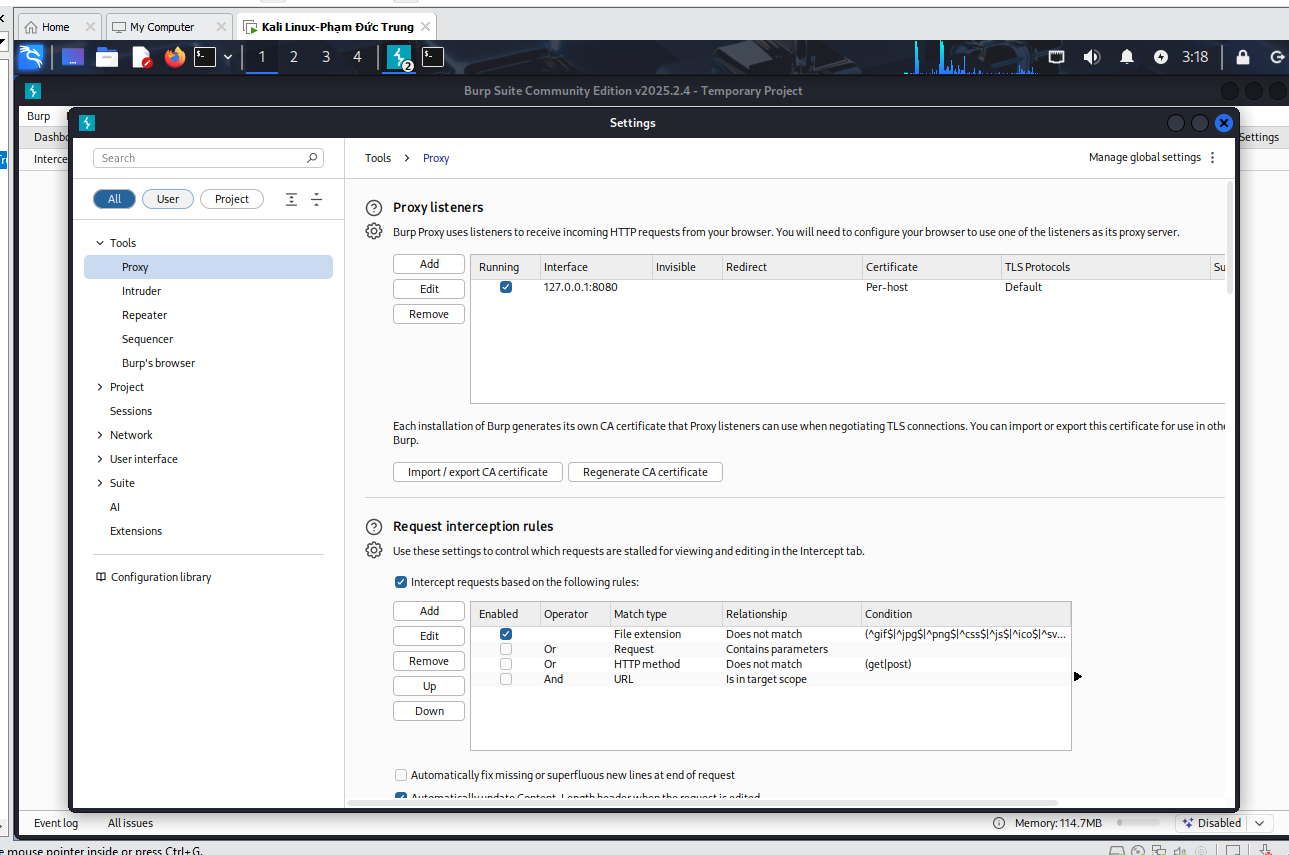
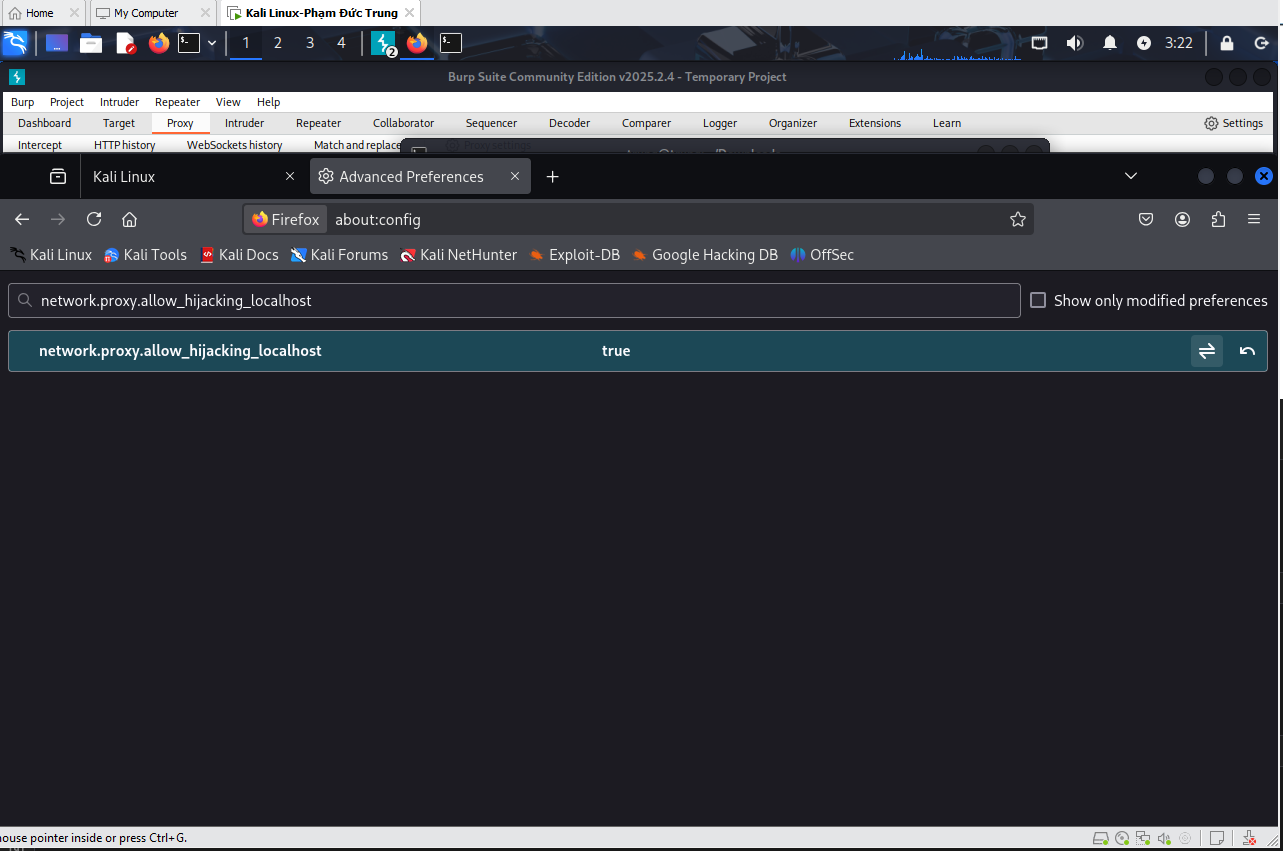
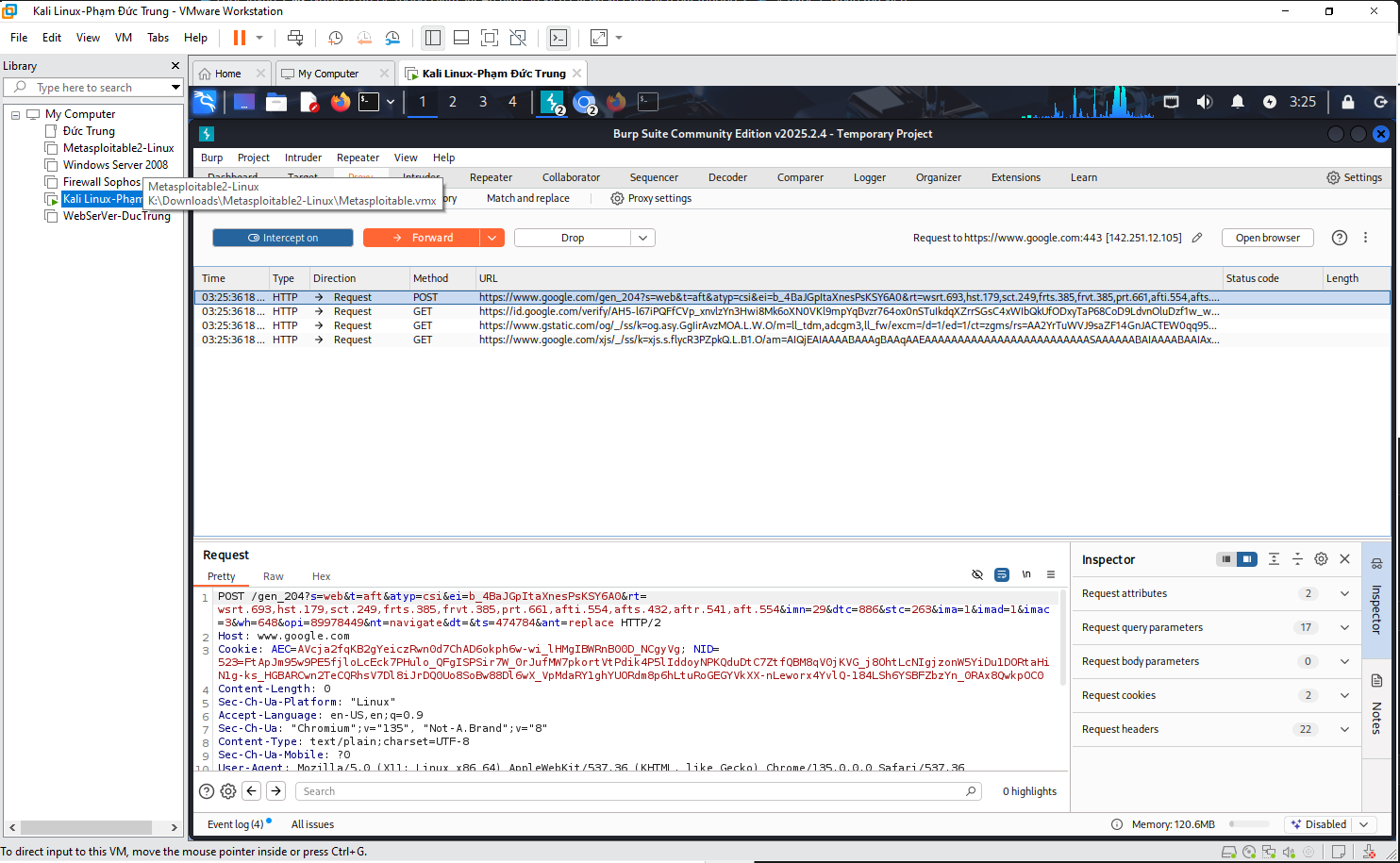
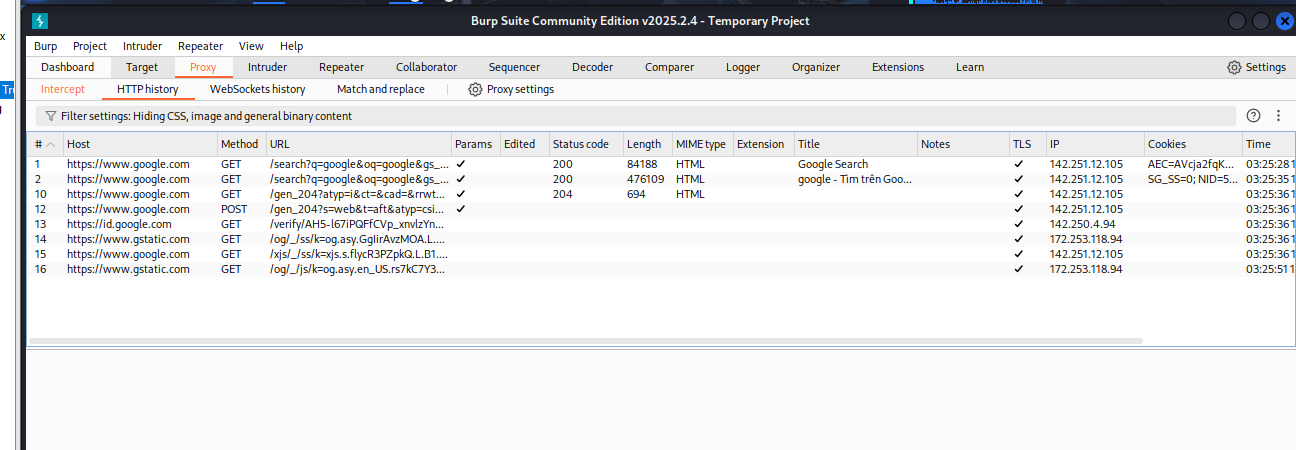


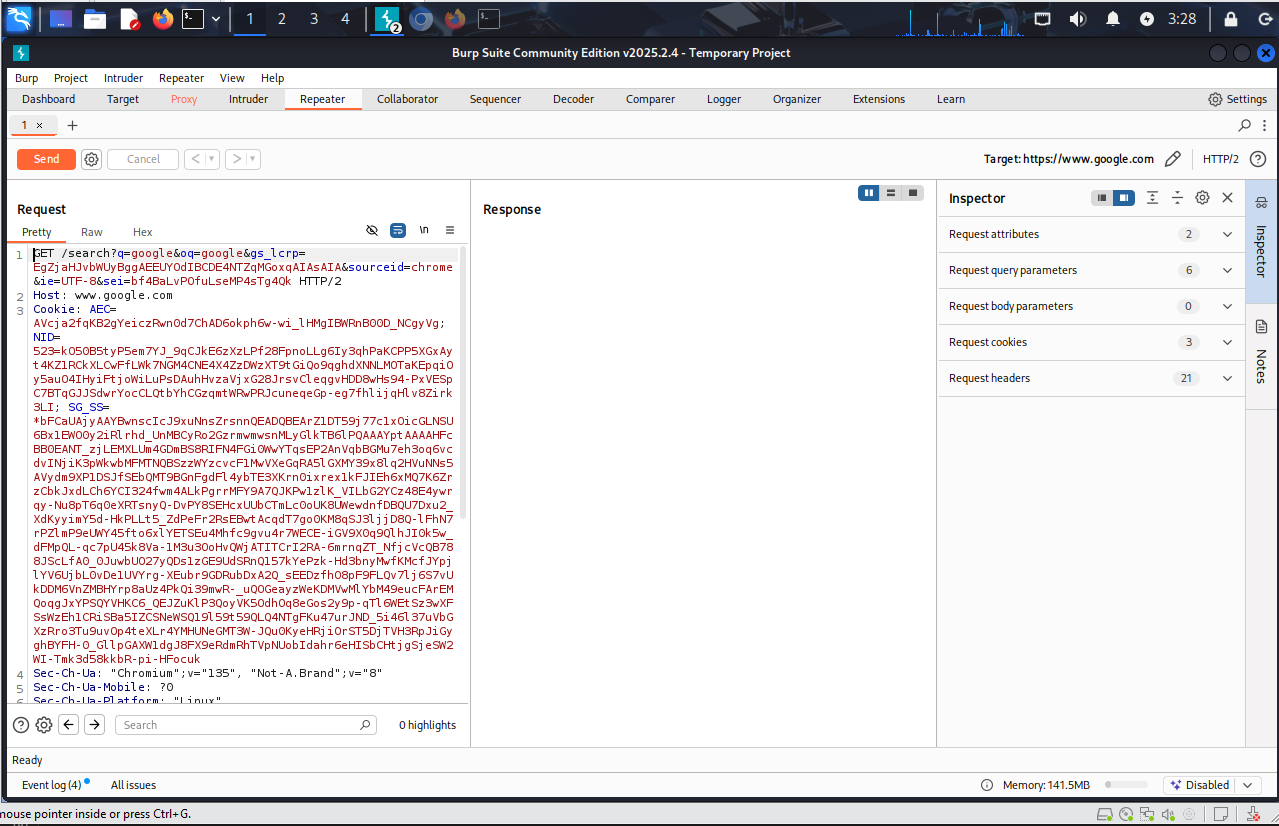
Địa chỉ web Server là 92.168.84.130

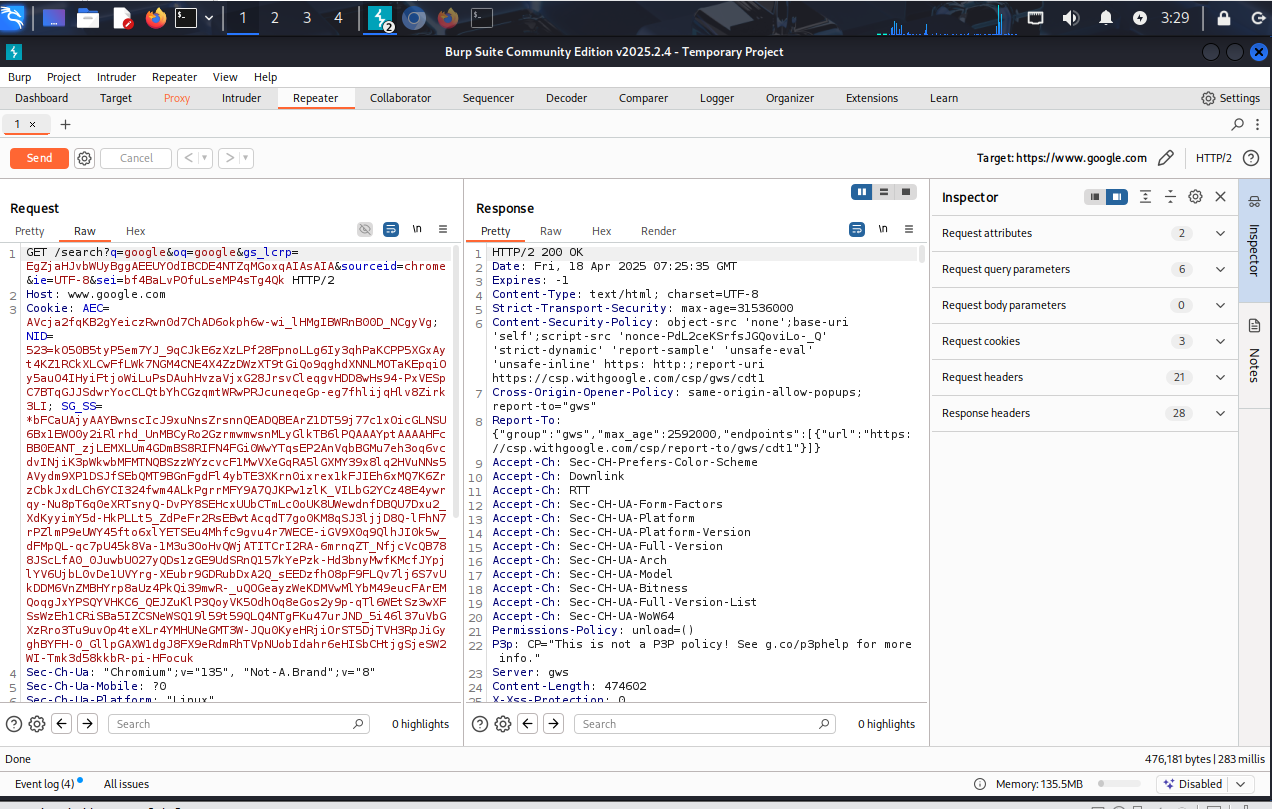


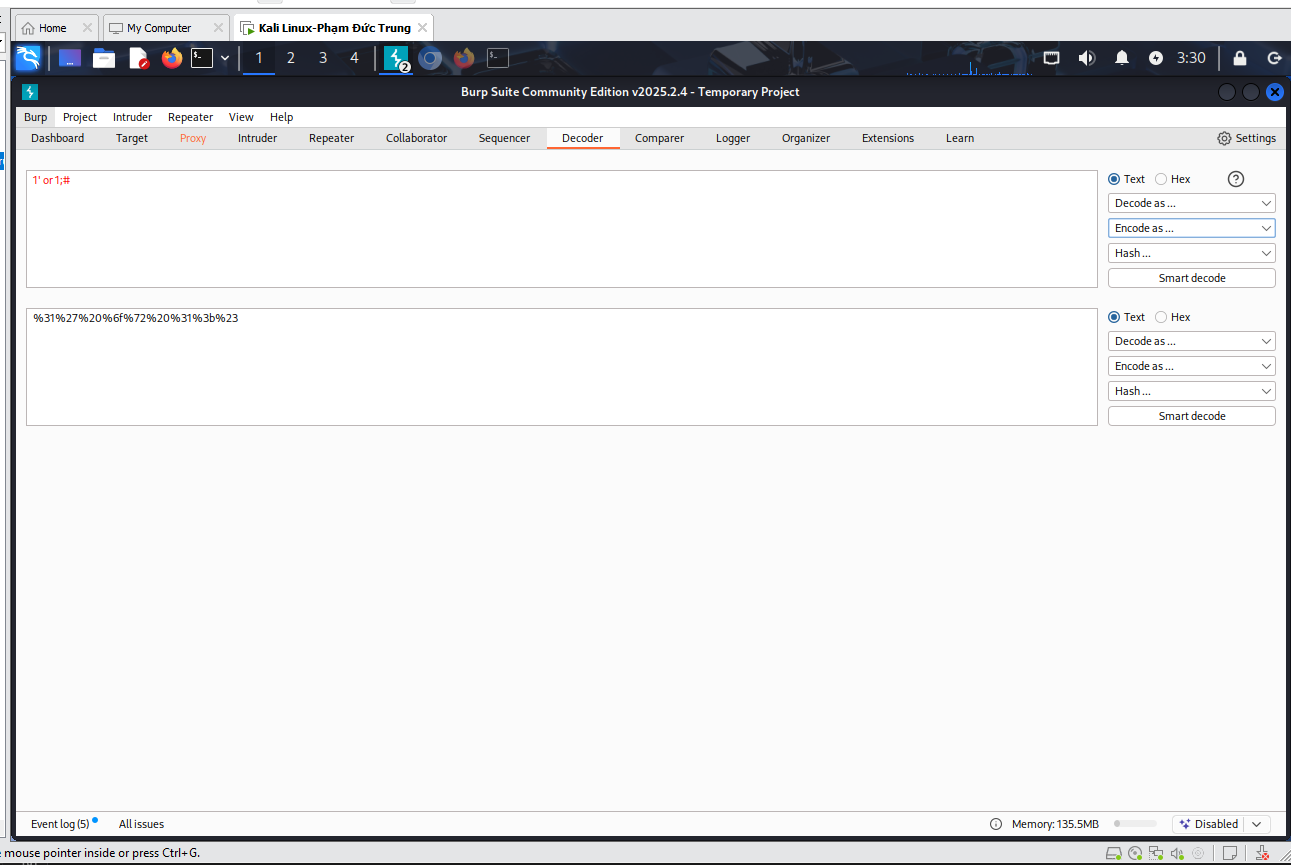












Tới đây em chịu chả thấy Param mà thay thế

**1. Xác định các tham số đầu vào và cách thức chúng được gửi tới server (0.25 điểm)**

**Tham số đầu vào**:

* **Biểu mẫu (Forms)**: Các trường nhập như username, password trong trang đăng nhập, hoặc các trường tìm kiếm (search, keyword) trên trang web. Những trường này thường được gửi qua phương thức **POST** hoặc **GET**.
* **Tham số URL**: Các tham số trong query string, ví dụ: http://example.com/page.php?id=1 hoặc http://example.com/list.php?cat=2. Những tham số này thường được gửi qua **GET**.
* **Cookies hoặc Headers**: Một số ứng dụng sử dụng dữ liệu từ cookies (session\_id) hoặc HTTP headers (User-Agent) làm đầu vào cho truy vấn SQL.

**Cách thức gửi tới server**:

* **GET**: Tham số được truyền qua URL (query string), dễ dàng kiểm thử bằng cách sửa đổi trực tiếp URL. Ví dụ: http://example.com/login.php?username=admin&password=pass.
* **POST**: Tham số được gửi trong phần thân (body) của yêu cầu HTTP, thường qua các biểu mẫu HTML. Cần sử dụng công cụ như Burp Suite hoặc Postman để sửa đổi dữ liệu.
* **Cookies/Headers**: Dữ liệu được gửi trong tiêu đề HTTP hoặc cookie, cần kiểm tra bằng cách sửa đổi các giá trị này.

**Mục đích**: Xác định tất cả các điểm đầu vào mà người dùng có thể kiểm soát để kiểm tra xem chúng có được sử dụng trực tiếp trong truy vấn SQL hay không.

**2. Phân tích và phán đoán kiểu truy vấn (0.25 điểm)**

**Các kiểu truy vấn SQL phổ biến**:

* **SELECT**: Dùng để truy xuất dữ liệu, ví dụ: SELECT \* FROM users WHERE username = 'input' AND password = 'input'. Đây là kiểu truy vấn thường gặp trong các trang đăng nhập hoặc tìm kiếm.
* **INSERT**: Dùng để thêm dữ liệu, ví dụ: INSERT INTO users (username, password) VALUES ('input', 'input'). Thường thấy trong biểu mẫu đăng ký.
* **UPDATE**: Dùng để cập nhật dữ liệu, ví dụ: UPDATE users SET password = 'input' WHERE user\_id = 'input'.
* **DELETE**: Dùng để xóa dữ liệu, hiếm gặp nhưng có thể xuất hiện trong các chức năng quản trị.

**Phán đoán**:

* Nếu trang có biểu mẫu đăng nhập, truy vấn thường là SELECT để kiểm tra thông tin người dùng.
* Nếu trang hiển thị danh sách hoặc nội dung (như index.php?id=1), truy vấn có thể là SELECT với điều kiện WHERE dựa trên tham số id.
* Nếu có biểu mẫu tìm kiếm, truy vấn có thể là SELECT với điều kiện LIKE hoặc WHERE.

**Mục đích**: Phán đoán kiểu truy vấn giúp xác định loại payload SQL Injection phù hợp (ví dụ: sử dụng UNION cho truy vấn SELECT hoặc payload lỗi cho truy vấn bất kỳ).

**3. Phân tích và phán đoán vị trí của các giá trị tham số đầu vào trong câu truy vấn (0.5 điểm)**

**Phân tích**:

* **Tham số trong điều kiện WHERE**: Thường gặp trong truy vấn SELECT. Ví dụ: SELECT \* FROM articles WHERE id = 'input'. Tham số input nằm trong mệnh đề WHERE.
* **Tham số trong giá trị cột**: Thường gặp trong INSERT hoặc UPDATE. Ví dụ: INSERT INTO users (username) VALUES ('input').
* **Tham số trong tên bảng hoặc cột (hiếm)**: Nếu ứng dụng động tạo truy vấn, tham số có thể được sử dụng để chỉ định bảng hoặc cột, ví dụ: SELECT \* FROM input.

**Phán đoán vị trí**:

* **GET parameters**: Thường xuất hiện trong WHERE, ví dụ: id=1 trong SELECT \* FROM pages WHERE id = 1.
* **POST parameters**: Thường xuất hiện trong WHERE (đăng nhập) hoặc VALUES (đăng ký). Ví dụ: username=admin trong SELECT \* FROM users WHERE username = 'admin'.
* **Cookies/Headers**: Có thể được sử dụng trong WHERE để kiểm tra phiên hoặc xác thực.

**Cách kiểm tra**:

* Thử chèn ký tự đặc biệt như ' hoặc " vào tham số. Nếu server trả về lỗi SQL (ví dụ: You have an error in your SQL syntax), tham số có khả năng được sử dụng trực tiếp trong truy vấn mà không được làm sạch.
* Thử payload như 1 OR 1=1 để xem phản hồi có thay đổi (ví dụ: trả về tất cả dữ liệu).

**Mục đích**: Xác định vị trí tham số giúp xây dựng payload chính xác để khai thác lỗ hổng.

**4. Xác định mức độ ảnh hưởng của lỗ hổng trên mỗi giá trị đầu vào (2.5 điểm)**

**Các giá trị kiểm thử và mục đích**:

1. **Ký tự đặc biệt (', ", ;)**:
   * **Mục đích**: Kiểm tra xem tham số có được chèn trực tiếp vào truy vấn SQL mà không được làm sạch hay không.
   * **Kết quả và suy luận**:
     + Nếu server trả về lỗi SQL (ví dụ: Syntax error near '''), tham số được chèn trực tiếp và có khả năng dễ bị khai thác.
     + Nếu không có lỗi và ứng dụng hoạt động bình thường, tham số có thể được làm sạch hoặc không liên quan đến truy vấn SQL.
     + **Mức độ ảnh hưởng**: Cao nếu lỗi SQL xuất hiện, vì kẻ tấn công có thể thao túng cú pháp truy vấn.
2. **Payload logic (1 OR 1=1, ' OR '1'='1)**:
   * **Mục đích**: Kiểm tra xem có thể bỏ qua xác thực (ví dụ: đăng nhập mà không cần mật khẩu đúng) hoặc truy xuất toàn bộ dữ liệu.
   * **Kết quả và suy luận**:
     + Nếu đăng nhập thành công hoặc trả về tất cả bản ghi, tham số nằm trong mệnh đề WHERE và không được làm sạch.
     + Ví dụ: Truy vấn SELECT \* FROM users WHERE username = '' OR '1'='1' sẽ trả về tất cả người dùng.
     + **Mức độ ảnh hưởng**: Rất cao, vì kẻ tấn công có thể truy cập trái phép hoặc lấy toàn bộ dữ liệu.
3. **Payload UNION (UNION SELECT 1,2,3)**:
   * **Mục đích**: Kiểm tra khả năng nối kết quả từ các bảng khác hoặc truy xuất dữ liệu nhạy cảm.
   * **Kết quả và suy luận**:
     + Nếu server trả về dữ liệu bổ sung (ví dụ: 1,2,3 hoặc dữ liệu từ bảng khác), lỗ hổng cho phép truy xuất dữ liệu ngoài ý định.
     + Cần xác định số cột trong truy vấn gốc bằng cách thử ORDER BY 1, ORDER BY 2, v.v., cho đến khi lỗi xuất hiện.
     + **Mức độ ảnh hưởng**: Cực kỳ cao, vì kẻ tấn công có thể lấy toàn bộ cơ sở dữ liệu (bảng, cột, dữ liệu).
4. **Payload lỗi (Error-based) (' AND EXTRACTVALUE(1,CONCAT(0x7e,(SELECT VERSION())))--)**:
   * **Mục đích**: Kích hoạt lỗi SQL để lấy thông tin cấu trúc cơ sở dữ liệu (phiên bản, tên bảng, v.v.).
   * **Kết quả và suy luận**:
     + Nếu lỗi trả về thông tin như phiên bản MySQL hoặc tên bảng, lỗ hổng có thể bị khai thác để thu thập cấu trúc cơ sở dữ liệu.
     + **Mức độ ảnh hưởng**: Cao, vì thông tin này giúp kẻ tấn công xây dựng các cuộc tấn công tiếp theo.
5. **Payload mù (Blind SQL Injection) (1 AND SLEEP(5))**:
   * **Mục đích**: Kiểm tra lỗ hổng khi server không trả về lỗi hoặc dữ liệu trực tiếp.
   * **Kết quả và suy luận**:
     + Nếu phản hồi bị trì hoãn 5 giây, tham số có thể bị khai thác bằng kỹ thuật mù (boolean-based hoặc time-based).
     + **Mức độ ảnh hưởng**: Trung bình đến cao, vì khai thác mù tốn thời gian hơn nhưng vẫn có thể lấy dữ liệu.

**Mức độ ảnh hưởng tổng thể**:

* **Đăng nhập trái phép**: Bỏ qua xác thực, ảnh hưởng nghiêm trọng đến bảo mật người dùng.
* **Truy xuất dữ liệu**: Lấy thông tin nhạy cảm (tên người dùng, mật khẩu, email), dẫn đến vi phạm dữ liệu.
* **Thay đổi/xóa dữ liệu**: Sử dụng INSERT, UPDATE, hoặc DELETE để sửa đổi hoặc xóa dữ liệu, gây mất dữ liệu hoặc gián đoạn dịch vụ.
* **Thực thi lệnh hệ thống**: Trong một số trường hợp (ví dụ: với MySQL và quyền cao), kẻ tấn công có thể thực thi lệnh hệ thống, dẫn đến chiếm quyền server.

**Câu 2: Kiểm thử lỗ hổng SQL Injection trên** [**http://webvul.bkcs.vn/login.php**](http://webvul.bkcs.vn/login.php)

**1. Xác định các tham số đầu vào và cách thức chúng được gửi tới server (0.25 điểm)**

**Giả định**: Trang login.php chứa một biểu mẫu đăng nhập với các trường username và password.

**Tham số đầu vào**:

* username: Tên người dùng nhập vào trường đăng nhập.
* password: Mật khẩu nhập vào trường đăng nhập.
* Có thể có thêm các tham số ẩn như csrf\_token hoặc redirect.

**Cách thức gửi**:

* **POST**: Thông thường, biểu mẫu đăng nhập gửi dữ liệu qua phương thức POST. Ví dụ:

text

Sao chép

POST /login.php HTTP/1.1

Host: webvul.bkcs.vn

Content-Type: application/x-www-form-urlencoded

username=admin&password=pass

* **GET** (ít gặp): Nếu tham số được gửi qua URL, ví dụ: http://webvul.bkcs.vn/login.php?username=admin&password=pass.

**Mục đích**: Xác định các tham số có thể được sửa đổi để kiểm tra lỗ hổng SQL Injection.

**2. Phân tích và phán đoán kiểu truy vấn, vị trí của các tham số đầu vào (0.25 điểm)**

**Phán đoán kiểu truy vấn**:

* Trang đăng nhập thường sử dụng truy vấn SELECT để kiểm tra thông tin đăng nhập. Ví dụ:

sql

Sao chép

SELECT \* FROM users WHERE username = 'input\_username' AND password = 'input\_password'

* Có thể là SELECT với hàm băm mật khẩu, ví dụ:

sql

Sao chép

SELECT \* FROM users WHERE username = 'input\_username' AND password = MD5('input\_password')

**Phán đoán vị trí tham số**:

* username và password được sử dụng trong mệnh đề WHERE của truy vấn SELECT.
* Nếu ứng dụng không làm sạch đầu vào, các tham số này được chèn trực tiếp vào truy vấn. Ví dụ:

sql

Sao chép

SELECT \* FROM users WHERE username = 'admin' AND password = 'pass'

**Mục đích**: Hiểu kiểu truy vấn và vị trí tham số để xây dựng payload kiểm thử phù hợp.

**3. Xác định mức độ ảnh hưởng của lỗ hổng trên mỗi giá trị đầu vào (3 điểm)**

**Các giá trị kiểm thử và mục đích**:

1. **Ký tự đặc biệt (') trên username**:
   * **Payload**: username='&password=anything
   * **Mục đích**: Kiểm tra xem tham số username có gây ra lỗi cú pháp SQL hay không.
   * **Kết quả và suy luận**:
     + Nếu server trả về lỗi như You have an error in your SQL syntax, tham số username được chèn trực tiếp vào truy vấn mà không được làm sạch.
     + Truy vấn có thể trở thành: SELECT \* FROM users WHERE username = ''' AND password = 'anything', gây lỗi cú pháp.
     + **Mức độ ảnh hưởng**: Cao, vì kẻ tấn công có thể thao túng truy vấn.
2. **Payload logic (' OR '1'='1) trên username**:
   * **Payload**: username=' OR '1'='1&password=anything
   * **Mục đích**: Bỏ qua xác thực đăng nhập để kiểm tra xem có thể đăng nhập mà không cần mật khẩu đúng.
   * **Kết quả và suy luận**:
     + Nếu đăng nhập thành công, truy vấn có thể là:

sql

Sao chép

SELECT \* FROM users WHERE username = '' OR '1'='1' AND password = 'anything'

Điều kiện OR '1'='1' luôn đúng, trả về tất cả người dùng.

* + - **Mức độ ảnh hưởng**: Cực kỳ cao, vì kẻ tấn công có thể truy cập tài khoản bất kỳ, bao gồm tài khoản quản trị.

1. **Payload UNION trên username**:
   * **Payload**: username=' UNION SELECT 1,2,3--&password=anything
   * **Mục đích**: Kiểm tra khả năng nối kết quả từ các bảng khác hoặc truy xuất dữ liệu nhạy cảm.
   * **Kết quả và suy luận**:
     + Nếu server trả về dữ liệu bổ sung (ví dụ: 1,2,3), lỗ hổng cho phép truy xuất dữ liệu ngoài ý định.
     + Cần thử ORDER BY 1, ORDER BY 2, v.v., để xác định số cột trong truy vấn gốc.
     + **Mức độ ảnh hưởng**: Cực kỳ cao, vì kẻ tấn công có thể lấy toàn bộ cơ sở dữ liệu.
2. **Payload lỗi trên username**:
   * **Payload**: username=' AND EXTRACTVALUE(1,CONCAT(0x7e,(SELECT VERSION())))--&password=anything
   * **Mục đích**: Thu thập thông tin cơ sở dữ liệu qua lỗi SQL.
   * **Kết quả và suy luận**:
     + Nếu lỗi trả về phiên bản MySQL (ví dụ: 5.7.32), kẻ tấn công có thông tin để xây dựng cuộc tấn công tiếp theo.
     + **Mức độ ảnh hưởng**: Cao, vì thông tin này hỗ trợ khai thác sâu hơn.
3. **Payload mù trên username**:
   * **Payload**: username=admin' AND SLEEP(5)--&password=anything
   * **Mục đích**: Kiểm tra lỗ hổng mù khi không có lỗi hoặc dữ liệu trả về.
   * **Kết quả và suy luận**:
     + Nếu phản hồi bị trì hoãn 5 giây, tham số có thể bị khai thác bằng kỹ thuật mù.
     + **Mức độ ảnh hưởng**: Trung bình đến cao, vì khai thác mù tốn thời gian nhưng vẫn nguy hiểm.

**Mức độ ảnh hưởng tổng thể**:

* **Đăng nhập trái phép**: Có thể truy cập tài khoản quản trị hoặc người dùng khác, dẫn đến chiếm quyền hệ thống.
* **Truy xuất dữ liệu**: Lấy thông tin nhạy cảm như tên người dùng, mật khẩu băm, hoặc email.
* **Thao túng dữ liệu**: Nếu có quyền INSERT hoặc UPDATE, kẻ tấn công có thể thêm người dùng giả hoặc sửa đổi dữ liệu.
* **Chiếm quyền server**: Nếu cơ sở dữ liệu cho phép thực thi lệnh hệ thống, kẻ tấn công có thể kiểm soát toàn bộ server.

**Câu 3: Kiểm thử lỗ hổng SQL Injection trên** [**http://webvul.bkcs.vn/index.php**](http://webvul.bkcs.vn/index.php)

**1. Xác định các tham số đầu vào và cách thức chúng được gửi tới server (0.25 điểm)**

**Giả định**: Trang index.php hiển thị nội dung động (ví dụ: bài viết, danh mục) dựa trên tham số URL.

**Tham số đầu vào**:

* id, cat, hoặc page: Tham số trong query string, ví dụ: http://webvul.bkcs.vn/index.php?id=1.
* Có thể có biểu mẫu tìm kiếm với trường search hoặc keyword, gửi qua **POST** hoặc **GET**.

**Cách thức gửi**:

* **GET**: Tham số trong URL, ví dụ: http://webvul.bkcs.vn/index.php?id=1.
* **POST**: Nếu có biểu mẫu tìm kiếm, dữ liệu được gửi trong body:

Sao chép

POST /index.php HTTP/1.1

Host: webvul.bkcs.vn

Content-Type: application/x-www-form-urlencoded

search=phone

**Mục đích**: Xác định các tham số có thể bị thao túng để kiểm tra lỗ hổng.

**2. Phân tích và phán đoán kiểu truy vấn, vị trí của các tham số đầu vào (0.25 điểm)**

**Phán đoán kiểu truy vấn**:

* Trang index.php thường sử dụng truy vấn SELECT để lấy nội dung. Ví dụ:

sql

Sao chép

SELECT \* FROM articles WHERE id = 'input\_id'

* Nếu có chức năng tìm kiếm, truy vấn có thể là:

sql

Sao chép

SELECT \* FROM products WHERE name LIKE '%input\_search%'

**Phán đoán vị trí tham số**:

* Tham số id hoặc cat thường nằm trong mệnh đề WHERE, ví dụ: WHERE id = '1'.
* Tham số search thường nằm trong LIKE hoặc WHERE, ví dụ: WHERE name LIKE '%phone%'.

**Mục đích**: Xác định vị trí tham số để xây dựng payload kiểm thử.

**3. Xác định mức độ ảnh hưởng của lỗ hổng trên mỗi giá trị đầu vào (2.5 điểm)**

**Các giá trị kiểm thử và mục đích**:

1. **Ký tự đặc biệt (') trên id**:
   * **Payload**: http://webvul.bkcs.vn/index.php?id='
   * **Mục đích**: Kiểm tra lỗi cú pháp SQL.
   * **Kết quả và suy luận**:
     + Nếu server trả về lỗi SQL, tham số id được chèn trực tiếp vào truy vấn.
     + **Mức độ ảnh hưởng**: Cao, vì kẻ tấn công có thể thao túng truy vấn.
2. **Payload logic (1 OR 1=1) trên id**:
   * **Payload**: http://webvul.bkcs.vn/index.php?id=1 OR 1=1
   * **Mục đích**: Kiểm tra khả năng truy xuất tất cả dữ liệu.
   * **Kết quả và suy luận**:
     + Nếu trang trả về tất cả bài viết, truy vấn có thể là:

sql

Sao chép

SELECT \* FROM articles WHERE id = 1 OR 1=1

* + - **Mức độ ảnh hưởng**: Rất cao, vì kẻ tấn công có thể lấy toàn bộ dữ liệu.

1. **Payload UNION trên id**:
   * **Payload**: http://webvul.bkcs.vn/index.php?id=-1 UNION SELECT 1,2,3--
   * **Mục đích**: Truy xuất dữ liệu từ bảng khác.
   * **Kết quả và suy luận**:
     + Nếu trả về 1,2,3 hoặc dữ liệu khác, lỗ hổng cho phép lấy thông tin nhạy cảm.
     + **Mức độ ảnh hưởng**: Cực kỳ cao, vì có thể lấy toàn bộ cơ sở dữ liệu.
2. **Payload lỗi trên id**:
   * **Payload**: http://webvul.bkcs.vn/index.php?id=1 AND EXTRACTVALUE(1,CONCAT(0x7e,(SELECT VERSION())))--
   * **Mục đích**: Lấy thông tin cơ sở dữ liệu qua lỗi.
   * **Kết quả và suy luận**:
     + Nếu trả về phiên bản MySQL, kẻ tấn công có thông tin để tấn công tiếp.
     + **Mức độ ảnh hưởng**: Cao.
3. **Payload mù trên id**:
   * **Payload**: http://webvul.bkcs.vn/index.php?id=1 AND SLEEP(5)
   * **Mục đích**: Kiểm tra lỗ hổng mù.
   * **Kết quả và suy luận**:
     + Nếu phản hồi bị trì hoãn, lỗ hổng tồn tại và có thể khai thác mù.
     + **Mức độ ảnh hưởng**: Trung bình đến cao.

**Mức độ ảnh hưởng tổng thể**:

* **Truy xuất dữ liệu**: Lấy nội dung nhạy cảm từ cơ sở dữ liệu.
* **Thao túng hiển thị**: Thay đổi nội dung trang web bằng cách sửa đổi kết quả truy vấn.
* **Chiếm quyền hệ thống**: Nếu có quyền cao, kẻ tấn công có thể thực thi lệnh hệ thống.