Shape, square

Description automatically generated

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🕯✡🕮🕮✡🕯**

|  |  |
| --- | --- |
|  | |
|  |  |

A picture containing text

Description automatically generatedLogo

Description automatically generated

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**BẢO MẬT WEB**

**XÂY DỰNG WEBSITE KINH DOANH SÁCH**

**METIS’S BOOK**

**GVHD: Ths. Lê Thị Minh Châu**

**Nhóm sinh viên thực hiện :**

Hà Vĩ Khang 20110657

Nguyễn Đức Khải 20110655

Nguyễn Đức Thịnh 20110256

Lê Nguyễn Tuấn Kiệt 20110286

**Tp. Hồ Chí Minh, tháng 5 năm 2023**

***Nhận xét của giảng viên***

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

*Tháng 5 năm 2023*

**Giáo viên chấm điểm**

# LỜI CẢM ƠN

Lời nói đầu tiên, nhóm thực hiện xin được gửi đến cô **Lê Thị Minh Châu** – giảng viên bộ môn lời cảm ơn chân thành và sâu sắc nhất.

Nhóm thực hiện xin cảm ơn sự quan tâm và giúp đỡ tận tình của cô trong suốt quá trình giảng dạy. Cảm ơn cô đã luôn giải đáp những thắc mắc cũng như đưa ra những nhận xét, góp ý giúp nhóm thực hiện cải thiện chất lượng công việc của nhóm.

Vì khả năng còn hạn chế nên trong quá trình thực hiện báo cáo không tránh khỏi sai sót, kính mong nhận được những ý kiến đóng góp từ thầy để nhóm có thể cải thiện hơn sau này.

Nhóm thực hiện xin chân thành cảm ơn.

# MỤC LỤC

[LỜI CẢM ƠN 3](#_Toc136254861)

[LỖ HỔNG QUÉT ĐƯỢC 7](#_Toc136254862)

[PHÂN CHIA CÔNG VIỆC 8](#_Toc136254863)

[Phần 1: Giới thiệu đề tài 1](#_Toc136254864)

[1. Giới thiệu 1](#_Toc136254865)

[*1.1. Đối tượng của đề tài* 1](#_Toc136254866)

[1.*2.Chức năng* 1](#_Toc136254867)

[2. Lược đồ usecase 4](#_Toc136254868)

[3. Lược đồ quan hệ ERD 5](#_Toc136254869)

[4. Hướng dẫn cài đặt và chạy 5](#_Toc136254870)

[4.1. Sơ lược về project 5](#_Toc136254871)

[4.2. Hướng dẫn cài đặt 6](#_Toc136254872)

[4.3. Hướng dẫn chạy 7](#_Toc136254873)

[Phần 2: Kiểm tra thủ công 8](#_Toc136254874)

[1. Hidden Field Manipulation 8](#_Toc136254875)

[1.1. Khái niệm 8](#_Toc136254876)

[1.2. Tấn công project 8](#_Toc136254877)

[1.3. Nguyên nhân 10](#_Toc136254878)

[1.4. Khắc phục 11](#_Toc136254879)

[2. Insecure Direct Object Reference ( IDOR ) 12](#_Toc136254880)

[2.1. Khái niệm 12](#_Toc136254881)

[2.2. Tấn công project 12](#_Toc136254882)

[2.3. Nguyên nhân 14](#_Toc136254883)

[2.4. Khắc phục 14](#_Toc136254884)

[3. SMTP credentials exposed 15](#_Toc136254885)

[3.1. Khái niệm 15](#_Toc136254886)

[3.2. Lỗ hổng trong project 16](#_Toc136254887)

[3.3. Nguyên nhân 18](#_Toc136254888)

[3.4. Khắc phục 19](#_Toc136254889)

[4. Brute Force Attack ( Admin Password ) 21](#_Toc136254890)

[4.1. Khái niệm 21](#_Toc136254891)

[4.2. Tấn công project 21](#_Toc136254892)

[4.3. Nguyên nhân 25](#_Toc136254893)

[4.4. Khắc phục 26](#_Toc136254894)

[Phần 3: Quét bằng ZAP 28](#_Toc136254895)

[1. Giới thiệu: 28](#_Toc136254896)

[2. Absence of Anti-CSRF Tokens 28](#_Toc136254897)

[2.1. Khái niệm 28](#_Toc136254898)

[2.2. Tấn công project 29](#_Toc136254899)

[2.3. Nguyên nhân 31](#_Toc136254900)

[2.4. Khắc phục 31](#_Toc136254901)

[3. Content Security Policy (CSP) Header Not Set 32](#_Toc136254902)

[3.1. Khái niệm 32](#_Toc136254903)

[3.2. Lỗ hổng trong project 33](#_Toc136254904)

[3.3. Khắc phục 36](#_Toc136254905)

[4. Vulnerable Library 36](#_Toc136254906)

[4.1. Khái niệm 36](#_Toc136254907)

[4.2. Lỗ hổng trong project 37](#_Toc136254908)

[4.3. Khắc phục 37](#_Toc136254909)

[Phần 4: Quét bằng Snyk 38](#_Toc136254910)

[1. Giới thiệu 38](#_Toc136254911)

[2. Improper Access Control 38](#_Toc136254912)

[2.1. Khái niệm 38](#_Toc136254913)

[2.2. Tấn công project 39](#_Toc136254914)

[2.3. Nguyên nhân 40](#_Toc136254915)

[2.4. Khắc phục 41](#_Toc136254916)

[Phần 5: Quét bằng Vega 44](#_Toc136254917)

[1. Giới thiệu 44](#_Toc136254918)

[2. Integer Overflow 44](#_Toc136254919)

[2.1. Khái niệm 44](#_Toc136254920)

[2.2. Tấn công project – False positive 45](#_Toc136254921)

[2.3. Nguyên nhân 45](#_Toc136254922)

[3. Session Cookie Without Secure Flag 46](#_Toc136254923)

[3.1. Khái niệm 46](#_Toc136254924)

[3.2. Lỗ hổng trong project 47](#_Toc136254925)

[3.3. Khắc phục 47](#_Toc136254926)

[Phần 6: Các lỗ hổng không tìm thấy 48](#_Toc136254927)

[1. SQL Injection 48](#_Toc136254928)

[1.1. Khái niệm 48](#_Toc136254929)

[1.2. Quét bằng SQLMap 48](#_Toc136254930)

[2.2. Sự bảo vệ của Spring Framework 50](#_Toc136254931)

[2. Cross-Site Scripting (XSS) 51](#_Toc136254932)

[2.1. Khái niệm 51](#_Toc136254933)

[2.2. Sự bảo vệ của Thymeleaf 52](#_Toc136254934)

[2.3. Cải thiện 53](#_Toc136254935)

# LỖ HỔNG QUÉT ĐƯỢC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thành viên** | **Tên** | **Công cụ** | **Trạng thái** |
| 1 | Khang | Hidden Field Manipulation | Thủ công | Phát hiện |
| 2 | Thịnh | Insecure Direct Object Reference | Thủ công | Phát hiện |
| 3 | Kiệt | SMTP credentials exposed | Thủ công | Phát hiện |
| 4 | Kiệt | Brute Force Attack | Burp Suite | Phát hiện |
| 5 | Khải | Absence of Anti-CSRF Tokens | ZAP | Phát hiện |
| 6 | Thịnh | Content Security Policy (CSP) Header Not Set | ZAP | Phát hiện |
| 7 | Khang | Vulnerable Library | ZAP | Phát hiện |
| 8 | Khải | Improper Access Control | Snyk | Phát hiện |
| 9 | Kiệt | Integer overflow | Vega | Dương tính giả |
| 10 | Khang | Session Cookie Without Secure Flag | Vega | Phát hiện |
| 11 | Khải | SQL injection | SQL Map | Không phát hiện |
| 12 | Cả nhóm | XSS | Tất cả các công cụ | Không phát hiện |

# PHÂN CHIA CÔNG VIỆC

|  |  |  |
| --- | --- | --- |
| Họ và tên | MSSV | Mức độ hoàn thành |
| Hà Vĩ Khang | 20110657 | 100% |
| Nguyễn Đức Khải | 20110655 | 100% |
| Lê Nguyễn Tuấn Kiệt | 20110286 | 100% |
| Nguyễn Đức Thịnh | 20110256 | 100% |

# Phần 1: Giới thiệu đề tài

## 1. Giới thiệu

### *1.1. Đối tượng của đề tài*

- Khách hàng

- Quản lý

- Nhân viên

### 1.*2.Chức năng*

#### 1.2.1. Khách hàng

* Đăng ký tài khoản: Người dùng tiến hành đăng ký tài khoản, thông tin cần cung cấp để đăng ký bao gồm:
  + - Thông tin bắt buộc: tên đăng nhập, mật khẩu, tên người dùng, gmail, số điện thoại.
    - Thông tin không bắt buộc: Địa chỉ, giới tính, ngày tháng năm sinh.
* Đăng nhập tài khoản: Người dùng nhập tên tài khoản và mật khẩu để đăng nhập vào tài khoản.
* Tìm kiếm sản phẩm: Người dùng có thể tìm kiếm các loại sách dựa vào tên sách, tên tác giả, thể loại bằng khung tìm kiếm và lọc sách theo các tiêu chí.
* Cập nhập giỏ hàng: Người dùng có thể thêm, xóa, sửa các sản phẩm trong giỏ hàng.
* Mua hàng: Người dùng có thể chọn các sản phẩm trong giỏ hàng và hoàn thành các thông tin trong đơn hàng để mua hàng.
* Lựa chọn phương thức thanh toán: Người dùng có thể thanh toán đơn hàng qua các phương thức thanh toán: Card(ATM, MoMo, Zalopay,…), thanh toán bằng tiền mặt.
* Lựa chọn phương thức vận chuyển: Người dùng có thể lựa chọn các phương thức vận chuyển như là vận chuyển hỏa tốc, vận chuyển nhanh, vận chuyển thường.
* Đánh giá sản phẩm: Người dùng sau khi trải nghiệm sản phẩm có thể để lại đánh giá đóng góp ý kiến về sản phẩm mình đã mua.
* chia sẻ sản phẩm: Người dùng có thể review giới thiệu sản phẩm thông qua việc chia sẻ đường dẫn liên kết của sản phẩm qua các nền tảng xã hội như là: Facebook, Zalo, Twitter.
* Gửi yêu cầu sản phẩm: Người dùng có thể gửi yêu cầu về loại sách chưa có thông qua tên sách và tên tác giả.
* Thay đổi thông tin tài khoản: Người dùng có thể thay đổi thông tin tài khoản cá nhân.

#### 1.2.2. Nhân viên

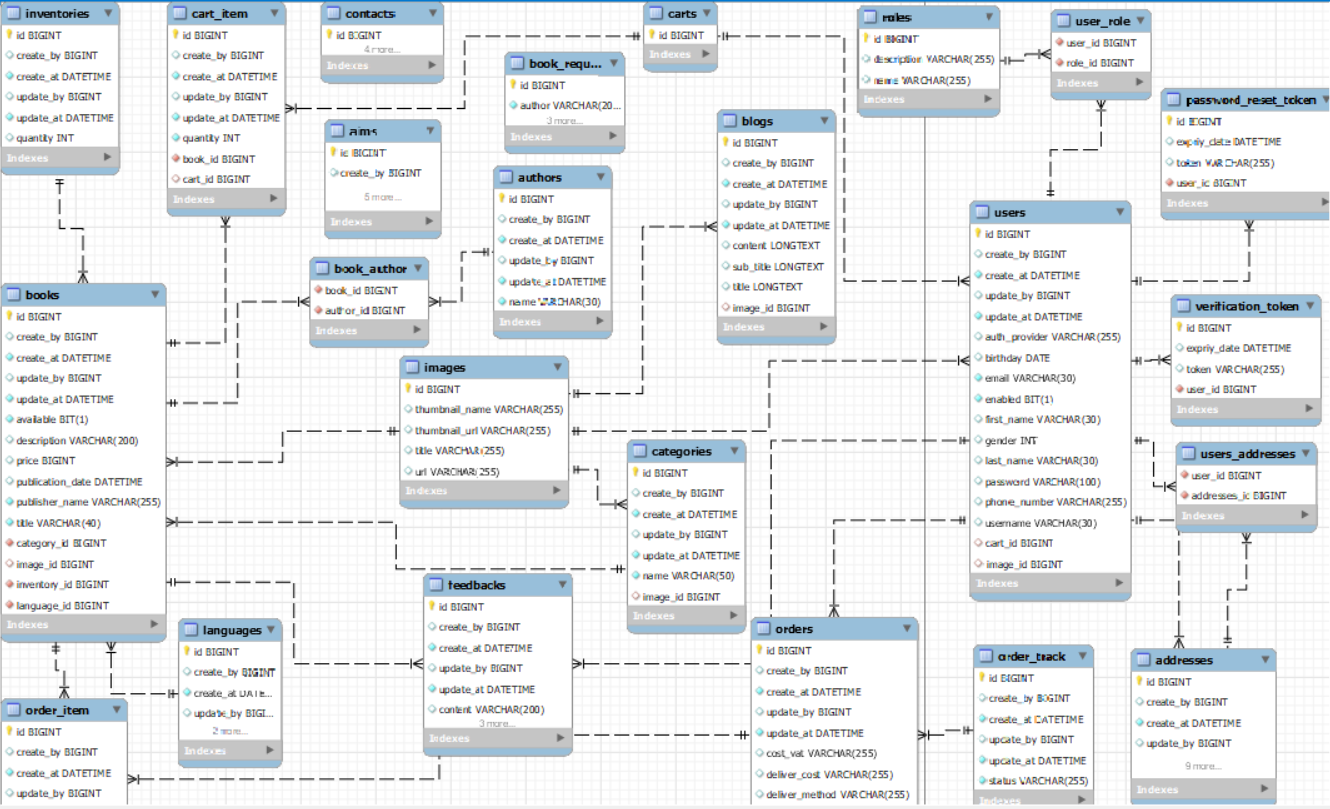
* Quản lý đơn hàng: Nhân viên có thể tiếp nhận đơn hàng, sửa đổi, xóa đơn hàng, chuẩn bị hàng để gửi.
* Quản lý sản phẩm: Nhân viên có thể quản lý các thông tin sản phẩm theo yêu cầu của người quản lý, thêm số lượng sản phẩm, sửa đổi thông tin, xóa sản phẩm.
* Quản lý bài đăng: Nhân viên có thể viết bài, sửa bài, xóa bài và đăng bài lên các mục quảng bá của trang web cũng như các trang thông tin của cửa hàng.
* Phản hồi đánh giá: Nhân viên có thể thêm, xóa, sửa và phản hồi đánh giá của khách hàng.

#### 1.2.3. Quản lý

* Quản lý tài khoản khách hàng: Quản lý có thể xem, thêm, sửa, xóa các thông tin liên quan đến tài khoản khách hàng.
* Quản lý sách: Quản lý có thể xem và chỉnh sửa thông tin sách, xem yêu cầu sách.
* Quản lý tài khoản nhân viên: Quản lý có thể xem, thêm, sửa, xóa các thông tin liên quan đến tài khoản nhân viên.
* Phân tích đánh giá lịch sử mua hàng: Quản lý có thể xem các thống kê về lịch sử mua hàng của một khách hàng cụ thể, lịch sử mua hàng của tổng thể, thu nhập, lịch sử hoạt động của nhân viên.

## Diagram Description automatically generated2. Lược đồ usecase

## 3. Lược đồ quan hệ ERD



## 4. Hướng dẫn cài đặt và chạy

### 4.1. Sơ lược về project

- Link project cũ: <https://github.com/KhaiNguyenDuc/Nhom4_NonFix_Security>

- Link project sau khi fix: <https://github.com/KhaiNguyenDuc/Nhom4_BookStore_Security>

- Ngôn ngữ lập trình: Java

- Framework: Spring boot

- Cơ sở dữ liệu: MySQL

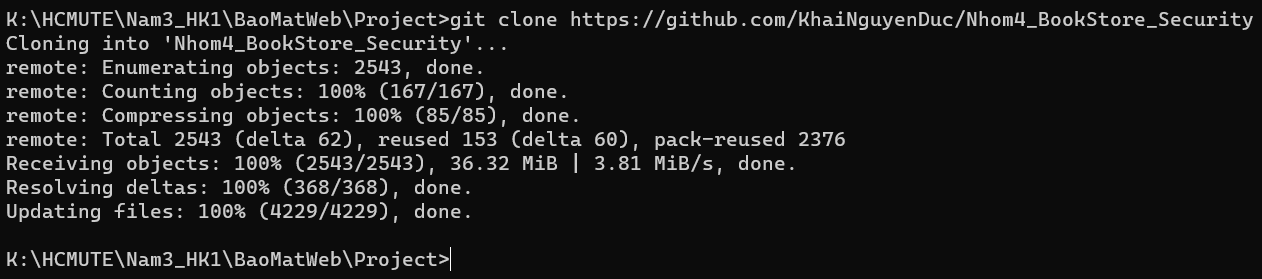
- Version control: GitHub

- IDE: Spring tool suite 4.0 for Eclipse, Eclipse, Intelliji

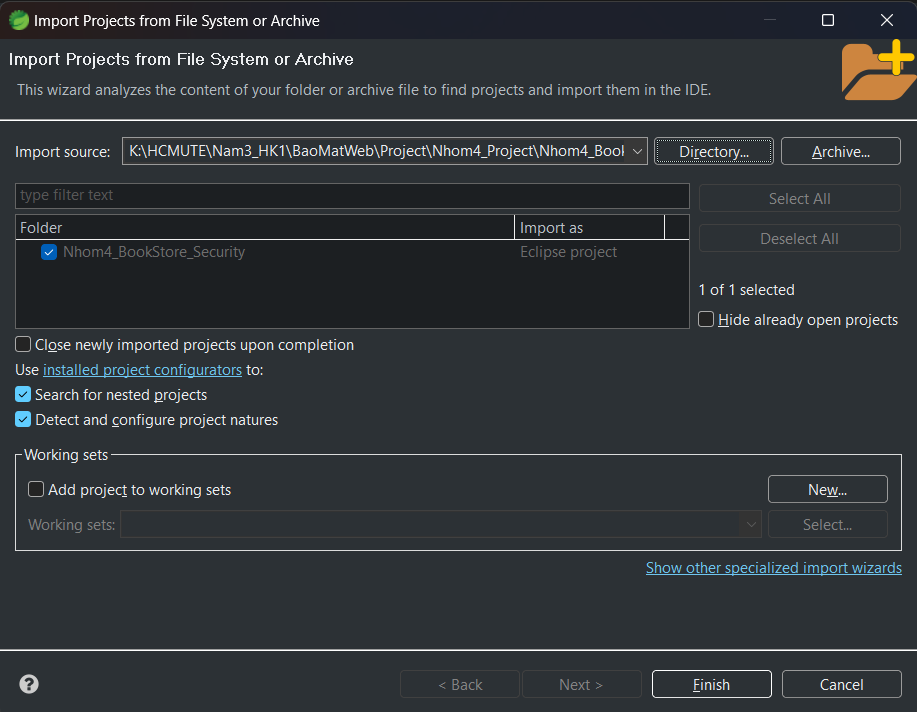
### 4.2. Hướng dẫn cài đặt

**Bước 1:** Tải project từ github về bằng câu lệnh:

git clone <https://github.com/KhaiNguyenDuc/Nhom4_BookStore_Security>

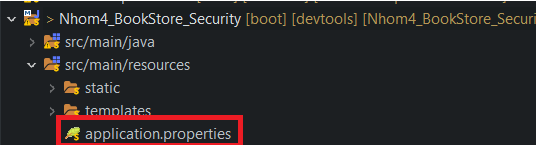


**Bước 2:** Import project vào IDE

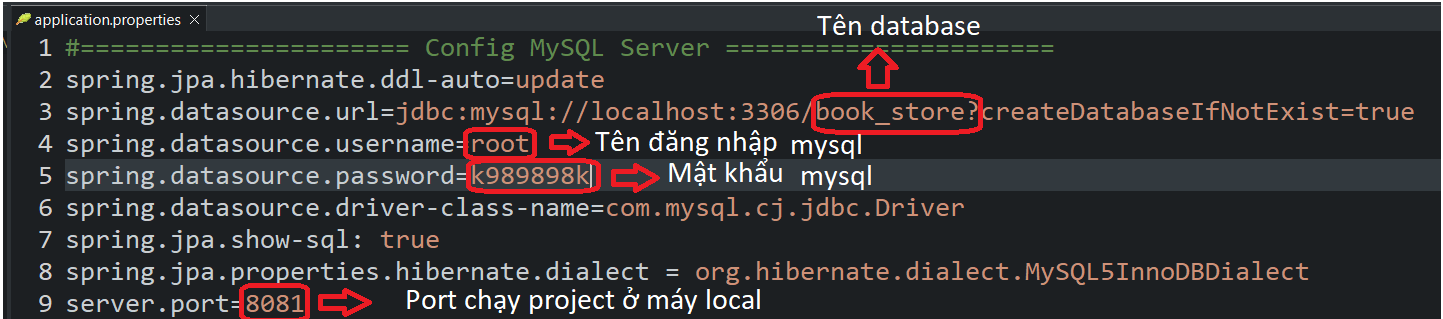
Mở 1 IDE bất kỳ hỗ trợ java ( Nên dùng STS cho eclipse ). Chọn file -> Open Project From file System -> Directory -> Chọn project -> Finish

### 4.3. Hướng dẫn chạy

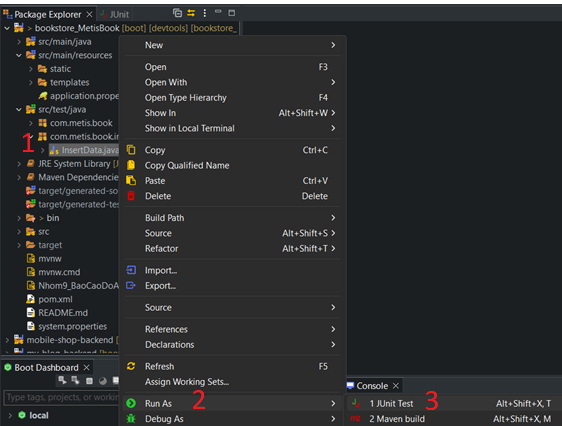
**Bước 1**: Sau khi tải project, mở file application.properties trong đường dẫn /src/main/resources/application.properties



**Bước 2**: Thay đổi cấu hình tùy chỉnh trong application.properties



**Bước 3:** Nháy chuột phải vào file Insert Data nằm trong thư mục src/test/java trong package com.metis.book.insertData. Sau đó chọn Run as và chọn Junit test

****

Sau khi chạy file InsertData, database và dữ liệu mẫu sẽ được tự động thêm vào database được cài đặt trong file application.properties

# Phần 2: Kiểm tra thủ công

## Hidden Field Manipulation

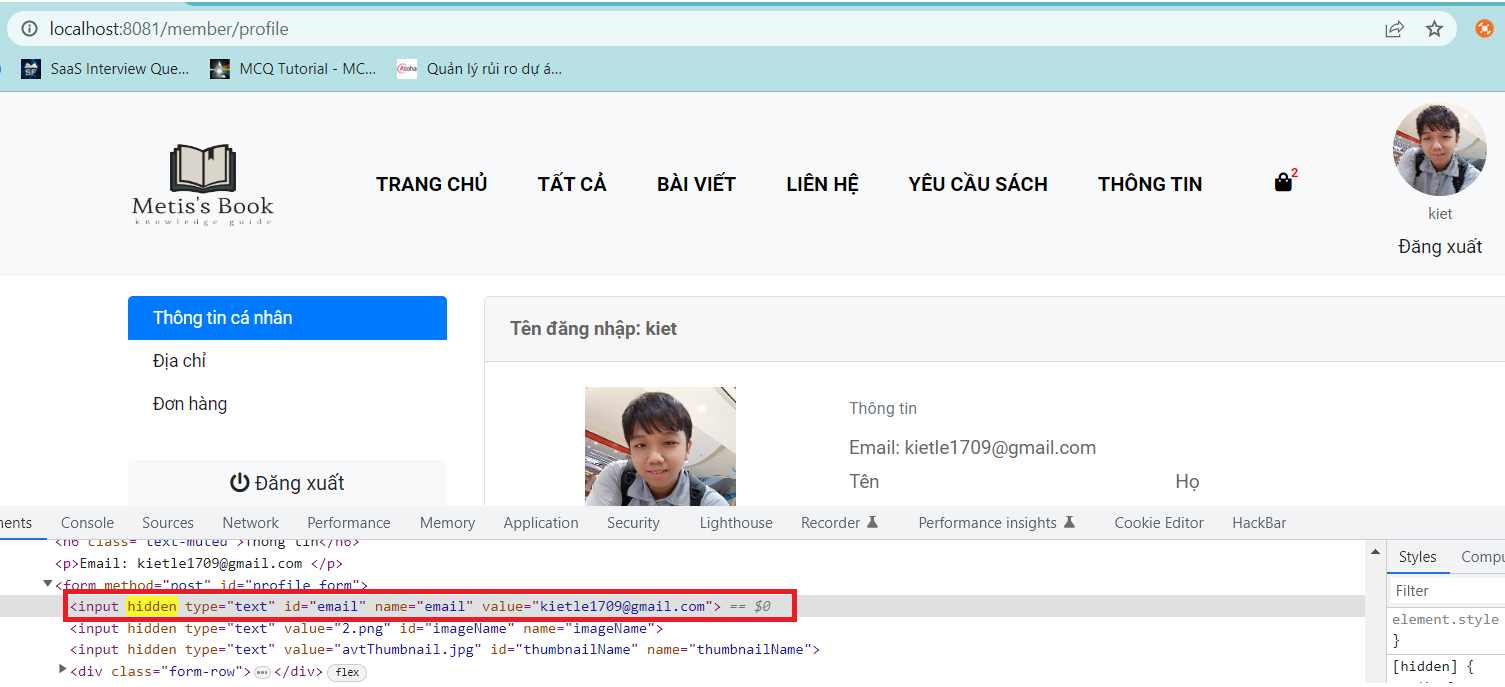
### Khái niệm

Là thể loại tấn công web trong đó lợi dụng các ô input được ẩn đi bởi các lập trình viên. Các ô input này thường chứa giá trị như: id, session id,... Hacker sẽ sử thay đổi giá trị các ô input này để vượt qua cơ chế bảo mật của website và gửi yêu cầu đến phía server. Nếu server kiểm tra dữ liệu lỏng lẻo, không có cơ chế xác thực người dùng. Hacker có thể thay đổi được các dữ liệu quan trọng

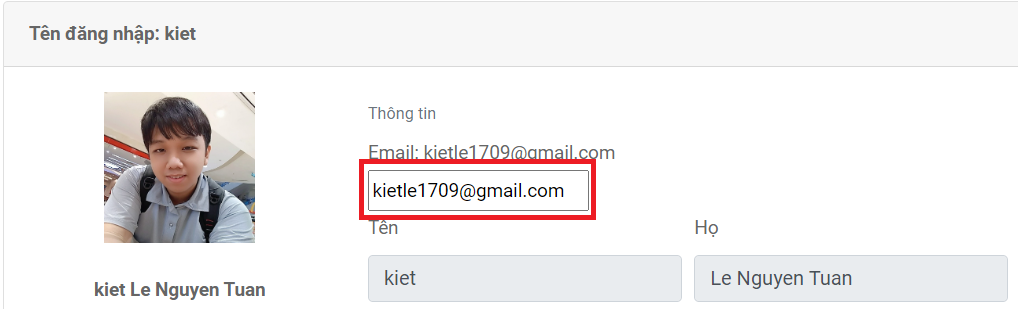
### Tấn công project

Trong project, tại trang quản lý thông tin cá nhân, người dùng có thể tùy ý chỉnh sửa thông tin tài khoản của mình tại form chỉnh sửa.

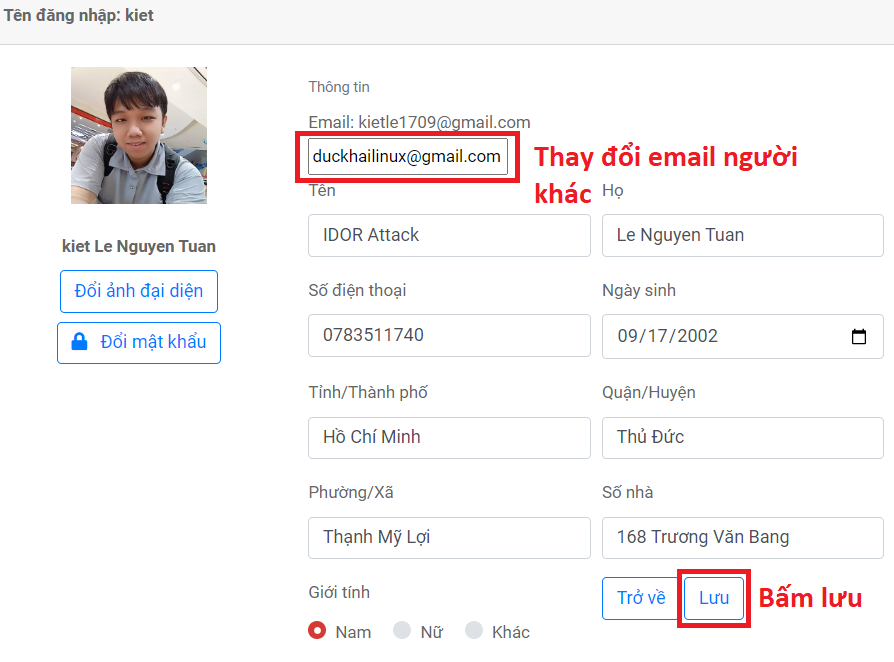
Tuy nhiên khi kiểm tra bằng F12, ta có thể nhận ra tồn tại thẻ input với thuộc tính là hidden, thẻ input này cho phép thay đổi email của yêu cầu chỉnh sửa



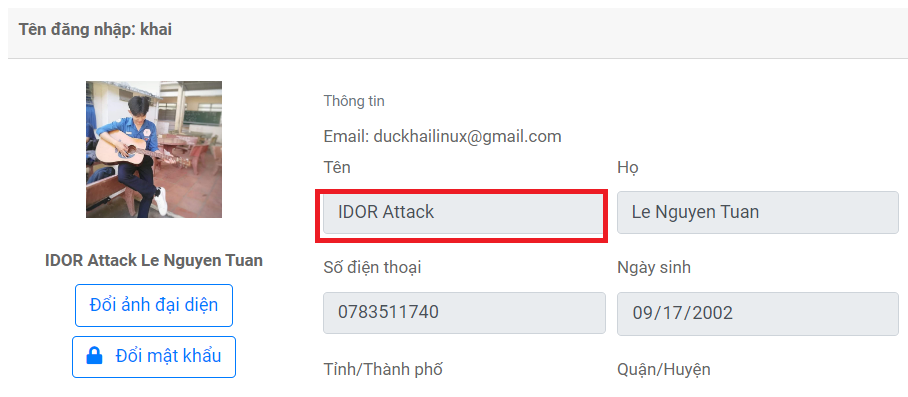
Khi ta loại bỏ thuộc tính “hidden”, 1 ô input hiện ra cho phép ta chỉnh sửa email:



Sau đó, thay đổi email ở ô hidden thành email của người dùng khác, sau đó chỉnh sửa thông tin mình mong muốn ở form chỉnh và bấm nút “Lưu”



Ta kiểm tra bằng cách đăng nhập vào tài khoản nạn nhân ( [duckhailinux@gmail.com](mailto:duckhailinux@gmail.com)). Và thông tin của nạn nhân đã bị chỉnh sửa trái phép bởi người dùng khác



### Nguyên nhân

Khi xem code project ở phần chỉnh sửa thông tin cá nhân, ta thấy server không hề xác thực người dùng mà chỉ đơn thuần là nhận trục tiếp email trong ô input có thuộc tính “hidden” ở phía client làm đối tượng thay đổi thông tin.



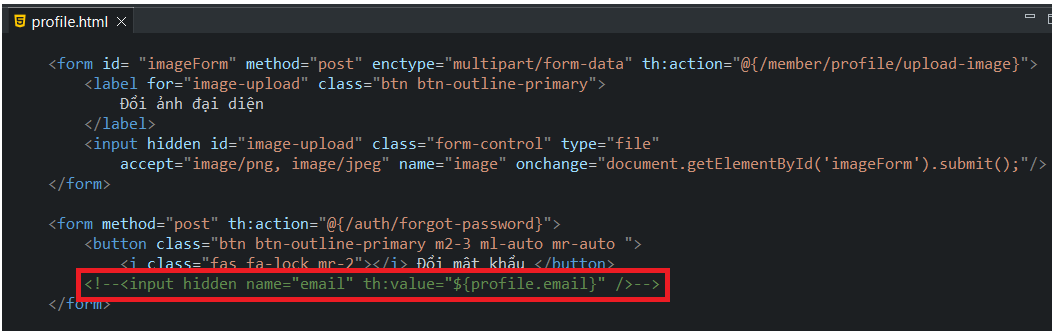
Vì vây, khi người dùng thay đổi email ở ô input, sẽ làm thay đổi đối tượng được yêu cầu chỉnh sửa thông tin.

### Khắc phục

Để khắc phục lỗ hổng này, ta sẽ tiến hành xác thực người dùng ở phía server và không sử dụng trực tiếp giá trị “email” từ ô input hidden nữa:



Đồng thời loại bỏ đi ô input hidden:



Nguyên nhân nằm ở cơ chế phân quyền, vì vậy việc phát triển ứng dụng tuân thủ chặt chẽ việc phân quyền và kiểm tra quyền truy cập tới dữ liệu là điều quan trọng nhất. Luôn kiểm tra quyền truy cập của mỗi người dùng ở tất cả các request. Việc kiểm tra phân quyền luôn luôn cần thực hiện trên phía server, không kiểm tra ở phía client. Đồng thời không nên sử dụng trực tiếp các giá trị quan trọng từ phía client mà phải qua cơ chế xác thực

## Insecure Direct Object Reference ( IDOR )

### Khái niệm

IDOR là cụm từ viết tắt của Insecure Direct Object Reference (Tham chiếu đối tượng trực tiếp không an toàn). Lỗ hổng này nằm trong TOP 10 lỗ hổng nguy hiểm mà OWASP đã đưa ra IDOR. Lỗ hổng này xảy ra khi chương trình cho phép người dùng truy cập tài nguyên (dữ liệu, file, thư mục, database..) một cách trực tiếp thông qua dữ liệu do người dùng cung cấp nhưng kém an toàn.

Nguyên nhân sâu xa của lỗ hổng IDOR nằm ở cơ chế phân quyền và kiểm soát truy cập người dùng trên website. Việc website không phân quyền đúng cũng như kiểm soát chặt chẽ việc truy cập tài nguyên của user trên hệ thống dẫn đến việc truy cập trái phép từ kẻ tấn công.

IDOR không chỉ cho phép truy cập thông tin trái phép của người dùng khác mà còn có thể cho phép kẻ tấn công thực hiện thay đổi hoặc xóa dữ liệu của người dùng khác. Vì vậy lỗ hổng này có mức độ nguy hiểm rất cao, gây hậu quả nghiêm trọng tới người dùng. Đặc biệt là các hệ thống có dữ liệu quan trọng của người dùng như: mạng xã hội, tài chính - ngân hàng, y tế..

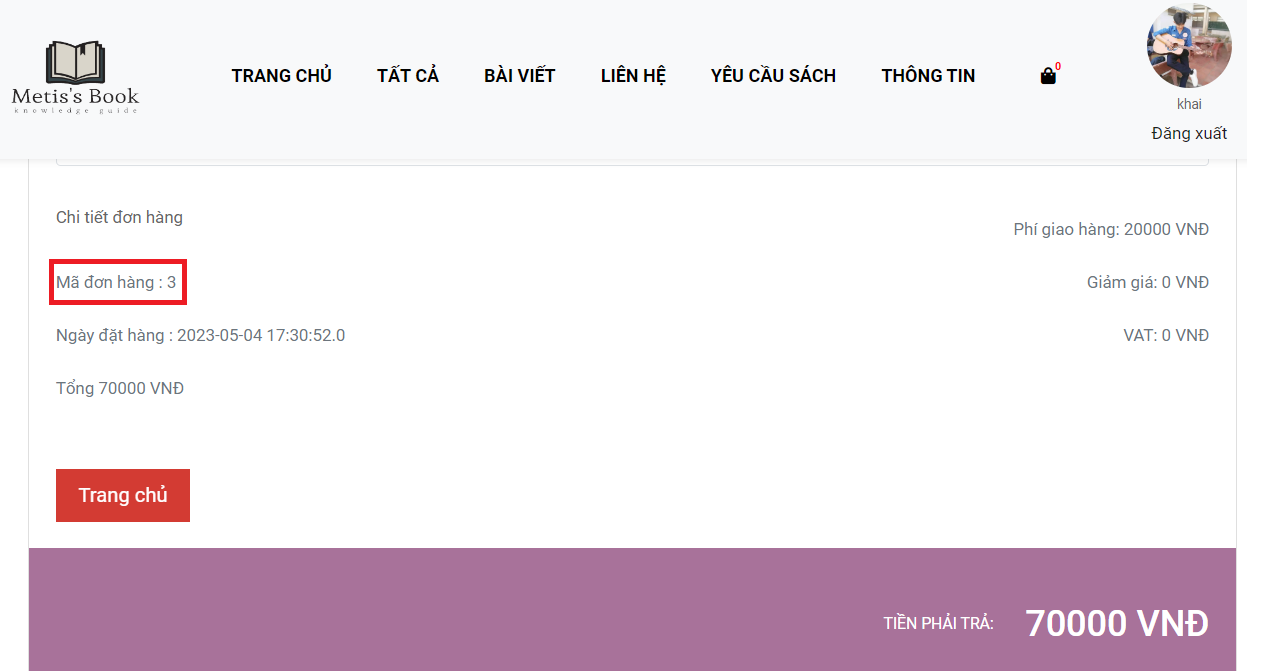
### Tấn công project

Trong project, tại trang xem thông tin hóa đơn, để ý trên url người dùng sẽ thấy có thể tìm kiếm hóa đơn theo id bằng cách thay đổi trực tiếp giá trị “orderId” :



Tuy nhiên vấn đề xảy ra khi người dùng thay đổi giá trị “orderId” không phải của mình.

Lấy ví dụ tài khoản của người dùng với email: [duckhailinux@gmail.com](mailto:duckhailinux@gmail.com). Có 1 hóa đơn với id là 3:



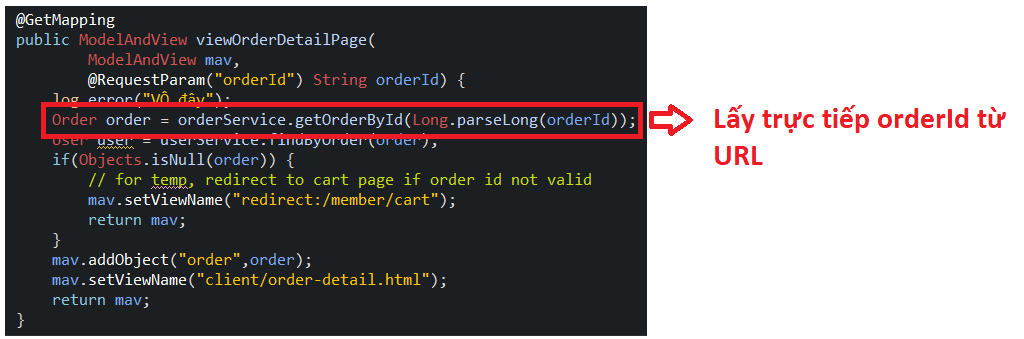
Người dùng với email [kietle1709@gmail.com](mailto:kietle1709@gmail.com). Khi thay đổi giá trị “orderId” trên url có thể truy cập và xem thông tin hóa đơn với id là 3:



Như vậy, các thông tin hóa đơn nhạy cảm có thể bị đọc bởi người dùng khác.

### Nguyên nhân

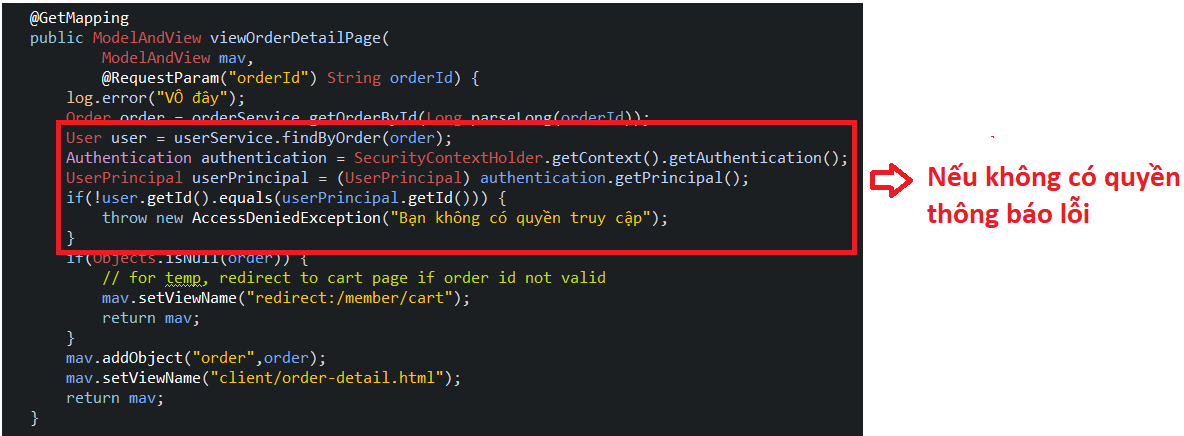
Nguyên nhân đến từ phía server đã không tiến hành xác thực người dùng trước khi cho phép truy cập vào thông tin hóa đơn. Xem lại code, ta thấy:



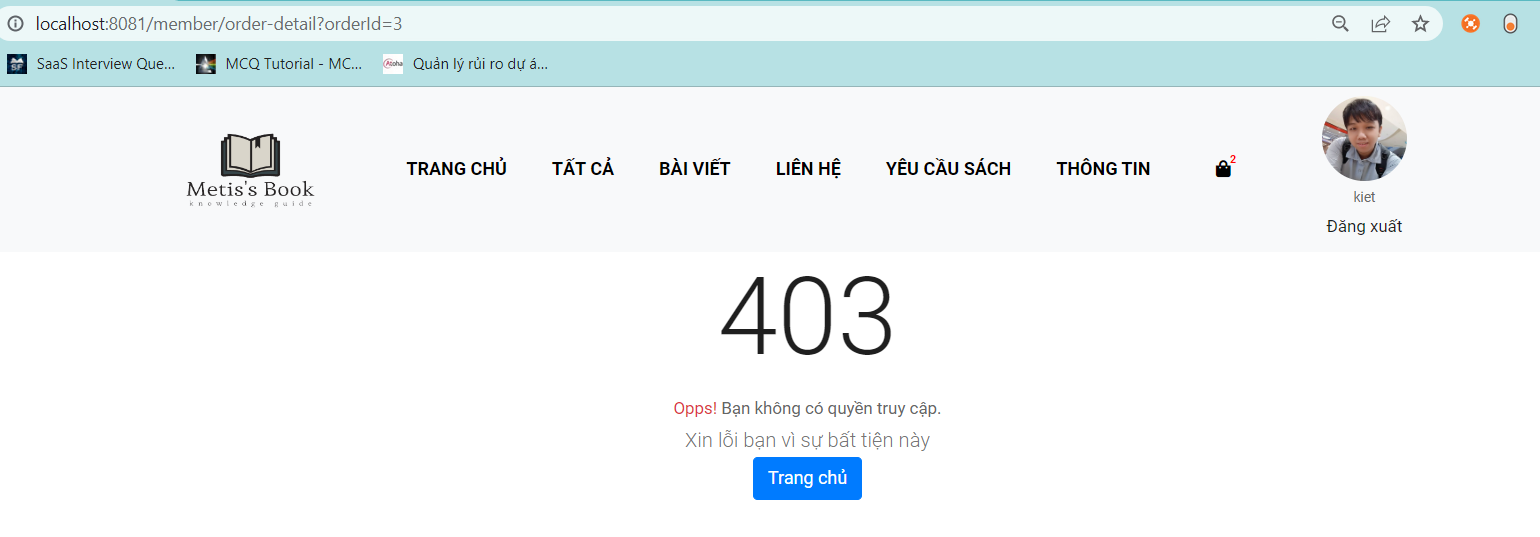
Việc giá trị orderId được lấy trực tiếp từ đường dẫn mà không qua bất kỳ cơ chế xác thực nào, đã khiến cho thông tin hóa đơn có thể dễ dàng bị truy cập bởi người dùng trái phép

### Khắc phục

Vì nguyên nhân bắt nguồn từ việc không xác thực người dùng ở phía server. Nên để khắc phục, ta sẽ tiễn hành xác thực người dùng:



Khi người dùng cố ý thay đổi orderId để truy cập vào thông tin hóa đơn người khác, website sẽ dẫn đến màn hình lỗi:



IDOR là một lỗ hổng nguy hiểm có thể đe dọa quyền riêng tư của người dùng và tính toàn vẹn của ứng dụng của bạn. Việc tuân thủ các cơ chế phân quyền chặt chẽ và kiểm tra ứng dụng kĩ sẽ giúp các bạn hạn chế được nguy cơ về lỗ hổng bảo mật IDOR.

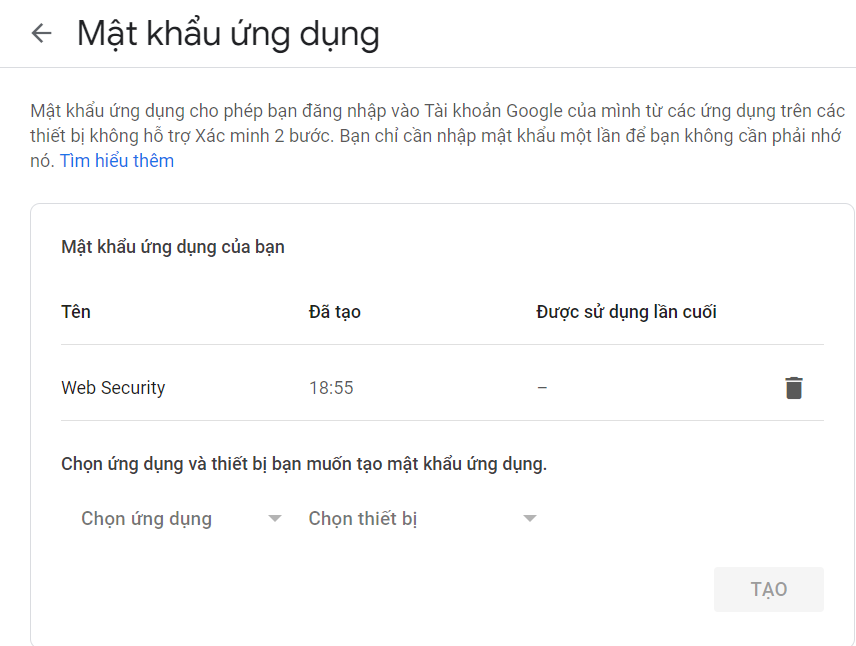
## SMTP credentials exposed

### Khái niệm

SMTP credentials exposed (hay còn gọi là "lộ thông tin đăng nhập SMTP") là khi thông tin đăng nhập cho giao thức SMTP (Simple Mail Transfer Protocol) bị tiết lộ hoặc bị phát hiện bởi người không được ủy quyền hoặc bởi một bên thứ ba không đáng tin cậy. Thông tin đăng nhập này bao gồm Id và secretId, được sử dụng để gửi email từ địa chỉ email đó.

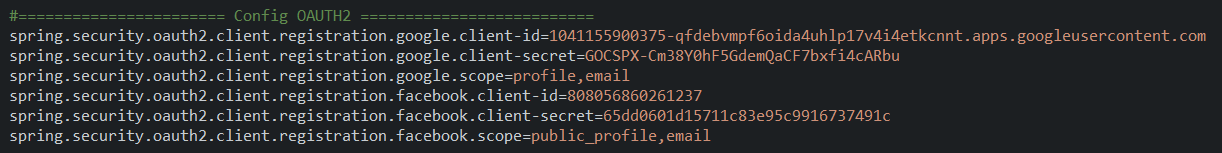
Khi SMTP credentials exposed, kẻ tấn công có thể thực hiện các hành động gửi email giả mạo, lừa đảo hoặc tấn công khác để gây hại cho người dùng hoặc các liên hệ của họ. Vì vậy, việc bảo vệ thông tin đăng nhập SMTP là rất quan trọng để bảo vệ tài khoản email của người dùng.

Truy cập vào google account, ta có thể tạo một cặp mật khẩu cho SMTP, cặp mật khẩu này dùng để tự động gửi mail trong cái ứng dụng, ví dụ như thư viện Java Mail Sender.

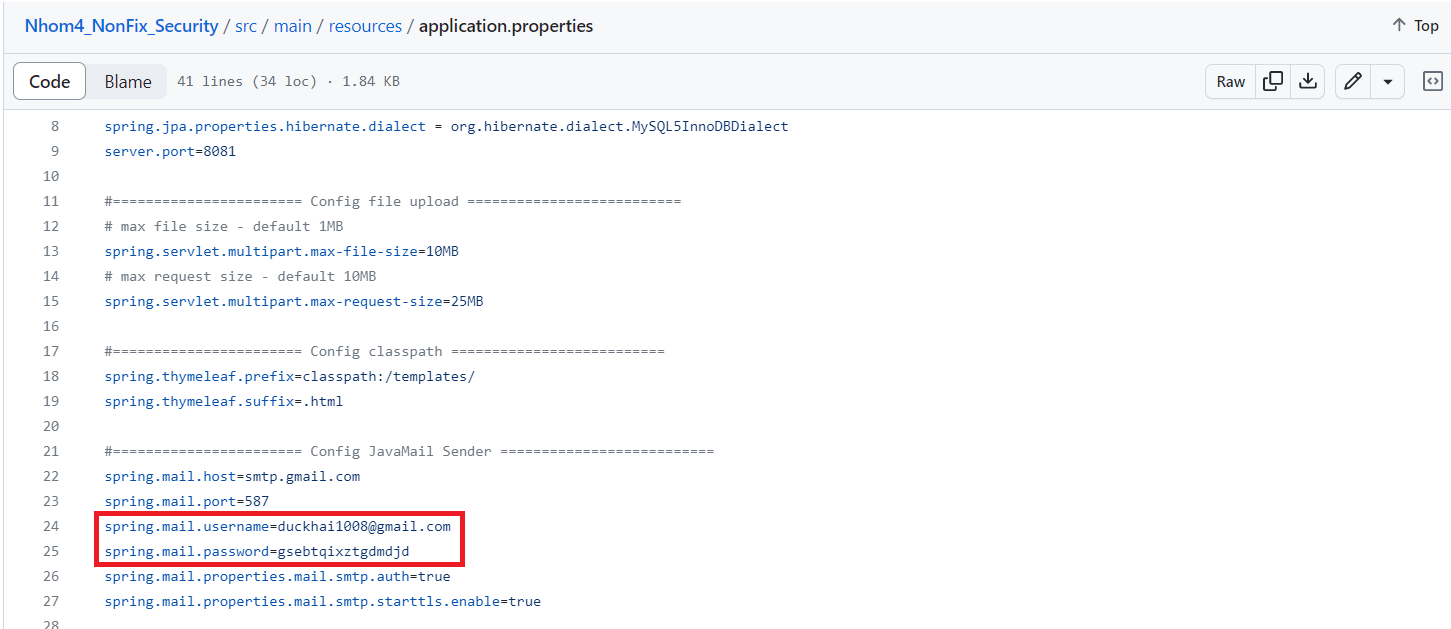


### Lỗ hổng trong project

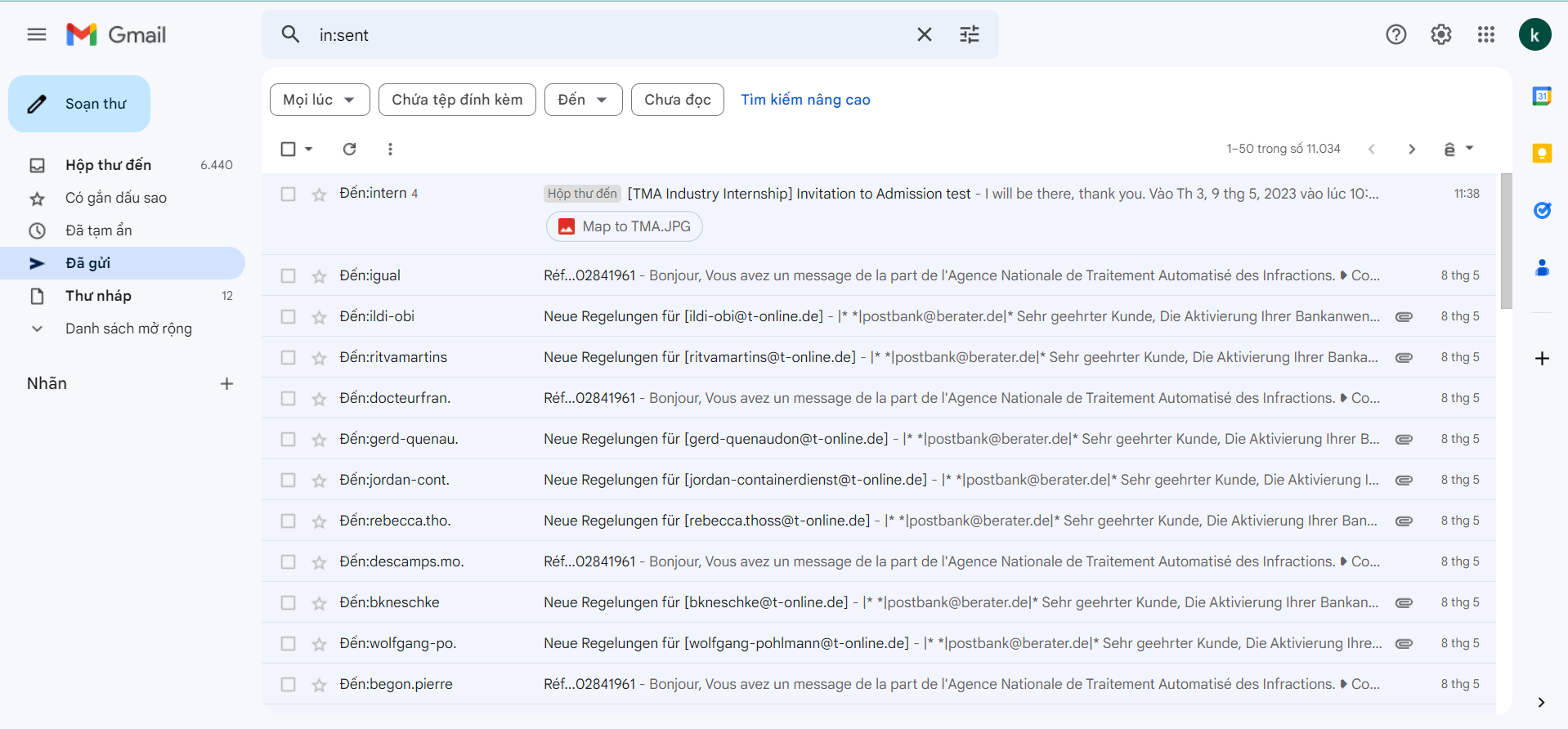
Trong project, tại file cài đặt, nhóm lập trình đã cấu hình tài khoản và mật khẩu của SMTP phục vụ cho quá trình tự động gửi mail của website ( xác nhận tài khoản, lấy lại mật khẩu ), trong file application.properties:



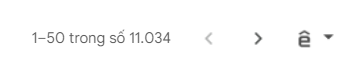
Tuy nhiên, nhóm lập trình sử dụng github làm nơi lưu trữ mã nguồn mà không che giấu đi thông tin quan trọng này. Khiến lộ thông tin trên github:



Hacker có thể lợi dụng và sử dụng SMTP để làm những hành động xấu. Điển hình là với email của bạn Khải: [duckhailinux@gmail.com](mailto:duckhailinux@gmail.com). Hacker đã sử dụng tài khoản SMTP này để gửi mail spam cho nhiều người:



Có thể thấy trong suốt quá trình bị lộ thông tin, hacker đã gửi email spam tổng cộng gần 11000 lần:



Nội dung email chủ yếu là tin nhắn spam đại diện cho 1 ngân hàng nào đó ở nước Đức:



Có thể thấy, việc làm lộ thông tin SMTP đã mang lại rất nhiều phiền toái, chưa kể, trong số email được gửi đi, có thể có email lừa đảo ( Tuy nhiên rất may mắn là việc đó không xảy ra ).

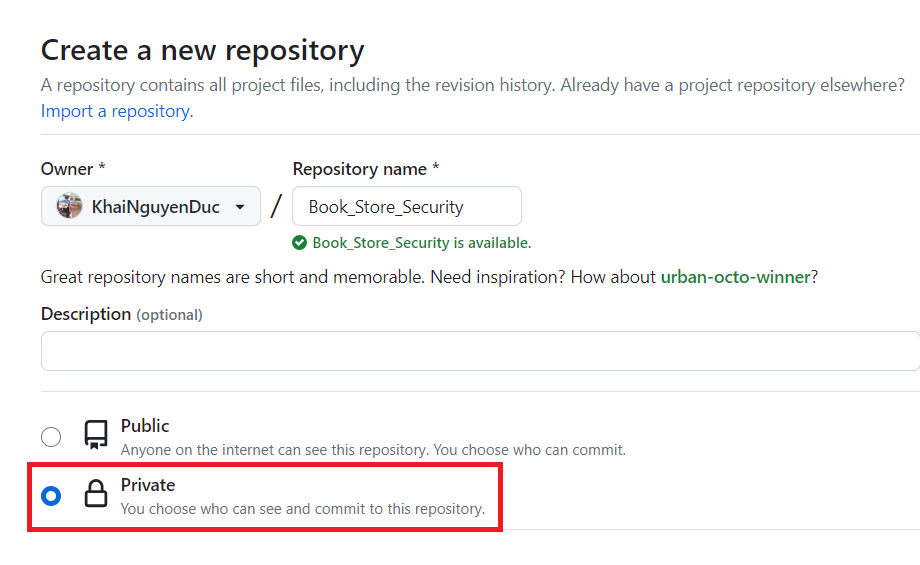
### Nguyên nhân

Nguyên nhân chủ yếu từ việc không cẩn thận che giấu thông tin nhạy cảm trong quá trình công khai mã nguồn. Github là nơi lưu trữ mã nguồn mở, và nhóm đã không cẩn thận khi upload code lên đây.

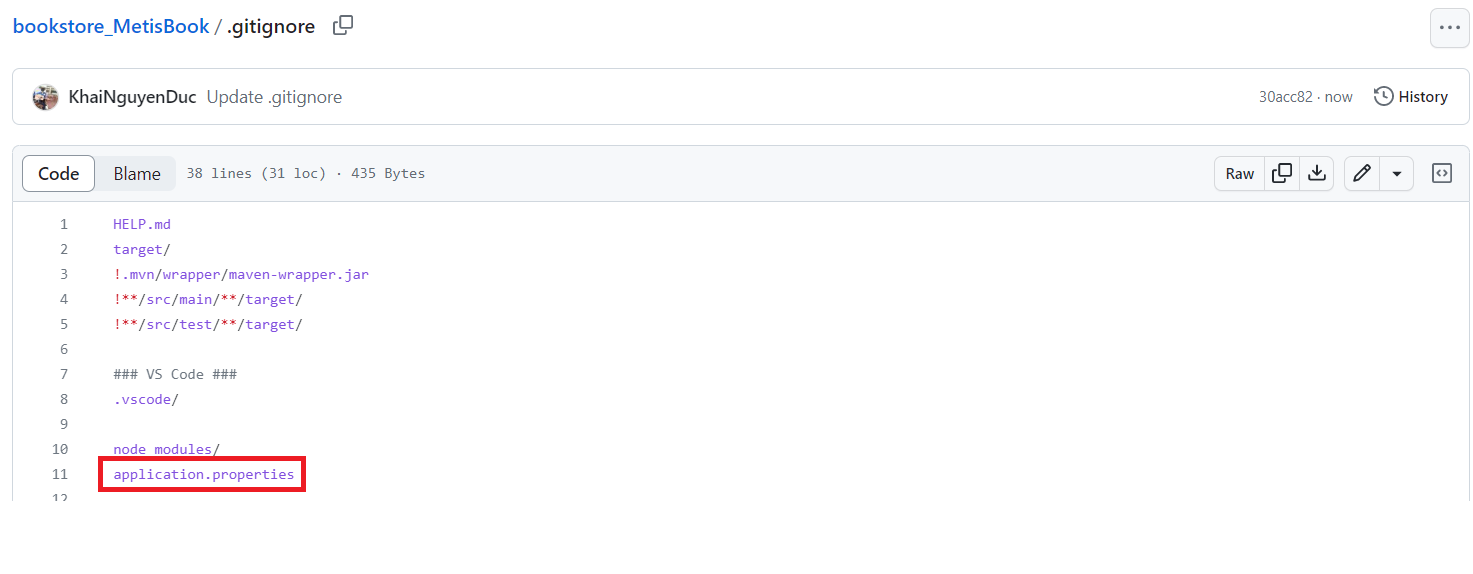
### Khắc phục

Có nhiều cách để khắc phục lỗ hổng này.

**Cách 1**: Sử dụng github repository private. Github cung cấp mỗi tài khoản ( không thuộc loại vip ) một nơi lưu trữ mã nguồn riêng tư. Khi phát triển những dự án quan trọng, ta nên đưa code lên nơi lưu trữ riêng tư để tránh người khác xem code và đánh cắp dữ liệu quan trọng.



**Cách 2:** Sử dụng file .gitignore. Trong trường hợp ta cần công khai code trên github. Hãy thêm tên những file quan trọng trong file .gitignore:



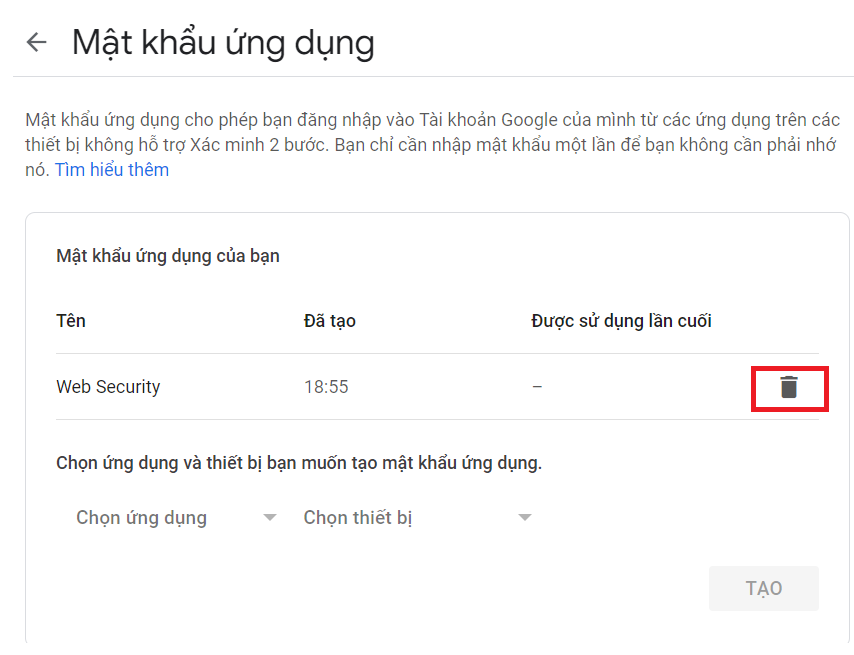
Khi đưa code lên github, những file được liệt kê trong .gitignore sẽ được ẩn đi và không được đưa lên cùng mã nguồn. Điều này khiến mã SMTP không còn được public ra nữa.

Dù sử dụng cách nào đi nữa, một khi đã bị lộ mã SMTP, ta phải lập tức xóa mật khẩu ứng dụng tại trang <https://www.google.com/account/about/>. Điều này làm cho mã SMTP hiện tại không còn sử dụng được nữa.

Tìm kiếm mật khẩu ứng dụng:



Nhấn vào biểu tương sau để xóa:



Như vậy, nhóm thực hiện đã thành công khắc phục được lỗ hổng này, và rút được kinh nghiệm quý báu về tầm quan trọng của bảo mật thông tin

## Brute Force Attack ( Admin Password )

### Khái niệm

Tấn công brute-force là một phương pháp bẻ khóa phổ biến . Hacker sẽ 'đoán' tên người dùng và mật khẩu để truy cập trái phép vào hệ thống thông qua việc sử dụng phương pháp thử và sai để cố gắng đoán thông tin đăng nhập hợp lệ.

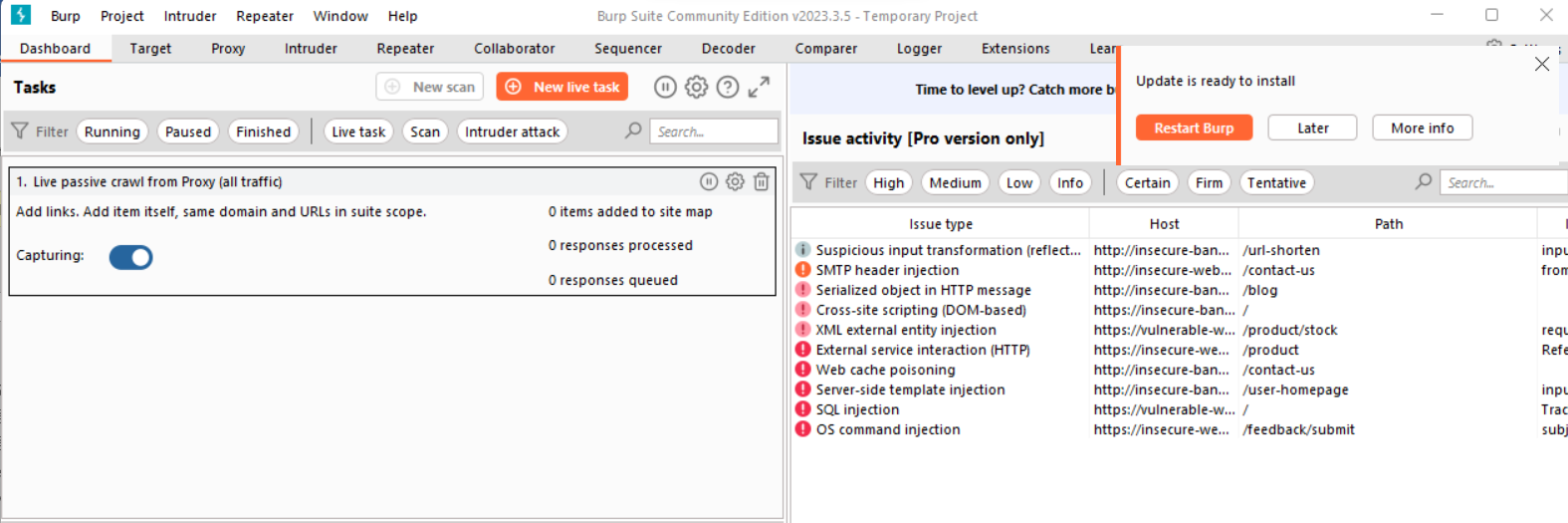
Các cuộc tấn công này thường được tự động hóa bằng cách sử dụng danh sách các mật khẩu thông dụng thường được gọi là “dictionary”. Một dictionary có thể bao gồm hàng triệu mật khẩu. Vì vậy, hacker sẽ sử dụng nhưng công cụ chuyên dụng có khả năng cho phép hacker thực hiện đăng nhập 1 cách tự động nhiều lần với tốc độ cao.

Nếu mật khẩu sử dụng tất cả các chữ cái thường và không có ký tự đặc biệt hoặc chữ số, chỉ mất 2-10 phút là một cuộc tấn công brute force có thể crack mật khẩu này. Ngược lại, một mật khẩu có sự kết hợp của cả chữ hoa và chữ thường cùng với một vài chữ số (giả sử có 8 chữ số) sẽ mất hơn 14-15 năm để bị crack.

Để kiểm tra độ mạnh của mật khẩu. Ta có thể kiểm tra tại các trang web như: <https://www.passwordmonster.com/>

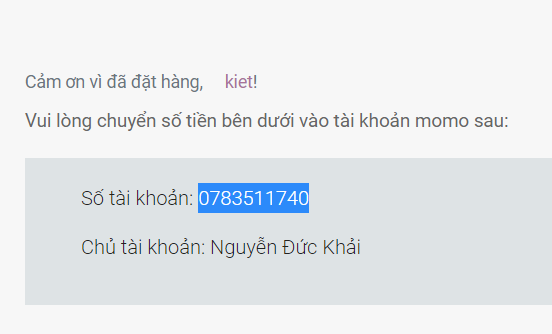
### Tấn công project

Ta sẽ tiến giả làm hacker tiến hành tấn công bằng phân mềm burp suite. Đây là một công cụ cực kỳ mạnh mẽ trong việc kiểm thử và tấn công thông qua việc gửi và chỉnh sửa các yêu cầu ( request )

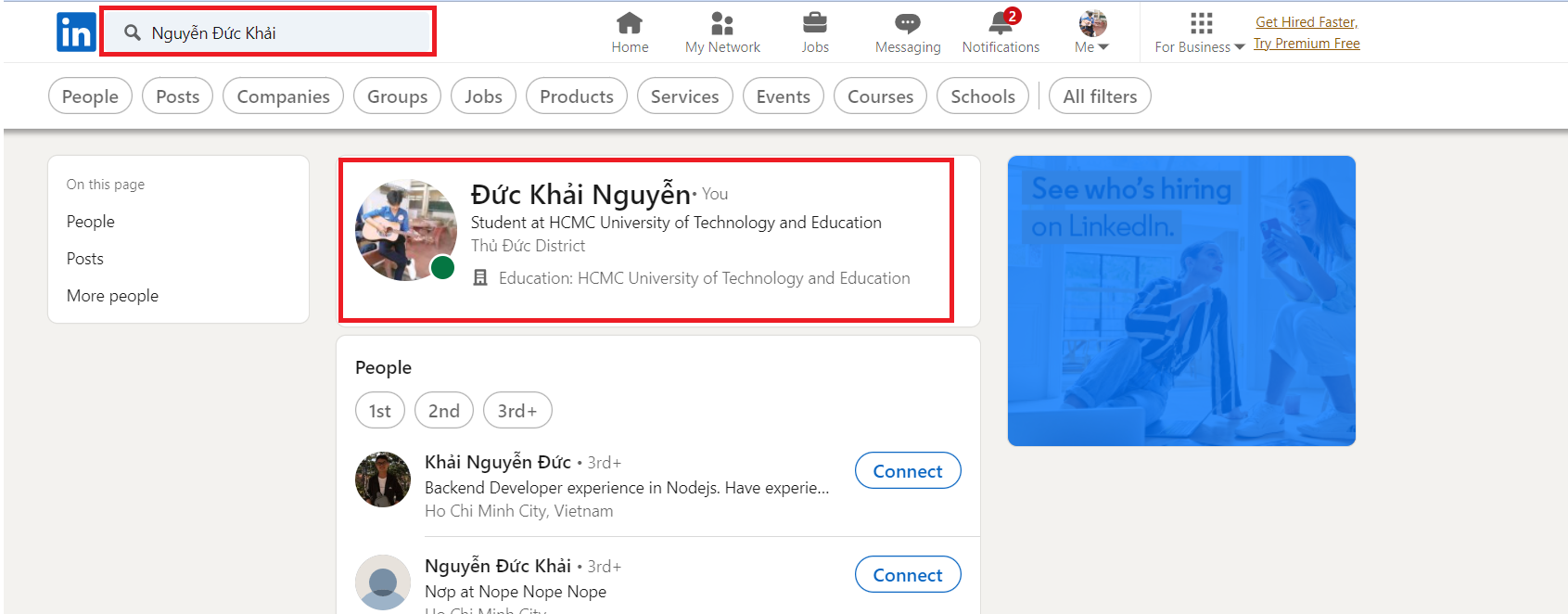


Đầu tiên, ta biết được trang đăng nhập của website chạy ở đường dẫn: <http://localhost:8081/auth/login>. Tiếp theo, cần tìm email của admin để tiến hành tấn công vét cạn mật khẩu.

Để ý thấy, khi người dùng đặt hàng thông qua phương thức thanh toán là momo. Số điện thoại và tên của admin sẽ được thông báo:

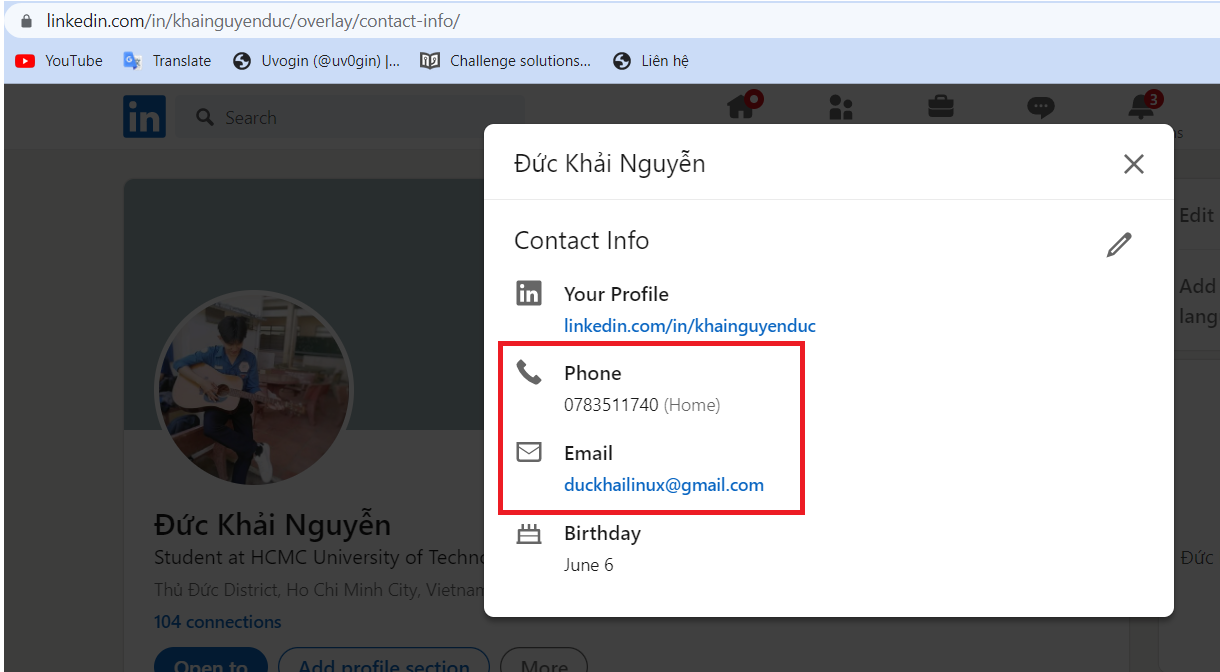


Ta đã biết được tên và số điện thoại của admin. Thông qua kỹ thuật OSINT, ta tìm được thông tin địa chỉ email của admin thông qua mạng xã hội linkedlink:



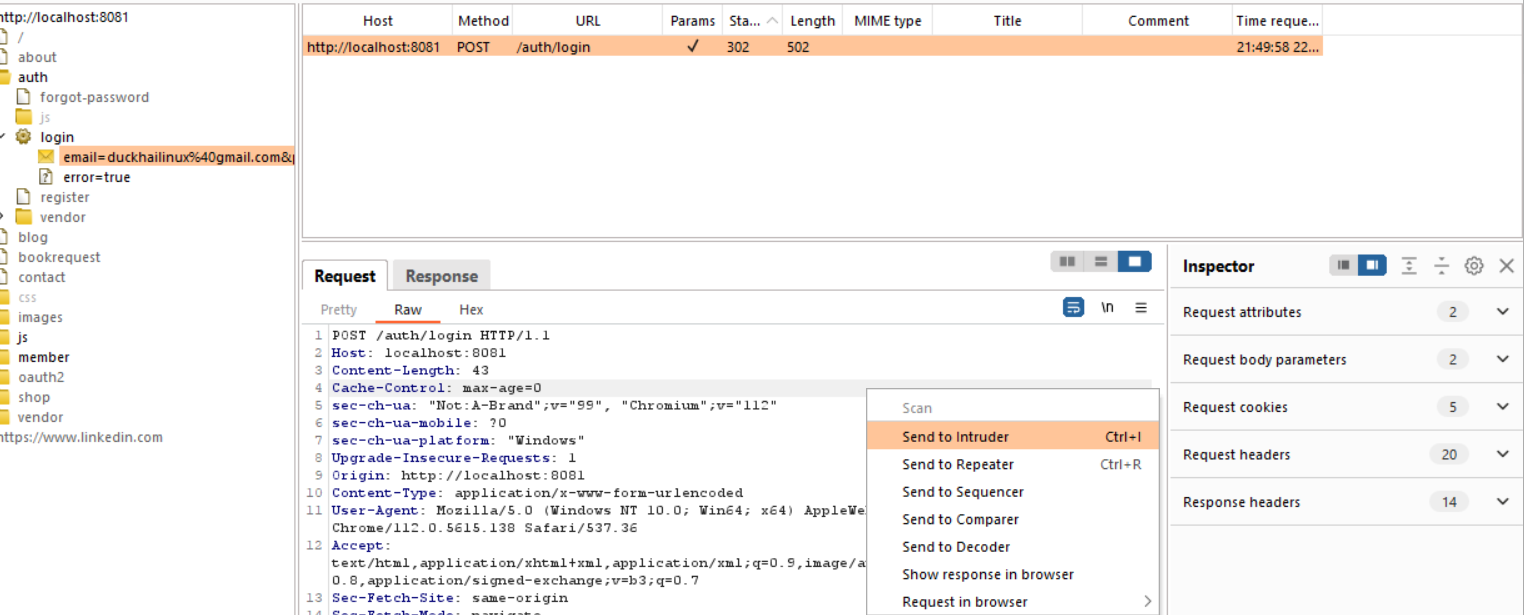
Khi vào xem trang cá nhân: <https://www.linkedin.com/in/khainguyenduc/overlay/contact-info/>

Ta thấy được email của người dùng là: [duckhailinux@gmail.com](mailto:duckhailinux@gmail.com) và khớp với số điện thoại ở trên website

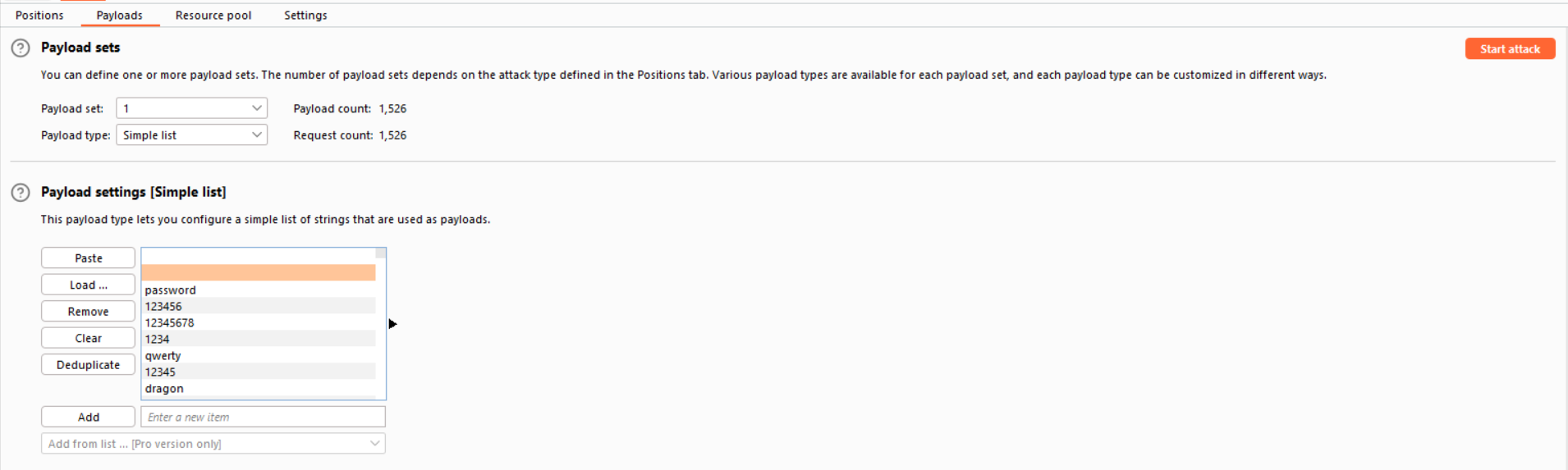


Vì vậy email của admin là [duckhailinux@gmail.com](mailto:duckhailinux@gmail.com)

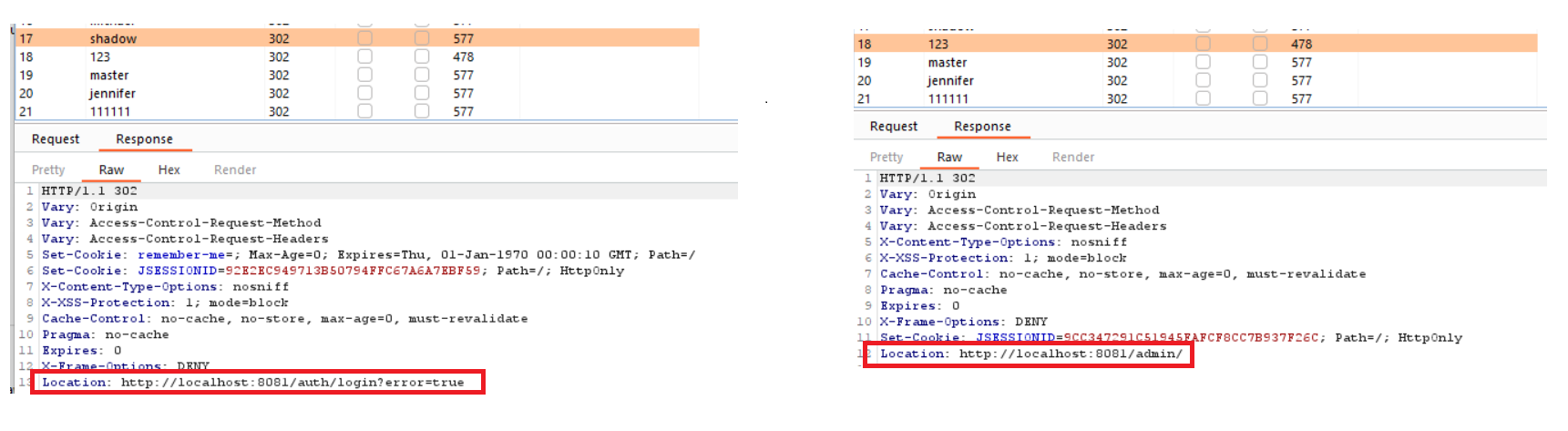
Tiếp theo, để có thể tấn công vét cạn ( Brute force ) mật khẩu. Ta sử dụng phần mềm burp suite. Gửi request đăng nhập với email là [duckhailinux@gmail.com](mailto:duckhailinux@gmail.com) đến intruder:



Tiếp theo, tiến hành đưa file dictionary chứa các mật khẩu thông dụng vào công cụ:

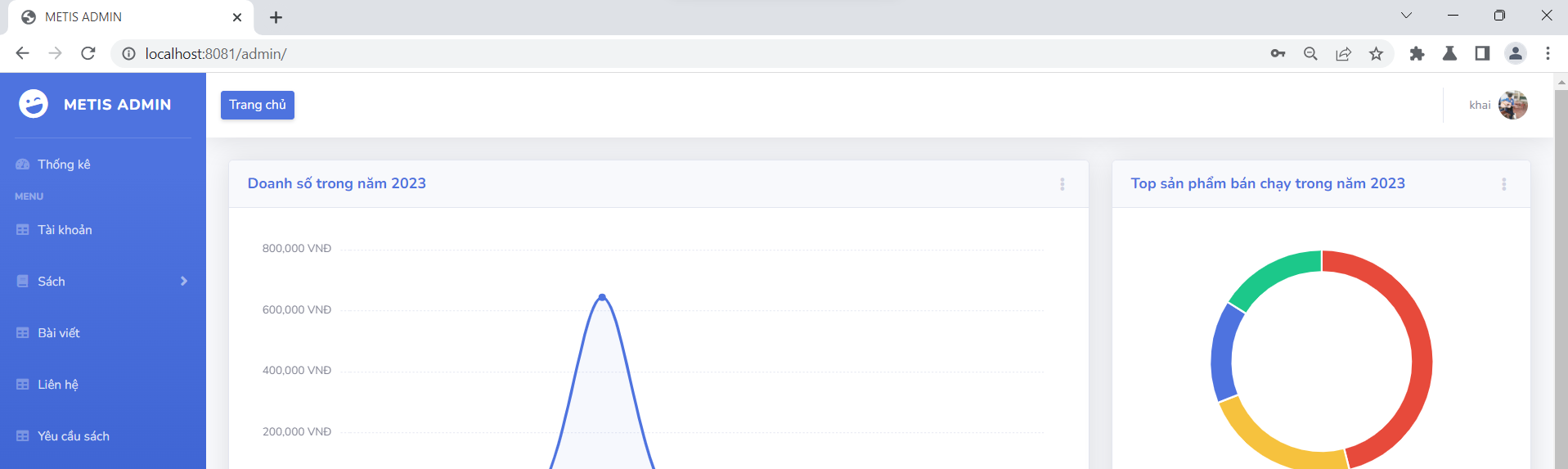


Tiến hành tấn công brute force. Để ý thấy tại mật khẩu là “123”, đường dẫn trả về là trang admin. Còn các mật khẩu khác, đường dẫn trả về là trang error:



Vì vậy, ta biết được tài khoản và mật khẩu của admin là [duckhailinux@gmail.com](mailto:duckhailinux@gmail.com) | 123

Tiến hành truy cập và đã vào được trang admin:



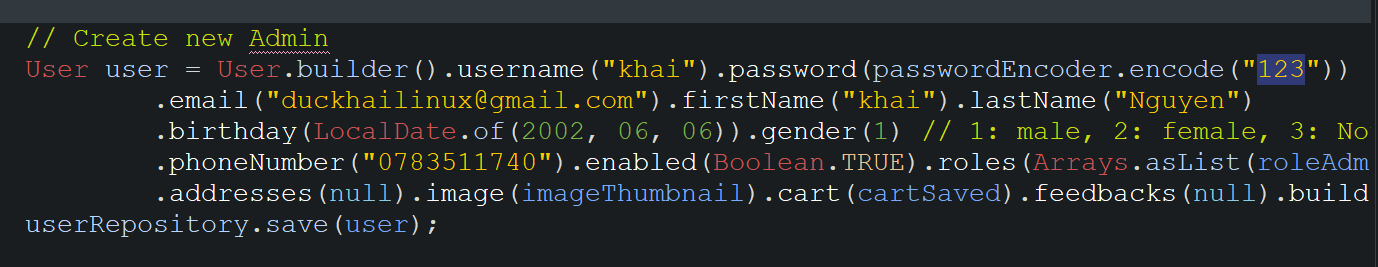
### Nguyên nhân

Có 2 nguyên nhân chính dẫn đến việc trang web dễ dàng bị tấn công brute force.

*Nguyên nhân đầu tiên*, là việc email của admin bị lộ. Việc bảo vệ sự bí mật của tài khoản admin là rất quan trọng, bất kể là email hay mật khẩu. Trang web đã sử dụng email được công khai trên mạng xã hội để làm tài khoản admin. Việc này có thể bị hacker tìm thấy thông qua một số kỹ thuật OSINT cơ bản.

*Nguyên nhân thứ 2*, cũng là nguyên nhân quan trọng nhất là mật khẩu của admin quá yếu. Chỉ tốn 1-2ph là có thể tìm ra được mật khẩu thông qua tấn công vẹt cạn. “123” là mật khẩu nằm trong top các mật khẩu dễ phá và thường được sử dụng nhất mà nhóm thực hiện xin phép được demo.

Có thể thấy trong mã nguồn, mật khẩu mặc định cho admin là “123”

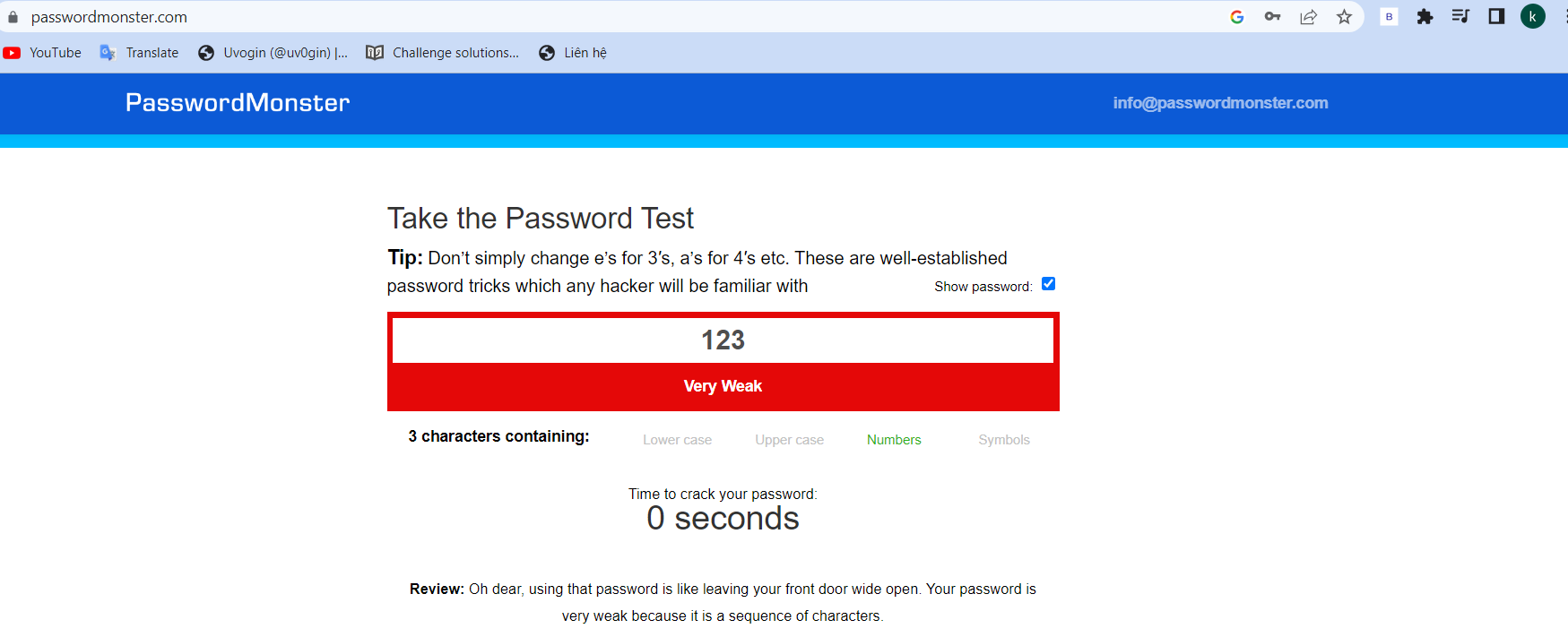


### Khắc phục

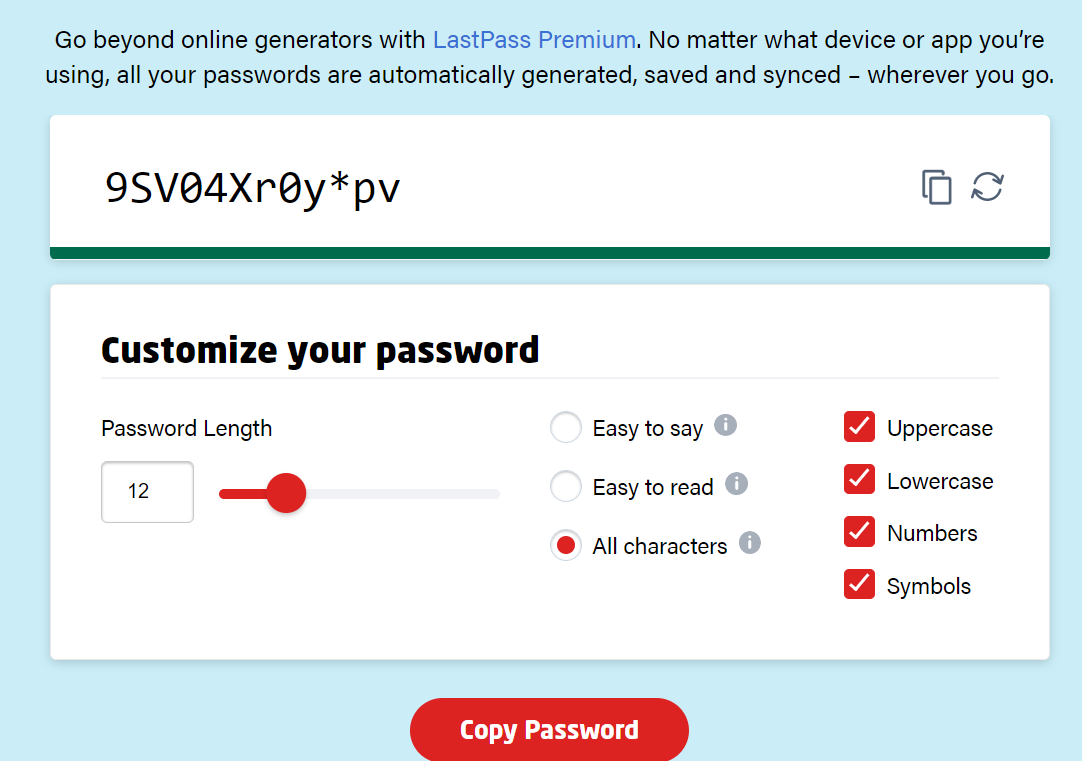
Đây là loại tấn công khai thác thông tin nhạy cảm của trang web. Hậu quả để lại rất nặng thế nhưng nhiều người dùng vẫn “Vô tư” đặc mật khẩu quá dễ đoán và dễ tấn công.

Để khắc phục được vấn đề này. Điều đầu tiên cần làm là nâng cao nhận thức của người dùng. Khiến họ thực sự cảm thấy việc đặc mật khẩu yếu kém là 1 mối đe dọa lớn đối với bản thân chính họ.

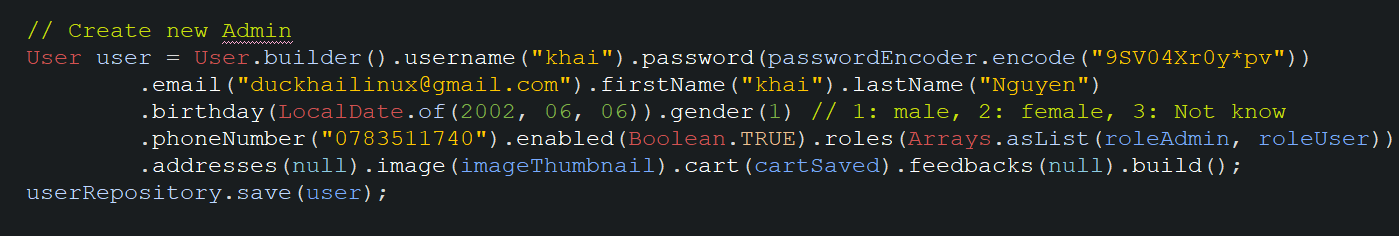
Tiếp theo, cần phải thay đổi mật khẩu mạnh hơn, ta có thể kiểm tra độ mạnh của mật khẩu ở một số trang web uy tín như: <https://www.passwordmonster.com/>:



Ta có thể sử dụng trình đặt mật khẩu tự động, để có 1 mật khẩu đủ mạnh như: <https://www.lastpass.com/features/password-generator> :



Đồng thời đặt lại mật khẩu của admin:

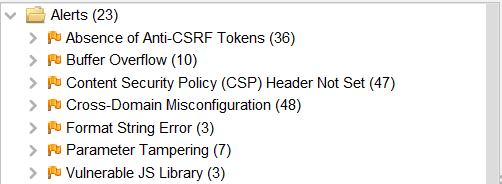


Việc chọn 1 mật khẩu khó có thể tấn công rất quan trọng. Nó giúp phần ngăn chặn việc bị tấn công vét cạn và do ra mật khẩu

# Phần 3: Quét bằng ZAP

## Giới thiệu:

Sử dụng ZAP ( Zed Attack Proxy ) quét được các lỗ hổng sau:



## Absence of Anti-CSRF Tokens

### Khái niệm

Thiếu token xác thực CSRF là một lỗ hổng trong đó form với phương thức POST không được cài đặt các token xác thực CSRF. Điều này có thể dẫn đến việc hệ thống bị tấn công CSRF.

CSRF ( Cross Site Request Forgery) là kỹ thuật tấn công bằng cách sử dụng quyền chứng thực của người dùng đối với một website. CSRF là kỹ thuật tấn công vào người dùng, dựa vào đó hacker có thể thực thi những thao tác phải yêu cầu sự chứng thực. Hiểu một cách nôm na, đây là kỹ thuật tấn công dựa vào mượn quyền trái phép.

Lấy ví dụ người dùng Kiệt đăng một bài đăng với ý đồ xấu, trong bài post đó chứa đường dẫn như sau:

<img height="0" width="0" src="http://www.webapp.com/project/1/destroy">

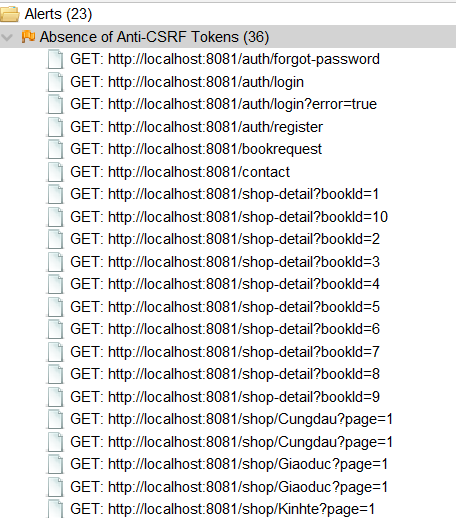
Khi người dùng Khải bấm vào xem bài đăng đó và vô tình bấm vào tấm ảnh, đường link trong tấm ảnh sẽ gọi đến request xóa project của Khải.

Để tăng hiệu quả che dấu, đoạn mã trên có thể được thêm các thông điệp bình thường để người dùng không chú ý. Thêm vào đó thẻ img sử dụng trong trường hợp này có kích thước 0x0 pixel và người dùng sẽ không thể thấy được.

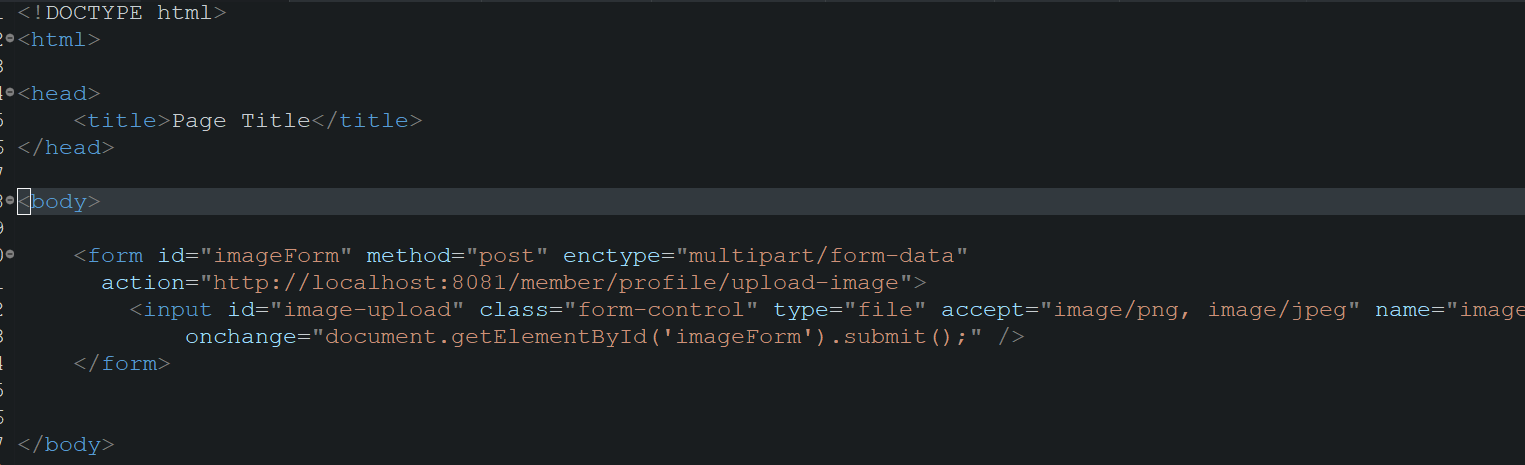
Các kĩ thuật CSRF rất đa dạng, lừa người dùng click vào link, gửi email chứa các đoạn mã độc đến người dùng…

### Tấn công project

Sử dụng công cụ ZAP, ta biết được rất nhiều request có thể tấn công CSRF. Ta sẽ tiến hành thử tấn công với request <http://localhost:8081/member/profile/upload-image>



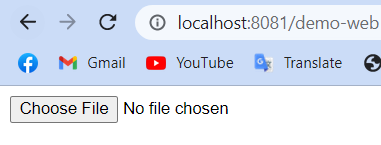
Giả sử, ta tạo ra 1 trang web giả mạo chứa 1 form upload hình như sau:



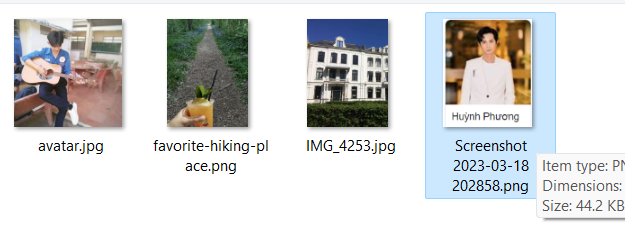
Trong form này, khi người dùng upload 1 tấm ảnh, hệ thống sẽ gửi 1 post request upload ảnh đến trang web với url: <http://localhost:8081/member/profile/upload-image>.

Giả sử người dùng Kiệt upload 1 tấm ảnh ở Website lừa đảo. Trang web sẽ tiến hành gửi request đó đến website metisBookstore. Nếu Kiệt chưa đăng xuất khỏi website và vẫn còn lưu cookies của mình trong website bookstore. Hình ảnh đại diện của Kiệt sẽ bị thay đổi:

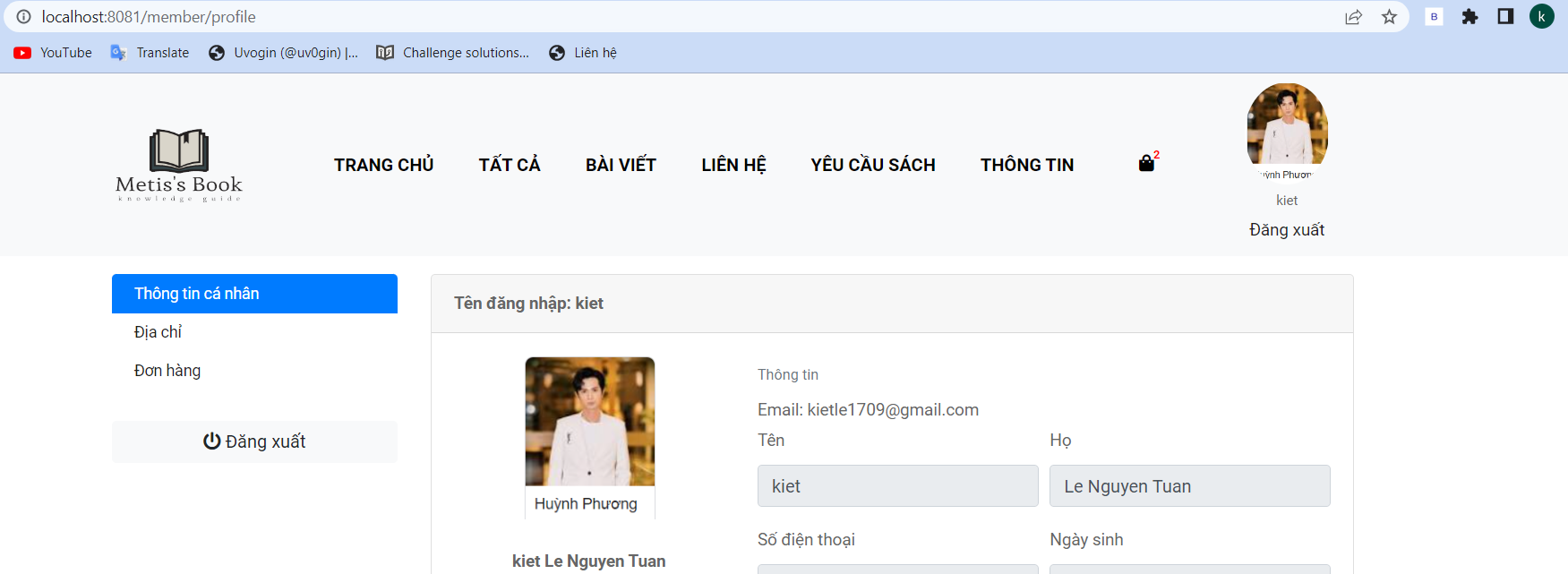
**Bước 1:** Người dùng Kiệt bấm vào nút upload hình ảnh ở trang web lừa đảo



**Bước 2:** Người dùng Kiệt chọn hình ảnh để upload:



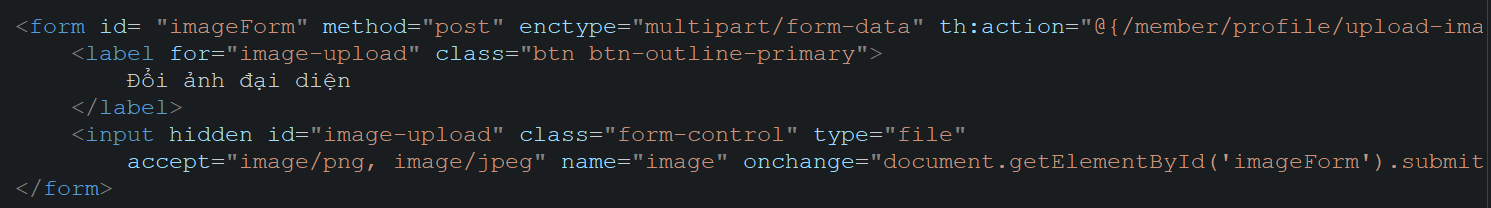
**Bước 3:** Trang cá nhân trang metis book của Kiệt bị thay đổi hình ảnh:



### Nguyên nhân

Nguyên nhân xuất phát từ việc ở trang web metis book. Các form với phương thức là Post không có CSRF token để phòng chống tấn công CSRF.

Form upload hình ảnh không hề có CSRF token:



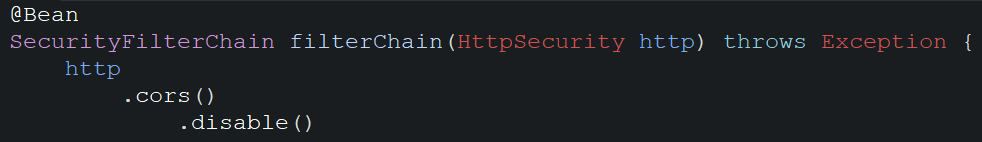
Trong file cấu hình bảo mật, cơ chế CSRF cũng bị tắt đi



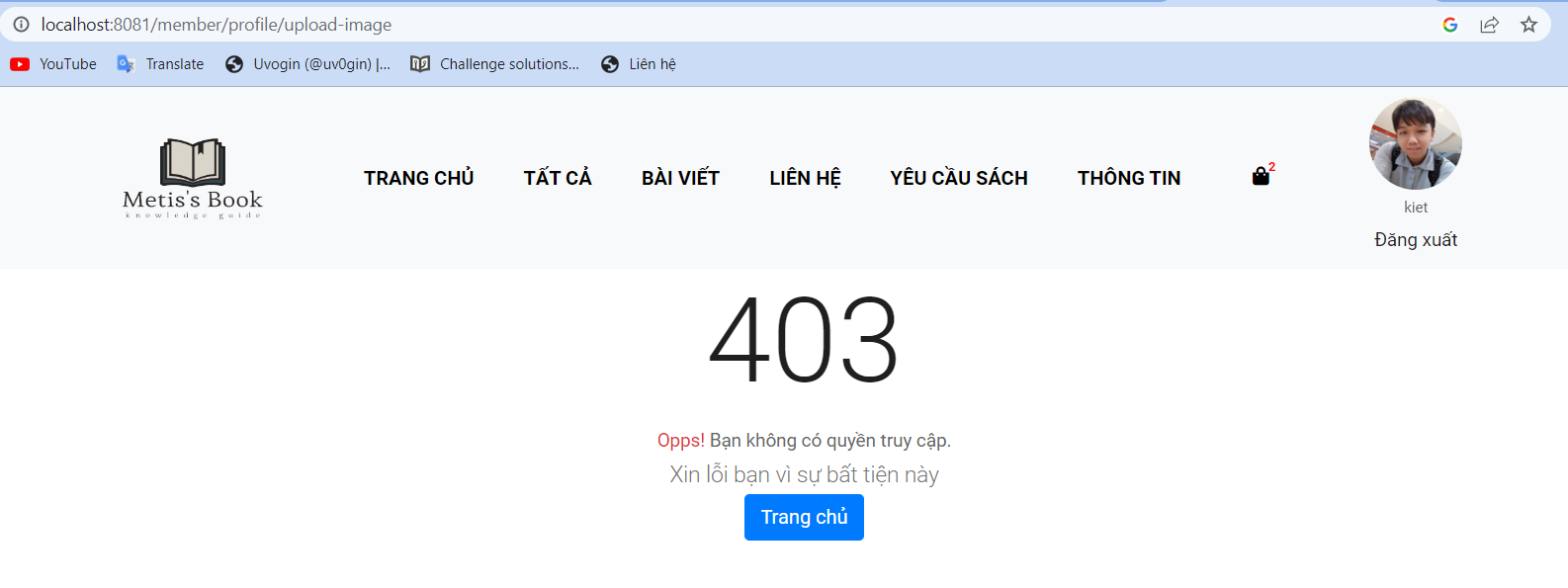
Trong quá trình lập trình, nhóm thực hiện đã tắt đi tính năng bảo vệ CSRF nhằm thuận lợi cho việc test request và api. Việc thiếu đi CSRF Token sẽ khiến sự toàn vẹn của dữ liệu bị mất đi. Hacker có thể lợi dụng đều này để tấn công URL từ một request ở trang web khác. Nếu người dùng chưa kiệp đăng xuất khỏi trang web, các thông tin quan trọng của người dùng có thể bị lấy mất hoặc sửa đổi vì không có cơ chế xác thực request đấy đến từ trang web nào.

### Khắc phục

Để trang web không bị tấn công CSRF, ta cần bật chế độ bảo mật CSRF ( tắt disable ) ở trong file cài đặt:

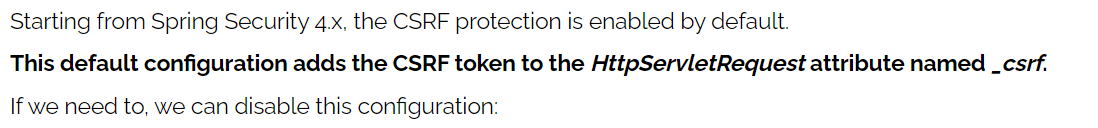


Như vậy, cơ chế bảo mật CSRF đã bị tắt đi và không còn có thể thay đổi dữ liệu của người dùng từ 1 trang khác nữa:



Nhóm sử dụng spring security, nếu ta không tắt đi cơ chế bảo mật csrf ( Đây là lỗi trong quá trình lập trình của nhóm ). Spring Security sẽ tự động có cơ chế bảo mật bằng CSRF.

Theo như document của spring có để cập cơ chế bảo mật CSRF sẽ tự động được bật nếu không bị tắt đi: <https://www.baeldung.com/spring-security-csrf#:~:text=To%20protect%20MVC%20applications%2C%20Spring,token%20from%20their%20own%20page>.



## Content Security Policy (CSP) Header Not Set

### 3.1. Khái niệm

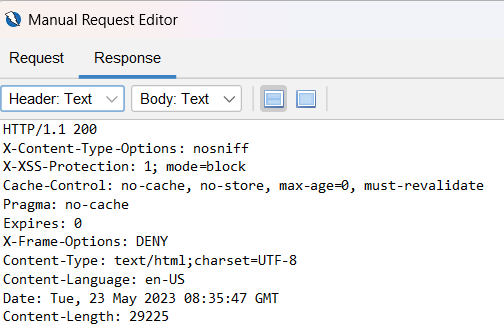
Content Security policy là một biện pháp bảo mật từ phía trình duyệt, giúp ngăn chặn những request xấu được gọi từ website của chúng ta. CSP là tập hợp một danh sách an toàn (whitelist) những domain hay kiểu script, style, image, frame mà trình duyệt được load trên website của chúng ta. Điều này đảm bảo chỉ có những script, style, image, frame được chúng ta chỉ định có thể được tải, những request không nằm trong whitelist sẽ bị chặn ngay tức khắc.

Nếu một hacker xâm nhập được vào hệ thống của ta mà chèn một thẻ <script> dẫn tới một tập tin javascript có ý đồ xấu nằm ở tên miền của hắn, ví dụ xauxa.com/jquery.min.js. Đây là dạng tấn công XSS Stored rất phổ biến mà các website hay gặp phải. Nhờ CSP header, **xauxa.com** không nằm trong whitelist nên không cách nào tải và thực thi được, từ đó bảo vệ website và người truy cập vào website.

### 3.2. Lỗ hổng trong project

Theo ZAP đã quét, trang web không chứa CSP trên header, điều này có thể dẫn đến khả năng cao trang web bị tấn công bởi các phương thức như XSS.

Khi truy cập vào 1 request bất kỳ và bắt request đó để kiểm tra bằng ZAP, ta thấy trong request trả về không hề có cấu hình CSP:



Điều này có thể lý giải khi trong file cấu hình bảo mật, không cấu hình CSP ở header. Xem file tại: <https://github.com/KhaiNguyenDuc/Nhom4_NonFix_Security/blob/main/src/main/java/com/metis/book/security/SecurityConfig.java>

Trong file cấu hình không có cài đặt CSP:

@Bean

SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http

.cors()

.disable()

.formLogin()

.usernameParameter("email")

.loginPage("/auth/login")

.loginProcessingUrl("/auth/login")

.successHandler(myLoginSuccessHandler())

.failureHandler(myLoginFailureHandler())

.and()

.requestCache().requestCache(requestCache())

.and()

.logout()

.logoutUrl("/auth/logout")

.logoutSuccessUrl("/")

.deleteCookies("JSESSIONID","remember-me")

.clearAuthentication(true)

.invalidateHttpSession(true)

.and()

.oauth2Login()

.loginPage("/auth/login")

.successHandler(myOAuthLoginSuccessHandler())

.failureHandler(myOAuthLoginFailureHandler())

.userInfoEndpoint()

.userService(auth2UserService())

.and()

.and()

.rememberMe()

.tokenValiditySeconds(1209600) // 14 days

.and()

.authorizeRequests()

.antMatchers(allowURL)

.permitAll()

.antMatchers("/member/\*\*")

.hasAuthority("ROLE\_USER")

.antMatchers("/admin/user/\*\*")

.hasAuthority("ROLE\_ADMIN")

.antMatchers("/admin/\*\*")

.hasAnyAuthority("ROLE\_ADMIN","ROLE\_STAFF")

.anyRequest()

.permitAll();

return http.build();

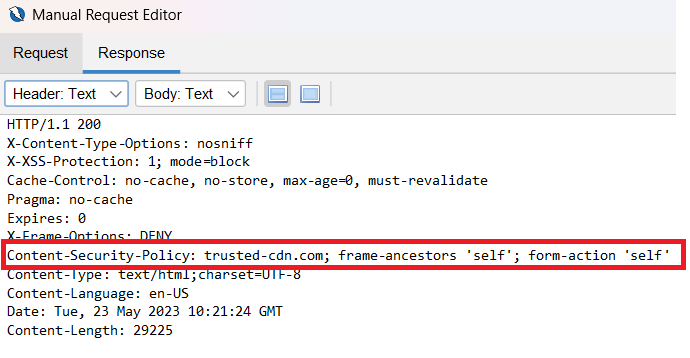
}

### 3.3. Khắc phục

Để tăng tính bảo mật cho trang web, ta sẽ cầu hình CSP ở headers. CSP sẽ tạo 1 whitelist ( bộ lọc các file, url ). Nhằm ngăn chặn các request chứa nội dung độc hại:



Kiểm tra lại bằng cách truy cập vào trang web và bắt request tại ZAP. Ta thấy lúc này ở phần Header đã có CSP:



Tuy nhiên vẫn còn tồn tại CSP style-src và script-src unsafe inline, vì nếu thêm vào policy này sẽ phá hỏng giao diện của trang web

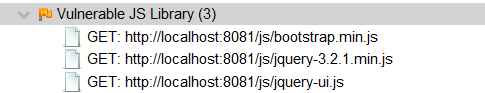
## Vulnerable Library

### Khái niệm

Vulnerabe library là các thành phần thư viện chứa các lỗi bảo mật có thể bị tấn công bởi hacker. Ngày nay, các trang web đều sử dụng rất nhiều các gói thư viện được xây dựng sẵn từ các bên thứ 3. Nếu lập trình viên không chú ý kỹ khi sử dụng các thư viện này. Rất có thể các thư viện chứa các lỗ hổng bảo mật chưa được phát hiện hoặc được phát hiện và được vá ở các phiên bản tiếp theo. Nhưng trang web vẫn sử dụng các gói thư viện cũ và dẫn đến các lỗi không đáng có.

### Lỗ hổng trong project

ZAP đã quét được 1 số thư viện chứa lỗ hổng bảo mật trong trang web như:



Đầu tiên là thư viện *bootstrap.min.js phiên bản 4.1.1*. Tại phiên bản này, chứa lỗ hổng có thể bị tấn công XSS thông qua các plugin: tooltip, collapse, scrollspy.

Link tham khảo: <https://security.snyk.io/package/npm/bootstrap/4.1.1>

Tiếp theo là thư viện *jquery-3.2.1.min.js*. Tại phiên bản này, chứa các lỗ hổng có thể bị tấn công như:

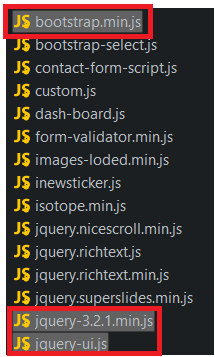
+ Bị tấn công XSS với các phương thức Jquerry DOM như: .html(), .append()

+ Lỗ hổng Prototype Polution, kẻ tấn công có thể sửa đổi Object.prototype

Cuối cùng là thư viện *jquery-ui phiên bản 1.11.4*. Chứa các lỗ hổng như:  
+ Bị tấn công XSS: <https://www.cybersecurity-help.cz/vdb/SB2022072727>

### Khắc phục

Trang web sử dụng các thư viện cũ, lỗi thời, chứa các lỗ hổng đã được thông báo và sửa lỗi ở các phiên bản tiếp theo. Vì vậy, việc cập nhật phiên bản thư viện là cần thiết đối với 3 thư viện này.



Tải Boostrap phiên bản mới nhất tại: <https://getbootstrap.com/docs/5.0/getting-started/download/>

Tải Jquerry phiên bản mới nhất tại: <https://code.jquery.com/jquery-3.7.0.min.js>

Tải Jquerry-ui phiển bản mới nhất tại: <https://jqueryui.com/>

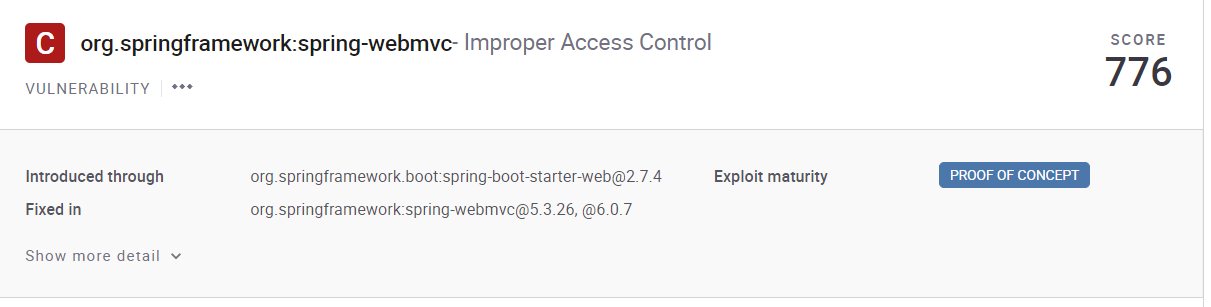
# Phần 4: Quét bằng Snyk

## Giới thiệu

Snyk là một nền tảng bảo mật mã nguồn mở tự động được cấp phép theo giấy phép Apache.Snyk phát hiện các lỗ hổng và tăng tốc sửa chữa trong suốt quá trình phát triển.

Snyk có một loạt các tính năng tuyệt vời giúp biến bảo mật thành một phần của quá trình phát triển như khả năng phát hiện các lỗ hổng từ bên trong IDE. Snyk cũng cung cấp một cổng bảo mật để ngăn các lỗ hổng mới đi qua quy trình xây dụng và môi trường sản phẩm để kiểm tra môi trường chạy nhằm đảm bảo không mắc phải các lỗ hổng hiện có.

Lỗ hổng quét được:



## Improper Access Control

### Khái niệm

Improper Access Control là lỗ hổng xảy ra khi hệ thống không được cấu hình đúng cách để ngăn chặn những truy cập trái phép vào tài nguyên hệ thống như files, thư mục, hình ảnh, các phương thức trong mã nguồn,…

Bao gồm:

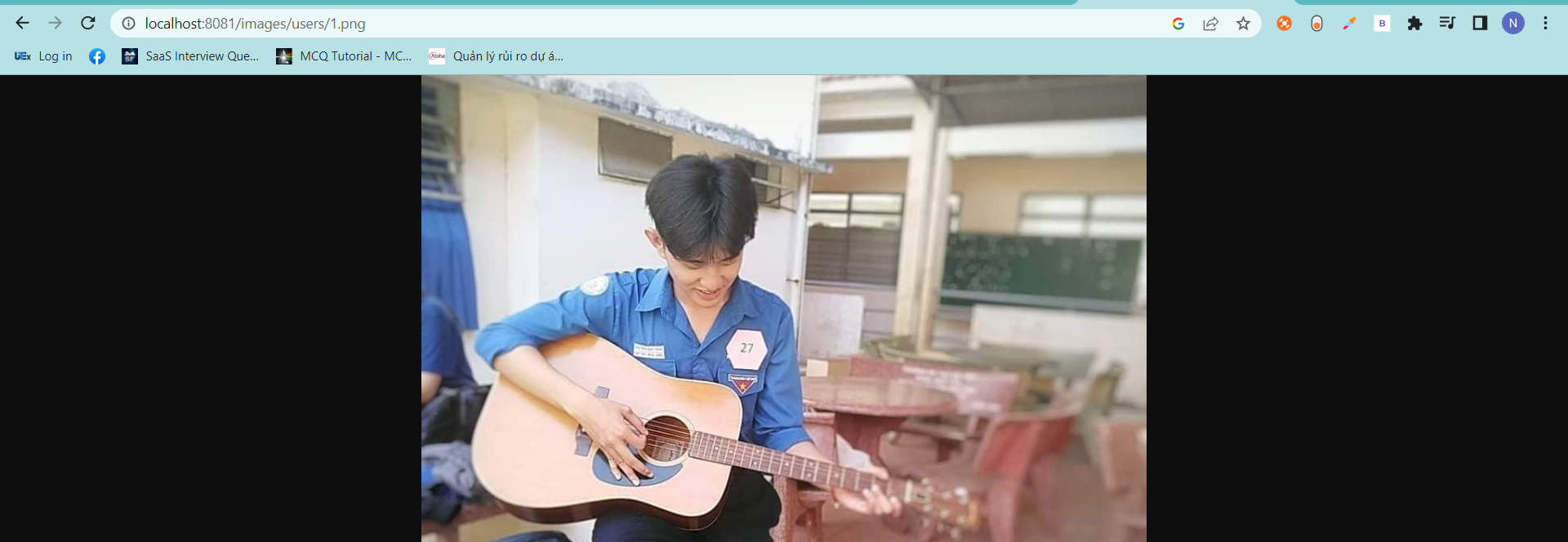
*Week access control*: Khi kiểm soát truy cập còn yếu và dễ bị vượt qua. Hacker có thể dễ dàng truy cập vào nhưng thông tin nhạy cảm của hệ thống

*Insufficient authorization checks*: Khi không xác thực đúng cách hoặc thiếu đi bước xác thực người dùng. Hệ thống có thể sẽ dễ dàng bị truy cập trái phép vào các thông tin nhạy cảm, hoặc thực hiện các hành động trái phép đối với hệ thống

### Tấn công project

Dễ nhận thấy dự án được thực hiện bằng công nghệ Spring Boot framework với Thymeleaf là frontend thông qua quá trình khai thác. Cấu trúc thường thấy của Spring Boot là các file tài nguyên hệ thống sẽ nằm trong thư mục *src/main/resource/static.*

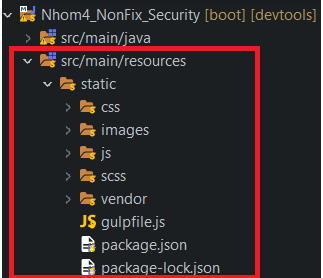
Nếu hệ thống không được kiểm soát truy cập đúng cách, ta có thể truy cập trực tiếp các file tài nguyên thông qua đường dẫn. Tiến hành kiểm thử bằng cách truy cập vào đường dẫn <http://localhost:8081/images/users/1.png> :



Ta thấy, hình ảnh profile của người dùng admin được lấy trực tiếp từ hệ thống. Ngoài file trên, trang web còn có thể bị khai thác rất nhiều tài nguyên nhạy cảm khác.

### Nguyên nhân

Trong project, nhóm thực hiển bỏ các file static như hình ảnh, các thư viện javascript như bootstrap, Jquery và đặt biệt là hình ảnh được upload ở trang web vào thư mục src/main/resource/static:

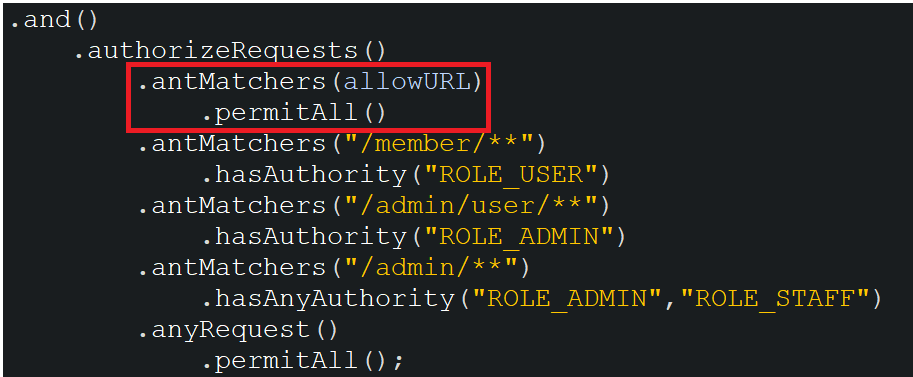


Đây là các file thư viện quan trọng trong hệ thống, nó thể hiện phiên bản của thư viện hệ thống sử dụng, đặc biệt trong project này, ở thư mục images còn chứa hình ảnh được upload trực tiếp từ phía trang web ( đây là đều không nên khi lập trình ).

Trong quá trình cấu hình, tại file cài đặt bảo mật, nhóm thực hiện tạo một allowURL. Các đường dẫn được đặt trong danh sách này sẽ được truy cập mà không cần xác thực. Lý do là vì khi lập trình frontend, cần cho phép truy cập vào những tài nguyên này để dễ dàng kiểm thử:



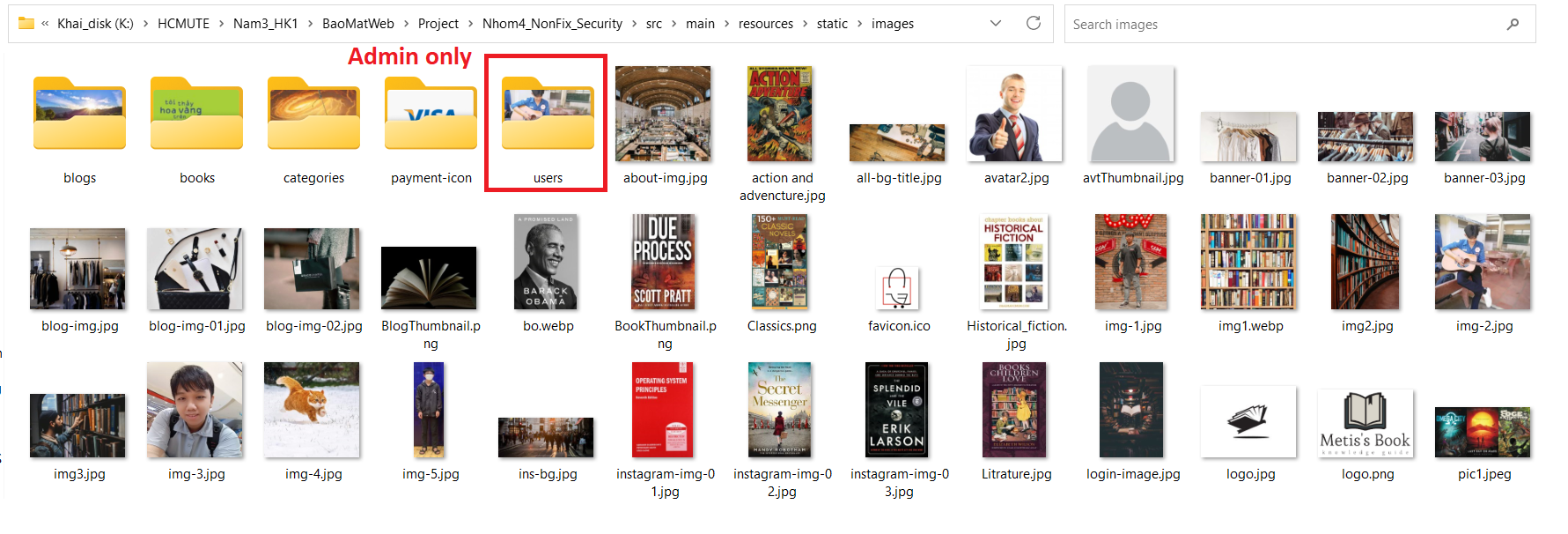
Dễ nhận thấy, trong allowURL chứa đường dẫn đến thư mục /static/\*\*. Dấu [\*\*] tương trưng cho 2 cấp thư mục kế tiếp tính từ static. Ví dụ /static/images/users. Đều này làm cho tài nguyên của hệ thống sẽ bị truy cập trái phép từ bất kỳ nơi đâu và không cần thông qua bất kỳ sự xác thực nào. Có thể thấy allowURL được cấu hình để truy cập mà không cần xác thực:



Vì vậy, nếu kẻ tấn công có kiến thức về cấu trúc thư mục hoặc sử dụng các công cụ để vét cạn. Rất dễ để tìm ra các tài nguyên nhạy cảm.

### Khắc phục

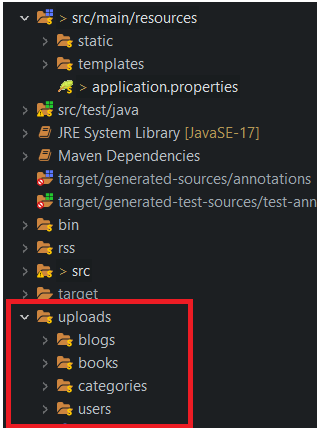
*Đầu tiên*, không nên đặt tài nguyên có mức độ kiểm soát truy cập khác nhau trong cùng 1 thư mục. Ví dụ trong thư mục images có 2 mức độ kiểm soát truy cập cho user ( những file js, css, scss ) và cho admin ( hình được upload từ trang web ):



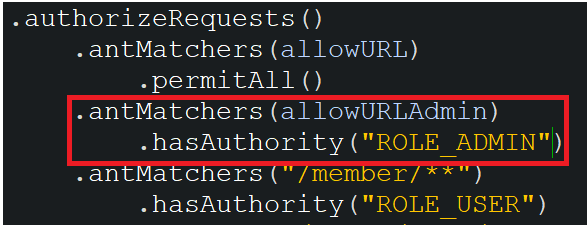
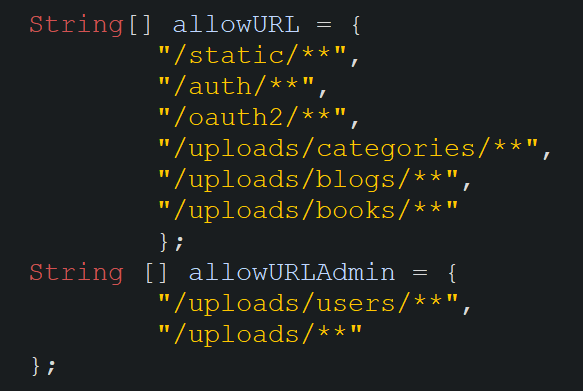
Vì khi kiểm soát truy cập ở cấp độ thư mục, toàn bộ thư mục sẽ bị phụ thuộc vào quyền mà ta cấu hình. Đều này dẫn đến 1 số file hình ảnh không cần xác thực bị yêu cầu xác thực:



Vì vậy, ta sẽ tiến hành tách ra thư mục upload riêng, không nằm ở static nữa. Điều này cũng góp phần giúp server reload lại hình ảnh khi ta tiến hành update hình ảnh mới:

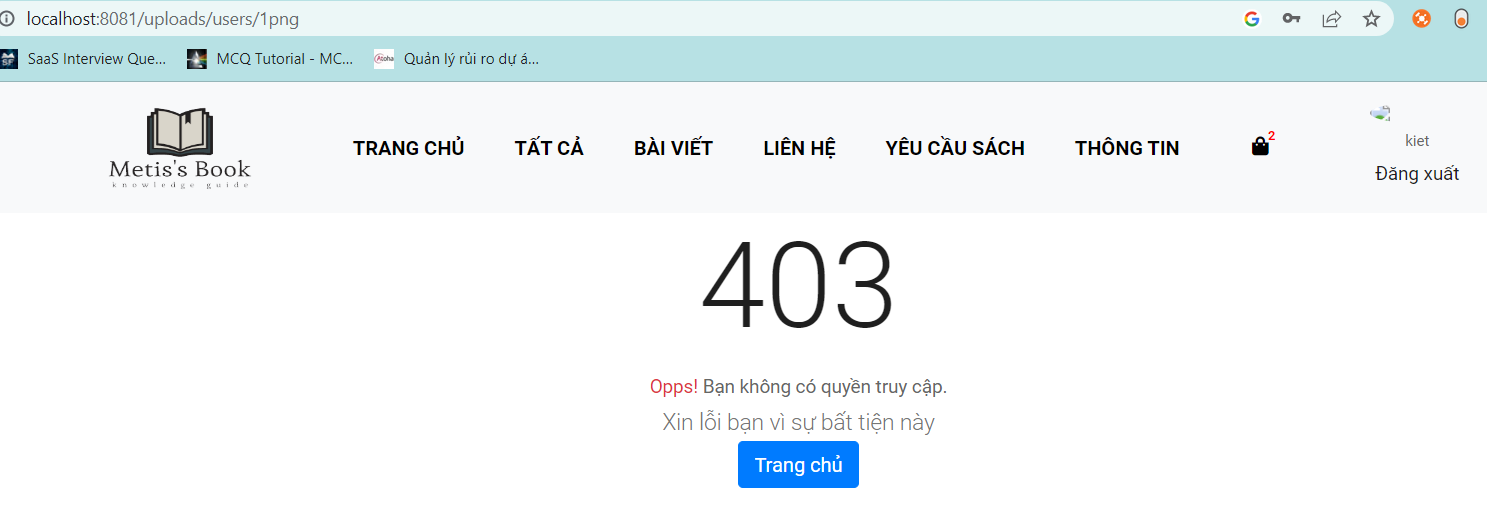


*Tiếp theo,* là không để các đường dẫn đến tài nguyên hệ thống quan trọng trong allowURL. Mà muốn truy cập cần phải thông qua cơ chế xác thực quyền admin mới có thể truy cập thông qua allowURLAdmin:.

 Trao truy cập allowURLAdmin cho admin

Trong đó các thông tin nhạy cảm trong thư mục hình ảnh của users sẽ chỉ được truy cập bởi admin.

Như vậy khi có 1 request truy cập đến tài nguyên hệ thống quan trọng, nếu request đó không được xác minh là admin sẽ bị chặn lại và quăng lỗi 403 ( Forbidden ).



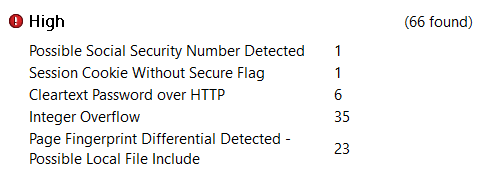
# Phần 5: Quét bằng Vega

## Giới thiệu

Vega là một platform để kiểm tra tính bảo mật của một ứng dụng web applications. Vega được viết bằng Java, và chạy trên nhiều nền tảng khác nhau như Linux, OS X, Windows, được phát triển bởi Subgraph.

Vega có thể hoạt động như một proxy, hoặc như một ứng dụng quét tự động. Các modules có thể tự động gửi các requests để kiểm tra các lỗ hổng như XSS, SQL injection...

Các lỗ hổng vega quét được:



Trong đó, lỗ hổng Integer Overflow và Session Cookie Without Secure Flag là có thể khai thác được

## Integer Overflow

### Khái niệm

Overflow hay còn gọi là lỗi tràn, một thuật ngữ trong ngành Công nghệ thông tin. Trong máy tính, lỗi tràn bộ nhớ xảy ra khi chương trình nhận được một số, giá trị hoặc biến nằm ngoài phạm vi xử lý. Đây là lỗi khá phổ biến trong các chương trình, đặc biệt là khi sử dụng số nguyên, hay các loại số khác của các biến.

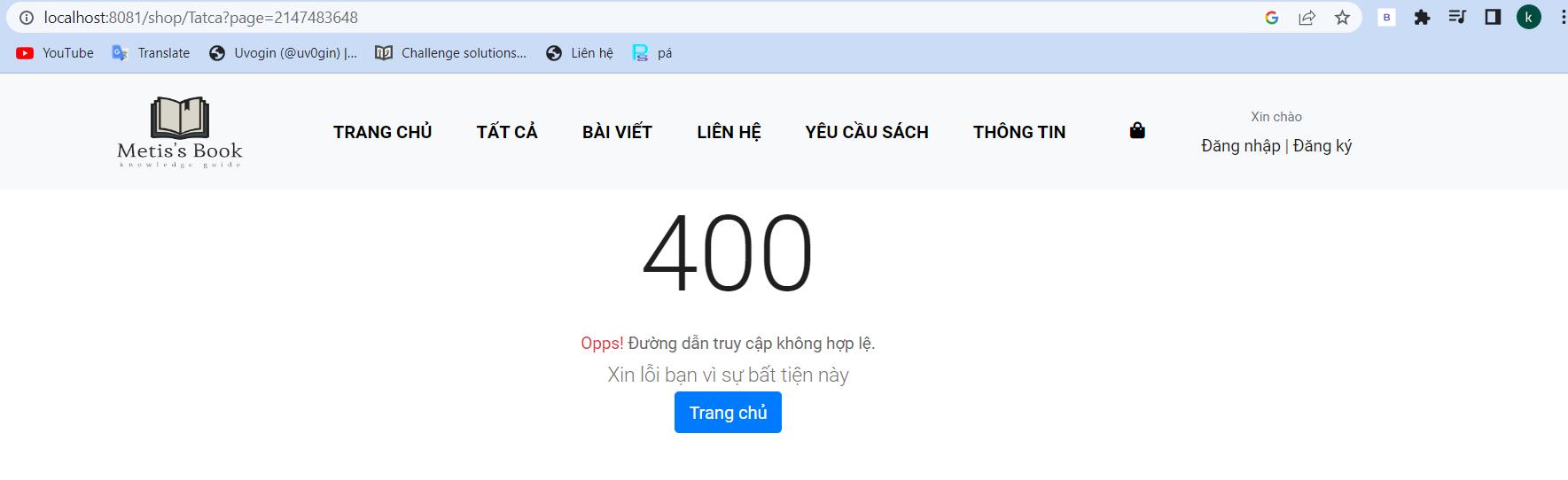
Lỗi tràn số nguyên ( Integer Overflow ): Là lỗi xảy ra khi một số nguyên được lưu trữ trong một biến có giới hạn kích thước và vượt quá giới hạn này, dẫn đến mất mát dữ liệu hoặc sự sai lệch giá trị.

Theo như vega quét được thì việc thêm tham số đầu vào cho các URL có phân trang quá lớn sẽ dẫn đến tràn bộ đệm số nguyên.



### Tấn công project – False positive

Truy cập vào URL <http://localhost:8081/shop/Tatca?page=2147483648>



Ta thấy với tham số đầu vào là 2147483648, trang web phản hồi về lỗi

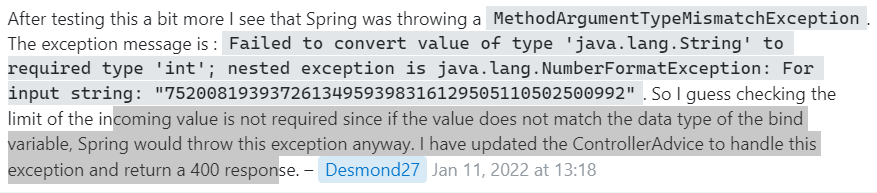
### Nguyên nhân

Được biết project nhóm sử dụng java, và với java, phạm vi của kiểu dữ liệu Integer là:



Như vậy nếu ta nhập input đầu vào lớn hơn giá trị số nguyên cho phép sẽ dẫn đến lỗi tràn bộ nhớ đệm.

Tuy nhiên, sau khi kiểm tra source code, ta nhận thấy vega chỉ đang báo động giả. Việc kiểm tra buffer overflow đã được spring sử dụng có các input đầu vào: <https://stackoverflow.com/questions/70664159/integer-overflow-attack-mitigation-in-spring-rest-api>:



Và ta cũng không cần thiết phải giải quyết lỗi này. Chỉ cần handle exception đầy đủ.

## Session Cookie Without Secure Flag

### Khái niệm

Để bảo mật thông tin cookie và quá trình truyền đi giữa server và client. Cookies sử dụng HttpOnly để bảo về cookie khỏi việc truy cập trái phép từ browser. Chỉ lưu và gửi kèm cookie phản hồi từ client tới server. Việc hạn chế sự can thiệp từ trình duyệt giúp hạn chế rủi ro từ các cuộc tấn công đánh cắp cookie.

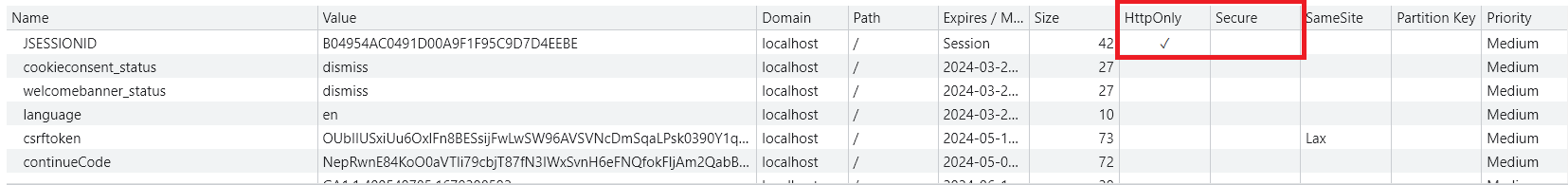
Nếu việc bảo mật ở phía client và browser tốt, nhưng lại sử dụng một phương thức truyền tin kém (HTTP) thì cookie chưa tới được tay người sử dụng đã bị mất (kết hợp Man in the Middle và Session Hijacking).

Secure flag có tác dụng làm cho trình duyệt phải sử dụng kết nối secure/encrypted tức là kết nối bảo mật, được mã hóa

### Lỗ hổng trong project

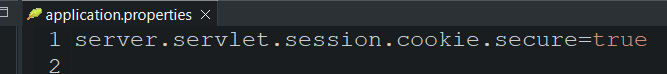
Sau khi đăng nhập, tiến hành kiểm tra cờ HttpOnly và Secure thông qua F12

Ta thấy ở phần secure, cờ secure không hề được bật lên, đều này làm giảm đáng kể sự bảo mật cookies trong quá trình truyền giữa server và client



### Khắc phục

Để bật secure flag, tiến hành cấu hình trong file application.properties:



Sau khi chạy lại chương trình, ta thấy cờ secureFlag đã được bật:



# Phần 6: Các lỗ hổng không tìm thấy

## SQL Injection

### Khái niệm

SQL Injection là một kỹ thuật lợi dụng những lỗ hổng về câu truy vấn của các ứng dụng. Được thực hiện bằng cách chèn thêm một đoạn SQL để làm sai lệnh đi câu truy vấn ban đầu, từ đó có thể khai thác dữ liệu từ database.

Thay vì nhập dữ liệu đúng, kẻ tấn công lợi dụng lỗ hổng để insert và thực thi các câu lệnh SQL bất hợp pháp để lấy dữ liệu của người dùng… SQL Injection được thực hiện với ngôn ngữ lập trình SQL. SQL (Structured Query Language) được sử dụng để quản lý dữ liệu được lưu trữ trong toàn bộ cơ sở dữ liệu.

SQL injection có thể gây ra những tác hại như:

* Hack tài khoản cá nhân.
* Ăn cắp hoặc sao chép dữ liệu của trang web hoặc hệ thống.
* Thay đổi dữ liệu nhạy cảm của hệ thống.
* Xóa dữ liệu nhạy cảm và quan trọng của hệ thống.
* Người dùng có thể đăng nhập vào ứng dụng với tư cách người dùng khác, ngay cả với tư cách quản trị viên.

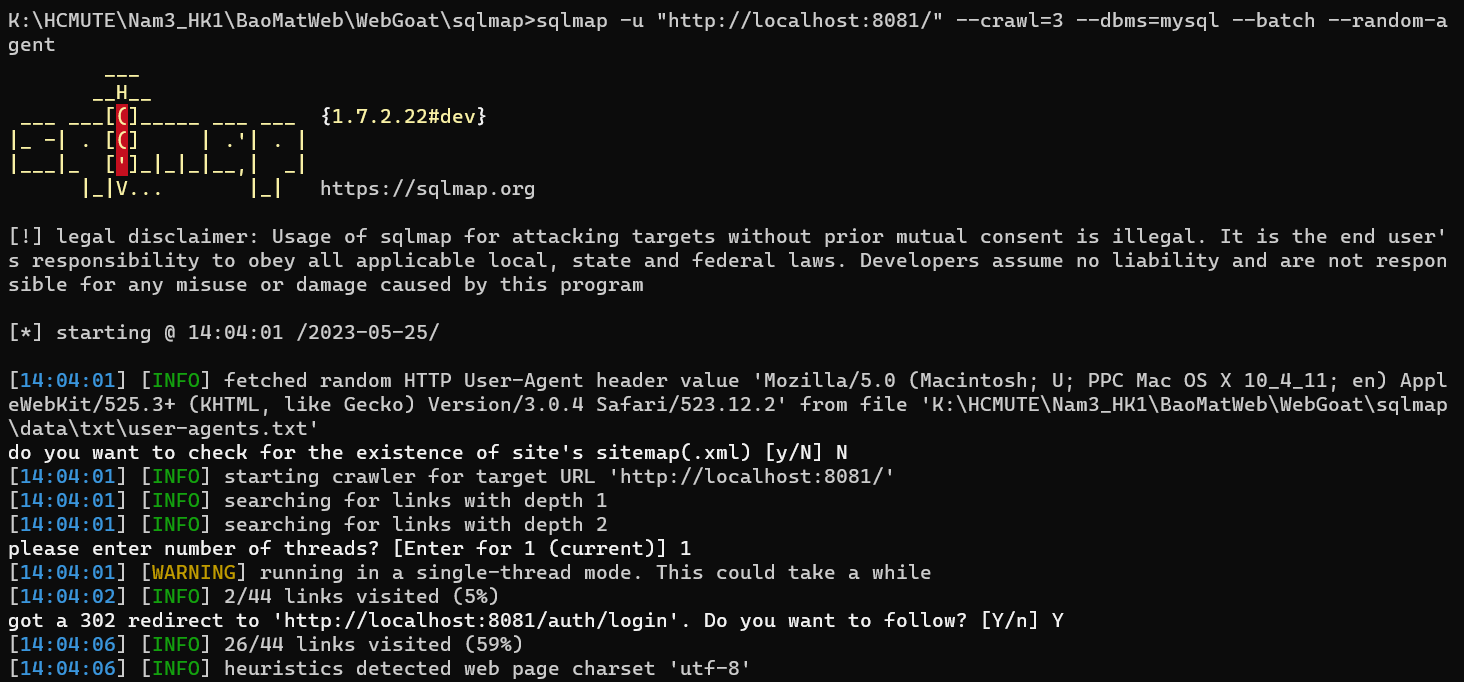
### Quét bằng SQLMap

SQLMAP là công cụ khai thác những lổ hỏng của cơ sở dữ liệu SQL. Công cụ này được xem là công cụ khai thác SQL tốt nhất hiện nay. Được giới bảo mật và giới hacker sử dụng thường xuyên.

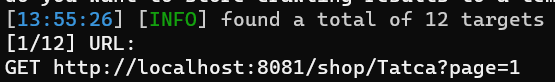
Đây là công cụ mã nguồn mở, tự động hóa quá trình phát hiện và khai thác lỗ hổng SQL. Nó đi kèm với một công cụ phát hiện mạnh mẽ, nhiều tính năng thích hợp cho trình kiểm tra thâm nhập cuối cùng

Sử dụng câu lệnh để tiến hành quét tất cả URL có trên đường dẫn <http://localhost:8081/> bằng SQLMap:

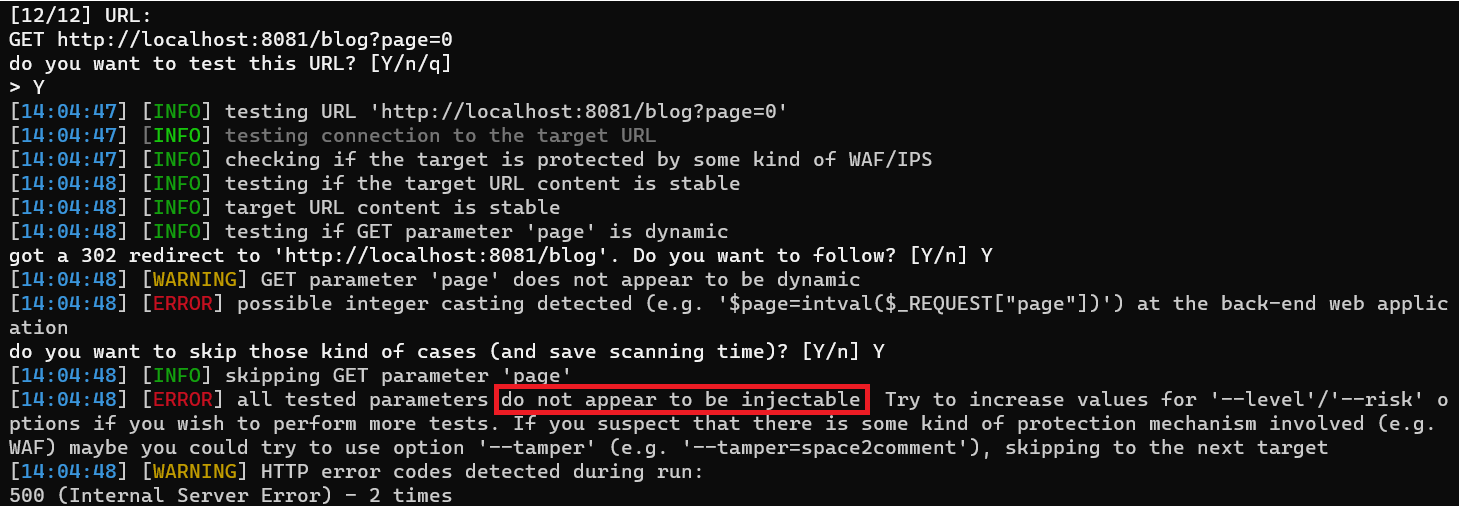
sqlmap -u "http://localhost:8081/" --crawl=3 --dbms=mysql --batch --random-agent



Tìm thấy 12 URL có thể khai thác:



Tuy nhiên, Sqlmap không tìm thấy bất kỳ lỗ hổng Sqlinjection nào. Vì vậy trang web không có khả năng mắc lỗi này.



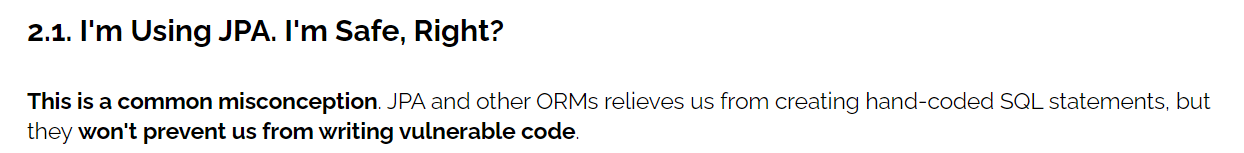
### 2.2. Sự bảo vệ của Spring Framework

Project nhóm sử dụng công nghệ Spring data JPA làm nền tảng đề truy vấn dữ liệu trong cơ sở dữ liệu

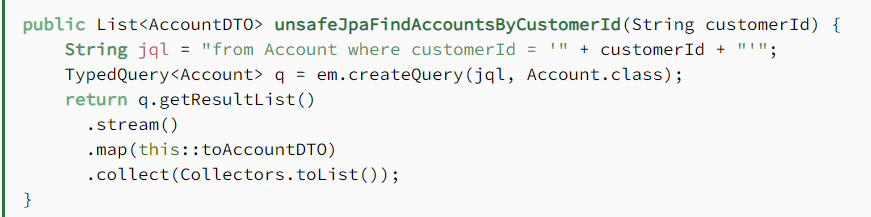
Spring Data JPA giúp ngăn chặn SQL injection thông qua việc sử dụng câu truy vấn được xây dựng sẵn như findAll(), findById(), findByName(),... Những thư viện này mặc định đã được cài đặt để ngăn chặn SQl injection. Trong project sử dụng những hàm mặc định này, do đó không có khả năng bị tấn công SQl injection.

Tuy nhiên, JPA không hoàn toàn có thể ngăn chặn Sql injection nếu ta sử dụng các hàm thủ công

Theo document: <https://www.baeldung.com/sql-injection> . JPA có thể ngăn chặn SQL injection thông qua các hàm có sẵn. Tuy nhiên nếu người dùng sử dụng các hàm thủ công thì có thể sẽ bị tấn công SQL injection nếu không có biện pháp phòng chống:



Một ví dụ cho việc hàm truy vấn dữ liệu có lỗ hổng SQL injection:



Tuy nhiên, trong project nhóm không sử dụng bất kỳ hàm nào tương tự. Toàn bộ đều sử dụng các hàm được xây dựng và kiểm thử bởi Spring framework. Vì vậy sẽ không xuất hiện lỗi như trên.

Dưới đây toàn bộ đều là phương thức có sẵn của JPA:



Vì vậy khả năng xảy ra SQL injection là không có.

## Cross-Site Scripting (XSS)

### Khái niệm

Cross Site Scripting (XSS) là một trong những tấn công phổ biến và dễ bị tấn công nhất Nó được coi là một trong những tấn công nguy hiểm nhất đối với các ứng dụng web và có thể mang lại những hậu quả nghiêm trọng.

Hacker sẽ chèn mã độc thông qua các đoạn script để thực thi chúng ở phía Client. Thông thường, các cuộc tấn công XSS được sử dụng để vượt qua truy cập và mạo danh người dùng.

Mục đích chính của cuộc tấn công này là ăn cắp dữ liệu nhận dạng của người dùng như: cookies, session tokens và các thông tin khác. Trong hầu hết các trường hợp, cuộc tấn công này đang được sử dụng để ăn cắp cookie của người khác. Như chúng ta biết, cookie giúp chúng tôi đăng nhập tự động. Do đó với cookie bị đánh cắp, chúng tôi có thể đăng nhập bằng các thông tin nhận dạng khác. Và đây là một trong những lý do, tại sao cuộc tấn công này được coi là một trong những cuộc tấn công nguy hiểm nhất.

### Sự bảo vệ của Thymeleaf

Project nhóm sử dụng Thymeleaf làm giao diện người dùng. Thymeleaf là một Java template engine dùng để xử lý và tạo HTML, XML, Javascript, CSS và text.

Mục tiêu chính của thymeleaf là mang lại các template tự nhiên, đồng nhất, đơn giản (nature templates) cho công việc phát triển.

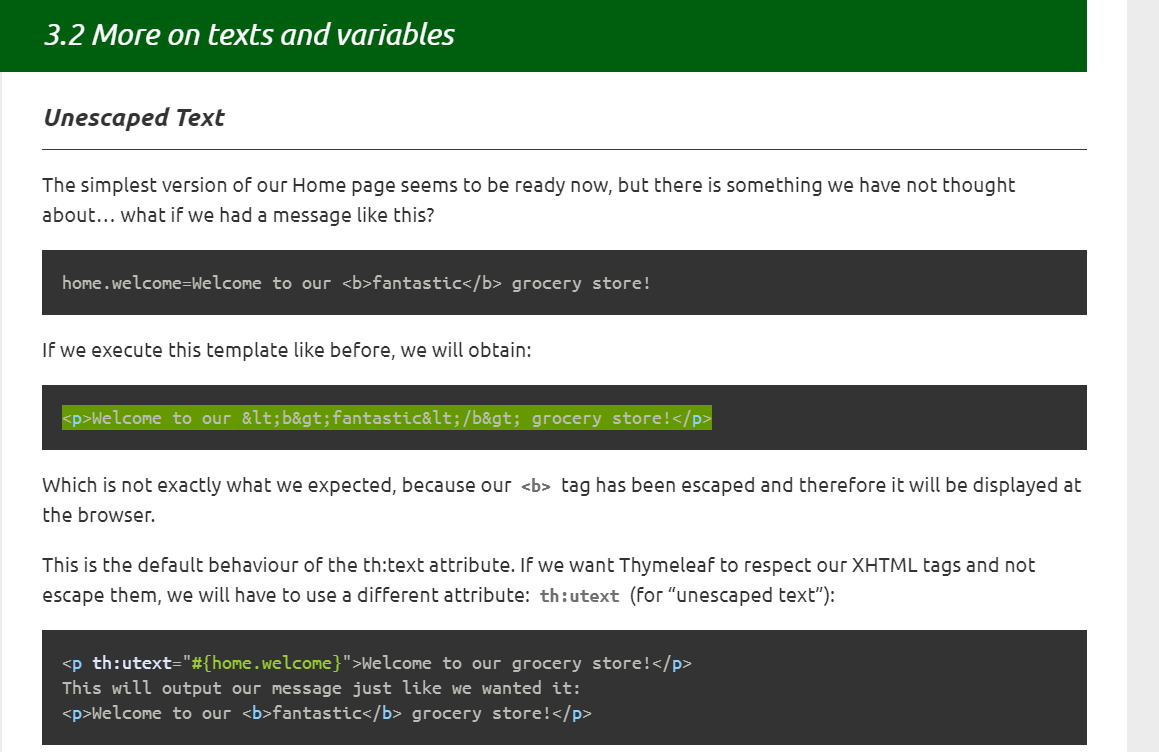
Thymeleaf sử dụng bộ escape text thông qua keyword “th:text”. Bất kỳ dữ thiệu nào được nhập vào ô này sẽ tự động bị encode.

Ví dụ khi người dùng nhập vào

Welcome to our <b>fantastic</b> grocery store!

Trang web sẽ hiển thị:

<p>Welcome to our &lt;b&gt;fantastic&lt;/b&gt; grocery store!</p>

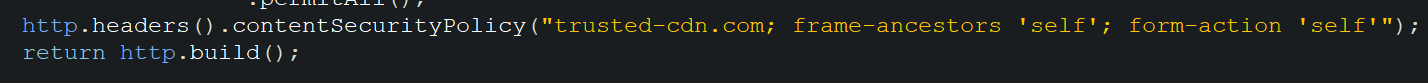


Link tài liệu : <https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html#:~:text=%3C/html%3E-,3.2%20More%20on%20texts%20and%20variables,-Unescaped%20Text>

Như vậy khi bất kỳ dữ liệu nào được nhập vào chứa từ khóa của XSS. Sẽ tự động bị encode và không thể thực thi được nữa

### Cải thiện

Ngoài cơ chế “escaped” mặc định của thymeleaf. Nhóm thực hiện cũng đã sử cụng CSP ( content security policy ) ở phần 2 mục 3. CSP được bổ sung trong file SecurityConfig.java như sau:



Việc này sẽ tạo 1 allowList và chỉ cho phép các đối tượng nằm trong chính sách được thực thi.

Ngoài ra, ta còn có thể cải thiện thông qua cấu hình ở trên như sau:



Việc cấu hình .xssProtection() ở header. Spring security sẽ tự động ngăn chặn các request có từ khóa trong 1 cuộc tấn công XSS