BÀI THỰC HÀNH SỐ 8

KIỂM THỬ LỖ HỔNG SQL INJECTION

***Họ và tên: Nguyễn Thanh Trường***

***MSSV: 1050080082***

**Câu 1*(3.5 điểm)***

Kiểm thử lỗ hổng SQL Injection trên trang web http://webvul.bkcs.vn/register.php

* Xác định các tham số đầu vào và cách thức chúng được gửi tới server **(0.25 điểm)**
* Trang register.php yêu cầu người dùng nhập:
  + username
  + password
  + confirm\_password
* Phân tích và phán đoán kiểu truy vấn **(0.25 điểm)**
  + Dựa vào chức năng, ta có thể phán đoán câu truy vấn dạng:
    - INSERT INTO users (username, password) VALUES ('$username', '$password');
    - Trong đó, $username và $password là các biến lấy trực tiếp từ đầu vào người dùng.
* Phân tích và phán đoán vị trí của các giá trị tham số đầu vào được sử dụng trong câu truy vấn **(0.5 điểm)**
  + $username và $password được sử dụng trực tiếp trong truy vấn SQL, không qua xử lý hay kiểm tra đầu vào kỹ lưỡng.
  + Điều này có thể tạo điều kiện để chèn thêm các câu lệnh SQL thông qua giá trị username.
* Xác định mức độ ảnh hưởng của lỗ hổng trên mỗi giá trị đầu vào **(2.5 điểm)**
  + testuser -> Kiểm tra đăng ký hợp lệ -> Thành công, người dùng được thêm
  + testuser');-- -> Chèn dấu ' và comment để kết thúc câu truy vấn -> SQL báo lỗi nếu không có filter
  + testuser', '123');-- -> Tạo SQL hoàn chỉnh: INSERT INTO users (username, password) VALUES ('testuser', '123');--') -> Có thể đăng ký thành công dù không đúng logic
  + ' OR 1=1-- -> Mục tiêu chèn để bỏ qua kiểm tra trùng lặp username -> Nếu không xử lý đúng, có thể gây trùng lặp, phá vỡ dữ liệu
  + abc'); DROP TABLE users;-- -> Thử phá hủy bảng users -> Nếu thành công => lỗ hổng nghiêm trọng

**Câu 2 *(3.5 điểm)***

Kiểm thử lỗ hổng SQL Injection trên trang web http://webvul.bkcs.vn/login.php

* Xác định các tham số đầu vào và cách thức chúng được gửi tới server **(0.25 điểm)**
* Phân tích và phán đoán kiểu truy vấn, phân tích và phán đoán vị trí của các tham số đầu vào được sử dụng trong câu truy vấn **(0.25 điểm)**
* Xác định mức độ ảnh hưởng của lỗ hổng trên mỗi giá trị đầu vào **(3 điểm)**

*Lưu ý:*

* *Nói rõ mục đích sử dụng các giá trị kiểm thử*
* *Giải thích rõ các phán đoán và suy luận từ kết quả khi sử dụng mỗi giá trị kiểm thử*

**Câu 3 *(3 điểm)***

Kiểm thử lỗ hổng SQL Injection trên trang web http://webvul.bkcs.vn/index.php

* Xác định các tham số đầu vào và cách thức chúng được gửi tới server **(0.25 điểm)**
* Phân tích và phán đoán kiểu truy vấn, vị trí của các tham số đầu vào được sử dụng trong câu truy vấn **(0.25 điểm)**
* Xác định mức độ ảnh hưởng của lỗ hổng trên mỗi giá trị đầu vào **(2.5 điểm)**

*Lưu ý:*

* *Nói rõ mục đích sử dụng các giá trị kiểm thử*
* *Giải thích rõ các phán đoán và suy luận từ kết quả khi sử dụng mỗi giá trị kiểm thử*

**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

14

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 hỗng 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.

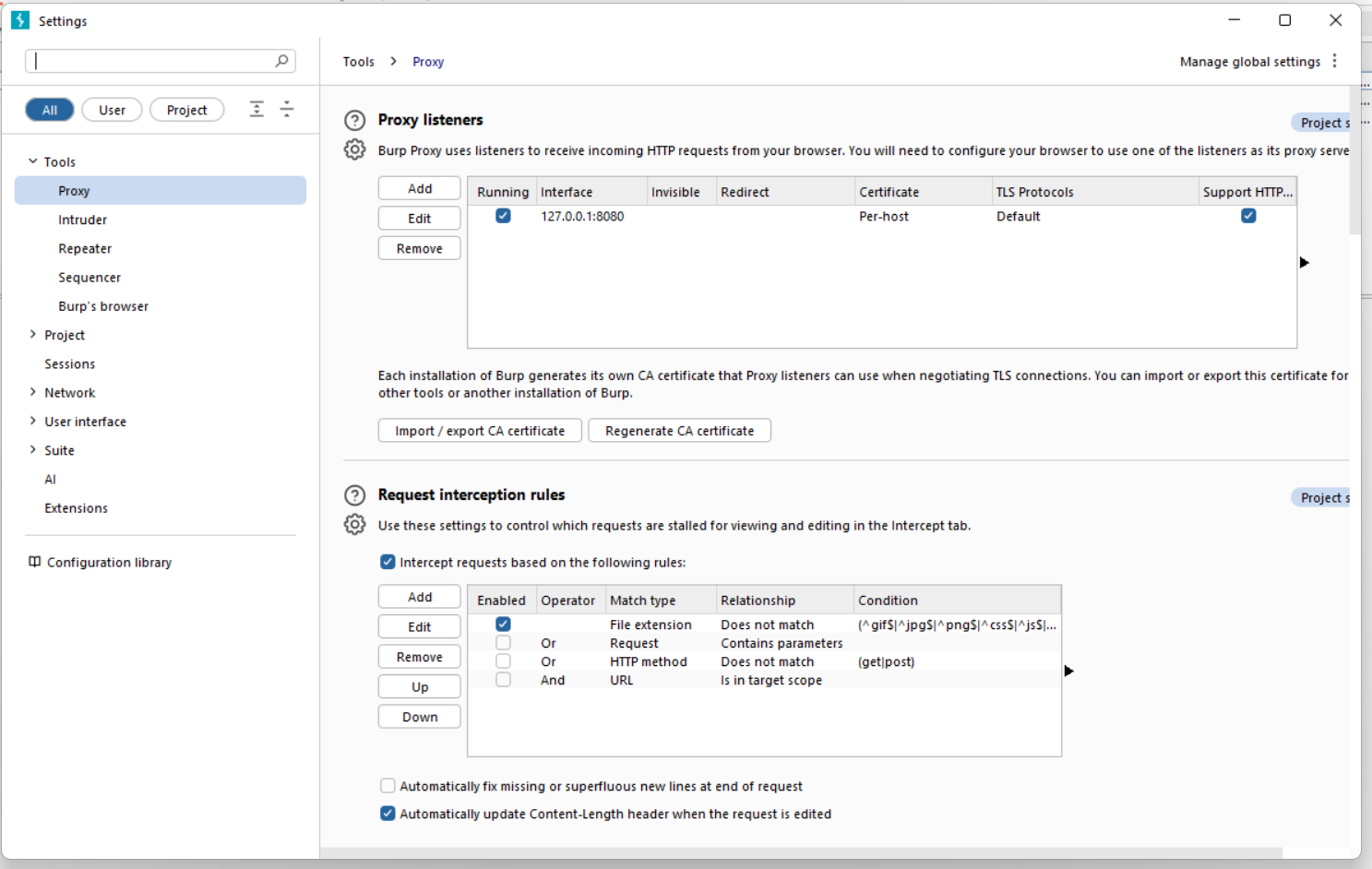
**Kiểm tra hoạt động Burp**

Burp Proxy đóng vai trò là chương trình trung chuyển các HTTP Request/Respone giữa trình duyệt và ứng dụng web, gọi là Intercepting Proxy. Burp cho phép người dùng toàn quyền điều khiển việc gởi/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.

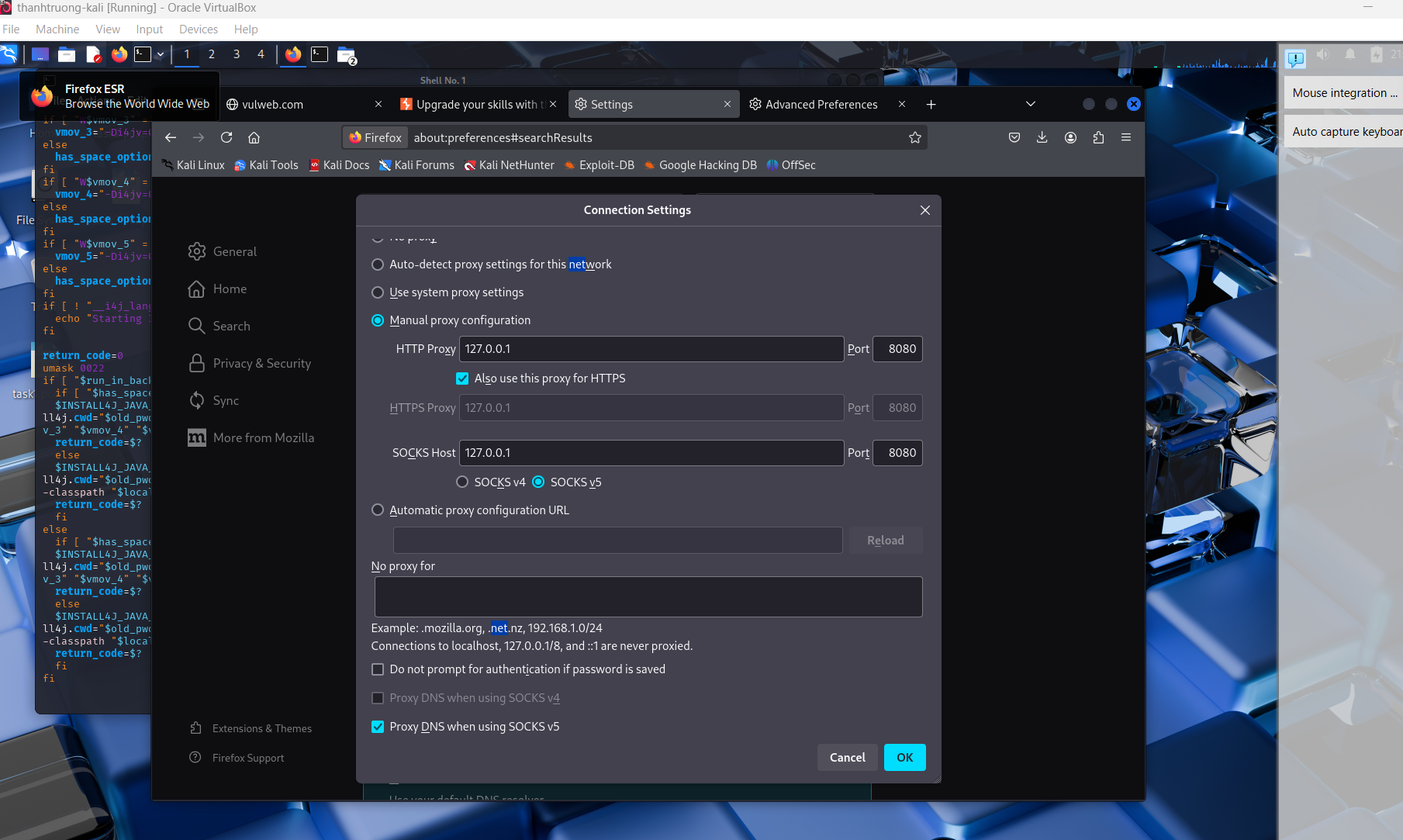
**Cấu hình tại Burp Proxy**

15

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.

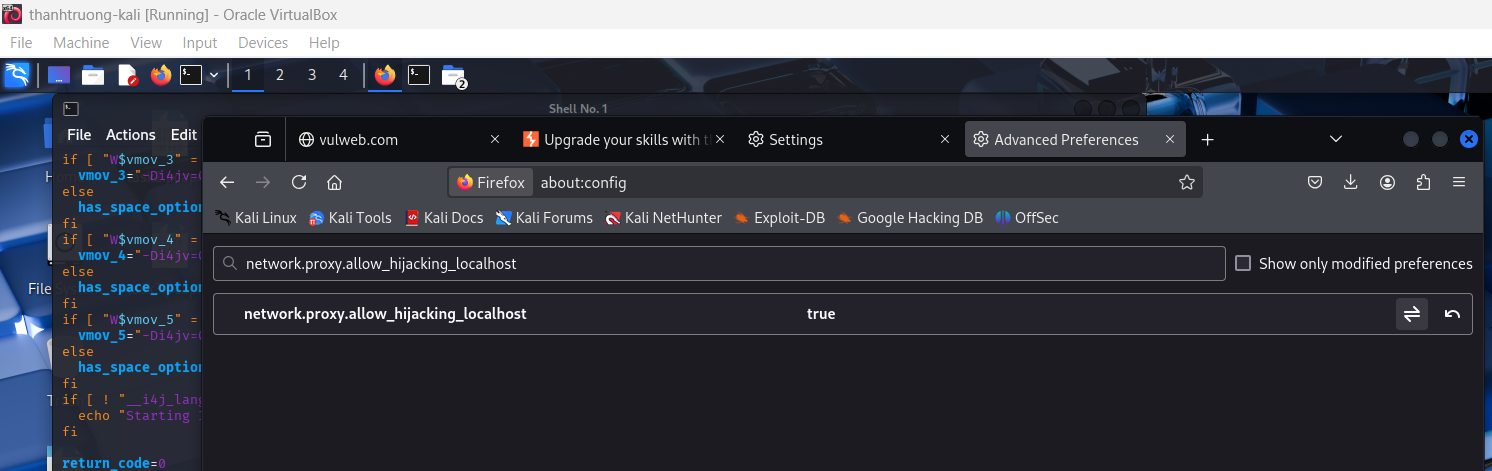
**Cấu hình tại trình duyệt**

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



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

**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

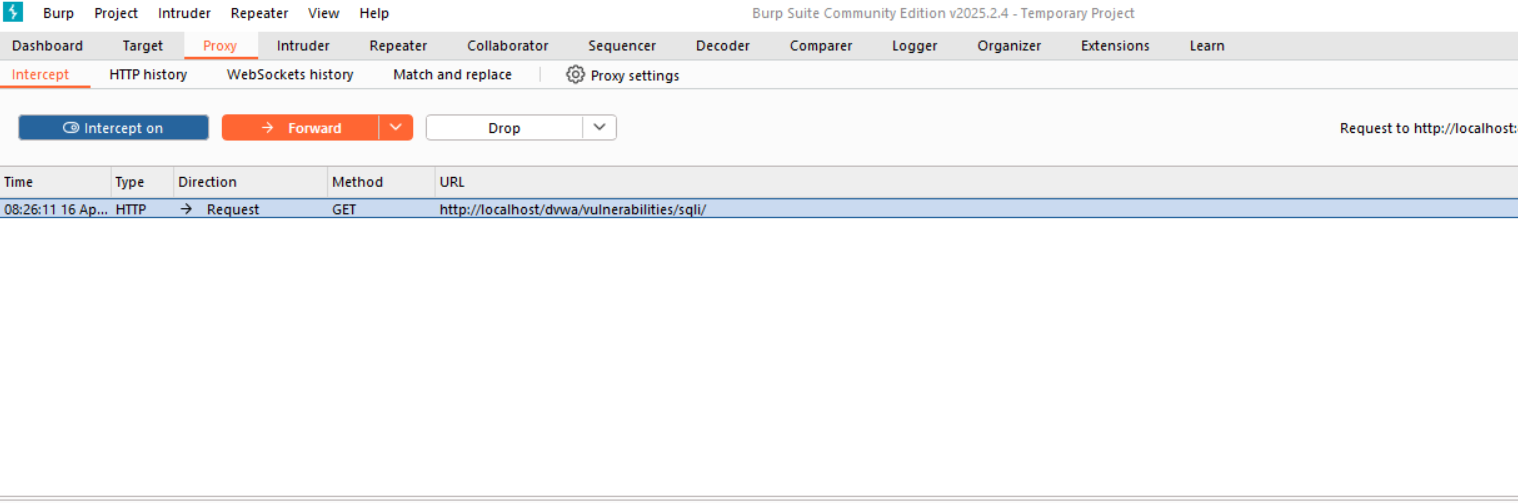
- Bước 3: Mở địa chỉ trang Web cần kiểm thử. Ví dụ dưới đây là giao diện kiểm tra lỗ hổng SQL Injecion trên website DVWA.

http://localhost/dvwa/vulnerabilities/sqli/

- Bước 4: Điền các giá trị bất kỳ nào đó và gửi yêu cầu từ trình duyệt

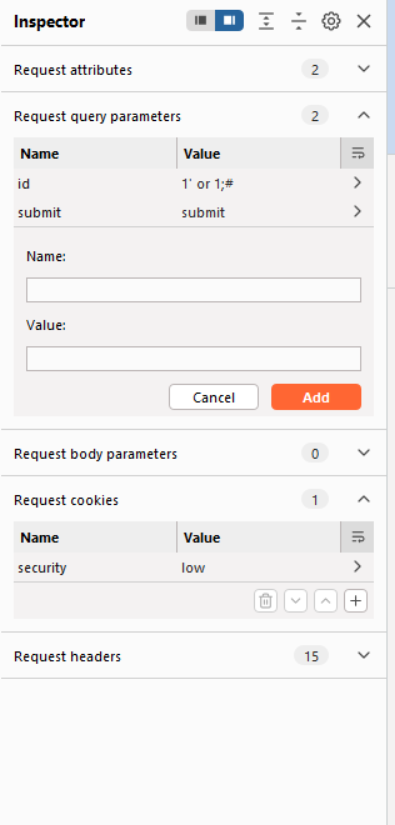
- Bước 5: Trên thẻ Proxy → Intercept, nhấn nút Forward để chuyển tiếp yêu cầu

17



- 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

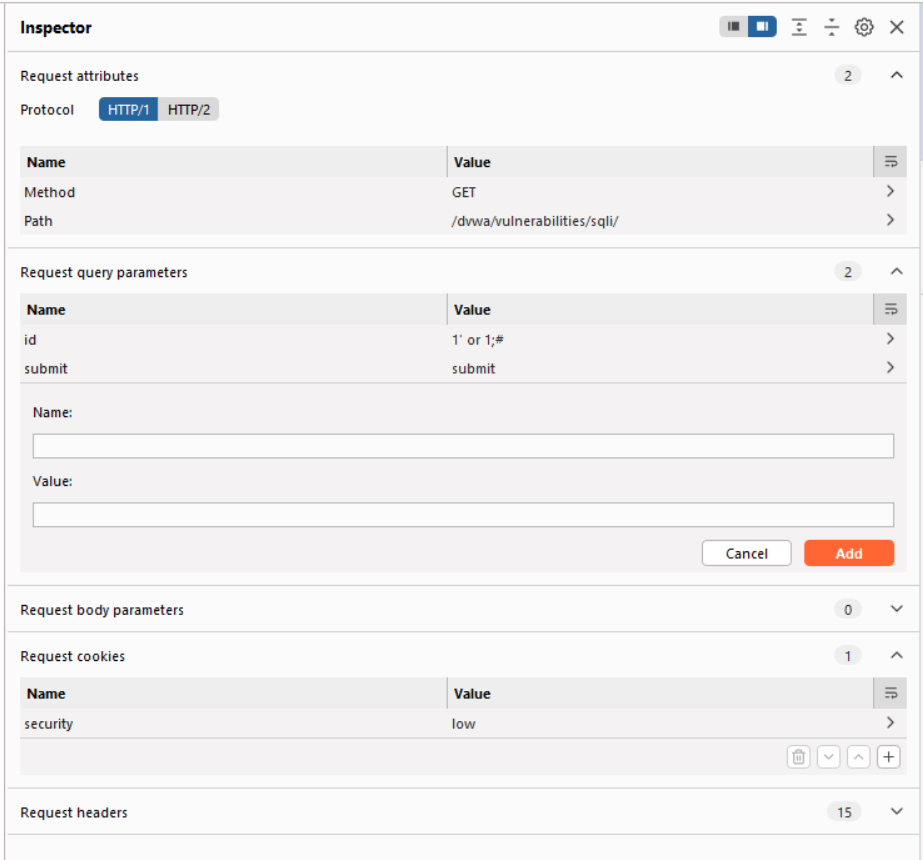
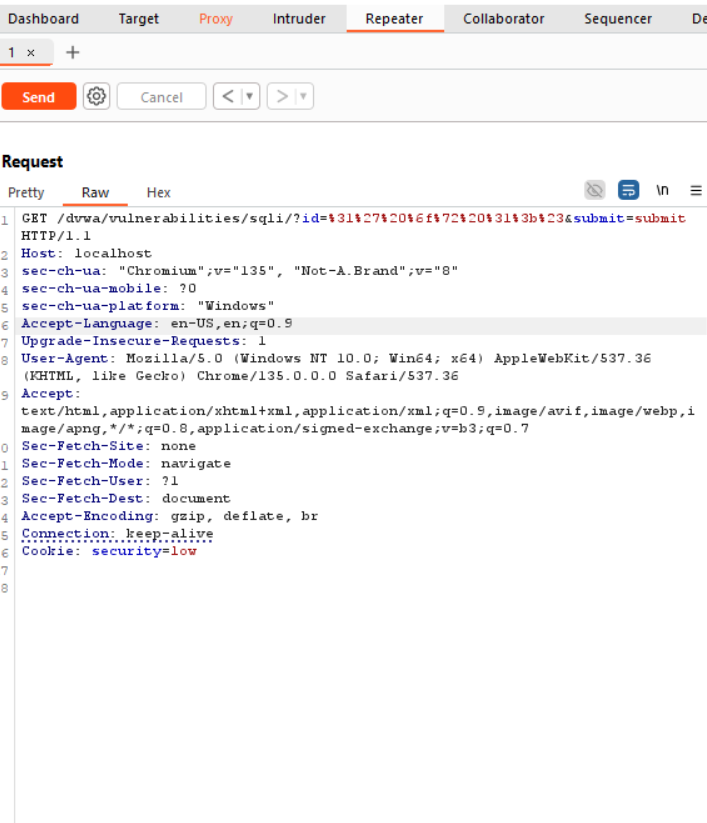
- Bước 7: Chọn thẻ Repeater. Thẻ này cho phép chúng ta thay đổi nội dung của HTTP Request và phát lại tới máy chủ. Thẻ con Params liệt kê danh sách các giá trị trên HTTP Header có thể là tham số đầu vào.



- 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)*

18



**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

- 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

19

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.



- 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



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.

20

- 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:



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 giang, 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 đề 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**

21

- Bước 1: Truy cập địa chỉ http://<Địa chỉ máy ảo>/sqli/sqli1.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.



- 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.



Ngoài ra với kết quả như trên, mặc dù chưa chắc chắn nhưng dựa trên ngữ nghĩa, ta cũng xác địn 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.

22

- 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 đề 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.

23