hưởng khi truy vấn bị khai thác. Ví dụ hai câu truy vấn sau cho kết quả khác nhau:

```
SELECT DISTINCT supplier
FROM tbl_product
WHERE category = Tham_số_1 AND supplier NOT IN (
    SELECT DISTINCT supplier
    FROM tbl_product
    WHERE category = Tham_số_2);
```

```
SELECT DISTINCT supplier
FROM tbl_product
WHERE category = Tham_số_2 AND supplier NOT IN (
    SELECT DISTINCT supplier
    FROM tbl_product
    WHERE category = Tham_số_1);
```

### 3.5. Phát hiện lỗ hổng

Để phán đoán lỗ hổng có xảy ra hay không, cách thông thường nhất là chúng ta thêm các ký tự đặc biệt vào sau các giá trị tham số bình thường mà chúng ta đã thử ở trên. Các ký tự thường được sử dụng như ', ;, ), /\*, \*/, #, --. Lưu ý rằng cần phải kiểm thử lần lượt từng tham số đầu vào, trong khi đó sử dụng hằng số cho những tham số đầu vào khác. Các tình huống có thể xảy ra:

- Một thông báo lỗi của hệ quản trị CSDL. Đây là dấu hiệu rõ ràng nhất cho thấy có thể đã có lỗ hổng SQL Injection. Các thông báo lỗi này còn có ý nghĩa cho phép ta phán đoán, do thám được thông tin về quản trị cơ sở dữ liệu.
- Một thông báo lỗi được soạn thảo bởi người lập trình. Trong tình huống này có thể sẽ có lỗ hổng Blind SQL Injection hoặc không.
- Một trang chứa thông tin truy vấn hoặc kết quả thực hiện truy vấn. Trong tình huống này có thể có lỗ hổng SQL Injection hoặc giá trị đầu vào đã được làm sạch (Input Sanitization).

Việc sử dụng ký tự đặc biệt còn cho phép chúng ta phán đoán cách thức các giá trị tham số kiểu số được xử lý như thế nào trong câu truy vấn. Bảng sau đây thống kê các tình huống với X là giá trị số được truyền cho tham số đầu vào:

Giá trị kiểm thử Xử lý trong truy vấn	X'#	X')#	X'))#
X	Lỗi	Lỗi	Lỗi
'X'	Không lỗi	Lỗi	Lỗi
('X')	Lỗi	Không lỗi	Lỗi
((X'))	Lỗi	Lỗi	Không lỗi

Để khẳng định rõ ràng hơn về lỗ hổng cũng như mức độ ảnh hưởng, ta có thể kiểm thử thêm bằng một số kỹ thuật tấn công.

### 3.6. Kiểm thử mức độ ảnh hưởng

- Tautology-based: Thay đổi ý nghĩa của biểu thức kiểm tra điều kiện trong mệnh đề WHERE.
- Batched query (Stacked query): Đây là phương pháp áp dụng khả năng thực thi cùng lúc nhiều câu lệnh SQL của một số hệ quản trị cơ sở dữ liệu và khả năng hỗ trợ của ngôn ngữ lập trình. Phương pháp này rất mạnh mẽ và gây nguy hiểm ngay với hệ thống. Bằng cách thêm vào một dòng lệnh Update, Insert hay Delete, dữ liệu trong cơ sở dữ liệu của ứng dụng web không còn toàn vẹn nữa.

<i>Support</i>	<i>ASP</i>	<i>ASP.NET</i>	<i>PHP</i>
<i>MySQL</i>	<i>No</i>	<i>Yes</i>	<i>No</i>
<i>PostgreSQL</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>
<i>MS SQL</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>

*Bảng 2: Khả năng hỗ trợ batched query trong các ngôn ngữ và cơ sở dữ liệu*

- Sử dụng mệnh đề UNION:
  - Thăm dò cấu trúc của CSDL
  - Đánh cắp dữ liệu bằng cách gộp kết quả truy vấn khai thác vào kết quả truy vấn gốc.
- Sử dụng mệnh đề ORDER BY để thăm dò cấu trúc của CSDL
- Blind SQL Injection: Đây là kỹ thuật được sử dụng trong trường hợp không có dữ liệu thực sự nào được trả về khi thực hiện các kỹ thuật khai thác ở trên. Có 2 kỹ thuật khai thác chính:

- Boolean-based: Suy luận dựa trên kết quả trả về khi chèn các lệnh kiểm tra điều kiện trong giá trị đầu vào
- Time-based: Suy luận dựa trên thời gian thực hiện truy vấn
- **Second-order Injection:** Đây là kỹ thuật ít được sử dụng vì rất khó để nhận biết một ứng dụng web có bị mắc lỗi này hay không. Kỹ thuật này được mô tả như sau : Trước hết, hacker “*inject*” vào cơ sở dữ liệu một đoạn mã. Đoạn mã này chưa hề gây nguy hiểm cho hệ thống nhưng nó sẽ được sử dụng làm bàn đạp cho lần inject tiếp theo của hacker. Chúng ta hãy xem một ví dụ cụ thể để hiểu hơn về kỹ thuật này.

Một hacker truy cập vào một ứng dụng web và cố gắng đăng ký một tài khoản có username là “**administrator**’ --”. Sau đó hacker này thực hiện thao tác thay đổi mật khẩu. Thao tác thay đổi mật khẩu được ứng dụng web xử lý như sau :

update Account set pwd = "" + newpwd + "" where username = "" + username + "";

Với *username* đã đăng ký ở trên, câu truy vấn trên trở thành :

update Account set pwd = "" + newpwd + "" where username = 'administrator'--';

Như vậy, *hacker* đã thay đổi được *password* của tài khoản *administrator* và hoàn toàn có thể đăng nhập dưới tên tài khoản *administrator*. Ý đồ của hắn đã thành công.

### Một số cách thức vòng tránh

- Sử dụng các dạng in hoa, in thường: select, SELECT, SleCt
- Bỏ qua hoặc thêm ký tự dấu cách. Ví dụ:

or 'a'='a'

or 'a' = 'a'

- Thêm vào ký tự ký tự tab, ký tự xuống dòng. Ví dụ:

or  
'a'=  
'a'

- Sử dụng null byte: %00 trước ký tự nghi ngờ bị lọc. Ví dụ:

%00' UNION SELECT password FROM Users WHERE username='admin'--

- Sử dụng ký tự chú thích: /\*\*/. Ví dụ:

'/\*\*/UNION/\*\*/SELECT/\*\*/password/\*\*/FROM/\*\*/Users/\*\*/WHERE/\*\*/name/\*\*/LIKE/\*\*/'admin'

- Sử dụng hàm char() cho mã ASCII. Ví dụ:

' UNION SELECT password FROM Users WHERE name=char(114,111,111,116)--

## 4 Sử dụng công cụ hỗ trợ kiểm thử Burpsuite

Khi thực hiện kiểm thử SQL Injection, các giá trị kiểm thử không phải lúc nào cũng có thể truyền qua form nhập liệu hoặc tham số trên địa chỉ URL. Khi đó, việc sử dụng các công cụ hỗ trợ cho phép phát hiện dễ dàng hơn các điểm nhận giá trị đầu vào của ứng dụng Web và thay đổi các giá trị đó. Trong hầu hết các trường hợp, việc sử dụng tốt các công cụ hỗ trợ sẽ giúp quá trình kiểm thử thực hiện nhanh

chóng hơn. Chúng ta sẽ làm quen với công cụ Burpsuite để hỗ trợ kiểm thử lỗ hổng SQL Injection cho website.

Burpsuite là một ứng dụng được tích hợp nhiều tính năng phục vụ kiểm tra bảo mật ứng dụng web. Các tính năng này sẽ phục vụ kiểm tra bảo mật các thành phần khác nhau trong ứng dụng web hiện đại ngày nay. Burpsuite có thể giúp người dùng đánh giá các tiêu chí bảo mật web như: Kiểm tra cơ chế xác thực, kiểm tra các vấn đề về phiên người dùng hoặc liệt kê và đánh giá các tham số đầu vào của ứng dụng web v.v... Chương trình không những hỗ trợ mạnh mẽ trong quy trình đánh giá bảo mật thủ công mà còn bao gồm công cụ quét lỗ hổng bảo mật được tích hợp trong phiên bản có phí.

Burp được phát triển bởi Công ty PortSwigger Ltd và được phân phối thành hai phiên bản là Burp Free và Burp Professional. Download công cụ tại địa chỉ sau:

<https://portswigger.net/burp/freedownload>

Mặc dù trong phiên bản thương mại được tích hợp chức năng quét lỗ hổng ứng dụng tự động và một số tính năng nâng cao, tuy nhiên với sự kết hợp thành thạo các tính năng trong phiên bản miễn phí sẽ giúp người dùng tận dụng tối đa hiệu quả trong đánh giá bảo mật ứng dụng mà không cần đầu tư chi phí cho công cụ.

#### 4.1. Hướng dẫn cấu hình cơ bản

Người dùng Windows có thể thực thi chương trình bằng cách mở tập tin burpsuite\_free\_vxx.jar sau khi đã download và bảo đảm rằng Java Runtime đã được cài đặt trên máy tính.

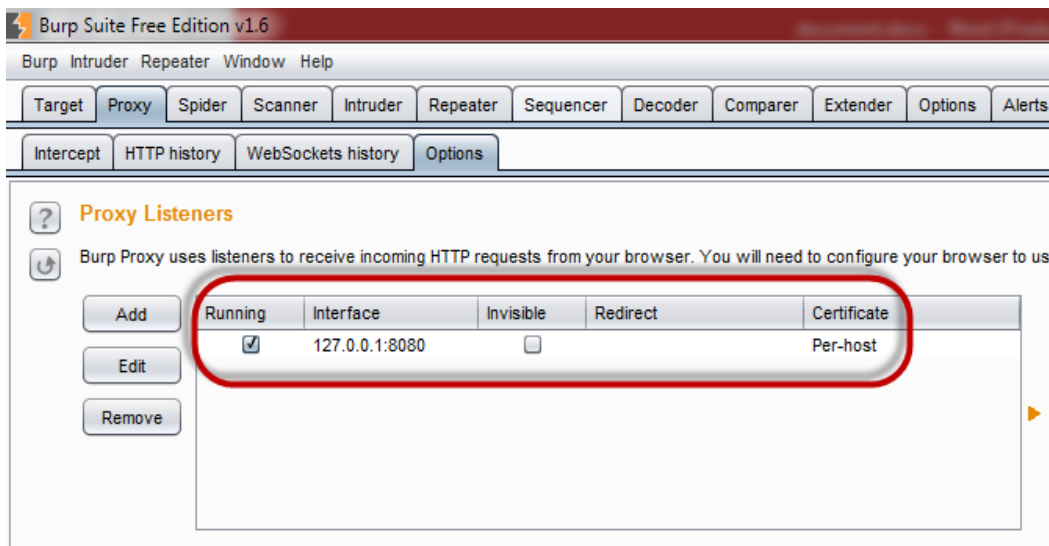


#### Kiểm tra hoạt động Burp

Burp Proxy đóng vai trò là chương trình trung chuyển các HTTP Request/Response giữa trình duyệt và ứng dụng web, gọi là Intercepting Proxy. Burp cho phép người dùng toàn quyền điều khiển việc gửi/nhận dữ liệu HTTP/s đến máy chủ và trình duyệt phục vụ việc đánh giá bảo mật ứng dụng web một cách cụ thể cho từng lỗ hổng bảo mật.

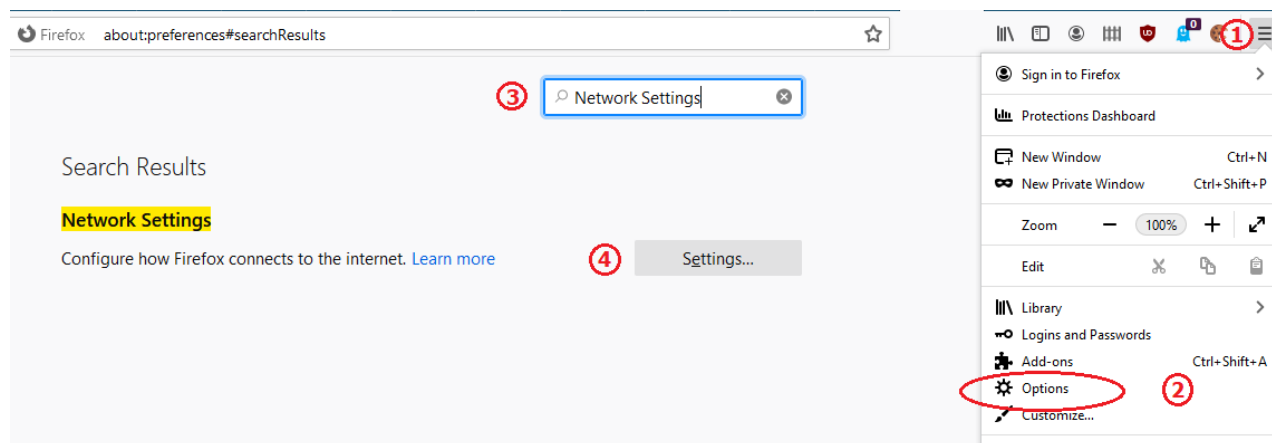
#### Cấu hình tại Burp Proxy

Theo mặc định, Burp Proxy được cấu hình lắng nghe trên cổng 8080/TCP. Để kiểm tra chắc chắn rằng không có chương trình hoặc dịch vụ nào khác đang lắng nghe trên cùng cổng 8080/TCP, bạn thực hiện kiểm tra tại thẻ Proxy | Options.



## Cấu hình tại trình duyệt

Thực hiện cấu hình Proxy tại trình duyệt Firefox như ảnh minh họa





**Configure Proxy Access to the Internet**

☐ No proxy  
☐ Auto-detect proxy settings for this network  
☐ Use system proxy settings  
☒ Manual proxy configuration

HTTP Proxy  Port   
☒ Also use this proxy for FTP and HTTPS

HTTPS Proxy  Port  Port  Port  SOCKS v4 ☒ SOCKS v5

Thiết lập cấu hình để cho phép proxy hoạt động với các các lưu lượng cục bộ (localhost):

- Bước 1: Mở một tab mới và truy cập tới trang about:config. Bỏ qua các cảnh báo nếu có
- Bước 2: Điền giá trị `network.proxy.allow_hijacking_localhost` vào ô tìm kiếm và thiết lập giá trị **true** cho nó

<b>network.proxy. allow_hijacking_localhost</b>	<b>true</b>	<input type="button" value="↗"/> <input type="button" value="5"/>
---	-------------	---

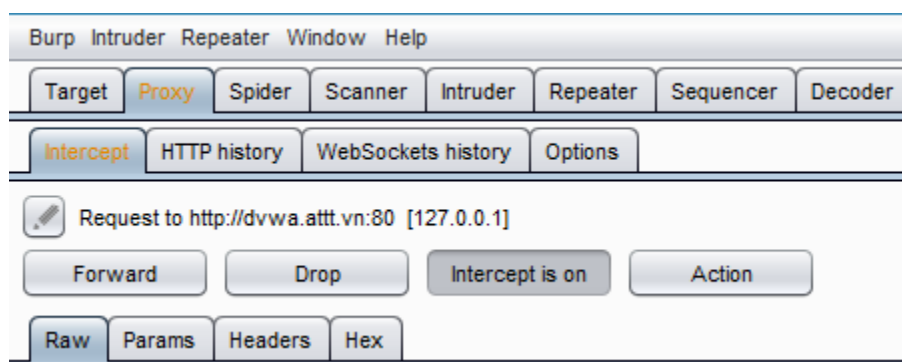
## 4.2. Sử dụng Burpsuite hỗ trợ kiểm thử SQL Injection

Phần này sẽ minh họa việc sử dụng Burpsuite để tương tác với website khi kiểm thử SQL Injection

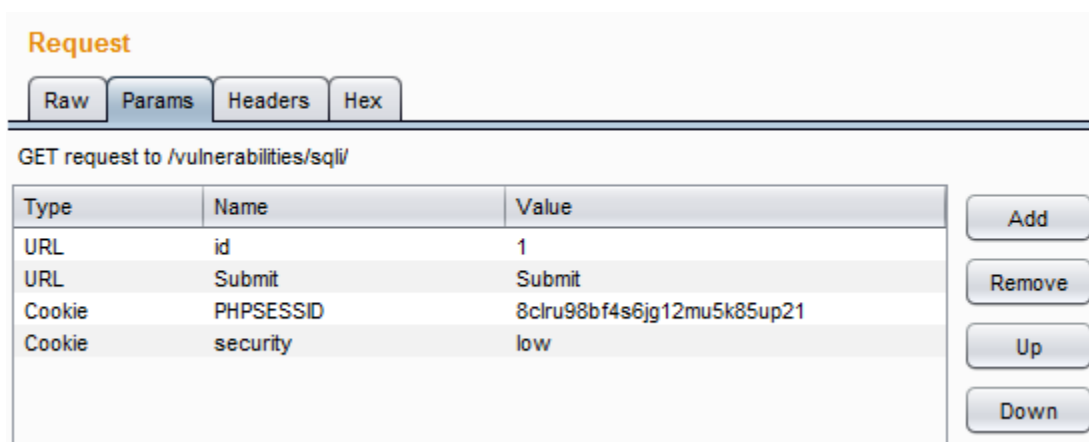
- Bước 1: Cấu hình và khởi động Burpsuite như hướng dẫn ở phần trước
- Bước 2: Trên cửa sổ công cụ Burpsuite, mở thẻ Proxy → Intercept và chắc chắn tính năng Intercept is on đã được bật
- Bước 3: Mở địa chỉ trang Web cần kiểm thử. Ví dụ dưới đây là giao diện kiểm tra lỗ hổng SQL Injection trên website DVWA.

<http://localhost/dvwa/vulnerabilities/sqli/>

- Bước 4: Điền các giá trị bất kỳ nào đó và gửi yêu cầu từ trình duyệt
- Bước 5: Trên thẻ Proxy → Intercept, nhấn nút Forward để chuyển tiếp yêu cầu



- Bước 6: Mở thẻ Proxy → HTTP history, chúng ta sẽ thấy danh sách các thông điệp HTTP mà Burpsuite đã bắt được. Chọn thông điệp HTTP Request tương ứng ở bước 4, nhấn chuột phải và chọn Send to Repeater
- Bước 7: Chọn thẻ Repeater. Thẻ này cho phép chúng ta thay đổi nội dung của HTTP Request và phát lại tới máy chủ. Thẻ con Params liệt kê danh sách các giá trị trên HTTP Header có thể là tham số đầu vào.



- Bước 8: Để thực hiện kiểm thử cho tham số đầu vào id chúng ta sẽ sửa trực tiếp trên thẻ con Raw. Chọn thẻ Decoder, điền chuỗi 1' or 1;# và chọn Encode as... → URL. Kết quả encode cho chúng ta xâu %31%27%20%6f%72%20%31%3b%23. Chọn lại thẻ Repeater, thay xâu giá trị này vào cho tham số id và nhấn nút Go. Thông điệp HTTP Response trả về từ server được hiển thị theo nhiều dạng khác nhau trong phần Response.

*(Giá trị các tham số cũng có thể sửa trực tiếp từ thẻ Params mà không cần qua encode)*

Raw Params Headers Hex

```

GET /vulnerabilities/sqli/?id=%31%27%20%6f%72%20%31%3b%23&Submit=Submit
HTTP/1.1
Host: dvwa.attt.vn
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:55.0) Gecko/20100101 Firefox/55.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://dvwa.attt.vn/vulnerabilities/sqli/
Cookie: PHPSESSID=8clru98bf4s6jg12mu5k85up21; security=low
Connection: close
Upgrade-Insecure-Requests: 1
DNT: 1

```

Request

Raw Params Headers Hex

GET request to /vulnerabilities/sqli/

Type	Name	Value
URL	id	1' or 1;#
URL	Submit	Submit
Cookie	PHPSESSID	8clru98bf4s6jg12mu5k85up21
Cookie	security	low

Add
Remove
Up
Down

## 5. Minh họa cách thức kiểm thử hộp đen

### 5.1. Cài đặt môi trường

- Bước 1: Download mã nguồn và cơ sở dữ liệu từ địa chỉ sau.

[https://users.soict.hust.edu.vn/tungbt/it4263/lab05\\_tut.zip](https://users.soict.hust.edu.vn/tungbt/it4263/lab05_tut.zip)

- Bước 2: Giải nén file download. Sử dụng WinSCP hoặc công cụ tương tự để upload thư mục lab05\_tut vừa giải nén được vào thư mục /home/bkcs của máy ảo

- Bước 3: Khởi động và truy cập vào máy ảo.

- Bước 4: Trên cửa sổ dòng lệnh, chuyển thư mục làm việc

```
cd /home/bkcs/lab05_tut
```

- Bước 3: Sao chép thư mục sqli vào thư mục /var/www/html

```
sudo cp -rf sqli /var/www/html
```

- Bước 4: Tạo cơ sở dữ liệu có tên là webvul và import file webvul.sql vào. Tài khoản truy cập mysql là root và mật khẩu là 123456

```
mysql -u root -p webvul < webvul.sql
```

## 5.2. Luyện tập

### 5.2.1. Ví dụ 1

- Bước 1: Truy cập địa chỉ `http://<Địa chỉ máy ảo>/sqli/sqli1.html`
- Bước 2: Lựa chọn một liên kết bất kỳ để truy cập
- Bước 3: Trang kết quả hiển thị cho thấy thông tin của một người dùng. Từ thanh địa chỉ có thể quan sát thấy có một tham số truyền vào theo phương thức GET là `cat` với giá trị được truyền là `1`.

`192.168.1.124/sqli/sqli1.php?cat=laptop`

- Bước 4: Sử dụng Burp Suite để chắc chắn rằng chỉ có tham số `cat` ở trên là duy nhất.
- Bước 5: Thực hiện thử truy cập bằng một số đường dẫn khác từ trang ban đầu, ta có thể đoán nhận kiểu truy vấn là `SELECT`
- Bước 6: Thử giá trị tham số là **`laptop'`** ta thấy có thông báo lỗi

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "laptop" at line 1

Từ thông báo này cho thấy có khả năng website đã có lỗ hổng SQL Injection. Hơn nữa, từ thông báo ta có thể thấy phần mềm CSDL là MySQL. Để khẳng định rõ hơn mức độ ảnh hưởng của lỗ hổng này, chúng ta có thể thực hiện thêm một số bước kiểm thử khác.

- Bước 7: Thử giá trị tham số là **`laptop';%23`**, trong đó `%23` là mã URL Encode của ký tự chú thích `#`. Ta thấy giá trị này được chấp nhận và có trang kết quả trả về. Từ đó ta đoán nhận được câu truy vấn có thể có dạng:

`SELECT <Danh sách cột>`

`FROM <Tên bảng dữ liệu>`

`WHERE <Tên cột> = 'Tham số đầu vào' <phần còn lại của câu truy vấn>`

Tham số đầu vào cần được đặt trong cặp dấu vì đây là phép so sánh xâu. Lưu ý rằng, SQL còn sử dụng toán tử `LIKE` để so sánh xâu. Khi đó, xâu đối sánh có thể chứa thêm các ký tự đại diện là `?` và `%`. Các bạn tự thực hiện kiểm thử để cho thấy, nếu toán tử `LIKE` được sử dụng thì các ký tự đại diện này không có mặt.

- Bước 8: Sử dụng giá trị đầu vào là **`any' or 1;%23`**. Kết quả cho thấy có danh sách sản phẩm hiển thị. Ở bước này, ta có thể khẳng định website có lỗ hổng SQL Injection. Lỗ hổng này có thể bị khai thác bởi kỹ thuật Tautology-based SQL Injection.

- Bước 9: Sử dụng toán tử ORDER BY để xác định số cột được truy vấn. Truyền giá trị tham số đầu vào là **laptop' ORDER BY 2;%23**. Trang kết quả trả về thành công cho thấy truy vấn có ít nhất 2 cột trong mệnh đề SELECT. Các bạn tự thực hiện với các giá trị khác để xác định số cột chính xác của truy vấn. Kết quả này cho thấy lỗ hổng có thể bị khai thác bởi mệnh đề ORDER BY.

- Bước 10: Sử dụng toán tử UNION để khai thác thông tin khác trong CSDL. Trong bước trên, khi thực hiện khai thác bằng mệnh đề ORDER BY 2 chúng ta thấy các sản phẩm được sắp xếp theo thông tin Vendor. Do đó, có thể phán đoán rằng cột thứ 2 chứa thông tin Vendor sẽ được hiển thị. Sử dụng giá trị đầu vào **laptop' UNION SELECT 1, database(), 3, 4, 5;%23**, chúng ta thấy trong danh sách sản phẩm cuối cùng có tên của CSDL mà website đang sử dụng:

Vendor : webvul

Model : 3

Price :4\$

Như vậy, có thể thấy lỗ hổng này có thể bị khai thác bởi kỹ thuật dùng mệnh đề UNION.

- Bước 11: Sử dụng kỹ thuật khai thác Boolean-based Blind SQL Injection.

- Sử dụng giá trị **laptop' and substring(database(),1,1)='a';%23** ta thấy không có danh sách sản phẩm trong trang kết quả trả về. Như vậy, tên CSDL mà website đang sử dụng không bắt đầu bằng chữ cái 'a'.
- Thực hiện tiếp tục cho tới giá trị **laptop' and substring(database(),1,1)='w';%23**, ta thấy có danh sách sản phẩm trong trang kết quả trả về. Như vậy tên CSDL mà website đang sử dụng bắt đầu bằng chữ cái 'w'

Như vậy, có thể thấy lỗ hổng này có thể bị khai thác bởi kỹ thuật Boolean-based Blind SQL Injection.

- Bước 11: Sử dụng kỹ thuật khai thác Timed-based Blind SQL Injection.

- Sử dụng giá trị **laptop' UNION SELECT 1, IF(SUBSTRING(database(),1,1) = 'a',BENCHMARK(500000,ENCODE('a','b')),null), 3, 4, 5;%23** ta thấy danh sách sản phẩm hiển thị ngay. Như vậy, hàm BENCHMARK() không thực hiện, tức là biểu thức trong lệnh IF là sai. Do đó, tên CSDL mà website đang sử dụng không bắt đầu bằng chữ cái 'a'.
- Sử dụng giá trị **laptop' UNION SELECT 1, IF(SUBSTRING(database(),1,1) = 'w',BENCHMARK(500000,ENCODE('a','b')),null), 3, 4, 5;%23** ta thấy mất một khoảng thời gian, danh sách sản phẩm hiển thị. Như vậy, hàm BENCHMARK() được thực hiện, tức là biểu thức trong lệnh IF là đúng. Do đó, tên CSDL mà website đang sử dụng không bắt đầu bằng chữ cái 'w'.

Như vậy, có thể thấy lỗ hổng này có thể bị khai thác bởi kỹ thuật Time-based Blind SQL Injection.

Ví dụ trên đã minh họa các bước cơ bản để kiểm thử lỗ hổng SQL Injection trên website với kiểu truy vấn SELECT. Có thể thấy rằng, các giá trị đầu vào hoàn toàn không được kiểm soát. Trong các trường hợp khác, chúng ta cần phải thử thêm các giá trị đầu vào với các kỹ thuật vòng tránh.

### 5.2.2. Ví dụ 2

- Bước 1: Truy cập địa chỉ `http://<Địa chỉ máy ảo>/sql/sql1.html`

- Bước 2: Điền các giá trị để thử chức năng. Từ kết quả, ta có thể phán đoán kiểu truy vấn như sau

`INSERT INTO <Tên bảng> (<Danh sách cột>)`

`VALUES (<Danh sách giá trị đầu vào>)`

- Bước 3: Sử dụng Burp Suite ta có thể xác định được có 4 tham số đầu vào từ form nhập dữ liệu truyền tới server bằng phương thức POST.

```
POST /sql/sql2.php HTTP/1.1
Host: 192.168.117.25
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
Accept: text/html,application/xhtml+xml,application/x
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 52
Origin: http://192.168.117.25
Connection: close
Referer: http://192.168.117.25/sql/sql2.html
Upgrade-Insecure-Requests: 1
```

`cat=keyboard&vendor=anyVendor&model=anyModel&price=0`

POST request to /sql/sql2.php		
Type	Name	Value
Body	cat	keyboard
Body	vendor	anyVendor
Body	model	anyModel
Body	price	0

- Bước 4: Quan sát từ trang kết quả, ta thấy cả 4 giá trị của tham số đầu vào đều được sử dụng trong câu truy vấn `INSERT INTO`.

Sau đây, ta sẽ kiểm tra xem có lỗ hổng SQL Injection trên tham số vendor không.

- Bước 5: Nhập giá trị `anyVendor'` cho mục Vendor, các mục khác sử dụng lại các giá trị như cũ. Kết quả nhận được là một thông báo lỗi “Error! Cannot add product”. Thông báo này chưa khẳng định được có lỗ hổng SQL Injection không.

- Bước 6: Ta thực hiện phán đoán vị trí của giá trị tham số vendor trong danh sách giá trị. Nhập giá trị **anyVendor’);#** cho mục Vendor, các mục khác sử dụng lại các giá trị như cũ. Kết quả nhận được là thông báo lỗi.

- Bước 7: Thử lần lượt các giá trị **anyVendor', '1');#**, **anyVendor', '1', '2');#** ta đều nhận được thông báo lỗi.

- Bước 8: Thử giá trị **anyVendor', '1', '2', '3');#** ta nhận được trang kết quả cho thấy một sản phẩm mới được thêm vào. Như vậy, có thể kết luận website có lỗ hổng SQL Injection ở vị trí tham số vendor.

Category	Vendor	Model	Price
3	anyVendor	1	2

Ngoài ra với kết quả như trên, mặc dù chưa chắc chắn nhưng dựa trên ngữ nghĩa, ta cũng xác định được thứ tự lần lượt giá trị của các tham số trong mệnh đề `VALUES` lần lượt là vendor, model, price, cat. Ta có thể thực hiện thêm một số kiểm thử để xác định mức độ ảnh hưởng của lỗ hổng này.

- Bước 9: Với vị trí của các tham số đã được xác định ở bước trên, ta có thể phán đoán câu truy vấn có thể như sau:

```
INSERT INTO <Tên bảng> (<Danh sách cột>)
```

```
VALUES (<Danh sách giá trị đầu vào khác>, 'vendor', 'model', 'price', 'cat')
```

Hoặc

```
INSERT INTO <Tên bảng> (<Danh sách cột>)
```

```
VALUES (<Danh sách giá trị đầu vào khác>, 'vendor', 'model', price, 'cat')
```

Do đó, ta sử dụng giá trị sau để kiểm thử **anyVendor', database(), '2', '3');**#. Trên trang kết quả, ta có thể thấy tên cơ sở dữ liệu là **webvul** đã được hiển thị.

Có thể sử dụng các hàm `version()`, `user()`,... để thấy lỗ hổng này có thể bị khai thác để do thám thông tin về cơ sở dữ liệu

- Bước 10: Tiếp tục sử dụng giá trị sau **anyVendor', (select 'anyModel'), '2', '3');**#. Trang kết quả có một sản phẩm mới được thêm vào. Như vậy có thể thấy lỗ hổng còn có thể bị khai thác bằng truy vấn **SELECT**. Điều này cho phép kẻ tấn công khai thác để lấy ra dữ liệu tùy ý tùy thuộc quyền truy cập của tài khoản trên phần mềm quản trị cơ sở dữ liệu.

- Bước 9: Thực hiện tương tự, ta cũng có thể xác định được ở vị trí các tham số còn lại cũng gặp lỗ hổng tương tự.

Ví dụ trên đã minh họa các bước cơ bản để kiểm thử lỗ hổng SQL Injection trên website với kiểu truy vấn **INSERT**. Có thể thấy rằng, các giá trị đầu vào hoàn toàn không được kiểm soát. Trong các trường hợp khác, chúng ta cần phải thử thêm các giá trị đầu vào với các kỹ thuật vòng tránh.

## 5 Nội dung thực hành

Kiểm thử lỗ hổng SQL Injection trên website **http://webvul.bkcs.vn**. Với mỗi nhiệm vụ, trình bày cách thức thực hiện và giải thích kết quả kiểm thử.