**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC VÀ KỸ THUẬT THÔNG TIN**

---------

**BÁO CÁO ĐỒ ÁN**

**TRIỂN KHAI CÁC VÍ DỤ CHO LỖ HỔNG BẢO MẬT WEB THEO OWASP (A01-A05)**

**Sinh viên thực hiện:**

22521227 – Võ Minh Quyền

22521192 – Võ Anh Quân

22521329 – Nguyễn Cao Thắng

**Giảng viên:**

Tiến sĩ Nguyễn Tấn Cầm

Thành phố Hồ Chí Minh, tháng 12 năm 2024

**MỤC LỤC**

[DANH MỤC HÌNH ẢNH 1](#_Toc184331351)

[CHƯƠNG 1: GIỚI THIỆU 2](#_Toc184331352)

[CHƯƠNG 2: LÝ THUYẾT VÀ PHÂN TÍCH 3](#_Toc184331353)

[1. Về OWASP Top 10. 3](#_Toc184331354)

[2. Về OWASP A01-A05. 3](#_Toc184331355)

[CHƯƠNG 3: TRIỂN KHAI ỨNG DỤNG WEB 5](#_Toc184331356)

[1. Tổng quan về ứng dụng 5](#_Toc184331357)

[2. Triển khai các lỗ hổng bảo mật trên ứng dụng. 5](#_Toc184331358)

[2.1. A01:Broken Access Control. 5](#_Toc184331359)

[2.2. A02:2021-Cryptographic Failures 8](#_Toc184331360)

[2.3. A03:Injection 10](#_Toc184331361)

[2.4. A04:Insecure Design 14](#_Toc184331362)

[2.5. A05:Security Misconfiguration 15](#_Toc184331363)

[3. Chatbot hỗ trợ người dùng. 17](#_Toc184331364)

[CHƯƠNG 4: KẾT LUẬN 19](#_Toc184331365)

[TÀI LIỆU THAM KHẢO 20](#_Toc184331366)

# DANH MỤC HÌNH ẢNH

[Hình 1. Trang chủ của ứng dụng 5](#_Toc184331263)

[Hình 2. Mã triển khai lỗ hổng bảo mật của Lab 1 - A01 6](#_Toc184331264)

[Hình 3. Mã triển khai lỗ hổng bảo mật của Lab 2 - A01 7](#_Toc184331265)

[Hình 4. Mã triển khai lỗ hổng bảo mật của Lab 3 - A01 8](#_Toc184331266)

[Hình 5. Mã triển khai lỗ hổng bảo mật của Lab 1 - A02 8](#_Toc184331267)

[Hình 6. Mã triển khai lỗ hổng bảo mật của Lab 2 - A02 9](#_Toc184331268)

[Hình 7. Mã triển khai lỗ hổng bảo mật của Lab 3 - A02 10](#_Toc184331269)

[Hình 8. Mã triển khai lỗ hổng bảo mật của SQL Injection - A03 11](#_Toc184331270)

[Hình 9. Mã triển khai lỗ hổng bảo mật của Lab 1 Command Injection - A03 12](#_Toc184331271)

[Hình 10. Mã triển khai lỗ hổng bảo mật của Lab 2 Command Injection - A03 13](#_Toc184331272)

[Hình 11. Mã triển khai lỗ hổng bảo mật của Template Injection 14](#_Toc184331273)

[Hình 12. Mã triển khai lỗ hổng bảo mật của Insecure Design 15](#_Toc184331274)

[Hình 13. Mã triển khai lỗ hổng bảo mật của Lab 1 - A05 15](#_Toc184331275)

[Hình 14. Mã triển khai lỗ hổng bảo mật của Lab 2 - A05 16](#_Toc184331276)

[Hình 15. Mã triển khai lỗ hổng bảo mật của Lab 3 - A05 17](#_Toc184331277)

[Hình 16. Chatbot hỗ trợ người dùng 18](#_Toc184331278)

# CHƯƠNG 1: GIỚI THIỆU

Trong đề tài này, nhóm chúng em đã thực hiện triển khai các lỗ hổng bảo mật OWASP A01-A05 (2021) trên một ứng dụng web dễ bị tấn công. Ứng dụng web được triển khai trên nền tảng Django, với cách thức triển khai dựa theo ứng dụng OWASP PyGoat [1].

Trong đồ án này, chúng em chú trọng trong việc xây dựng các bài tập để làm quen với việc nhận diện các lỗ hổng bảo mật web OWASP Top 10 (phiên bản 2021) từ A01 đến A05 bao gồm [2]:

* A01:2021-Broken Access Control;
* A02:2021-Cryptographic Failures;
* A03:2021-Injection (bao gồm SQL Injection, Command Injection, Template Injection);
* A04:2021-Insecure Design;
* A05:2021-Security Misconfiguration;

Ứng dụng web được xây dựng có một số điểm đã được cải tiến khác biệt với PyGoat như: chuyển ngữ các template sang tiếng Việt cũng như giải thích các thuật ngữ, yêu cầu bài tập cho dễ hiểu hơn; tạo một chatbot hỗ trợ cho việc giải các bài tập cũng như hướng dẫn cách xử lý các lỗ hổng; nâng độ khó cho một số bài tập, từ đó giúp người sử dụng ứng dụng có cái nhìn sâu sắc hơn về các lỗ hổng bảo mật có thể vô tình ở trên ứng dụng web.

# CHƯƠNG 2: LÝ THUYẾT VÀ PHÂN TÍCH

## Về OWASP Top 10.

OWASP Top 10 là một tài liệu do OWASP (Open Web Application Security Project) xuất bản, nhằm cung cấp danh sách 10 lỗ hổng bảo mật web phổ biến và nguy hiểm nhất [2]. Đây là tài liệu được các nhà phát triển phần mềm và chuyên gia bảo mật sử dụng rộng rãi để hiểu và giảm thiểu rủi ro trong các ứng dụng web.

Lần cập nhật gần đây nhất của tài liệu này là vào năm 2021, có ba hạng mục mới, bốn hạng mục có thay đổi về tên và phạm vi, và một số hợp nhất trong Top 10 cho năm 2021 so với năm 2017.

## Về OWASP A01-A05.

A01:2021-Broken Access Control: Việc kiểm soát truy cập là hành vi thực thi các chính sách để người dùng không thể thực hiện các tương tác với dữ liệu ngoài các quyền hạn đã được quy định. Tuy nhiên, với một ứng dụng mắc phải lỗ hổng A01 sẽ dẫn đến một số hậu quả như tiết lộ thông tin trái phép, sửa đổi hoặc xóa mọi dữ liệu quan trọng nằm ngoài quyền hạn của người dùng đó, ... [3]

A02:2021-Cryptographic Failures: Lỗi mã hóa là nguyên nhân trực tiếp dẫn đến việc các thông tin nhạy cảm của người dùng bị lộ. Lỗ hổng này cho thấy các điểm yếu trong mã hóa dữ liệu, chủ yếu do việc không mã hóa hoặc thuật toán mã hóa quá dễ để giải mã [4].

A03:2021-Injection: Lỗ hổng này xảy ra khi một ứng dụng không kiểm tra hoặc làm sạch dữ liệu đầu vào đúng cách, dẫn đến việc kẻ tấn công có thể gửi mã độc để thực thi các hành động như đọc, sửa đổi, hoặc xóa dữ liệu nhạy cảm; thực thi mã trên máy chủ hoặc hệ thống của nạn nhân; ...[5]

Có 3 loại lỗ hổng Injection phổ biến:

* Command Injection: Lỗ hổng xảy ra khi một ứng dụng thực thi các lệnh hệ thống (OS command) dựa trên đầu vào của người dùng mà không kiểm tra đầy đủ.
* SQL Injection: Xảy ra khi ứng dụng thực hiện truy vấn SQL bằng cách ghép nối các câu truy vấn lại với nhau mà không sàng lọc. Kẻ tấn công có thể chèn mã SQL độc hại để truy cập hoặc thao túng cơ sở dữ liệu.
* Template Injection: Xảy ra khi kẻ tấn công chèn mã độc vào các template (mẫu giao diện) được ứng dụng sử dụng để hiển thị nội dung, do không kiểm tra đúng cách đầu vào.

A04:2021-Insecure Design: Thiết kế không an toàn thực chất là một lỗ hổng bảo mật không xuất phát từ vấn đề kỹ thuật mà bắt nguồn ngay từ khâu thiết kế, tức là lỗi này chủ yếu bắt nguồn từ sai sót trong cách nghĩ của người thiết kế ứng dụng. Trong thiết kế ban đầu có thể có những sai sót như không xác thực, phân quyền đầy đủ; không che giấu thông tin nhạy cảm; bỏ qua việc kiểm tra các dữ liệu nhận vào; bản thiết kế bị lộ cho kẻ xấu,... Lỗ hổng này không thể khắc phục trong quá trình triển khai, mà phải khắc phục ngay từ khâu nhận thức của đội ngũ thiết kế [6].

A05:2021-Security Misconfiguration: Security Misconfiguration là lỗ hổng xảy ra khi ứng dụng hoặc hệ thống không được cấu hình bảo mật đúng cách. Đây là một trong những lỗ hổng phổ biến và nguy hiểm, bởi một cấu hình sai có thể tạo điều kiện cho kẻ tấn công xâm nhập, khai thác hệ thống hoặc đánh cắp dữ liệu [7].

Các lỗi cấu hình bảo mật thường do các nguyên nhân: cấu hình mặc định chưa được thay đổi; thiếu cập nhật phần mềm, thư viện; cấu hình sai dịch vụ hoặc tính năng bảo mật; quản lý quyền truy cập không phù hợp.

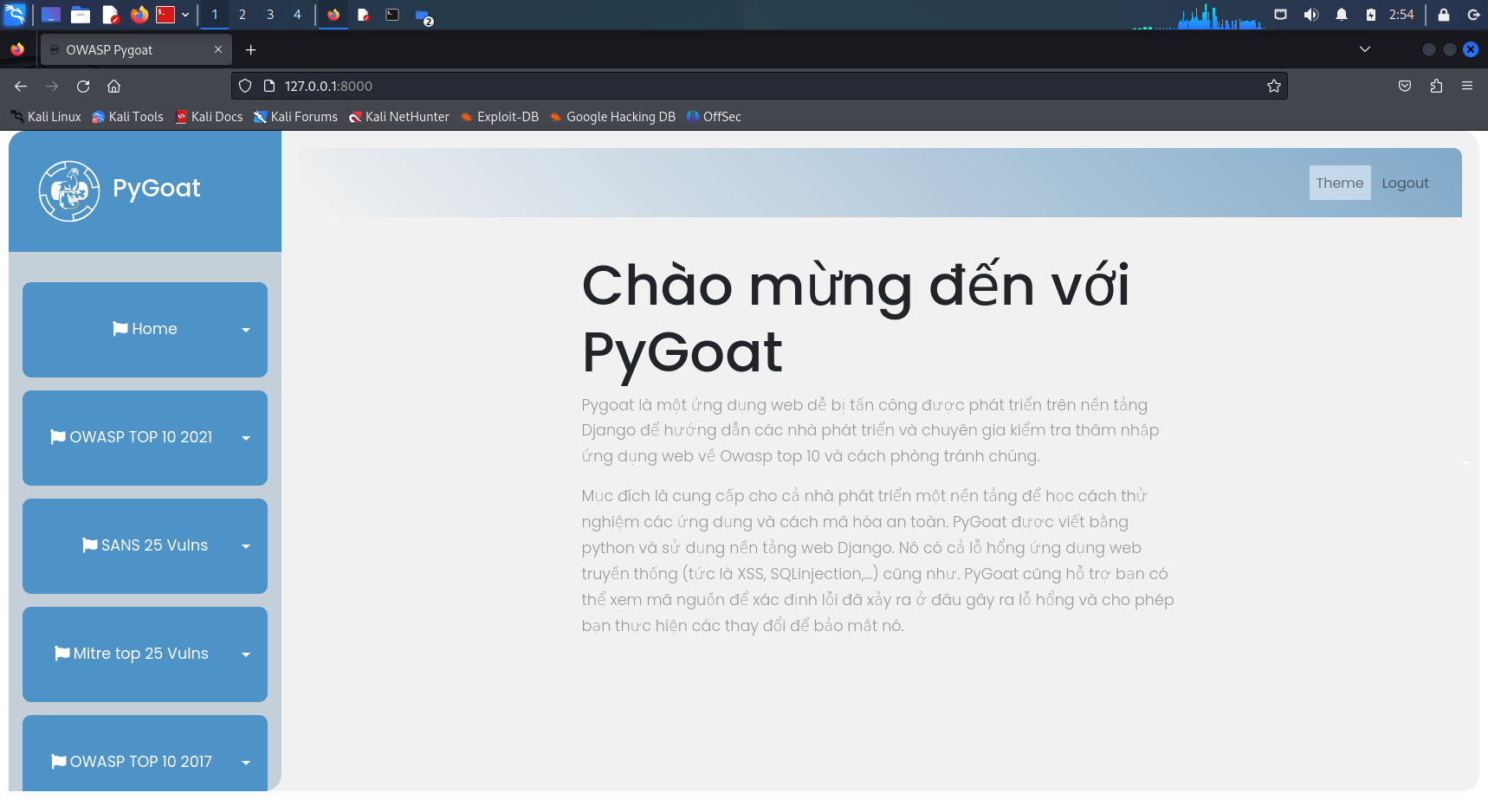
# CHƯƠNG 3: TRIỂN KHAI ỨNG DỤNG WEB

## Tổng quan về ứng dụng

Ứng dụng này được xây dựng để minh họa các lỗ hổng bảo mật OWASP Top 10 A01-A05. Ứng dụng bao gồm nhiều bài tập khác nhau để người dùng có thể làm quen và nhận biết dấu hiệu tồn tại một lỗ hổng bảo mật trên ứng dụng web, được viết dựa trên mã nguồn của PyGoat.

Ứng dụng web được triển khai trên localhost, nên triển khai một môi trường Python ảo trước khi sử dụng để tránh sự xung đột giữa các thư viện cần thiết. Hệ quản trị CSDL được sử dụng là SQLite. Phần fornt-end của ứng dụng được thiết kế bằng DTL nhúng vào các template HTML, CSS và JavaScript. Ứng dụng được viết bằng ngôn ngữ Python, triển khai trên framework Django. Trong ứng dụng cũng sử dụng một chatbot với API từ Gemini, model Gemini 1.5 Flash 8B Latest. Module L10N cũng được ứng dụng để chuyển ngôn ngữ trong các template về tiếng Việt.

Có thể cài đặt ứng dụng này bằng Docker hoặc có thể giải nén file .zip (chúng em đã nộp), thực hiện cài đặt các thư viện cần thiết (nên cài đặt trên môi trường ảo) bằng lệnh ***pip install -r requirements.txt***, sau đó áp dụng các migration bằng lệnh ***python manage.py migrate***, và chạy ứng dụng trên localhost bằng ***python manage.py runserver***, có thể sử dụng ứng dụng tại port 8000 của localhost: [http://127.0.0.1:8000](http://127.0.0.1:8000/).



Hình 1. Trang chủ của ứng dụng

## Triển khai các lỗ hổng bảo mật trên ứng dụng.

### A01:Broken Access Control.

Minh hoạ cho lỗ hổng A01 bao gồm 3 bài lab với độ khó tăng dần:

#### Lab 1

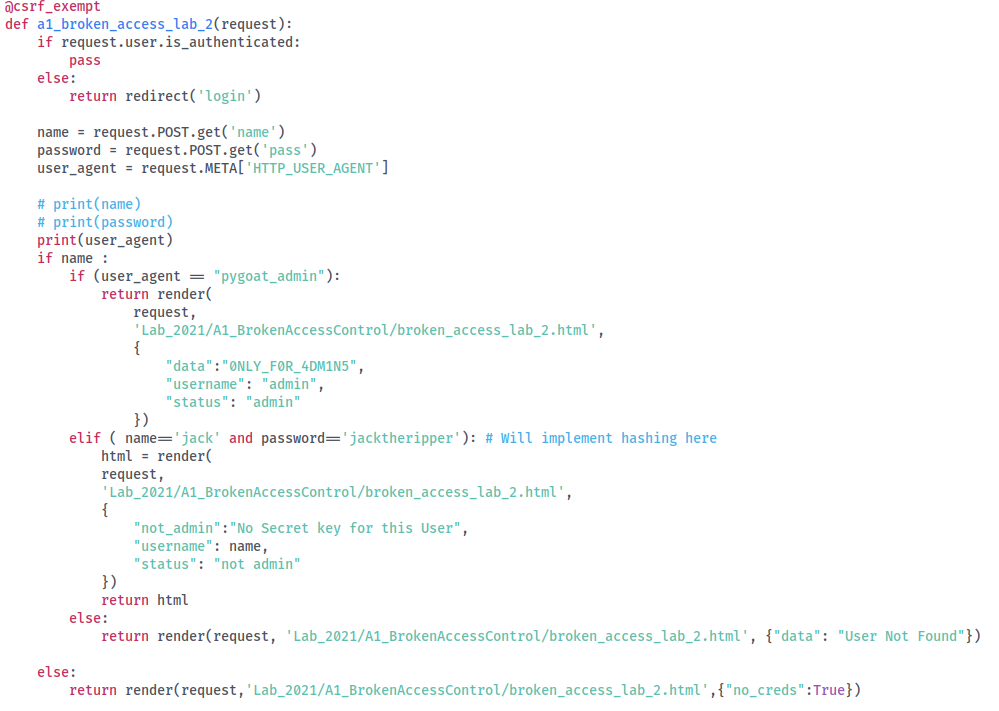
* Khi truy cập bài lab, người dùng được cung cấp một trang đăng nhập đơn giản yêu cầu tên đăng nhập và mật khẩu. Thông tin đăng nhập cho người dùng Jack là jack:jacktheripper. Sử dụng thông tin trên để đăng nhập. Mục tiêu chính của bài lab này là đăng nhập với quyền quản trị để lấy khóa bí mật.
* Mã triển khai lỗ hổng bảo mật: Đoạn mã kiểm tra quyền truy cập của người dùng dựa trên giá trị của cookie admin. Nếu người dùng đã đăng nhập, mã sẽ lấy thông tin name và password từ POST request. Nếu cookie admin có giá trị "1", người dùng được coi là admin và sẽ thấy thông tin dành riêng cho admin. Nếu name là "jack" và password là "jacktheripper", người dùng sẽ được thiết lập cookie admin với giá trị "0" và không có quyền admin. Nếu không, thông báo "User Not Found" sẽ được hiển thị. Lỗ hổng ở đây là người dùng có thể thay đổi giá trị của cookie admin để truy cập vào các chức năng dành cho admin.



Hình 2. Mã triển khai lỗ hổng bảo mật của Lab 1 - A01

#### Lab 2

* Bài lab này giúp chúng ta hiểu một trong những lỗ hổng xác thực dẫn đến việc kẻ tấn công chiếm quyền kiểm soát trái phép một tài khoản. Thông tin đăng nhập cho người dùng Jack là jack:jacktheripper. Sử dụng thông tin trên để đăng nhập. Mục tiêu chính của bài lab này là đăng nhập với quyền quản trị để lấy khóa bí mật.
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng dựa trên giá trị của user\_agent. Nếu người dùng đã đăng nhập, mã sẽ lấy thông tin name, password từ POST request và user\_agent từ HTTP header. Nếu user\_agent là "pygoat\_admin", người dùng được coi là admin và sẽ thấy thông tin dành riêng cho admin. Nếu name là "jack" và password là "jacktheripper", người dùng sẽ không có quyền admin và sẽ thấy thông báo tương ứng. Nếu không, thông báo "User Not Found" sẽ được hiển thị. Lỗ hổng ở đây là việc kiểm tra quyền truy cập dựa trên giá trị của user\_agent, mà giá trị này có thể dễ dàng bị giả mạo bởi người dùng.



Hình 3. Mã triển khai lỗ hổng bảo mật của Lab 2 - A01

#### Lab 3

* Thông tin đăng nhập của quản trị viên: tên người dùng: admin, mật khẩu: admin\_pass và thông tin đăng nhập của người dùng bình thường: tên người dùng: John, mật khẩu: reaper. Bạn có thể truy cập nội dung trang quản trị bằng thông tin người dùng thường hoặc không cần thông tin đăng nhập không?
* Mã triển khai lỗ hổng bảo mật: Hàm a1\_broken\_access\_lab\_3 kiểm tra xem người dùng đã đăng nhập hay chưa và xử lý các yêu cầu GET và POST. Nếu là yêu cầu GET, nó hiển thị trang đăng nhập. Nếu là POST request, nó kiểm tra tên người dùng và mật khẩu. Nếu thông tin đăng nhập là "John" và "reaper", người dùng được xác định là không phải admin. Nếu thông tin đăng nhập là "admin" và "admin\_pass", người dùng được xác định là admin. Hàm a1\_broken\_access\_lab3\_secret hiển thị trang bí mật mà không kiểm tra quyền truy cập, tạo ra lỗ hổng bảo mật vì bất kỳ người dùng đã đăng nhập nào cũng có thể truy cập trang này mà không cần xác thực thêm.

A screenshot of a computer program

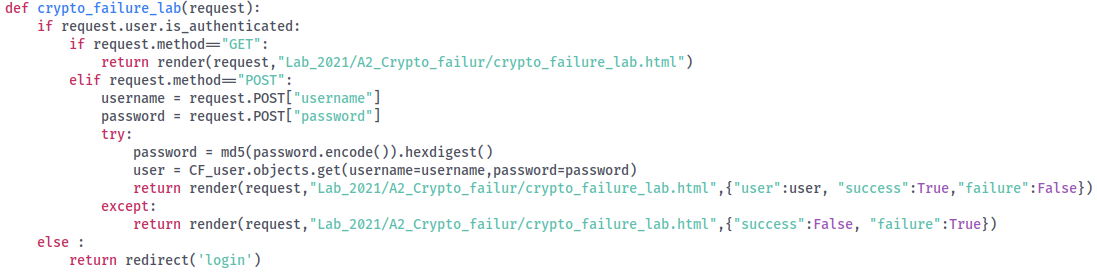
Description automatically generated

Hình 4. Mã triển khai lỗ hổng bảo mật của Lab 3 - A01

### A02:2021-Cryptographic Failures

#### Lab 1

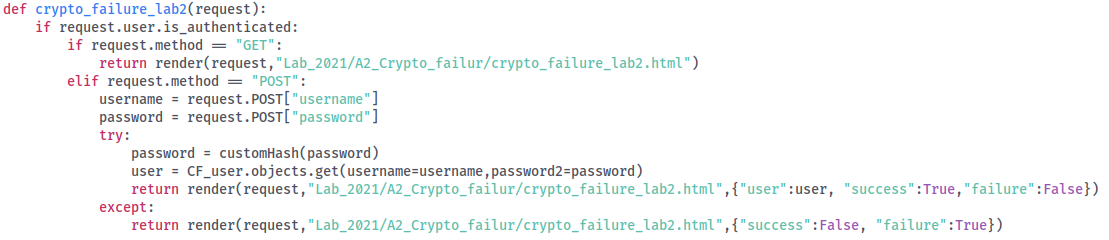
* Bạn có thể đăng nhập với tư cách quản trị viên không? Một hacker trước đây đã thực hiện một cuộc tấn công SQL Injection và lấy được cơ sở dữ liệu của bảng người dùng: alex,9d6edee6ce9312981084bd98eb3751ee; admin,c93ccd78b2076528346216b3b2f701e6; rupak,5ee3547adb4481902349bdd0f2ffba93
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng và xử lý yêu cầu GET và POST cho trang crypto\_failure\_lab. Nếu người dùng đã đăng nhập, yêu cầu GET sẽ hiển thị trang crypto\_failure\_lab.html. Với POST request, mã sẽ lấy username và password từ yêu cầu, mã hóa mật khẩu bằng hàm băm MD5, và kiểm tra thông tin đăng nhập trong cơ sở dữ liệu. Nếu thông tin đúng, trang sẽ hiển thị thông tin người dùng và thông báo thành công. Nếu thông tin sai, trang sẽ hiển thị thông báo thất bại. Nếu người dùng chưa đăng nhập, họ sẽ được chuyển hướng đến trang đăng nhập. Lỗ hổng bảo mật ở đây là việc sử dụng MD5 để mã hóa mật khẩu, một thuật toán không an toàn và dễ bị tấn công.



Hình 5. Mã triển khai lỗ hổng bảo mật của Lab 1 - A02

#### Lab 2

* Bạn có thể đăng nhập với tư cách quản trị viên không? Một hacker trước đây đã thực hiện một cuộc tấn công SQL Injection và lấy được cơ sở dữ liệu của bảng người dùng. Lần này, có vẻ họ đã sử dụng thuật toán băm tốt hơn: alex,2a280ba4ff0f8c763c5b0606f40effc3319dbc4c91d4361a39990292d4b7b0cd; admin,d953b4a47ce307fcb8b1b85fc6a0d34aea5585b6ad9188beb94c1eea9bbb5c7a; rupak,c17cde8d179a37cad4bd93e55355fdf240eb52d585e428c1cdfecc68123e192a
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng và xử lý yêu cầu GET và POST cho trang crypto\_failure\_lab2. Nếu người dùng đã đăng nhập, yêu cầu GET sẽ hiển thị trang crypto\_failure\_lab2.html. Với POST request, mã sẽ lấy username và password từ yêu cầu, mã hóa mật khẩu bằng hàm customHash, và kiểm tra thông tin đăng nhập trong cơ sở dữ liệu. Nếu thông tin đúng, trang sẽ hiển thị thông tin người dùng và thông báo thành công. Nếu thông tin sai, trang sẽ hiển thị thông báo thất bại. Nếu người dùng chưa đăng nhập, họ sẽ được chuyển hướng đến trang đăng nhập. Lỗ hổng bảo mật ở đây là việc sử dụng hàm băm tùy chỉnh customHash, thuật toán này sử dụng SHA256 để mã hóa mật khẩu, sau đó đảo ngược chuỗi đã mã hóa.



Hình 6. Mã triển khai lỗ hổng bảo mật của Lab 2 - A02

#### Lab 3

* Chúng tôi có thông tin đăng nhập người dùng cho trang này, nhưng bạn có thể đăng nhập với tư cách quản trị viên và lấy thông tin bí mật không? username : User; password : P@$$w0rd
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng và xử lý yêu cầu GET và POST cho trang crypto\_failure\_lab3. Nếu người dùng đã đăng nhập, yêu cầu GET sẽ kiểm tra cookie để xác định quyền truy cập. Cookie chứa tên người dùng và thời gian hết hạn. Nếu cookie hết hạn, trang sẽ hiển thị thông báo thất bại. Nếu cookie hợp lệ và tên người dùng là "admin", trang sẽ hiển thị thông tin dành cho admin. Với POST request, mã sẽ lấy username và password từ yêu cầu, kiểm tra thông tin đăng nhập. Nếu thông tin đúng, cookie mới sẽ được tạo với thời gian hết hạn là 60 phút và trang sẽ hiển thị thông báo thành công. Nếu thông tin sai, trang sẽ hiển thị thông báo thất bại. Lỗ hổng bảo mật ở đây là việc sử dụng cookie không an toàn để kiểm soát quyền truy cập, dễ bị tấn công và giả mạo.



Hình 7. Mã triển khai lỗ hổng bảo mật của Lab 3 - A02

### A03:Injection

#### SQL Injection

* Bài lab này giúp bạn khai thác loại lỗ hổng SQL Injection phổ biến, gây ra do thiếu xác thực đầu vào và đưa trực tiếp dữ liệu nhập vào truy vấn. Người dùng khi truy cập vào bài lab sẽ được đưa đến trang đăng nhập. Nhiệm vụ là đăng nhập với vai trò quản trị viên. Lỗ hổng SQL Injection có thể được phát hiện bằng cách chèn một ' vào bất kỳ trường nào. Nếu xuất hiện lỗi SQL, thì lỗ hổng SQL Injection đã được xác định.
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng và xử lý POST request cho trang injection\_sql\_lab. Nếu người dùng đã đăng nhập, mã sẽ lấy name và password từ POST request và tạo một truy vấn SQL để tìm kiếm người dùng trong cơ sở dữ liệu. Mã này cố tình chứa lỗ hổng SQL Injection vì truy vấn SQL được xây dựng bằng cách nối chuỗi trực tiếp từ đầu vào của người dùng. Sau đó, mã lưu một số bản ghi mẫu vào cơ sở dữ liệu và thực thi truy vấn SQL. Nếu truy vấn trả về kết quả, trang sẽ hiển thị thông tin người dùng. Nếu không, trang sẽ hiển thị thông báo lỗi. Nếu người dùng chưa đăng nhập, họ sẽ được chuyển hướng đến trang đăng nhập. Lỗ hổng bảo mật ở đây là việc sử dụng truy vấn SQL không an toàn, dễ bị tấn công SQL Injection, cho phép kẻ tấn công thực thi các lệnh SQL tùy ý.



Hình 8. Mã triển khai lỗ hổng bảo mật của SQL Injection - A03

#### Command Injection – Lab 1

* Khi truy cập lab, người dùng được cung cấp một tính năng tra cứu máy chủ tên miền (name server lookup) trên một domain đã cho. Người dùng cần cung cấp một domain, sau đó server sẽ thực hiện lệnh tra cứu và trả kết quả lại cho client. Nếu bạn đang chạy lab, bạn có thể chọn Windows hoặc Linux dựa trên hệ điều hành của mình.
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng và xử lý yêu cầu POST cho trang cmd\_lab. Nếu người dùng đã đăng nhập, mã sẽ lấy domain và os từ yêu cầu POST. Tên miền sẽ được chuẩn hóa bằng cách loại bỏ tiền tố "https://www.". Dựa trên hệ điều hành (os), mã sẽ tạo lệnh nslookup cho Windows hoặc dig cho các hệ điều hành khác. Sau đó, lệnh sẽ được thực thi bằng cách sử dụng subprocess.Popen, và kết quả đầu ra sẽ được thu thập từ stdout và stderr. Kết quả này sẽ được hiển thị trên trang cmd\_lab.html. Lỗ hổng bảo mật ở đây là việc sử dụng subprocess với tham số shell=True, dễ bị tấn công Command Injection, cho phép kẻ tấn công thực thi các lệnh tùy ý trên hệ thống.



Hình 9. Mã triển khai lỗ hổng bảo mật của Lab 1 Command Injection - A03

#### Command Injection – Lab 2

* Một hộp nhập liệu cho phép người dùng tính toán các biểu thức toán học như 1 + 1 hoặc 5 \* 5. Nhiệm vụ của bạn là khai thác lỗi này để thực thi lệnh trên hệ thống.
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng và xử lý yêu cầu POST cho trang cmd\_lab2. Nếu người dùng đã đăng nhập, mã sẽ lấy giá trị val từ yêu cầu POST và sử dụng hàm eval để thực thi giá trị này như một biểu thức Python. Kết quả của biểu thức sẽ được hiển thị trên trang cmd\_lab2.html. Nếu có lỗi xảy ra trong quá trình thực thi, trang sẽ hiển thị thông báo lỗi. Nếu người dùng chưa đăng nhập, họ sẽ được chuyển hướng đến trang đăng nhập. Lỗ hổng bảo mật ở đây là việc sử dụng hàm eval, dễ bị tấn công Code Injection, cho phép kẻ tấn công thực thi mã Python tùy ý trên máy chủ.

A screen shot of a computer code

Description automatically generated

Hình 10. Mã triển khai lỗ hổng bảo mật của Lab 2 Command Injection - A03

#### Template Injection

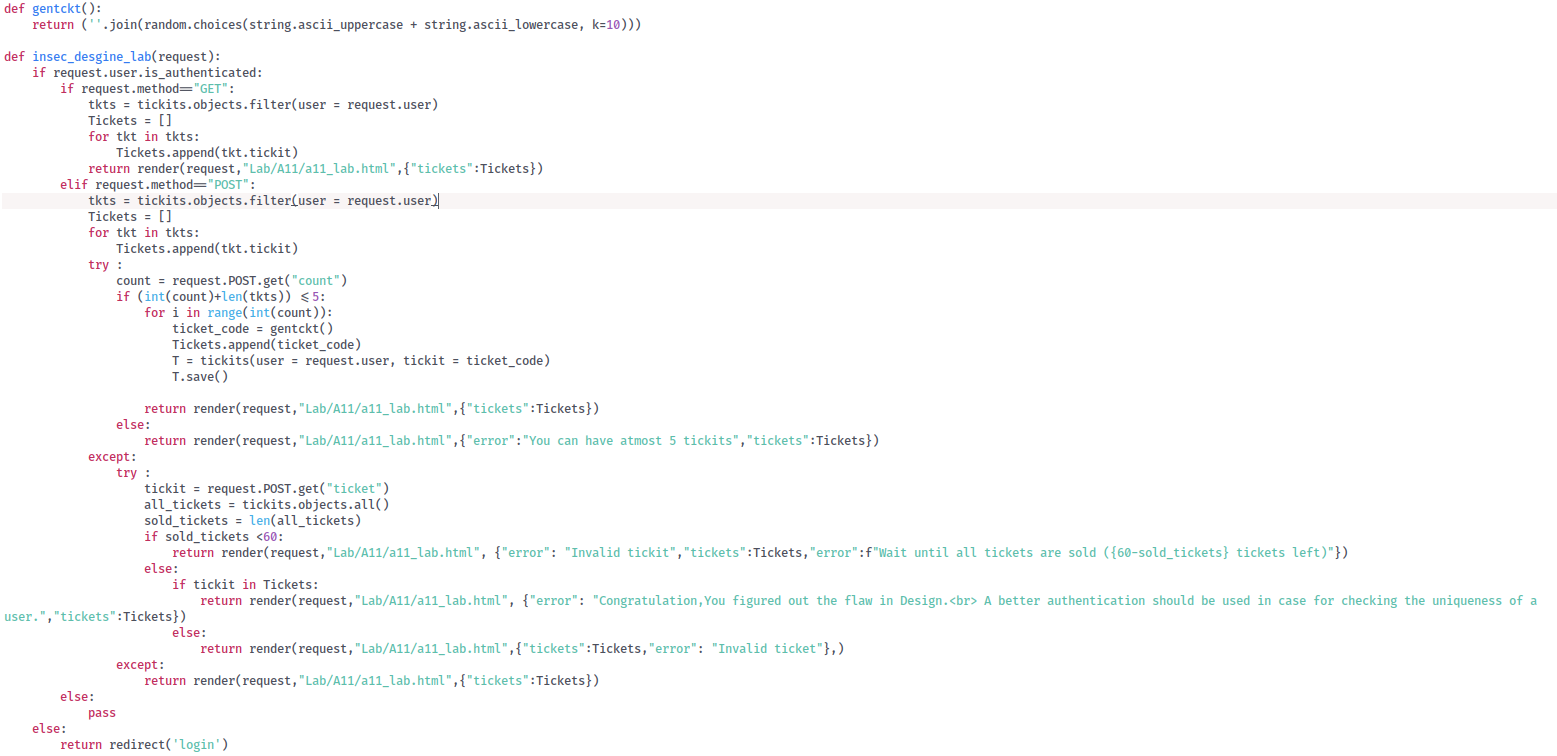
* Lab này sử dụng công cụ template mặc định của Django, và rõ ràng là đầu vào không được lọc đúng cách. Hãy thử khai thác lỗi này để lấy hash mật khẩu của admin.
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng và xử lý yêu cầu GET và POST cho trang ssti\_lab. Nếu người dùng đã đăng nhập, yêu cầu GET sẽ hiển thị các blog của người dùng. Với yêu cầu POST, mã sẽ lấy nội dung blog từ yêu cầu, tạo một ID duy nhất cho blog, và thêm mã HTML để định dạng blog. Sau đó, blog mới sẽ được lưu vào cơ sở dữ liệu và ghi vào một tệp HTML trong thư mục templates. Người dùng sẽ được chuyển hướng đến trang hiển thị blog mới. Hàm ssti\_view\_blog hiển thị nội dung blog dựa trên ID blog nếu người dùng đã đăng nhập. Nếu người dùng chưa đăng nhập, họ sẽ được chuyển hướng đến trang đăng nhập. Lỗ hổng bảo mật ở đây là việc xử lý nội dung blog mà không kiểm tra kỹ lưỡng, dễ bị tấn công Server-Side Template Injection (SSTI), cho phép kẻ tấn công chèn mã độc vào blog.



Hình 11. Mã triển khai lỗ hổng bảo mật của Template Injection

### A04:Insecure Design

* Trong trang tiếp theo, người dùng có thể nhận được 5 vé miễn phí cho một bộ phim. Tuy nhiên, họ phải đợi cho đến khi tất cả vé đã được bán hết. Trong tình huống này, chúng ta có thể tận dụng thiết kế không an toàn để lấy tất cả các vé của bộ phim.
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng và xử lý yêu cầu GET và POST cho trang insec\_desgine\_lab. Nếu người dùng đã đăng nhập, yêu cầu GET sẽ hiển thị các vé của người dùng. Với yêu cầu POST, mã sẽ lấy số lượng vé từ yêu cầu và kiểm tra xem tổng số vé của người dùng có vượt quá 5 hay không. Nếu không, mã sẽ tạo các mã vé mới bằng hàm gentckt và lưu chúng vào cơ sở dữ liệu. Nếu tổng số vé vượt quá 5, trang sẽ hiển thị thông báo lỗi. Ngoài ra, mã cũng kiểm tra tính hợp lệ của vé khi người dùng nhập vào. Nếu tổng số vé bán ra chưa đạt 60, trang sẽ yêu cầu người dùng chờ. Nếu vé hợp lệ, trang sẽ hiển thị thông báo thành công. Nếu vé không hợp lệ, trang sẽ hiển thị thông báo lỗi. Nếu người dùng chưa đăng nhập, họ sẽ được chuyển hướng đến trang đăng nhập. Lỗ hổng bảo mật ở đây là có kiểm tra xem tổng số vé của người dùng có vượt quá 5 hay không, nhưng không có cơ chế ngăn chặn người dùng tạo nhiều tài khoản để vượt qua giới hạn này, dẫn đến việc người dùng lạm dụng hệ thống để tạo ra nhiều vé hơn mức cho phép.



Hình 12. Mã triển khai lỗ hổng bảo mật của Insecure Design

### A05:Security Misconfiguration

#### Lab 1

* Lab này có một cấu hình bảo mật sai. Nó có một nút để tiết lộ khóa bí mật nhưng chỉ có thể truy cập nếu quản trị viên đang sử dụng nó. Làm thế nào để đăng nhập dưới tư cách quản trị viên? Có phải là cookie, hoặc một header trong yêu cầu không?
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng và xử lý yêu cầu cho hai trang sec\_mis\_lab và secret. Hàm sec\_mis\_lab kiểm tra xem người dùng đã đăng nhập hay chưa và hiển thị trang sec\_mis\_lab.html nếu đã đăng nhập, ngược lại sẽ chuyển hướng đến trang đăng nhập. Hàm secret kiểm tra giá trị của header X-Host trong yêu cầu. Nếu giá trị của X-Host là admin.localhost:8000, trang sẽ hiển thị một thông báo chứa "S3CR37K3Y". Nếu không, trang sẽ hiển thị thông báo rằng chỉ admin.localhost:8000 mới có thể truy cập và hiển thị giá trị của X-Host hiện tại. Lỗ hổng bảo mật ở đây là việc dựa vào header X-Host để xác thực quyền truy cập, dễ bị giả mạo bởi kẻ tấn công để truy cập thông tin bí mật.

A computer code on a white background

Description automatically generated

Hình 13. Mã triển khai lỗ hổng bảo mật của Lab 1 - A05

#### Lab 2

* Một trong những đặc điểm của việc thiết lập DEBUG=True là tiết lộ nhiều thông tin metadata từ môi trường của bạn, bao gồm toàn bộ cấu hình settings.py khi xảy ra lỗi. Bạn có thể kích hoạt lỗi 500 và lấy dữ liệu nhạy cảm không?
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng và xử lý các yêu cầu cho ba hàm: data\_exp\_lab, robots, và error. Hàm data\_exp\_lab kiểm tra xem người dùng đã đăng nhập hay chưa và hiển thị trang data\_exp\_lab.html nếu đã đăng nhập, ngược lại sẽ chuyển hướng đến trang đăng nhập. Hàm robots kiểm tra xem người dùng đã đăng nhập hay chưa và hiển thị nội dung của tệp robots.txt dưới dạng văn bản thuần túy nếu đã đăng nhập. Hàm error không thực hiện bất kỳ hành động nào và chỉ trả về None. Lỗ hổng bảo mật ở đây là việc hiển thị tệp robots.txt chỉ cho người dùng đã đăng nhập, có thể dẫn đến việc lộ thông tin nhạy cảm nếu tệp này chứa các chỉ dẫn quan trọng về cấu trúc trang web.

A screen shot of a computer

Description automatically generated

Hình 14. Mã triển khai lỗ hổng bảo mật của Lab 2 - A05

#### Lab 3

* Ứng dụng sử dụng JWT để quản lý phiên đăng nhập và lưu thông tin người dùng trong cookie. Tuy nhiên, cơ chế xác thực được triển khai có lỗ hổng bảo mật, cho phép một người dùng không phải admin giả mạo quyền truy cập admin chỉ bằng cách thay đổi giá trị của JWT. Bạn có thể khai thác lỗ hổng này để có được quyền truy cập admin không?
* Mã triển khai lỗ hổng bảo mật: Đoạn mã trên kiểm tra quyền truy cập của người dùng và xử lý yêu cầu cho trang sec\_misconfig\_lab3. Nếu người dùng chưa đăng nhập, họ sẽ được chuyển hướng đến trang đăng nhập. Mã sẽ cố gắng lấy giá trị của cookie auth\_cookie và giải mã nó bằng cách sử dụng khóa bí mật SECRET\_COOKIE\_KEY. Nếu giá trị của trường user trong payload là admin, trang sẽ hiển thị thông tin dành cho admin. Nếu không, trang sẽ hiển thị thông tin không phải admin. Nếu có lỗi xảy ra (ví dụ như cookie không tồn tại hoặc không thể giải mã), mã sẽ tạo một payload mới với giá trị user là not\_admin, mã hóa nó thành cookie mới và thiết lập cookie này cho người dùng. Lỗ hổng bảo mật ở đây là việc sử dụng cookie để xác thực quyền truy cập mà không có biện pháp bảo vệ mạnh mẽ, dễ bị tấn công và giả mạo, và kẻ tấn công có thể tạo ra các JWT hợp lệ với thông tin giả mạo, cho phép họ truy cập vào các chức năng dành cho admin.



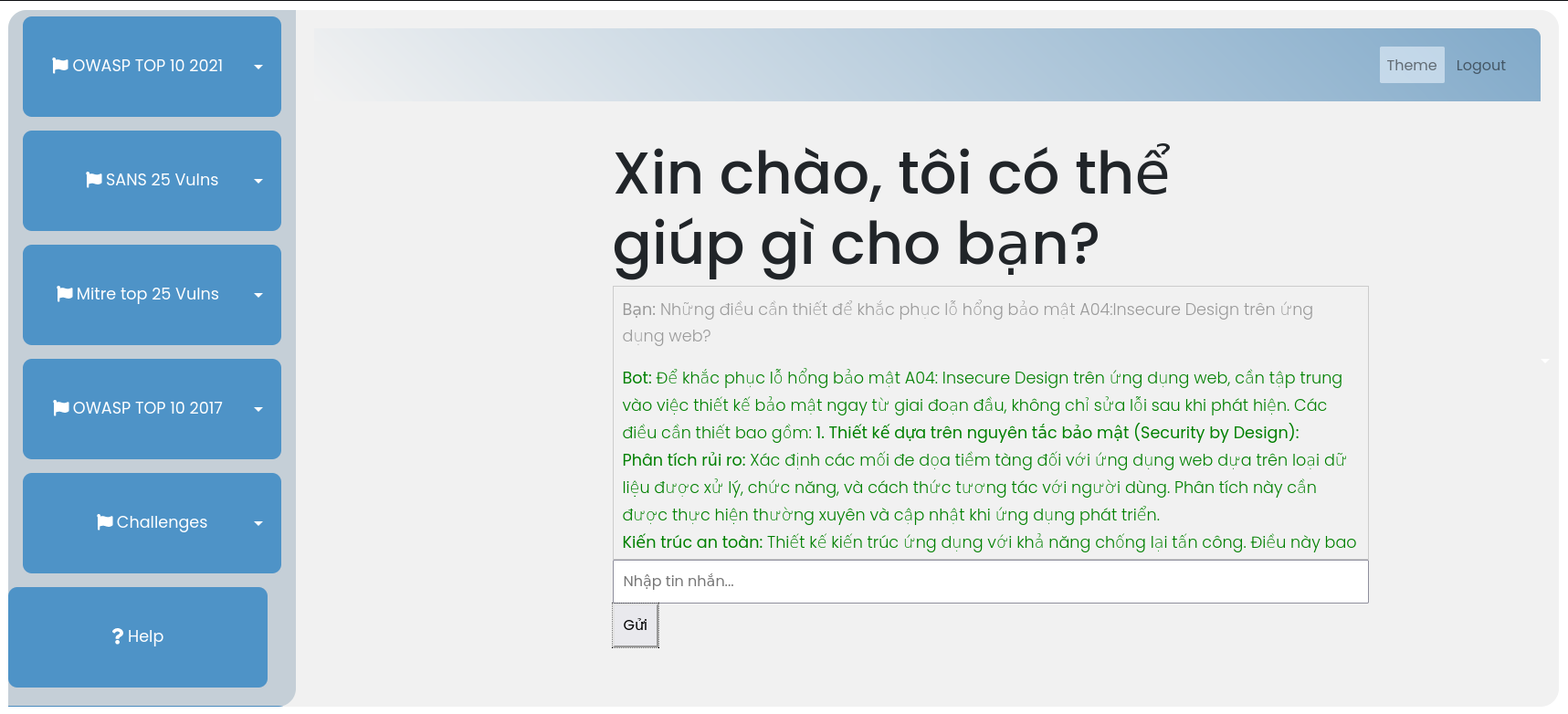
Hình 15. Mã triển khai lỗ hổng bảo mật của Lab 3 - A05

## Chatbot hỗ trợ người dùng.

Khi triển khai phiên bản OWASP PyGoat được cung cấp bởi OWASP, người dùng có thể nhận được một chatbot hỗ trợ cho một số ít câu hỏi cố định, cũng như là chatbot có sẵn đó không được cài đặt sẵn mà phải được chạy trên một trình có thể đọc file .ipynb. Để khắc phục điều này, trong đồ án này, chúng em đã tích hợp một AI chatbot sẵn bên trong ứng dụng.

Chatbot sử dụng ngôn ngữ lập trình Python, thư viện google.generativeai để kết nối với một mô hình ngôn ngữ lớn (LLM) của Google, ở đây chúng em chọn mô hình Gemini 1.5 Flash 8B Latest. Nền tảng chính ở đây là Google Cloud Platform, cung cấp dịch vụ API để tương tác với model.

Đầu tiên, đoạn mã thiết lập kết nối với API của Google và chọn một mô hình Gemini 1.5 Flash 8B Latest. Sau đó, khi người dùng nhập một câu hỏi, đoạn mã sẽ gửi câu hỏi đó đến mô hình và nhận lại câu trả lời. Cuối cùng, câu trả lời sẽ được hiển thị cho người dùng.



Hình 16. Chatbot hỗ trợ người dùng

# CHƯƠNG 4: KẾT LUẬN

Trong đề tài này, nhóm chúng em đã thực hiện triển khai các lỗ hổng bảo mật OWASP A01-A05 (2021) trên một ứng dụng web dễ bị tấn công. Sau khi thực hiện thì đã thành công tạo ra một ứng dụng web hoàn chỉnh theo yêu cầu đề ra

Trong quá trình thực hiện đồ án, từ mã nguồn có sẵn và các công cụ hỗ trợ thì nhóm chúng em đã cải thiện và đạt được kết quả đề ra gồm:

* Có thể triển khai được các lỗ hổng bảo mật từ A01-A05 trên ứng dụng web
* Thêm được phần chatbox hỗ trợ vào ứng dụng;
* Sử dụng module phiên dịch để chuyển ngôn ngữ sang tiếng Việt;
* Nâng độ khó của một vài phần so với phiên bản gốc.

Hạn chế trong quá trình:

* Chưa thể tự làm được một vài phần trong ứng dụng mà sử dụng từ mã nguồn có sẵn và các công cụ hỗ trợ.
* Chưa tối ưu được khả năng cài đặt của ứng dụng cũng như không cập nhật các thư viện mới và tinh giản lại dung lượng so với ứng dụng OWASP PyGoat.

# TÀI LIỆU THAM KHẢO

[1] “adeyosemanputra/pygoat at 411f9e9c32623fbe799b0b4fffecd768b0f250d3”. Truy cập: 28 Tháng Mười-Một 2024. [Online]. Available at: https://github.com/adeyosemanputra/pygoat/tree/411f9e9c32623fbe799b0b4fffecd768b0f250d3

[2] “OWASP Top Ten | OWASP Foundation”. Truy cập: 28 Tháng Mười-Một 2024. [Online]. Available at: https://owasp.org/www-project-top-ten/

[3] “A01 Broken Access Control - OWASP Top 10:2021”. Truy cập: 28 Tháng Mười-Một 2024. [Online]. Available at: https://owasp.org/Top10/A01\_2021-Broken\_Access\_Control/

[4] “A02 Cryptographic Failures - OWASP Top 10:2021”. Truy cập: 28 Tháng Mười-Một 2024. [Online]. Available at: https://owasp.org/Top10/A02\_2021-Cryptographic\_Failures/

[5] “A03 Injection - OWASP Top 10:2021”. Truy cập: 28 Tháng Mười-Một 2024. [Online]. Available at: https://owasp.org/Top10/A03\_2021-Injection/

[6] “A04 Insecure Design - OWASP Top 10:2021”. Truy cập: 28 Tháng Mười-Một 2024. [Online]. Available at: https://owasp.org/Top10/A04\_2021-Insecure\_Design/

[7] “A05 Security Misconfiguration - OWASP Top 10:2021”. Truy cập: 28 Tháng Mười-Một 2024. [Online]. Available at: https://owasp.org/Top10/A05\_2021-Security\_Misconfiguration/