# Xây dựng Web Demo kiểm thử OWASP 7 và 8

## Chuẩn bị bộ dữ liệu test và web demo

Web demo: Tạo ứng dụng Flask đơn giản với các chức năng như đăng nhập, đăng ký, quản lý mật khẩu, tải file (biểu hiện lỗ hổng OWASP 7 và 8).

## Chạy ứng dụng cục bộ

Chưa làm được chạy ứng dụng trên Docker, nên hiện tại các công việc xây dựng và kiểm thử sẽ tiến hành cục bộ.

## Các lỗ hổng sẽ được kiểm thử và phòng chống trong nội dung bài tập

### OWASP 7:

1. Không giới hạn số lần đăng nhập sai.

2. Không mã hóa dữ liệu đăng nhập.

3. Session không được bảo mật.

### OWASP 8:

4. Xử lý file tải lên không an toàn.

5. Thư viện không được kiểm tra tính toàn vẹn.

6. Injection thông qua các dependency không đáng tin.

## Test bảo mật với dữ liệu test

Viết script và hướng dẫn kiểm thử.

## Update bảo mật

Áp dụng các biện pháp phòng chống.

## Test bảo mật lần 2 và ghi lại kết quả

Lặp lại kiểm thử sau khi bảo mật.

## Cài đặt

pip install -r requirements.txt

python run.py

## Tiến hành test

### 1.Kiểm thử brute-force (OWASP 7)

python tests/test\_auth.py

Kết quả mong đợi: Script có thể tìm được mật khẩu hợp lệ do không giới hạn số lần thử.

Cách xử lý

Thêm limit

@limiter.limit("5 per minute") # Giới hạn 5 lần đăng nhập mỗi phút

Nếu nhập sai 5 lần thì sẽ bị khóa trong 1 phút

### 2.Kiểm thử lưu mật khẩu dạng plain-text (OWASP 7)

Hash mật khẩu

### 3.Kiểm thử bảo mật phiên (OWASP 7)

Session không an toàn

app.config["SESSION\_COOKIE\_SECURE"] = False

app.config["SESSION\_COOKIE\_HTTPONLY"] = False

Kẻ gian có thể dễ dàng viết Javascript xem phiên, ví dụ:

console.log(document.cookie);

Cách xử lý

Thêm config

app.config["SESSION\_COOKIE\_SECURE"] = True # Chỉ gửi cookie qua HTTPS

app.config["SESSION\_COOKIE\_HTTPONLY"] = True # Chặn JavaScript truy cập cookie

app.config["SESSION\_COOKIE\_SAMESITE"] = "Lax" # Giảm nguy cơ cross-site attack

Xóa hết session sau khi logout

### 4.Kiểm thử upload file không an toàn (OWASP 8)

Sử dụng Git Bash:

curl -X POST -F "file=@malicious.py" http://localhost:5000/upload

Kết quả mong đợi: File không được kiểm tra trước khi lưu, dễ bị tấn công thông qua file độc hại hoặc path traversal.

Kiểm tra qua giao diện Home

Giải pháp

Kiểm tra loại file được tải lên:

Chỉ cho phép các loại file hợp lệ dựa trên phần mở rộng (.txt, .jpg, .png, v.v.).

Kiểm tra MIME type của file để xác minh loại file thực sự.

Ngăn tấn công path traversal:

Ngăn người dùng tải file lên thư mục không mong muốn bằng cách lọc đường dẫn file.

Tạo tên file an toàn:

Sử dụng werkzeug.utils.secure\_filename để đảm bảo tên file không chứa ký tự nguy hiểm.

Giới hạn kích thước file:

Chặn file quá lớn để tránh tấn công từ chối dịch vụ (DoS).

### 5.Thư viện không được kiểm tra tính toàn vẹn (OWASP 8)

Mô phỏng: Thêm một thư viện bên ngoài không đáng tin cậy, như:

pip install some-untrusted-package

Giải pháp:

Sử dụng pip-audit để kiểm tra tính toàn vẹn:

pip install pip-audit

pip-audit

Mô phỏng: Sử dụng thư viện từ nguồn không đáng tin hoặc không khóa phiên bản cụ thể.

Cách triển khai lỗ hổng: Trong requirements.txt, dùng thư viện không khóa phiên bản:

flask

werkzeug

Thay vào đó, khóa phiên bản cụ thể trong requirements.txt

flask==2.0.3

werkzeug==2.0.3

### 6.Kiểm thử Dependency Injection (Injection thông qua dependency không đáng tin) (OWASP 8)

Cài đặt fake\_dependency vào môi trường:

cd fake\_dependency

pip install .

Sử dụng

@app.route("/vulnerable-dependency")

def test\_dependency():

from fake\_dependency.vulnerable\_function import vulnerable\_function

return vulnerable\_function()

Giải pháp

Dùng flag --only-binary :all: khi cài đặt để hạn chế cài đặt từ mã nguồn

pip install fake\_dependency --only-binary :all:

Dùng pip-audit

Kết quả

Name Skip Reason

--------------- ----------------------------------------------------------------------------

fake-dependency Dependency not found on PyPI and could not be audited: fake-dependency (0.1)

(docker build -t flask-demo .

docker run -p 5000:5000 flask-demo)