HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

VIENGOUDOM XAYAVONG

NGHIÊN CỨU KỸ THUẬT FUZZING TRONG KIỂM THỬ LỖ HỔNG BẢO MẬT WEBSITE NGÂN HÀNG

LUẬN VĂN THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI – 2022

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

VIENGOUDOM XAYAVONG

NGHIÊN CỨU KỸ THUẬT FUZZING TRONG KIỂM THỬ LỖ HỔNG BẢO MẬT WEBSITE NGÂN HÀNG

Chuyên ngành : Khoa học máy tính

Mã số : 8.48.01.01

LUẬN VĂN THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

Người hướng dẫn khoa học: PGS.TSKH. Hoàng Đăng Hải

HÀ NỘI - 2022

LỜI CAM ĐOAN

Tôi xin cam đoan nội dung trình bày luận văn này là do sự tìm hiểu và nghiên cứu của bản thân. Các kết quả nghiên cứu của các tác giả khác đều được trích dẫn cụ thể.

Luận văn này chưa được bảo vệ tại bất kỳ một hội đồng bảo vệ luận văn thạc sĩ nào trong nước và nước ngoài. Đồng thời, đến nay cũng chưa được công bố trên bất kỳ phương tiện thông tin truyền thông nào.

Tác giả luận văn

VIENGOUDOM XAYAVONG

LỜI CẢM ƠN

Lời đầu tiên tác giả xin được bày tỏ lòng biết ơn chân thành tới : **PGS.TSKH. Hoàng Đăng Hải** đã tận tình hướng dẫn và định hướng cho tôi trong suốt quá trình làm luận văn.

Tôi xin chân thành cảm ơn Ban lãnh đạo Học viện Công nghệ Bưu chính Viễn thông, Khoa Đào tạo Sau Đại học và quý thầy, cô và các bạn học viên đã tạo điều kiện tốt nhất và giúp đỡ tôi hoàn thành luận văn này.

Tôi xin bày tỏ sự biết ơn tới gia đình, bạn bè và đồng nghiệp đã thông cảm, động viên giúp đỡ cho tôi trong quá trình học tập và thực hiện luận văn.

Cuối cùng, mặc dù trong quá trình thực hiện luận văn này, tôi đã nỗ lực và cống gắng bằng tất cả khả năng của mình, nhưng không thể tránh khỏi những thiếu sót, tôi rất mong nhận được sự thông cảm và góp ý quý báu của quý thầy, cô và các bạn đọc.

Hà nội, ngày 07 tháng 07 năm 2022

Tác giả luận văn

VIENGOUDOM XAYAVONG

MỤC LỤC

DANH MỤC BẢNG, HÌNH

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Nghĩa Tiếng Anh	Nghĩa Tiếng Việt
HTTP	Hypertext Transfer Protocol	Giao thức truyền siêu văn bản
TCP	Transmission Control Protocol	Giao thức truyền TCP
HTML	Hypertext Markup Language	Ngôn ngữ đánh dấu siêu văn bản
XML	Extensible Markup Language	Ngôn ngữ đánh dấu mở rộng
SSL	Secure Sockets Layer	Lớp bảo mật socket
XSS	Cross Script Site	Lỗ hổng XSS
CSRF	Cross - Site Request Forgery	Lỗ hổng CSRF
URL	Uniform Resource Locator	Địa chỉ tài nguyên
RFI	Remote File Inclusion	Lỗ hổng RFI
LFI	Local File Inclusion	Lỗ hổng LFI
OWASP	The Open Web Application Security Project	Dự án nghiên cứu bảo mật ứng dụng web
GUI	Graphical User Interface	Giao diện đồ họa người dùng
CSDL	Database	Cơ sở dữ liệu

MỞ ĐẦU

1.Tính cấp thiết của đề tài

Website ngân hàng chứa nhiều thông tin nhạy cảm của khách hàng, cần được bảo vệ chống các xâm nhập của tin tặc. Bản chất các Website khi phát triển luôn tồn tại các lỗ hổng bảo mật. Do vậy, việc rà soát và kiểm tra các lỗ hổng bảo mật trên các Website ngân hàng luôn được quan tâm. Vì thế, bảo vệ thông tin trở thành một yêu cầu không thể thiếu trong mọi hoạt động nói chung và hoạt động điện tử nói riêng. Đặc biệt việc đảm bảo bảo mật Website ngân hàng trong thời đại số hiện nay càng trở nên cấp thiết và quan trọng. An toàn thông tin là bảo vệ thông tin và hệ thống thông tin nói chung bảo mật Website ngân hàng nói riêng khỏi các truy cập trái phép. Đây là biện pháp chống chiếm dụng, làm hỏng, chỉnh sửa hoặc thực hiện các thao tác gây mất thông tin quan trọng liên quan đến cá nhân, tổ chức.

Tuy nhiên, vấn đề bảo mật Website ngân hàng nói chung và ở Lào nói riêng vẫn chưa được các cơ quan, doanh nghiệp ở Lào chú trọng đầu tư. Không phải tất cả các tổ chức, doanh nghiệp đều có thể trang bị đầy đủ cũng như có thể đảm bảo an toàn, bảo mật thông tin một cách toàn diện. Việc kiểm thử lỗ hổng bảo mật Website ngân hàng rất cần thiết vì sẽ góp phần phát hiện các lỗ hổng và giúp khắc phục, vá các lỗi bảo mật này. Nhiều kỹ thuật kiểm thử, song kỹ thuật kiểm thử hộp đen (Black-Box) cụ thể là kỹ thuật Fuzzing (thuộc loại kiểm thử Black-Box) là quan trọng nhất. Mặc dù không mới, song mới được quan tâm từ vài năm nay do các ưu việt so với các kỹ thuật truyền thống đã biết.

Việc kiểm thử phần mềm hiện nay đa phần được thực hiện một cách thủ công, không có hiệu quả cao trong việc phát hiện những lỗ hổng an ninh tiềm tàng. Kỹ thuật Fuzzing trong kiểm thử lỗ hổng bảo mật Website ngân hàng chính là một giải pháp cho vấn đề trên. Với khả năng tự động hóa cao cùng với cơ chế phát hiện lỗ hổng hiệu quả, công nghệ này được rất nhiều hãng quan tâm sử dụng. Tuy kỹ thuật fuzzing đã được áp dụng trong nhiều lĩnh vực, song việc áp dụng kỹ thuật này vào kiểm thử lỗ hổng bảo mật website vẫn còn có những vấn đề cần nghiên cứu do sự phát triển của lĩnh vực an toàn thông tin.

Xuất phát từ thực tế trên, em đã lựa chọn đề tài **"*Nghiên cứu kỹ thuật Fuzzing trong kiểm thử lỗ hổng bảo mật Website ngân hàng*"** thuộc phạm vi các vấn đề đã nêu để làm luận văn. Mục đích của nghiên cứu là tìm hiểu, thử nghiệm kỹ thuật Fuzzing trong kiểm thử lỗ hổng bảo mật website ngân hàng. Do bản thân có nguyện vọng nghiên cứu kỹ thuật và áp dụng vào công tác tại một ngân hàng tại Lào, em chọn website ngân hàng làm môi trường cho thử nghiệm kỹ thuật này.

2. Tổng quan về vấn đề nghiên cứu

- **Về vấn đề lỗ hổng bảo mật của website**

Có thể nói lỗ hổng bảo mật là những điểm yếu trên hệ thống hoặc ẩn chứa trong một dịch vụ mà dựa vào đó kẻ tấn công có thể xâm nhập trái phép để thực hiện các hành động phá hoại hay chiếm đoạt các tài nguyên hợp pháp.

Lỗ hổng website là những điểm yếu của hệ thống website mà tin tặc 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 website.

- **Về kỹ thuật Fuzzing**

Trong lĩnh vực an ninh ứng dụng, Fuzzing hay kiểm thử mờ (fuzz testing) là một kỹ thuật thuộc kiểm thử hộp đen (black box), phát hiện lỗi của phần mềm bằng cách tự động hoặc bán tự động cung cấp dữ liệu đầu vào không hợp lệ, không mong đợi hay ngẫu nhiên vào phần mềm. Phần mềm sẽ được giám sát và ghi lại các trường hợp ngoại lệ như lỗi mà không được thực thi, tài nguyên thất thoát,... nhằm xác định các hành vi bất thường, phát hiện các lỗ hổng bảo mật tiềm ẩn của phần mềm. Dữ liệu không mong đợi thường là các giá trị vượt quá biên, các giá trị đặc biệt có ảnh hưởng tới phần xử lý, hiển thị của chương trình. Các chương trình và framework được dùng để tạo ra kỹ thuật fuzzing hoặc thực hiện fuzzing được gọi là Fuzzer. Tùy theo môi trường và ứng dụng cần kiểm tra mà người ta có các phương án khác nhau để xây dựng Fuzzer.

Fuzzing là một trong những kỹ thuật của kiểm thử hộp đen, không đòi hỏi quyền truy cập vào mã nguồn [3, 4]. Do đó, nó có khả năng tìm thấy lỗi một cách

nhanh chóng và tránh được việc phải xem mã nguồn.Fuzzing cũng giống như các kỹ thuật kiểm thử phần mềm, nhưng nó được sử dụng để phát hiện ra một loạt các vấn đề của web như: Cross Site Scripting, tràn bộ đệm, chèn câu truy vấn (SQL Injection),... [5, 15]

Hiện nay, đã có một số nghiên cứu và ứng dụng kỹ thuật Fuzzing trong kiểm thử phần mềm [5]. Tuy nhiên, việc ứng dụng kỹ thuật Fuzzing trong kiểm thử bảo mật tự động các Website nói chung và Website ngân hàng nói riêng chưa được quan tâm nhiều, song việc áp dụng kỹ thuật này vào kiểm thử lỗ hổng bảo mật website vẫn còn có những vấn đề cần nghiên cứu do sự phát triển của lĩnh vực an toàn thông tin [6, 9, 10].

3. Mục đích nghiên cứu

Mục tiêu nghiên cứu của luận văn là: Nghiên cứu, khảo sát, tìm hiểu về những lỗ hổng bảo mật phổ biến trong website, cụ thể là website một ngân hàng; tìm hiểu về kiểm thử lỗ hổng bảo mật website và kỹ thuật Fuzzing trong kiểm thử lỗ hổng bảo mật website ngân hàng; thực hiện thử nghiệm.

4. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Kỹ thuật Fuzzing trong kiểm thử lỗ hổng bảo mật trên website một ngân hàng.

Phạm vi nghiên cứu: Website một ngân hàng.

5. Phương pháp nghiên cứu

- Phương pháp nghiên cứu lý thuyết: Nghiên cứu tài liệu liên quan đến các lỗ hổng bảo mật đã được công bố hiện nay, các kỹ thuật Fuzzing trong bảo mật website.

- Phương pháp khảo sát: Nghiên cứu tìm hiểu các lỗ hổng bảo mật điển hình trên website, kỹ thuật kiểm thử; phân tích đánh giá công cụ thử nghiệm.

- Phương pháp nghiên cứu thực nghiệm: Thực hiện thử nghiệm với một công cụ kiểm thử.

CHƯƠNG 1. TỔNG QUAN VỀ KIỂM THỬ WEBSITE

1.1. Các khái niệm cơ bản về hoạt động của website (ngân hàng), lỗ hổng bảo mật trên website

1.1.1. Các khái niệm cơ bản về hoạt động của website (ngân hàng)

Website là một tập hợp các trang web, thường chỉ nằm trong một tên miền hoặc tên miền phụ trên World Wide Web của Internet.Một trang web là tập tin HTML hoặc XHTML có thể truy nhập dùng giao thức HTTP.Website có thể được xây dựng từ các tệp tin HTML (website tĩnh) hoặc vận hành bằng các CMS chạy trên máy chủ (website động). Website có thể được xây dựng bằng nhiều ngôn ngữ lập trình khác nhau (PHP, .NET, Java, Ruby on Rails…)[6, 12, 20].

Một Website thường được bao gồm bởi 04 phần chính:

- Source code: Mã nguồn website, chứa tệp lệnh trích xuất HTML.
- Hosting: Bộ nhớ lưu trữ website.
- Database: Dữ liệu nội dung website.
- Domain: Tên miền của website, thực chất một website không cần đến tên miền nó vẫn có thể hoạt động bình thường vì nó có địa chỉ IP. Bản chất của tên miền là nó được ánh xạ sang địa chỉ IP thông qua máy chủ DNS, tạo ra sự đơn giản cho người dùng dễ dàng truy cập vào web thông qua tên miền, thay vì phải nhớ địa chỉ IP của website.

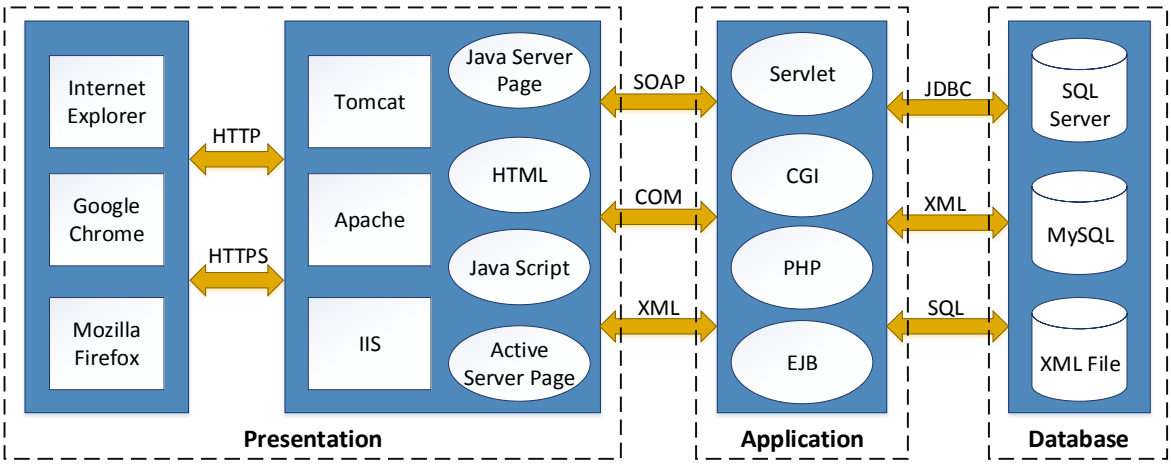
Ban đầu, các website chỉ bao gồm các nội dung văn bản, hình ảnh và video, chúng được liên kết với nhau thông qua các link. Tác dụng của website đơn giản chỉ là lưu trữ và hiển thị thông tin. Người dùng chỉ có thể đọc, xem, di chuyển đến các được dẫn giữa các page.Với công nghệ hiện nay, website không chỉ đơn giản là một trang tin cung cấp các thông tin. Trước sự ra đời của các ngôn ngữ server như: CGI, ASP, PHP,… các website đã trở nên linh hoạt, có thể tương tác với người dùng. Từ đây, người dùng có thể dùng web để thực hiện một công việc nào đó bằng máy tính, do đó ứng dụng web được ra đời.

Ứng dụng web là một ứng dụng chủ/khách sử dụng giao thức HTTP để tương tác với người dùng hay hệ thống khác [8, 13].

Trình khách là một trình duyệt web như: Internet Explorer, Chrome, FireFox hay có thể là một chương trình có chức năng như một trình duyệt web. Người dùng có thể gửi, nhận các dữ liệu từ máy chủ thông qua việc trao đổi luồng thông tin với web server và hiển thị nội dung trang web nhận được trên trình duyệt. Các ứng dụng web này có thể là các trang cổng thông tin điện tử, trao đổi thông tin, mua bán, các diễn đàn, các trang gửi nhận thư,...

Tốc độ phát triển các kỹ thuật xây dựng ứng dụng web cũng phát triển rất nhanh. Trước đây những ứng dụng web thường được xây dựng bằng CGI (Common Getaway Interface) được chạy trên các máy chủ web và kết nối với với các cơ sở dữ liệu đơn giản trên cùng một máy chủ. Ngày nay, ứng dụng web thường được viết bằng PHP, ASP.Net, JSP (hay các ngôn ngữ tương tự) và chạy trên máy chủ phân tán, kết nối đến nhiều nguồn dữ liệu.

Một ứng dụng web thường có kiến trúc gồm:



Hình 1.1. Kiến trúc một ứng dụng web

Trên hình 1.1 mô tả kiến trúc thông thường của một ứng dụng web bao gồm các lớp:

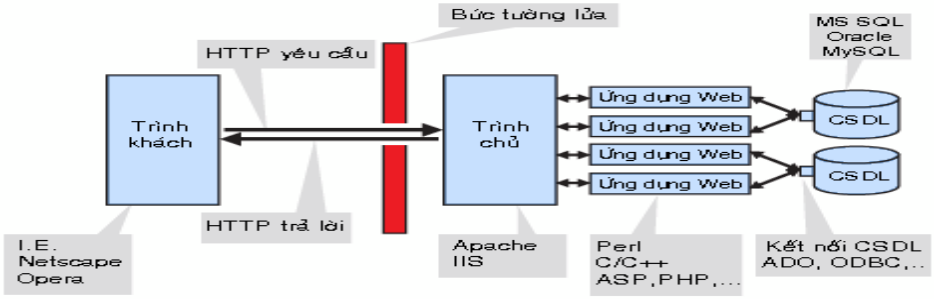
- Lớp trình diễn: Lớp này có chức năng hiển thị dữ liệu mà nó nhận được từ web server cho người dùng, ngoài ra còn có thể có chức năng tạo bố cục và giao diện cho trang web.

- Lớp ứng dụng: Đây là nơi xử lý của ứng dụng web. Nó sẽ xử lý thông tin yêu cầu từ người dùng, đưa ra quyết định, gửi kết quả đến lớp trình diễn. Lớp này thường được cài đặt bằng các kỹ thuật lập trình dựa trên các ngôn ngữ như CGI, Java, .NET, PHP,... và được triển khai trên host hoặc trên các dịch vụ của máy chủ như Apache của Linux, IIS của Windows Server,…

- Lớp dữ liệu: Lớp này là các hệ quản trị dữ liệu SQL như MySQL, SQL Server, Oracle,... chịu tránh nhiệm quản lý các file dữ liệu và quyền sử dụng dữ liệu của toàn bộ website. Thường được triển khai trực tiếp trên cùng hoặc tách biệt riêng với web server.

Việc phân lớp trong kiến trúc web tạo ra các hoạt động đơn giản nhưng có liên kết chặt chẽ giữa các lớp. Nó giúp cho người quản trị dễ dàng triển khai, vận hành và chủ động trong phòng, chống các cuộc tấn công.Ví dụ như lớp ứng dụng có lỗi nhưng hệ thống, cơ sở dữ liệu được cấu hình đảm bảo thì hacker khó có thể khai thác và làm ảnh hưởng tới hệ thống.

Hoạt động của một ứng dụng web là sự tương tác giữa trình khách với web server. Dưới đây là mô hình hoạt động của một ứng dụng web:



Hình 1.2. Mô hình hoạt động của một ứng dụng web

Tương ứng các lớp của một ứng dụng web, hoạt động của một website cũng có 3 phần:

- Trình khách (trình duyệt người dùng): Chrome, FireFox,…

- Trình chủ: Apache, IIS,…

- Hệ quản trị CSDL: Oracle, SQL Server, MySQL,…

Bên cạnh đó, một giải pháp dùng để bảo vệ hệ thống mạng thường được sử dụng là bức tường lửa (firewall), nó có vai trò như lớp rào chắn bên ngoài một hệ thống mạng, vì chức năng chính của firewall là kiểm soát luồng thông tin giữa các máy tính. Có thể xem firewall như một bộ lọc thông tin, nó xác định và cho phép một máy tính này có được truy xuất đến một máy tính khác hay không, hay một mạng này có được truy xuất đến mạng kia hay không [8].

Người ta thường dùng firewall vào mục đích:

- Cho phép hoặc cấm các dịch vụ truy xuất ra ngoài.

- Cho phép quy định cấm các hay cho phép dịch vụ từ bên ngoài truy xuất vào trong.

- Kiểm soát địa chỉ truy nhập, cấm địa chỉ truy nhập.

Firewall hoạt động dựa trên góp IP do đó kiểm soát được việc truy cập máy tính của người sử dụng.

- ***Mô tả hoạt động của website***

Trình duyệt tạo một HTTP Request gửi máy chủ web thông qua các phương thức GET, POST,… của giao thức HTTP, yêu cầu cung cấp hoặc xử lý tài nguyên thông tin. Địa chỉ của tài nguyên yêu cầu được xác định trong định dạng URL.

Sau khi nhận được truy vấn từ trình khách, máy chủ web xác định sự tồn tại của tài nguyên được yêu cầu. Nếu yêu cầu can thiệp các quyền truy cập của tài nguyên thì máy chủ web từ chối truy vấn và trả về cảnh báo thích hợp. Nếu yêu cầu là hợp lệ, lúc này máy chủ có thể cho thực thi một chương trình được xây dựng từ ngôn ngữ như Perl, C/C++,… hoặc máy chủ yêu cầu bộ biên dịch thực thi các trang PHP, ASP, JSP,… theo yêu cầu của máy khách. Tùy theo các tác vụ của chương trình được cài đặt mà nó xử lý, tính toán, kết nối đến cơ sở dữ liệu, lưu các thông tin do máy khách gửi đến.

Khi máy chủ web định danh được tài nguyên, nó thực hiện hành động chỉ ra trong request method và tạo ra response trả về cho máy khách 1 luồng dữ liệu có định dạng theo giao thức HTTP, nó gồm 2 phần:

- Header mô tả các thông tin về gói dữ liệu và các thuộc tính, trạng thái trao đổi giữa trình duyệt và WebServer.

- Body là phần nội dung dữ liệu mà Server gửi về Client, nó có thể là một file HTML, một hình ảnh, một đoạn phim hay một văn bản bất kì.

Khi giao dịch hoàn tất, máy chủ web thực hiện ghi vào tệp tin nhật ký mô tả giao dịch vừa thực hiện.

Với firewall, luồng thông tin giữa máy chủ và máy khách là luồng thông tin hợp lệ. Vì thế, nếu hacker tìm thấy vài lỗ hổng trong ứng dụng Web thì firewall không còn hữu dụng trong việc ngăn chặn hacker này. Do đó, các kỹ thuật tấn công vào một hệ thống mạng ngày nay đang dần tập trung vào những sơ suất (hay lỗ hổng) trong quá trình tạo ứng dụng của những nhà phát triển Web hơn là tấn công trực tiếp vào hệ thống mạng, hệ điều hành. Tuy nhiên, hacker cũng có thể lợi dụng các lỗ hổng Web để mở rộng sự tấn công của mình vào các hệ thống không liên quan khác [8, 18].

1.1.2. Lỗ hổng bảo mật website

❖ **Định nghĩa lỗ hổng bảo mật**

Lỗ hổng bảo mật trên một hệ thống là các điểm yếu có thể tạo ra sự ngưng trệ của dịch vụ, thêm quyền đối với người sử dụng hoặc cho phép các truy nhập không hợp pháp vào hệ thống. Các lỗ hổng cũng có thể nằm ngay các dịch vụ cung cấp như sendmail, web, ftp … Ngoài ra các lỗ hổng còn tồn tại ngay chính tại hệ điều hành như trong Windows XP, Windows NT, UNIX; hoặc trong các ứng dụng mà người sử dụng thường xuyên sử dụng như Word processing, các hệ databases… [7, 11, 12]

Có thể nói lỗ hổng bảo mật là những điểm yếu trên hệ thống hoặc ẩn chứa trong một dịch vụ mà dựa vào đó kẻ tấn công có thể xâm nhập trái phép để thực hiện các hành động phá hoại hay chiếm đoạt các tài nguyên hợp pháp.

Nguyên nhân gây ra lỗ hổng bảo mật là khác nhau:

- Do lỗi của bản thân hệ thống

- Do phần mềm cung cấp hoặc do người lập trình

- Do người quản trị yếu kém không hiểu sâu sắc các dịch vụ cung cấp.

❖ **Lỗ hổng website**

Lỗ hổng website là những điểm yếu của hệ thống website mà tin tặc 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 website [1, 19].

Lỗ hổng website có thể xuất phát từ nhiều nguyên nhân, tuy nhiên chủ yếu là do 3 nguyên nhân sau:

- Lỗi do người lập trình, phát triển ứng dụng tập trung vào chức năng và tốc độ mà không quan tâm đến an toàn. Ứng dụng không có thành phần kiểm tra hay kiểm tra yếu các dữ liệu đầu vào từ người dùng, từ đó, kẻ tấn công có thể lợi dụng lỗ hổng từ mã nguồn để khai thác và tấn công hệ thống.

- Lỗi do người quản trị cấu hình hệ thống yếu, cấu hình hệ thống mặc định, tài khoản mặc định, không thường xuyên cập nhật phiên bản mới cho các dịch vụ triển khai trên hệ thống.

- Lỗi nằm trong các giao thức, các nền tảng hay chuẩn xây dựng hệ thống đã được công khai. Ví dụ như giao thức HTTP hoạt động theo chuẩn mô hình client/server đơn giản và khi xây dựng giao thức này người ta chưa quan tâm đến vấn đề bảo mật.

❖ **Kiểm thử phần mềm**

Kiểm thử phần mềm được định nghĩa theo nhiều cách khác nhau, dưới đây là một số định nghĩa:

Kiểm thử phần mềm là quá trình khảo sát một hệ thống hay thành phần dưới những điều kiện xác định, quan sát và ghi lại các kết quả, và đánh giá một khía cạnh nào đó của hệ thống hay thành phần đó [8].

Kiểm thử phần mềm là quá trình thực thi một chương trình với mục đích tìm lỗi [7].

Kiểm thử phần mềm là hoạt động khảo sát thực tiễn sản phẩm hay dịch vụ phần mềm trong đúng môi trường chúng dự định sẽ được triển khai nhằm cung cấp

cho người có lợi ích liên quan những thông tin về chất lượng của sản phẩm hay dịch vụ phần mềm ấy. Mục đích của kiểm thử phần mềm là tìm ra các lỗi hay khiếm khuyết phần mềm nhằm đảm bảo hiệu quả hoạt động tối ưu của phần mềm trong nhiều ngành khác nhau [5, 7, 18].

Có thể định nghĩa một cách dễ hiểu như sau:

Kiểm thử phần mềm là một tiến trình hay một tập hợp các tiến trình được thiết kế và thực hiện nhằm đảm bảo cho hệ thống thực hiện theo đúng những yêu cầu mà chúng đã được thiết kế và không thực hiện những điều không mong muốn. Kiểm thử phần mềm là một pha quan trọng trong quá trình xây dựng và phát triển hệ thống, chúng giúp cho người phát triển hệ thống và các khách hàng thấy được hệ thống mới đã đáp ứng các yêu cầu đặt ra.

Các phương pháp kiểm thử phần mềm có thể chia làm 3 loại:

- Kiểm thử hộp đen (Black box testing)
- Kiểm thử hộp trắng (White box testing)
- Kiểm thử hộp xám (Gray box testing)

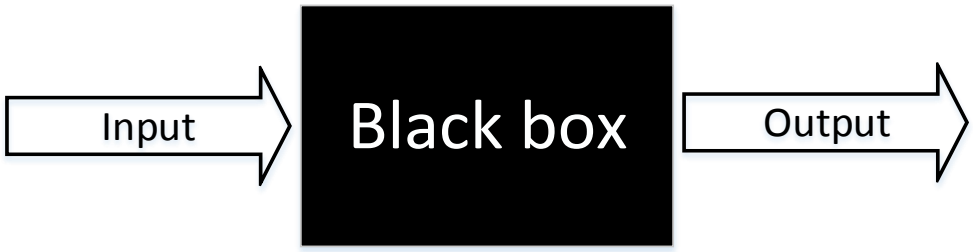
1.2. Nghiên cứu về các kỹ thuật kiểm thử (hộp trắng, hộp đen, hộp xám).

1.2.1. Kiểm thử hộp đen

Là phương pháp kiểm thử được thực hiện mà không biết được cấu trúc và hành vi bên trong của phần mềm, là cách kiểm thử mà hệ thống được xem như một chiếc hộp đen, không cách nào nhìn thấy phía bên trong cái hộp [14].

Một số phương pháp kiểm thử hộp đen:

- Kiểm thử fuzzing (Fuzz testing)
- Phân lớp tương đương (Equivalence partitioning)
- Phân tích giá trị biên (Boundary value analysis)
- Kiểm thử mọi cặp (All-pairs testing)
- Ma trận dấu vết (Traceability matrix)
- Kiểm thử thăm dò (Exploratory testing)



Hình 1.3. Kiểm thử hộp đen

Kiểm thử hộp đen không có mối liên quan nào tới mã lệnh, những kiểm thử viên hộp đen tìm ra lỗi mà những lập trình viên đã không tìm ra. Nhưng, mặt khác, người ta cũng nói kiểm thử hộp đen "giống như là đi trong bóng tối mà không có đèn", bởi vì kiểm thử viên không biết các phần mềm được kiểm tra thực sự được xây dựng như thế nào. Đó là lý do mà có nhiều trường hợp mà một kiểm thử viên hộp đen viết rất nhiều ca kiểm thử để kiểm tra một thứ gì đó mà đáng lẽ có thể chỉ cần kiểm tra bằng 1 ca kiểm thử duy nhất [6] .

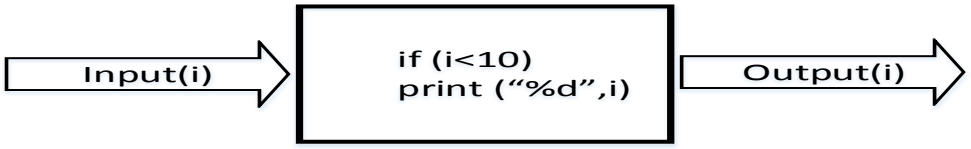
Do vậy, kiểm thử hộp đen có ưu điểm của một sự đánh giá khách quan, mặt khác nó lại có nhược điểm của một thăm dò mù.

1.2.2. Kiểm thử hộp trắng

Là phương pháp kiểm thử trái ngược hoàn toàn với kiểm thử hộp đen, nó cho phép kiểm tra cấu trúc bên trong của một phần mềm với mục đích đảm bảo rằng tất cả các mã lệnh, thuật toán và điều kiện sẽ được thực hiện ít nhất 1 lần.

Một số phương pháp kiểm thử hộp trắng:

- Kiểm thử giao diện lập trình ứng dụng (API testing)
- Bao phủ mã lệnh (Code coverage)
- Các phương pháp gán lỗi (Fault injection)
- Các phương pháp kiểm thử hoán chuyển (Mutation testing methods)
- Kiểm thử tĩnh (Static testing)



Hình 1.4. Kiểm thử hộp trắng

Kiểm thử hộp trắng có thể áp dụng tại cấp đơn vị, tích hợp hệ thống và các cấp độ của quá trình kiểm thử phần mềm. Mặc dù phương pháp này thiết kế kiểm thử có thể phát hiện ra nhiều lỗi hoặc các vấn đề, nhưng nó có thể không phát hiện các phần chưa thực hiện của các đặc điểm kỹ thuật hoặc yêu cầu thiếu sót [17].

1.2.3. Kiểm thử hộp xám

Là sự kết hợp của kiểm thử hộp đen và hộp trắng. Trong kiểm thử hộp xám, cấu trúc bên trong sản phẩm chỉ được biết một phần, người kiểm thử có thể truy cập vào cấu trúc dữ liệu bên trong và thuật toán của chương trình với mục đích là để thiết kế đầu vào, nhưng khi kiểm tra thì như ở mức hộp đen.

Hình 1.5. Kiểm thử hộp xám

Việc thao tác tới dữ liệu đầu vào và định dạng dữ liệu đầu ra là không rõ ràng, giống như một chiếc hộp xám, bởi vì đầu vào và đầu ra rõ ràng là ở bên ngoài hộp đen mà chúng ta vẫn gọi về hệ thống được kiểm tra [6].

1.2.4. Kiểm thử website

Kiểm thử website là một thành phần trong kiểm thử phần mềm nhưng tập trung vào các ứng dụng web, nhằm đảm bảo các ứng dụng web hoạt động một cách hiệu quả, chính xác và đáp ứng được nhu cầu của khách hàng. Hiện nay, nó đang là một trong những thành phần đang phát triển nhanh nhất của kiểm thử phần mềm.

Hoàn thành quá trình kiểm thử của một hệ thống web trước khi đi vào hoạt động là bước đầu để có được sự đảm bảo về khả năng các ứng dụng được xây dựng trên trang web đang hoạt động đúng. Nó giúp giải quyết các vấn đề như tính sẵn sàng, toàn vẹn, bảo mật của hệ thống web, đáp ứng cho số lượng ngày càng tăng cao người sử dụng và khả năng sống sót trong lưu lượng truy cập của người dùng. Việc bỏ qua các vấn đề trong kiểm thử trước khi đi vào hoạt động có thể ảnh hưởng đến khả năng hoạt động của chính website đó.

Sau khi thực hiện kiểm thử web, kiểm thử viên có thể tìm thấy các sự cố trong hệ thống trước khi chúng xảy ra trong môi trường người dùng.

1.2.5. Fuzzing

Trong lĩnh vực an ninh ứng dụng, Fuzzing hay kiểm thử mờ (fuzz testing) là một kỹ thuật thuộc kiểm thử hộp đen (black box), phát hiện lỗi của phần mềm bằng cách tự động hoặc bán tự động cung cấp dữ liệu đầu vào không hợp lệ, không mong đợi hay ngẫu nhiên vào phần mềm. Phần mềm sẽ được giám sát và ghi lại các trường hợp ngoại lệ như lỗi mà không được thực thi, tài nguyên thất thoát,... nhằm xác định các hành vi bất thường, phát hiện các lỗ hổng bảo mật tiềm ẩn của phần mềm. Dữ liệu không mong đợi thường là các giá trị vượt quá biên, các giá trị đặc biệt có ảnh hưởng tới phần xử lý, hiển thị của chương trình [4, 6, 16].

Các chương trình và framework được dùng để tạo ra kỹ thuật fuzzing hoặc thực hiện fuzzing được gọi là Fuzzer. Tùy theo môi trường và ứng dụng cần kiểm tra mà người ta có các phương án khác nhau để xây dựng Fuzzer.

Fuzzing là một trong những kỹ thuật của kiểm thử hộp đen, không đòi hỏi quyền truy cập vào mã nguồn. Do đó, nó có khả năng tìm thấy lỗi một cách nhanh chóng và tránh được việc phải xem mã nguồn.

Fuzzing cũng giống như các kỹ thuật kiểm thử phần mềm, nhưng nó được sử dụng để phát hiện ra một loạt các vấn đề của web như: Cross Site Scripting, tràn bộ đệm, chèn câu truy vấn (SQL Injection),... [6, 16]

1.3. Tìm hiểu về yêu cầu và khả năng của kỹ thuật Fuzzing áp dụng cho kiểm thử lỗ hổng bảo mật website ngân hàng

1.3.1. Lịch sử

Fuzzing có nguồn gốc từ năm 1988, bởi giáo sư Barton Miller, tại Đại học Wisconsin [3].

Ông cùng sinh viên của mình thực hiện một dự án mang tên “Operating System Utility Program Reliability - The Fuzz Generator” để kiểm tra mức độ chịu đựng của các ứng dụng Unix, độ tin cậy của mã nguồn. Dự án được thực hiện bằng cách thử nghiệm tấn công hệ thống với các dữ liệu đầu vào không hợp lệ, bất ngờ

hoặc ngẫu nhiên ở các cấp độ khác nhau, nhằm nỗ lực để khám phá các hành vi bất ngờ hoặc là thất bại của hệ thống, bao gồm: treo hệ thống, không khẳng định mã, rò rỉ bộ nhớ... Dự án cũng cung cấp bộ gỡ lỗi và công cụ để xác định nguyên nhân và thể loại của mỗi kết quả phát hiện.

Mã nguồn của công cụ, các dữ liệu kết quả thô đã được công bố công khai để các nhà nghiên cứu khác có thể để tiến hành các thử nghiệm tương tự với các phần mềm khác. Hiện nay, các kết quả nghiên cứu của dự án vẫn được cập nhật tại địa chỉ: http://pages.cs.wisc.edu/~bart/fuzz/.

Năm 1991, các công cụ crashme đã được phát hành, được dùng để kiểm tra độ tin cậy của hệ điều hành Unix bằng cách thực hiện lệnh máy ngẫu nhiên. Trong năm 1995, một fuzzer có giao diện GUI đã được sử dụng để thử nghiệm các công cụ, giao thức mạng và các API hệ thống thư viện.

Năm 2002, Microsoft đã quyết định đầu tư cho nhóm sáng lập PROTOS. Năm 2003, các thành viên của nhóm đã thành lập Codenomicon, một công ty chuyên thiết kế và phát triển các sản phẩm fuzzing thương mại.

Năm 2012, Google đã công bố ClusterFuzz, một hạ tầng kỹ thuật fuzzing dựa trên đám mây cho các thành phần bảo mật quan trọng của các trình duyệt web Chromium . Nghiên cứu bảo mật có thể tải lên các fuzzers riêng có và thu thập tiền thưởng lỗi nếu ClusterFuzz thấy một vụ tai nạn với fuzzer tải lên.

Năm 2016, Microsoft đã công bố dự án Springfield, một dịch vụ thử nghiệm Fuzzing dựa trên điện toán đám mây cho việc tìm kiếm an ninh lỗi nghiêm trọng trong phần mềm.

Năm 2016, Google đã công bố OSS-Fuzz, một chương trình mã nguồn mở được phát triển dựa trên 2 dự án ClusterFuzz và Springfield, cho phép fuzzing liên tục phần mềm mã nguồn mở. Giúp cho các mã nguồn mở đảm bảo an toàn, bảo mật.

Đến nay, không chỉ các hãng lớn thực hiện nghiên cứu mà còn có nhiều dự án mã nguồn mở đã được phát triển và ứng dụng rộng rãi trong cộng đồng người sử dụng.

1.3.2. Phân loại Fuzzing

Phân loại fuzzing có thể tùy thuộc vào bộ dữ liệu fuzz, mục tiêu fuzzing hay phương pháp fuzzing,…

Phân loại theo dữ liệu fuzz

Kiểm thử mờ dựa trên đột biến

Kiểm thử mờ dựa trên đột biến (Mutation Based Fuzzing) hay còn gọi là kiểm thử mờ câm (Dumb Fuzzing) là phương pháp kiểm thử mà dữ liệu fuzz được biến đổi từ mẫu dữ liệu hợp lệ hiện có để tạo thành dữ liệu kiểm thử cho mục tiêu fuzzing.

Một số đặc điểm đối với cách tiếp cận này [16]:

- Người thực hiện không cần có nhiều hiểu biết về cấu trúc của các yếu tố đầu vào.

- Tính dị thường được thêm vào đầu vào hợp lệ hiện có có thể hoàn toàn ngẫu nhiên hoặc theo một số chuẩn đoán về mặt kinh nghiệm.

- Dữ liệu cho thực hiện fuzzing hoàn toàn phụ thuộc vào các yếu tố đầu vào được sửa đổi.

- Yêu cầu ít hoặc việc thiết lập thời gian đơn giản hoặc không cần thiết.

Một số công cụ cho phép thực hiện fuzzing theo phương pháp này: Taof, GPF, ProxyFuzz, Peach Fuzzer...

Kiểm thử mờ dựa trên thế hệ

Kiểm thử mờ dựa trên thế hệ (Generation Based Fuzzing) hay còn gọi là kiểm thử mờ thông minh (Smart Fuzzing) là phương pháp kiểm thử mà dữ liệu fuzz được xây dựng mới hoàn toàn dựa trên các mô tả đặc điểm kỹ thuật, định dạng của mô hình đầu vào.

Đối với cách tiếp cận này [16]:

- Trường hợp thử nghiệm được tạo ra từ một số mô tả về các định dạng: RFC, các định dạng tài liệu.

- Tính dị thường được thêm vào mỗi điểm có thể có trong các đầu vào.

- Hỗ trợ kiến thức về giao thức nên cho kết quả tốt hơn so với fuzzing ngẫu nhiên.

- Có thể mất thời gian đáng kể để thiết lập.

Công cụ để thực hiện: SPIKE, Sulley, Mu-4000,...

Phân loại theo OWASP

The Open Web Application Security Project (OWASP) là một dự án phi lợi nhuận phát triển các dự án liên quan tới bảo mật ứng dụng Web hàng đầu thế giới, tổ chức này đưa ra 2 cách phân loại khác về Fuzzing hỗ trợ cho kiểm thử mờ các ứng dụng Web như sau:

Fuzzing đệ quy

Fuzzing đệ quy (Recursive Fuzzing) là phương pháp kiểm thử mà Fuzzer thực hiện duyệt qua bộ dữ liệu fuzz được xây dựng dựa trên tất cả các kết hợp của bộ chữ cái Alphabet.

Giả sử ta gởi một request là một chuỗi có dạng:

```
http://www.domain.com/2af8rb03
```

Nếu chọn "2af8rb03" như một một điểm đầu vào thì bộ dữ liệu fuzzing là một tập các chuỗi của bảng chữ cái Alphabet và số hệ thập lục phân (a-z, 0-9) thuộc loại fuzzing đệ quy. Như vậy, bộ dữ liệu fuzzing sẽ có 16^8 chuỗi và fuzzer sẽ thực hiện các request có dạng như sau:

```
http://www.domain.com/00000000
.....
http://www.domain.com/9999ffff
.....
http://www.domain.com/ffffffff
```

Fuzzing thay thế

Fuzzing thay thế (Replacive Fuzzing) là quá trình fuzzing mà một phần của yêu cầu được thực hiện thông qua việc thay thế nó bằng một tập giá trị mờ. Giá trị này được hiểu như một fuzz vector [16].

Xét trường hợp này:

```
http://www.example.com/2af8rb03
```

Để thực hiện kiểm tra sự tồn tại của lỗ hổng Cross Site Scripting (XSS), fuzzer thực hiện kiểm thử bằng cách gửi đến server các fuzz vector như sau:

```
http://www.example.com/>"><script>alert("XSS")</script>&
http://www.example.com/";!--"<XSS>=&{()}
```

Các fuzz vector được xây dựng dựa trên các mô tả về loại lỗ hổng cần kiểm thử. Tổng số lượng request mà fuzzer cần phải thực hiện phụ thuộc vào số lượng các fuzz vector xác định.

1.3.3. Ưu nhược điểm của Fuzzing

Ưu điểm

Như bất kỳ kỹ thuật kiểm thử an toàn nào khác, kiểm thử Fuzzing có ưu và nhược điểm của nó. Một trong những điểm mạnh của kiểm thử Fuzzing là các loại điểm yếu an toàn trong mã nguồn mà nó xác định được thường rất nghiêm trọng trong ứng dụng. Ví dụ, như tràn bộ đệm, lỗi số học số nguyên hay SQL injection, đều là những lỗ hổng cho phép một người sử dụng ác ý có thể nắm quyền kiểm soát hoàn toàn của một ứng dụng [7].

Những ưu điểm của kiểm thử fuzzing:

- Kết quả sử dụng kiểm thử Fuzzing hiệu quả hơn khi sử dụng các phương pháp kiểm thử khác. Kiểm thử Fuzzing tập trung vào việc sử dụng các giá trị đặc biệt như là đầu vào cho ứng dụng được kiểm thử, do đó giúp việc phát hiện các lỗi quan trọng mà có thể không được phát hiện bằng phương pháp tiếp cận dựa trên mô hình.

- Kiểm thử Fuzzing chỉ theo dõi các trường hợp mà kết quả trả về có sự bất thường hay hành vi không mong muốn. Điều này giúp nó có khả năng chạy hàng nghìn trường hợp thử nghiệm.

- Là một loại kiểm thử hộp đen nên có thể thực hiện kiểm thử cho các ứng dụng không biết mã nguồn bên trong, vì vậy nó thường tìm ra được các lỗ hổng nghiêm trọng và hầu hết là những lỗ hổng mà tin tặc thường khai thác.

- Các quá trình Fuzzing thường có lượng đầu vào thử nghiệm rất lớn, độ bao phủ rộng nên hiệu quả trong việc tìm kiếm các lỗ hổng.

Nhược điểm

Bên cạnh những ưu điểm giúp cho fuzzing được trở nên ưa chuộng thì nó cũng tồn tại những hạn chế:

- Khó có thể kiểm thử toàn diện và tìm thấy được tất cả các lỗi trong một chương trình lớn, những lỗi đòi hỏi kiểm thử viên phải thực hiện phân tích tĩnh.

- Fuzzing nằm trong phương pháp kiểm thử hộp đen nên không cung cấp nhiều kiến thức về hoạt động nội bộ của các phần mềm, vì vậy khó có thể tìm hiểu triệt để mà không hiểu chi tiết.

- Với chương trình có các đầu vào phức tạp để tìm ra các lỗi đòi hỏi phải tốn nhiều thời gian, bởi với mỗi biến đang fuzzing phải thử N vector fuzz và phải tạo ra một fuzzer đủ thông minh để phân tích các kết quả trả về.

- Fuzzing hoạt động không hiệu quả trong các chương trình có các kết quả trả về không có các mã lỗi hay các dấu hiệu bất thường.

1.3.4. Lựa chọn Fuzzing cho kiểm tra lỗ hổng website

Trong kiểm thử bảo mật website và kiểm thử bảo mật phần mềm không có quá nhiều điểm khác nhau nhưng đòi hỏi kiểm thử viên phải kết hợp với các kiến thức công nghệ bảo mật web, công nghệ mạng, lập trình web và kinh nghiệm thực tế về thâm nhập các hệ thống server. Vì vậy để xây dựng ứng dụng tự động phát hiện lỗ hổng bảo mật cho website, đòi hỏi phải có một phương pháp kiểm thử và phân tích đặc thù cho từng loại lỗ hổng trong bảo mật web.

Hiện nay, fuzzing là kỹ thuật được sử dụng rất nhiều trong việc kiểm thử cho các vấn đề về an ninh trong các phần mềm, hệ thống máy tính và các website dịch vụ. Ngoài ra, fuzzing là một trong những phương pháp phổ biến nhất được hacker sử dụng để tìm lỗ hổng của hệ thống.

Hệ thống Fuzzing sẽ gửi dữ liệu fuzz lên server chứa website hoặc truy cập thẳng vào đường link của website kèm theo dữ liệu gây lỗi, nhận dữ liệu từ website trả về và đưa vào bộ phân tích trước khi đưa ra kết luận về lỗ hổng. Dữ liệu fuzz là một tập hợp chứa dữ liệu nhận dạng, được kết hợp với một số thành phần của URL hoặc với những dữ liệu mà website xử lý.

Lựa chọn kỹ thuật Fuzzing, kiểm thử hộp đen để xây dựng ứng dụng quét lỗ hổng website, ta có thể quét bất kỳ một trang web hoặc một ứng dụng web, không phụ thuộc vào công nghệ hoặc các ngôn ngữ lập trình mà nó sử dụng. Nó chủ yếu kiểm thử một trang web hoặc một ứng dụng web mà không cần bất kỳ kiến thức về cách mà trang web làm việc, giống một kẻ tấn công thực sự. Nên khi các quản trị viên, những người trực tiếp quản lý và theo dõi tình hình hoạt động các website hoặc những người kiểm thử web sử dụng phương pháp này để kiểm thử sẽ giúp chương trình ngăn chặn trước được tấn công từ hacker.

Trong phạm vi đồ án, tôi sẽ đi sâu vào phân tích kỹ thuật fuzzing đặc thù cho việc kiểm tra, phát hiện lỗ hổng bảo mật ứng dụng web.

Kết luận chương 1

Chương đầu tiên đã trình bày toàn bộ cơ sở lý thuyết có liên quan tới website và kiểm thử website. Các nội dung này đã làm rõ và đưa ra được vấn đề nghiên cứu của toàn bộ đồ án, đó là lỗ hổng bảo mật website và kỹ thuật Fuzzing trong phát hiện các lỗ hổng bảo mật.

- Trình bày các khái niệm cơ bản có liên quan như website, lỗ hổng bảo mật, kiểm thử, fuzzing,.. Đây là các khái niệm cơ tạo nền tảng ban đầu cho các nghiên cứu và phát triển đồ án.

- Các loại lỗ hổng website, phần này đã trình bày về việc phân loại các lỗ hổng website, cách phát hiện và phòng chống với từng loại lỗ hổng. Đây là những đặc trưng phát hiện lỗ hổng cho việc xây dựng phần mềm. Phần này sẽ được nêu chi tiết trong chương 2.

- Kỹ thuật Fuzzing, phần này đã trình bày khái quát về lịch sử, phân loại và ưu nhược điểm của kỹ thuật Fuzzing trong kiểm thử bảo mật.

Từ những nội dung trình bày ở trên tôi đã trình bày lý do lựa chọn kỹ thuật Fuzzing cho các nghiên cứu trong kiểm thử bảo mật website. Các nội dung này là cơ sở lý thuyết cho việc nghiên cứu áp dụng kỹ thuật Fuzzing với các lỗ hổng web trong chương 2.

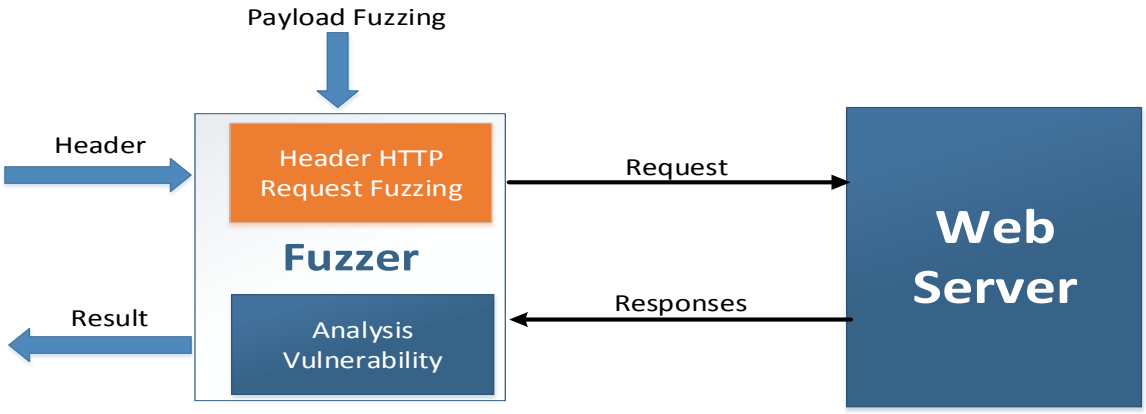
CHƯƠNG 2. KỸ THUẬT FUZZING TRONG KIỂM THỬ LỖ HỔNG BẢO MẬT WEBSITE

2.1. Trình bày cụ thể về mô hình, nguyên lý hoạt động của kỹ thuật Fuzzing

2.1.1. Mô hình Fuzzing

Một ứng dụng web thao tác với người dùng qua trình duyệt và sử dụng giao thức HTTP cổng 80 hoặc HTTPS cổng 443. Một hệ thống website sẽ chứa các điểm đầu vào của hệ thống bao gồm: các form cho người dùng nhập dữ liệu, các giá trị của biến được truyền trên các URL của website, các trường thông tin trong HTTP Headers [15].

Quá trình kiểm thử chủ yếu được thực trên các điểm đầu vào của hệ thống, cụ thể các trường dữ liệu của request headers được gửi qua phương thức truyền dữ liệu của HTTP, chủ yếu là phương thức GET, POST. Fuzzer sẽ phải thực hiện việc thu thập toàn bộ các điểm đầu vào của hệ thống trước khi thực hiện fuzzing. Mô hình được mô tả như hình 2.1:



Hình 2.1. Mô hình Fuzzing cho ứng dụng web

Mô hình kiểm thử fuzzing cho website cũng tương tự như mô hình fuzzing trong kiểm thử phần mềm, có 2 thành phần chủ yếu là fuzzer và web server:

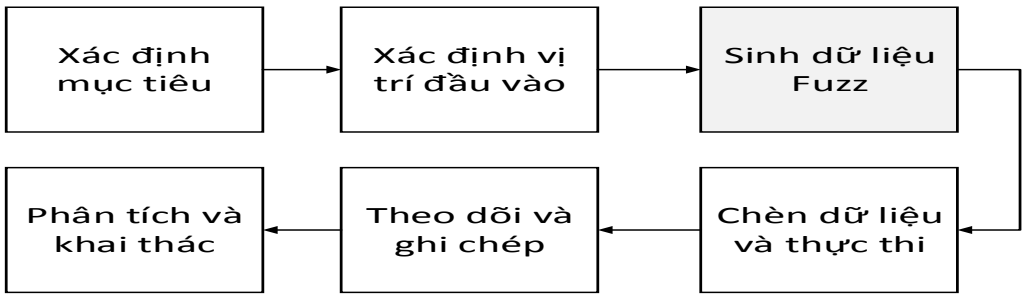
- Fuzzer là chương trình thực hiện kiểm thử tự động bằng kỹ thuật fuzzing.
- Web server là hệ thống máy chủ web được fuzzer thực hiện kiểm thử.

Với mỗi loại lỗ hổng website sẽ có những dữ liệu fuzz để thực hiện nhận dạng cho các ứng dụng đó, cụ thể chính là giá trị được thêm vào các biến trước khi

gửi tới hệ thống.Việc phân tích các Response trả về cũng được thực hiện dựa trên các đặc điểm của từng loại hệ thống Web Server có mã nguồn website được xây dựng trên loại ngôn ngữ lập trình nào.

2.1.2. Quy trình Fuzzing trong kiểm thử bảo mật website

Tùy thuộc vào các nhân tố khác nhau, việc lựa chọn cách tiếp cận Fuzzing có thể khác nhau. Tuy nhiên, về cơ bản Fuzzing có các giai đoạn như sau:



Hình 2.2. Quy trình Fuzzing

Các giai đoạn của Fuzzing được mô tả như sau:

1. Xác định mục tiêu

Mục tiêu được đánh giá có nguy cơ rủi ro cao gồm các lỗ hổng do lỗi của người lập trình hệ thống: SQL Injection, Code Injection, Cross Site Scripting, URL Redirect… Hoặc các lỗi do việc cấu hình hệ thống không an toàn như để đường dẫn vào trang quản trị hệ thống là mặc định, tài khoản mặc định…

Fuzzer cần có đầu vào là địa chỉ IP hay domain của website cho việc xác định đối tượng fuzzing.

2. Xác định vị trí đầu vào

Một ứng dụng web nhận các yêu cầu, dữ liệu từ người dùng thông qua các URL hoặc trường biểu mẫu. Các yêu cầu của người dùng được chuyển thành các gói tin theo giao thức HTTP và đưa tới Web Server. Các trường trong phần tiêu đề của gói tin sẽ được chèn dữ liệu fuzzing.

Fuzzer thực hiện cuộc thử nghiệm thông qua phần tiêu đề khác nhau của giao thức HTTP của website đó, do vậy, việc thực hiện thu thập được toàn diện các điểm đầu vào là các URL, trường biểu mẫu giúp cho Fuzzer đi vào được các ngóc ngách của website và thực hiện fuzzing.

3. Sinh dữ liệu Fuzz

Mục tiêu của fuzzing là cung cấp dữ liệu bất thường thông qua đầu vào cho mục tiêu mà nó thường không mong đợi nhận được.

Giai đoạn này được xem là quan trọng nhất trong fuzzing.Ngày nay, nó được nghiên cứu và phát triển đáng kể bởi các nhà khoa học.Mục đích của một fuzzer là để kiểm tra sự tồn tại của lỗ hổng bảo mật trên các đầu vào của ứng dụng. Do đó, fuzzer phải tạo ra dữ liệu thử nghiệm mà ở các mức độ mà sau đó nó có thể được thông qua vào mục tiêu ứng dụng đầu vào. Dữ liệu được tạo ra có thể dạng file nhị phân (Binary files), file văn bản (Text files) được tạo ra lặp đi lặp lại vào thời điểm bắt đầu của mỗi lần test [16].

4. Chèn dữ liệu fuzz và thực thi các truy vấn

Server nắm bắt dữ liệu từ tiêu đề được gửi bởi khách hàng để thực hiện một số nhiệm vụ ở phía máy chủ.Các dữ liệu lần lượt được chèn vào các trường phần tiêu đề của HTTP Request.

Các trường tiêu đề sau đây có thể được chèn dữ liệu fuzz:

- Query parameters
- Path
- Accept language
- Cookie
- User-Agent
- POST Data

Ví dụ, ứng dụng sẽ dựa vào giá trị user-agent là admin hay user để quyết định nội dung sẽ được chuyển lại cho người dùng.Nếu ứng dụng không thực hiện xác nhận hợp lệ đầu vào chuỗi user-agent, nó có thể bị kẻ tấn công khai thác Sau khi tham số đầu vào và dữ liệu fuzzing đã sẵn sàng, đó là lúc để gửi nó tới đích.

5. Theo dõi và ghi chép

Khi bộ fuzzer bắt đầu fuzzing, fuzzer theo dõi mục tiêu và đợi cho ứng dụng gặp phải tình trạng lỗi hay phản ứng bất thường do những dữ liệu không thích hợp được truyền đến. Tình trạng lỗi và dữ liệu gây ra lỗi sẽ được ghi lại.

Dựa vào các thông báo lỗi được phản hồi lại bởi ứng dụng và mã HTTP. Mã trạng thái 403 chỉ ra rằng tài nguyên mà bạn đang cố gắng truy cập bị hạn chế và bạn không được phép xem nó, mã lỗi 404 nói rằng trang web bạn đang cố gắng truy cập không khả dụng và mã lỗi 500 cho biết lỗi máy chủ nội bộ.

6. Phân tích và khai thác

Giai đoạn này, không đơn thuần các fuzzers phát hiện các lỗ hổng qua việc fuzzing mà phải định nghĩa các lỗ hổng được phát hiện. Điều này có ý nghĩa hết sức quan trọng trong việc phân tích và báo cáo lỗ hổng.Để báo cáo lỗ hổng đòi hỏi Fuzzer hiểu rõ về hoạt động xử lý và được tích hợp vào sự kiện phân loại lỗ hổng tự động.

Ứng dụng web sẽ phản hồi lại với các thông báo lỗi của ứng dụng như thông báo lỗi SQL cho lỗ hổng SQL Injection, các đặc trưng của dữ liệu fuzz trong lỗ hổng XSS, truy nhập thành công cho lỗ hổng LFI,... Bằng cách sử dụng này, Fuzzer căn cứ vào những đặc trưng quy định trước mà phát hiện lỗ hổng.

2.2. Cơ chế phát hiện lỗ hổng bảo mật với kỹ thuật Fuzzing

2.2.1. Thu thập các điểm đầu vào

Để có thể kiểm thử và phát hiện các lỗ hổng của hệ thống, fuzzer cần xác định được các điểm đầu vào của hệ thống. Các điểm đầu vào thường là các đường dẫn, các form nhập dữ liệu của hệ thống và các thông tin trên các trường của header của gói tin HTTP.

Từ tập hợp các điểm đầu vào của website, fuzzer mới có thể thực hiện kiểm tra và phát hiện các lỗ hổng tồn tại trên hệ thống.

2.2.1.1. Cơ chế trích xuất URL từ mã HTML

HTML là ngôn ngữ cho giao diện của website, chúng đánh dấu bằng thẻ (tag) và sử dụng các thẻ khác nhau để định dạng nội dung của một trang web. Những thẻ này được chứa trong hai dấu ngoặc đơn <tên thẻ>. Ví dụ, thẻ <html> có thẻ đóng tương ứng là </html> và thẻ <body> có thẻ đóng tương ứng là </body> ...

Thu thập thông tin (web crawler) là quá trình lấy thông tin từ website, trích xuất ra những thông tin người sử dụng cần, đồng thời cũng tìm những liên kết có

trong trang web đó và tự động truy cập vào những đường dẫn đó. Nó lần lượt đi từ liên kết này đến các liên kết khác và thu thập tất cả các dữ liệu của toàn bộ website.

Nguyên lý thu thập các điểm đầu vào của website cũng tương tự như vậy, nó là quá trình thu thập các URL và form nhập dữ liệu dựa trên việc phân tích các mã HTML trả về sau mỗi yêu cầu. Đơn giản nó là quá trình bóc tách từng thẻ trong mã HTML trả về để tìm các URL của website trong đó.

Quá trình thu thập đầu vào dựa trên các thuộc tính và thẻ trong HTML, danh sách các thuộc tính này được đưa ra trong bảng 2.1:

Bảng 2.1.Các thuộc tính và các thẻ đi kèm có chứa các URL của hệ thống

Thuộc tính	Các thẻ có chứa thông tin URL
href	Nằm trong mã HTML. Các thẻ mà chứa thuộc tính href thì giá trị của href chính là một URL.
src	Nằm trong mã HTML, mã javascript.
site	Nằm trong mã HTML, giá trị của biến có chứa site là một đường dẫn.
action	Nằm trong mã HTML, nằm trong thẻ <form>.
location	Nằm trong mã Javascript.
http://	Có chứa thông tin “http://” cũng xác định đường dẫn.

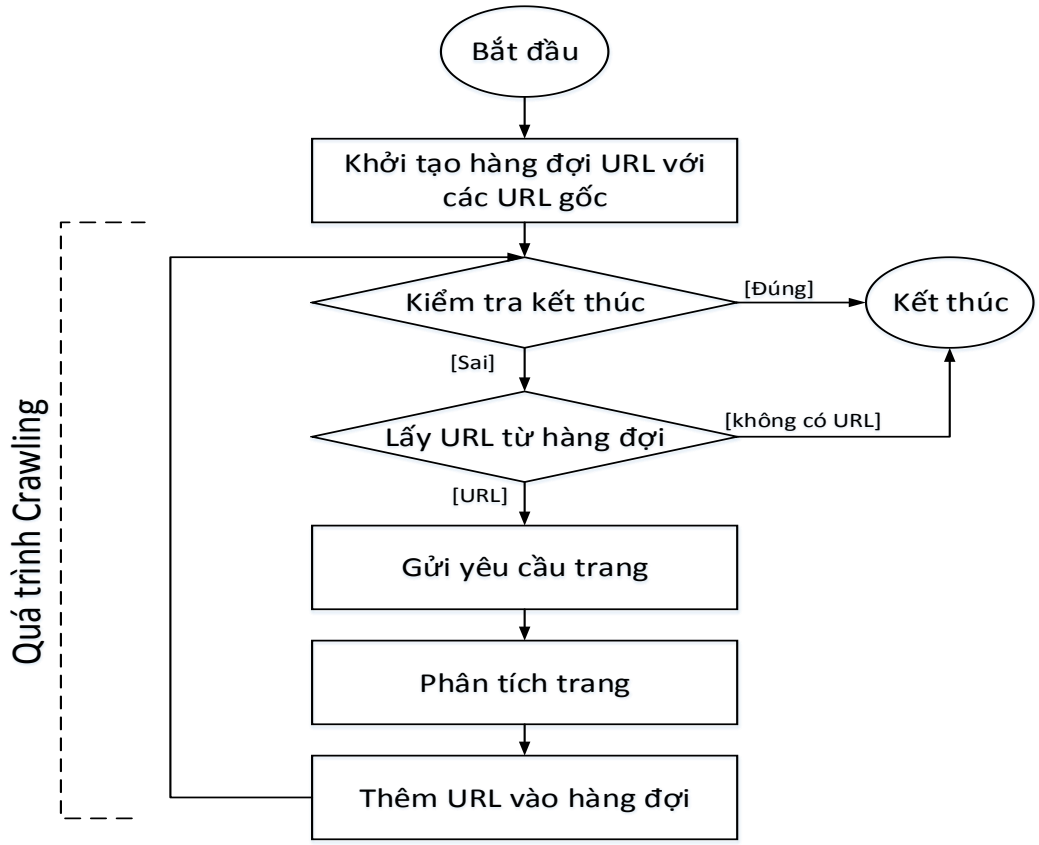
Thu thập các form trong các thẻ <form> của mã HTML, các thẻ <input> có các thuộc tính name trong form là các biến mang giá trị đầu vào cho liên kết trong thuộc tính action.

Với javascript, thực hiện tìm kiếm các liên kết dựa trên biểu thức chính quy (Regular Expression).

2.2.1.2. Phương pháp thu thập

Web crawler

Web crawler là các chương trình khai thác sơ đồ cấu trúc của web bằng cách chuyển từ trang web này sang trang web khác.



Hình 2.3. Sơ đồ của một crawler

Ban đầu, động cơ chủ yếu thúc đẩy việc thiết kế các web crawler là việc lấy ra nội dung các trang web và thêm chúng hoặc thể hiện của chúng vào các kho chứa cục bộ. Các kho chứa này, sau đó sẽ đáp ứng các ứng dụng cụ thể chẳng hạn một hệ thống tìm kiếm trên Web. Ở dạng đơn giản nhất, một chương trình crawler sẽ bắt đầu từ một địa chỉ nguồn khởi đầu nào đó và sử dụng các liên kết ngoài trong trang web đó để mở rộng ra các trang tiếp theo. Quá trình này tiếp tục với các trang web mới, các trang này lại cung cấp các liên kết ngoài khác để đi theo. Cứ như vậy cho tới khi đạt tới một số lượng trang web xác định hoặc một mục tiêu nào đó đạt được. Phía sau sự mô tả một cách đơn giản này là một mảng các vấn đề phức tạp có liên quan như việc kết nối mạng, các tiêu chuẩn về một URL, việc duyệt các trang HTML và cách thức để giao tiếp với các Server ở xa. Trên thực tế, các thế hệ web crawler gần đây, có thể coi là một trong những phần phức tạp nhất của hệ thống mà nó đi kèm [2].

Hình 2.3 biểu diễn đồ thị của một crawler tuần tự cơ bản. Một chương trình crawler bao gồm một danh sách các URL chưa được thăm gọi là hàng đợi URL. Danh sách này được khởi tạo bởi các URL hạt nhân đã được cung cấp bởi người dùng hoặc các chương trình khác. Mỗi vòng lặp crawling bao gồm:

- Lấy ra URL cần được duyệt tiếp theo từ hàng đợi URL, nạp trang web tương ứng với URL đó bằng giao thức HTTP.

- Duyệt trang web vừa tải về để lấy ra các từ URL và các thông tin mà ứng dụng cần.

- Cuối cùng, thêm các trang URL chưa được thăm vào hàng đợi URL và thực hiện vòng lặp tiếp theo.

Trước khi các URL được thêm vào frontier chúng sẽ được gán cho một độ đo thể hiện đánh giá hiệu quả khi thăm trang web tương ứng với URL đó.

Quá trình crawling có thể kết thúc khi một số lượng nhất định các trang web đã được tải. Nếu chương trình crawler đã sẵn sàng để duyệt một trang web khác và trạng thái của frontier là rỗng, một tín hiệu trạng thái kết thúc (dead-end) sẽ được gửi cho crawler. Chương trình crawler sẽ không có trang web mới để tải và nó sẽ dừng lại [2].

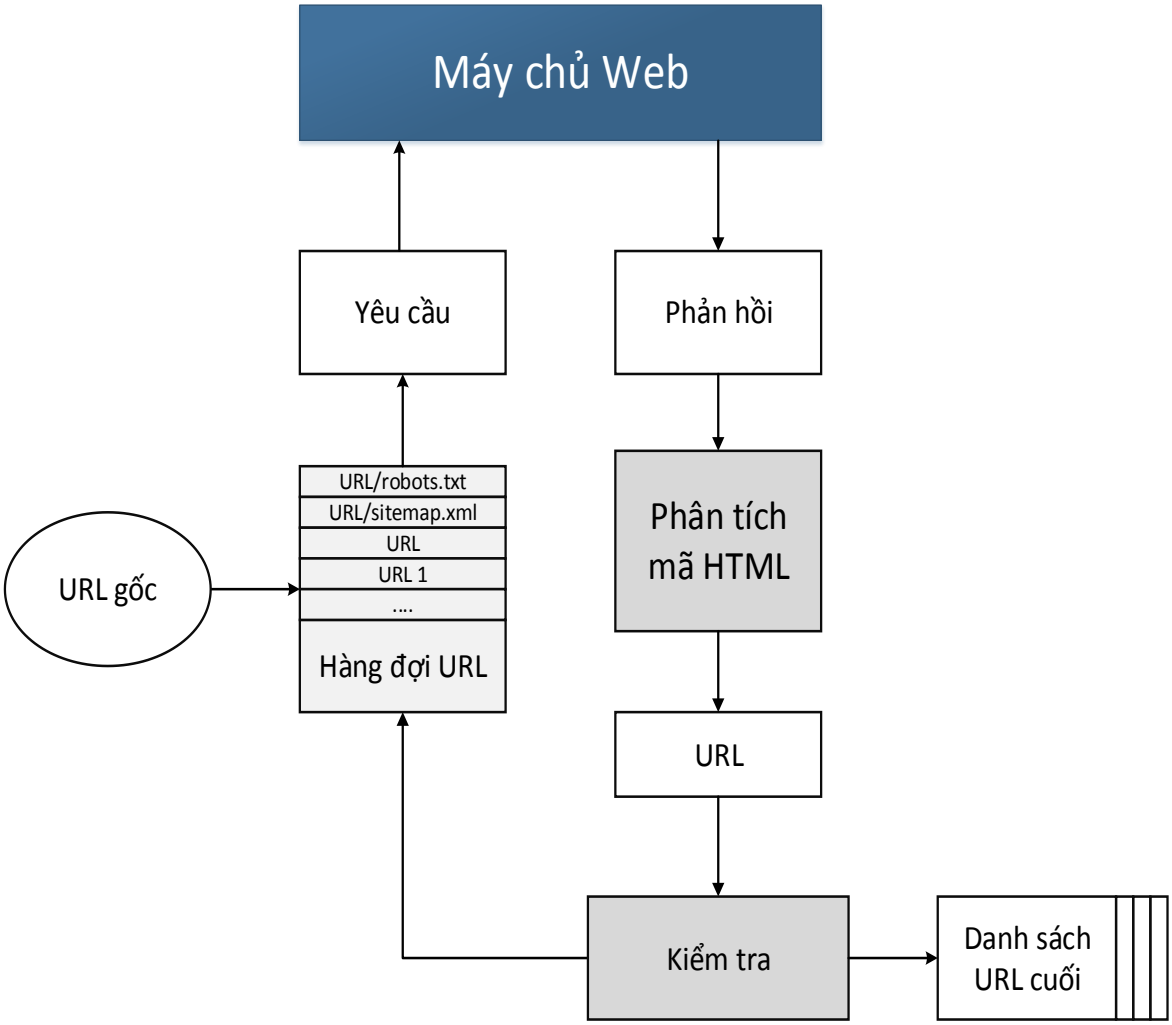
Quy trình thu thập điểm đầu vào

Một chương trình Fuzzer cần phải có tập hợp các điểm đầu vào (nơi thực hiện chèn dữ liệu fuzz) để phục vụ cho quá trình fuzzing và tìm kiếm lỗ hổng. Dựa trên mô hình web crawler, nguyên tắc thu thập toàn bộ các điểm đầu vào của một website cũng như vậy, hay nói cách khác Crawler là một phần của Fuzzer nhưng dữ liệu cần thu thập không chỉ URL mà cần thu thập các biến và dữ liệu truyền trên mỗi đường dẫn đó.

Ban đầu Fuzzer thực hiện duyệt trang web với URL gốc, sau khi trang web đã được tải về, Fuzzer duyệt nội dung của nó để lấy ra các thông tin sẽ được nạp trở lại và giúp định hướng việc đi theo các đường dẫn tiếp theo. Việc duyệt nội dung đơn giản chỉ bao hàm việc trích ra các URL mà trang web chỉ tới hay có thể bao gồm các bước để chuẩn hóa các URL được lấy ra.

- Input: Đường dẫn gốc của website (http://www.domain.com).
- Output: Toàn bộ các liên kết trong website (danh sách URL cuối).

Mô hình thu thập URL theo mã HTML được mô tả như trong hình 2.4.



Hình 2.4. Mô hình thu thập URL theo mã HTML

Hàng đợi URL là một hàng đợi chứa danh sách các trang web chưa được thăm hay nó là một danh sách chứa các nút chưa được mở rộng. Chúng được lưu trữ như là các cấu trúc dữ liệu trong bộ nhớ trong. Hàng đợi này có kích thước giới hạn nên cần một cơ chế để quyết định URL nào cần bị loại bỏ khi số lượng trong hàng đợi này đạt tới giới hạn và nó có thể bị đầy nhanh hơn so với số lượng trang web được duyệt. Với số lượng ước tính trung bình có khoảng 7 đến 10 liên kết trong một

trang web thì việc hàng đợi có thể bị đầy nhanh hơn là điều dễ hiểu. Do vậy, điều cần thiết là phải đảm bảo rằng không thêm các url lặp lại vào hàng đợi, cần có một cơ chế tìm kiếm và trích xuất các URL mới trong nội dung trang web đang được duyệt.

Tại một thời điểm, chương trình có thể gặp vấn đề thu được một số lượng lớn các URL khác nhau nhưng chúng cùng trỏ về một trang web. Có thể giải quyết vấn đề này bằng cách loại bỏ các trang web không nằm trên một tên miền xác định. Điều này đảm bảo rằng mọi chuỗi gồm k URL liên tiếp được lấy ra thì chỉ trích xuất một địa chỉ URL chuẩn hóa. Điều này sẽ tránh được việc phải truy cập và lấy nội dung một trang web quá nhiều lần và nội dung các trang web được tải có xu hướng khác biệt nhau.

Để thu được nội dung trang web, cần phải gửi một yêu cầu HTTP tới trang web yêu cầu và đọc các đáp ứng. Fuzzer cần phải có một thời gian quy định trước để tránh cho việc lãng phí quá nhiều thời gian để thực hiện truy cập tới máy chủ web có độ trễ cao hay kích thước nội dung web quá lớn. Trên thực tế, chương trình cần phải loại bỏ các tệp tin không liên quan có nội dung như ảnh, nhạc,... Chúng cũng duyệt các header để lấy mã trạng thái của trang web và lưu thời gian trễ để xác định thời gian cập nhật của website.

Các bước thu thập URL một hệ thống website theo mô hình 2.4:

- Bước 1: Khởi tạo hàng đợi với 1 phần tử là URL gốc. Khởi tạo danh sách URL cuối để lưu các URL cuối cùng của hệ thống.

- Bước 2: Fuzzer thêm vào URL gốc /robots.txt, /sitemap.xml,... và đưa vào hàng đợi. Thực hiện việc lấy URL từ hàng đợi và gửi yêu cầu đến web server.

- Bước 3: Phân tích mã HTML trả về từ Server và lọc lấy URL trong các thuộc tính của các thẻ trong mã HTML.

- Bước 4: Nhận URL thu được từ bước 3 và thực hiện kiểm tra (URL check) như sau:

+ Đưa vào trong hàng đợi nếu URL này không trùng hoặc tương đương với URL nào trong các URL đã duyệt và các URL trong hàng đợi.

+ Đưa vào trong danh sách URL cuối nếu URL này không trùng hoặc tương đương với URL nào trong danh sách các URL đã thu được (danh sách URL cuối).

+ Loại bỏ trong các trường hợp còn lại.

- Bước 5: Kiểm tra nếu hàng đợi rỗng thì kết thúc. Nếu hàng đợi không rỗng thì quay lại B2.

Mô hình được xây dựng đã dẫn tới một số vấn đề và điều đó cần thiết phải có các giải pháp nhằm giải quyết các vấn đề trong quá trình hoạt động của chương trình:

1. Thời gian giới hạn: Nếu server không trả lời thì chương trình sẽ bị đóng băng. Vì thế Fuzzer cần xử lý trường hợp máy chủ web không trả lời sau 1 khoảng thời gian quy định bằng cách đơn giản là quy định thời gian chờ.

2. Truy cập lặp lại: Xảy ra khi fuzzer thực hiện gửi yêu cầu lặp lại trang web đã được xử lý trước đó, chương trình có thể bị rơi vào vòng lặp vĩnh viễn. Vì thế cần phải có phương pháp đánh dấu những liên kết đã xử lý.Đơn giản nhất là lưu lại liên kết đã xử lý, trước khi thêm vào hàng đợi một liên kết mới thì so sánh với những liên kết đã xử lý trước.

3. Bỏ sót đường dẫn: Với việc chỉ một đường dẫn gốc duy nhất làm cho việc quét khó khăn hơn hoặc bỏ sót các đường dẫn mà nó không liên kết với đường dẫn ta đang có. Vì vậy, Fuzzer phải thực hiện mở rộng đường dẫn gốc bằng cách lấy các cấu trúc thư mục tại các tệp tin mặc định của website: robots.txt, sitemap.xml,...

Tệp tin robots.txt chứa các thư mục và tệp tin mà website quy định cho phép hay cấm các chương trình (bot) của các công cụ tìm kiếm đánh chỉ mục một khu vực nào đó trong website. Nguyên tắc này cung cấp cho các nhà quản trị web thông báo về quyền truy nhập tệp tin trên máy chủ.Nó cung cấp các chính sách truy cập cho các User-agent khác nhau. Vì vậy ta có thể lấy các đường dẫn tương ứng thông qua việc phân tích mã nguồn của tệp tin robots.txt.

Sitemap (còn được gọi là sơ đồ của một trang web) là một tập tin văn bản có chứa tất cả các URL của một trang web.Sitemap còn có thể chứa các siêu dữ liệu về mỗi URL thông báo sẽ được gửi đến cho bạn khi nó mới được cập nhật.

```
User-agent: *
Disallow: /ajax/
Disallow: /blog/
Disallow: /cache/
Disallow: /download/
Disallow: /embed/
Disallow: /html5/
Disallow: /json/
Disallow: /link/
Disallow: /log/
Disallow: /login/
Disallow: /share/
Disallow: /suggest/
Disallow: /test/
Disallow: /thongbao/
Disallow: /vip-free/
Disallow: /xhr/
Disallow: /xml/
Sitemap: http://mp3.zing.vn/sitemap.xml
```

Hình 2.5. Các đường dẫn từ tệp tin robots.txt

4. Đường dẫn tương đương: Liên tục truy xuất tới tất cả các đường dẫn tương tự nhau mà chỉ khác giá trị truyền vào của biến trên đường dẫn. Điều này làm tăng số lượng yêu cầu gửi không cần thiết.

Cấu trúc một đường dẫn:

http://buigiap.com/path1/index.php?a=1&b=2#endpage

Giao thức	Tên miền	Cổng	Đường dẫn	Truy vấn	Phân mảnh
http	buigiap.com	80	path1/index.php	var1=a & var2=b	endpage

Đường dẫn tương đương là các đường dẫn hoàn toàn giống nhau về các thành phần trên nó mà chỉ khác về các giá trị được truyền vào. Phần truy vấn và phân mảnh là khác nhau, tuy nhiên trong phần truy vấn các biến là giống nhau. Trong quá trình fuzzing, điều này giúp làm tránh các trường hợp kiểm thử các đường dẫn cùng mang lại một kết quả như nhau.

Fuzzer cần phải có bước kiểm tra xem trong danh sách URL cuối xem có tồn tại URL tương đương của nó không, nếu không tồn tại thì mới thực hiện việc thêm URL này vào URL cuối. Ví dụ xét 2 đường dẫn như sau:

Liên kết 1:

```
http://www.domain.com/index.php?var1=1&var2=abc#endpage
```

Liên kết 2:

```
http://www.domain.com/index.php?var1=2&var2=cde#endpage
```

Đây là 2 đường dẫn tương đương bởi có các thành phần giống nhau về giao thức, tên miền, đường dẫn và các biến.

Trong quá trình kiểm thử, fuzzer thực hiện chèn dữ liệu fuzz vào các biến trên đường dẫn, nên với các đường dẫn tương đương sau khi chèn dữ liệu fuzz thì chúng hoàn toàn giống nhau và kết quả cho cuộc tấn công kiểm thử là như nhau. Vì vậy, việc lọc và loại bỏ các đường dẫn tương đương là hết sức cần thiết trong quá trình thực hiện fuzzing.

Một số ví dụ về việc lọc và loại bỏ đường dẫn tương đương khi thực hiện fuzzing được trình bày chi tiết trong bảng 2.2:

Bảng 2.2. Ví dụ trong fuzzing đường dẫn tương đương

URL	Nội dung đường dẫn
URL1	http://www.domain.com/index.php?var1=1
URL2	http://www.domain.com/index.php?var1=2
URL3	http://www.domain.com/index.php?var1=3
Fuzzing 1	http://www.domain.com/index.php?var1=[Fuzz]
URL4	http://www.domain.com/index.php?action=home&var1=1
URL5	http://www.domain.com/index.php?action=news&var1=2
URL6	http://www.domain.com/index.php?action=main&var1=3
Fuzzing 2	http://www.domain.com/index.php?action=[Fuzz]&var1=[Fuzz]

Với các đường dẫn tương đương chúng được thay thế như sau:

(URL1, URL2, URL3) =>Fuzzing 1

(URL4, URL5, URL6) => Fuzzing 2

Việc loại bỏ các đường dẫn tương đương giúp cho quá trình kiểm thử website giảm thời gian đáng kể và giảm tổn hao tài nguyên của hệ thống.

2.2.2. Nguyên lý chèn dữ liệu fuzz

2.2.2.1. Chèn dữ liệu vào phương thức GET

Đường dẫn (URL) có 2 loại chính:

- Loại 1: URL có chứa các biến truyền giá trị vào cho web.

- Loại 2: URL không chứa các biến truy vấn mà chỉ trỏ đến các tệp tin trên hệ thống.

Bảng 2.3. Chèn dữ liệu fuzzing vào URL

Loại URL 1	
URL	http://localhost/index.php?var1=a
URL Fuzzing	http://localhost/index.php?var1=[Fuzzing]
Ví dụ (XSS)	http://localhost/index.php?var1=”onmouseover=alert(“signature”) foo=”
Loại URL 2	
URL	http://localhost/index.php
URL Fuzzing	http://localhost/index.php?[Fuzzing] hoặc phải đoán biến (id, act, page...) http://localhost/index.php?id=[Fuzzing]
Ví dụ (XSS)	http://localhost/index.php?”onmouseover=alert(“XSS”)foo=” http://localhost/index.php?id=”onmouseover=alert(“XSS”) foo=” http://localhost/index.php?act=”onmouseover=alert(“XSS”) foo=”

Với từng loại lỗ hổng, kiểm thử viên cần xây dựng riêng những tập dữ liệu fuzz cho từng loại lỗ hổng khai thác. Bộ dữ liệu có chất lượng và độ bao phủ càng cao thì càng dễ phát hiện các lỗ hổng. Để thực hiện việc kiểm tra và phát hiện các lỗ hổng, phải chèn tất cả các dữ liệu fuzzing vào tất cả các điểm đầu vào hệ thống thu được trước khi thực hiện việc gửi yêu cầu. Nguyên tắc chèn fuzzing vào các URL:

2.2.2.2. Chèn dữ liệu vào phương thức POST

Đối với các đường dẫn thu được là FORM POST (sử dụng phương thức POST để truyền dữ liệu) chúng ta có thể thực hiện hoàn toàn tương tự, dữ liệu Fuzzing được chèn vào các biến trong Form Data của gói tin request.

Nguyên tắc chèn dữ liệu vào data post:

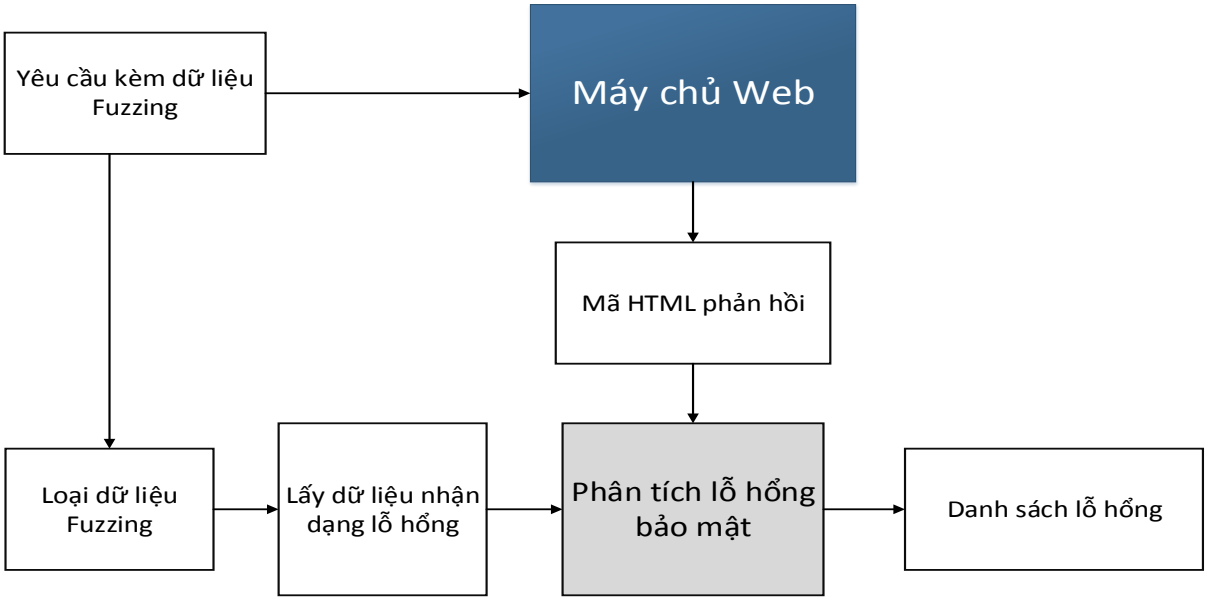
Bảng 2.4. Chèn dữ liệu fuzzing vào phương thức POST

Kiểu FORM POST	
URL	POST /index.php HTTP/1.1 Host: localhost User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) Accept: text/html; charset=utf-8 Accept-Encoding: gzip, deflate Accept-Language: vi act=a&id=1
URL Fuzzing	POST /index.php HTTP/1.1 Host: localhost User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) Accept: text/html Accept-Encoding: gzip, deflate Accept-Language: vi act=[Fuzzing]&id=[Fuzzing]
Ví dụ (SQLi)	POST /index.php HTTP/1.1 Host: localhost User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) Accept: text/html Accept-Encoding: gzip, deflate Accept-Language: vi act=[Fuzzing]&id=-1 or 1=1-- -

2.2.3. Phương pháp phát hiện lỗ hổng bảo mật

Sau khi đã thu thập được các điểm đầu vào của hệ thống, Fuzzer bắt đầu xử lý danh sách các mục tiêu đầu vào. Fuzzer thực hiện duyệt từng trường dữ liệu fuzz của từng loại lỗ hổng với từng biểu mẫu yêu cầu. Đối với mỗi biểu mẫu web, các địa chỉ được trích (hay mục tiêu) và phương thức (GET hoặc POST), chúng căn cứ dựa trên phân loại trong quá trình thu thập được sử dụng để gửi các nội dung yêu cầu. Sau một cuộc tấn công kiểm thử, thành phần phân tích lỗ hổng bảo mật thực hiện phân tích kết quả trả về với các phản ứng của máy chủ. Một thành phần phân tích sử dụng tiêu chuẩn đáp ứng tấn công cụ thể và từ khóa để tính toán giá trị tin cậy để đưa ra các quyết định về một cuộc tấn công có thành công và tồn tại lỗ hổng hay không.

Mô hình phát hiện lỗ hổng trong Fuzzing được mô tả như hình dưới:



Hình 2.6. Mô hình phân tích phát hiện lỗ hổng

Các bước thu thập URL một hệ thống website:

Bước 1: Xác định loại Fuzzing đang được thực hiện cho loại lỗ hổng nào, từ đó, lấy ra các mẫu nhận dạng loại lỗ hổng đó.

Bước 2: Nhận HTTP Response từ Web Server và thực hiện phát hiện lỗ hổng bảo mật bằng cách phân tích, đối sánh kết quả trả về với các mẫu nhận dạng loại lỗ hổng. Nếu trùng với mẫu nhận dạng thì kết luận có tồn tại lỗ hổng.

Bước 3: Đưa ra báo cáo lỗ hổng bảo mật website.

Căn cứ dựa trên loại lỗ hổng đang được kiểm tra mà Fuzzer thực hiện tìm kiếm các đặc trưng của lỗ hổng đó trong kết quả trả về. Fuzzer cũng cần phải duyệt header và lưu thời gian trả về để xác định trạng thái của website đó.Việc kiểm tra các mã đặc trưng và ngoại lệ là rất quan trọng trong quá trình phân tích và phát hiện lỗ hổng.Thu thập và thống kê về thời gian timeout và các mã trạng thái cũng rất hữu ích cho việc xác định các vấn đề nảy sinh.

Các ngôn ngữ lập trình hiện đại như C#, Java cung cấp các cơ chế đơn giản cùng nhiều giao diện lập trình để tải các trang web. Nhưng việc sử dụng ngôn ngữ bậc cao phải hết sức mềm dẻo bởi có thể sẽ khó tìm ra các lỗi ở bậc thấp.

2.2.4. Phát hiện lỗ hổng dựa trên đặc trưng

Với mỗi loại lỗ hổng chúng mang những đặc trưng khác, vì vậy cần phải có những cơ chế phân tích kết quả trả về khác nhau. Fuzzer dựa trên những đặc điểm nhận dạng về lỗ hổng mà đánh giá và đưa ra kết luận một giao dịch có tồn tại lỗ hổng hay không.

Dựa trên những đặc điểm của từng loại lỗ hổng mà bộ dữ liệu fuzz và phương pháp kiểm thử áp dụng cho chúng. Kỹ thuật của từng phương pháp này được mô tả như ở bảng 2.5:

Bảng 2.5. Cơ chế phát hiện các lỗ hổng hệ thống

Phương pháp	**Lỗ hổng áp dụng**	**Mô tả kỹ thuật**
Dựa trên thông báo mã trạng thái của hệ thống	File Inclusion, Path Traversal, Configuration, …	Dựa vào kết quả thông báo lỗi của hệ thống ta có thể biết được hệ thống có thực thi đoạn dữ liệu fuzzing đầu vào hay không hoặc URL đó có tồn tại hay không. Ví dụ: Khi tìm một URL mặc định của hệ thống. Nếu nó trả về giá trị lớn hơn hoặc bằng 200 và nhỏ hơn 300. Thì có nghĩa là URL đó là tồn tại.
Dựa trên các lỗi của hệ thống	SQL Injection, Xpath Injection, Code Injection, LDAP Injection, …	Với từng loại lỗ hổng tương ứng fuzzer phải phân tích và tìm kiếm các thông báo lỗi tương ứng với request trong các mã HTML trả về. Ví dụ: Các thông báo lỗi về SQL Injection được mô tả trong bảng 2.6.
Dựa trên sư	Cross Site	Chữ ký được đính kèm với dữ liệu fuzzing,

xuất hiện của chữ ký	Script	nếu dữ liệu fuzzing này được thực thi sẽ xuất hiện chữ ký đó. Ví dụ: Dữ liệu thực thi Fuzzing là: <script>alert("XSS");</script> Nếu đoạn script này được thực thi sẽ có hộp thoại thông báo chữ ký "XSS".
Dựa trên việc so sánh các kết quả của HTML nhận được.	Blind SQLi, Blind Xpath Injection, Blind Command Injection, …	Thực hiện so sánh kết quả của 2 request khi thực hiện chèn 2 đoạn dữ liệu fuzzing mang giá trị đối lập nhau. Ví dụ: SQL Injection với 2 mẫu đầu vào là: -1' or 1=1 -- - mang giá trị đúng -1' or 1=2 -- - mang giá trị sai
Dựa trên việc kiểm tra thời gian phản hồi từ hệ thống.	Blind SQL Injection	Thực hiện kiểm tra thời gian nhận được phản hồi của máy chủ sau khi thực thi yêu cầu có chèn dữ liệu fuzzing. Ví dụ: SQL Injection sử dụng kỹ thuật Time Base: ' and sleep(10) -- - Với việc chèn vào đoạn fuzzing trên nếu ứng dụng có lỗi Blind SQL injection trong biến này thì hệ thống sẽ bị sleep(10) giây.

Ví dụ về mẫu nhận dạng của lỗ hổng SQL Injection dựa trên kỹ thuật nhận dạng lỗi trả về từ hệ thống. Bảng 2.6 bao gồm những cụm ký tự đặc trưng cho tất cả các loại hệ thống là Apache, ISS, Tomcat,.. mà nó có thể trả về.

Bảng 2.6. Các mẫu thông báo lỗi từ SQL

Đầu vào	Các thông báo lỗi từ hệ thống
' " \xBF	1. mysql_fetch_array | mysql_num_rows | mysql_fetch_array | Error at line near | You have an error in your SQL syntax | mySQL error with query | on MySQL result index | mysql_query | supplied

') ") or 1=1 ') or 1 %27	argument is not a valid MySQL result resource in. 2. SQL command not properly ended | SQLException | Supplied argument is not a valid PostgreSQL result | Syntax error in query expression | The error occurred in | Unterminated string constant | invalid query | is not allowed to access. 3. \[Microsoft\]\[ODBC Microsoft Access Driver\] 4. ASP\.NET is configured to show verbose error messages | Microsoft OLE DB Provider for ODBC Drivers[\S\s]*error 5. java\.sql\.SQLException\: Syntax error or access violation 6. XPathException 7. Dynamic SQL Error 8. DB2 SQL error\:

2.2.5. Phát hiện lỗ hổng dựa trên cấu hình

Trong mỗi hệ thống luôn có những thành phần cần được bảo mật và nó ảnh hưởng tới sự sinh tồn của cả hệ thống. Một hệ thống website cũng như vậy, khi triển khai cần phải bảo mật những thông tin quan trọng, đặc biệt là các thành phần trang quản trị. Một số thành phần quan trọng và nguyên nhân cần làm lộ lọt thông tin hệ thống:

- Các đường dẫn tới tệp tin cấu hình hệ thống hiển thị trong mã HTML hay để mặc định khi sử dụng mã nguồn mở.

- Các tệp tin cấu hình hệ thống, tệp tin dự phòng không được phân quyền.

- Các đường dẫn mặc định trong các nền tảng ứng dụng (phpMyadmin, manager.html…)

- Danh sách các thư mục và tệp tin hiển thị ra bên ngoài do không có tệp tin index hay default mặc định.

- Cấu hình tài khoản, mật khẩu mặc định, không thay đổi so với ban đầu.

Một tập dữ liệu fuzz bao gồm tên các thư mục và tệp tin mặc định của hệ thống nó tạo thành các đường dẫn ngẫu nhiên để tìm kiếm chính xác trang quản trị, các tệp tin mở rộng dự phòng, các tệp tin cấu hình, mặc định của hệ thống. Khi đó

để phát hiện những lỗi về cấu hình này, fuzzer cần phải gửi các yêu cầu kèm theo các trường nằm trong bộ dữ liệu fuzz của lỗ hổng này, sau đó thực hiện kiểm tra mã trạng thái trả về của hệ thống và mã nguồn HTML. Mã trạng thái trả về 200 cho những truy cập thành công, 404 cho các truy cập thất bại hay 302 chuyển hướng truy cập do không tồn tại tài nguyên đó. Khi đó fuzzer hoàn toàn có thể phân tích, đánh giá việc tồn tại các lỗi về cấu hình mặc định này. Một số phương pháp phân tích kết quả trả về để phát hiện các lỗi cấu hình mặc định được trình này trong bảng 2.7:

Bảng 2.7. Phát hiện các lỗi do cấu hình

Kiểu lỗi	Mô tả kỹ thuật phát hiện
Directory Listing	Đường dẫn được phân tách thành các đường dẫn trỏ tới các thư mục nhằm thực hiện kiểm tra các đường dẫn này có hiển thị danh sách các tệp tin nằm trong nó hay không. Được xác định dựa trên 2 đặc điểm: - Mã trạng thái trả về từ 200 đến 299. - Trong mã HTML trả về có chứa các từ khóa “index of” hoặc “parent directory”.
Manager Path	Thực hiện tấn công phỏng đoán đường dẫn tới trang quản trị website bằng cách gửi các yêu cầu với các đường dẫn trỏ tới trang quản trị như /admin, /administrator, /manager,… Xác định dựa trên 2 đặc điểm chính là - Mã trạng thái trả về từ 200 đến 299. Một số trường hợp cần kiểm tra mã trạng thái từ 300 đến 399. - Trong mã HTML trả về có chứa form đăng nhập, mà trường nhận dạng chính là type = “password”.
Tệp tin cấu hình	Thực hiện gửi yêu cầu tới các đường dẫn thư mục và nối thêm tên các tệp tin cấu hình như .htaccess,... Truy cập vào đường dẫn file cấu hình. Xác định dựa trên đặc điểm: - Mã trạng thái trả về từ 200 đến 299 - Mã HTML có chứa các từ khóa tương ứng của các tệp tin cấu hình.
Tệp tin Back-up	Tương tự như tấn công thử nghiệm vào các tệp tin cấu hình. Các URL được trỏ đến tệp tin các extension mặc định của update tệp tin (.back, ~, .bak...)

	Đặc điểm nhận dạng chính là mã trạng thái trả về từ 200 đến 299.
Tài khoản, mật khẩu mặc định	Bản chất của cuộc tấn công vào tài khoản, mật khẩu của một website là nó cố gắng sử dụng thuật toán vét cạn cho thực hiện đăng nhập tới khi đạt được điều mong muốn. Nó chủ yếu được sử dụng để tấn công vào tài khoản quản trị, người dùng và tài khoản cơ sở dữ liệu. Xác định dựa trên đặc điểm: - Mã trạng thái trả về là 302 cho sự chuyển hướng khi xác thực thành công. - Mã HTML khác so với giao diện đăng nhập.

2.3. Trình bày về công cụ liên quan kỹ thuật Fuzzing.

Phương pháp kiểm tra fuzzer chủ yếu được ứng dụng trong việc thiết kế các tool để kiểm tra lỗi bảo mật Web.

2.3.1. Nguyên tắc thực hiện:

Dựa vào các phân tích các lỗ hổng ở trên mà các nhà phát triển sẽ thiết kế bộ dữ liệu fuzzing phù hợp nhằm phát hiện tối đã các lỗ hổng bảo mật website. Sau đó các công cụ sử dụng phương pháp fuzzing để kiểm tra lỗi bảo mật sẽ không thực hiện việc quét cấu trúc thư mục và tập tin của ứng dụng web, mà sẽ là tập hợp tất các lỗi đã được công bố đối với từng ứng dụng web cụ thể và thực hiện đệ trình đến ứng dụng web xem thử ứng dụng đó có bị mắc các lỗi bảo mật hay không? Ví dụ, một fuzzer kiểm tra tất cả các lỗi bảo mật liên quan đến ứng dụng cổng thông tin điện tử Joomla thì nó sẽ tập hợp tất cả các lỗi bảo mật liên quan đến ứng dụng Joomla và thực hiện đệ trình khi người kiểm tra cung cấp.

2.3.2. Một số Tool tiêu biểu

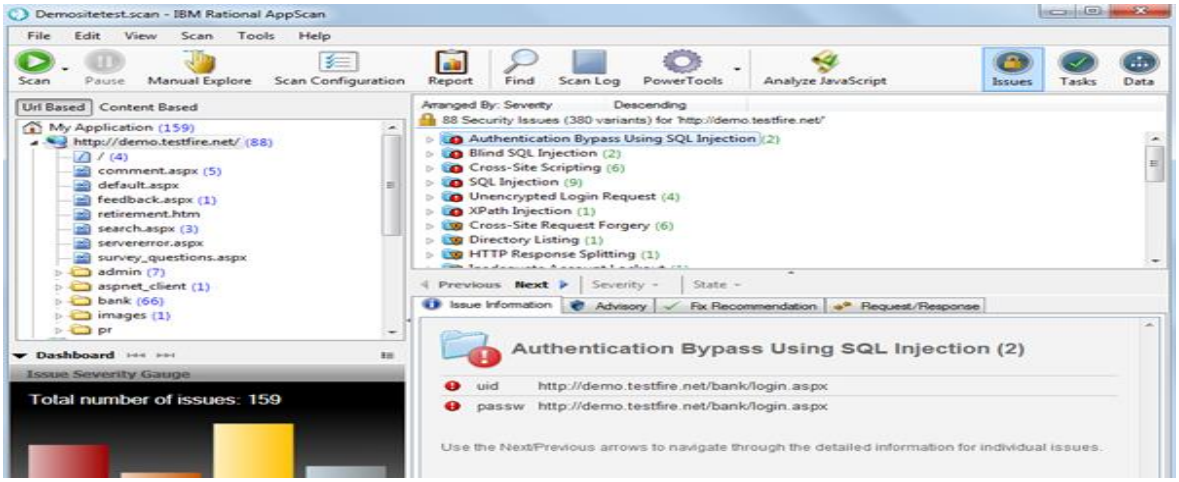
2.3.2.1. Nikto (http://cirt.net/nikto2)

Nikto [20] là một trong những công cụ thực hiện fuzzer các lỗi bảo mật tốt nhất và nhanh nhất hiện nay. Với cơ sở dữ liệu cập nhật hàng trăm lỗi bảo mật được xuất hiện hằng ngày. Đặc biệt là Nikto được sử dụng hoàn toàn miễn phí và có khả năng tùy biến cao. Nikto cũng là một trong những công cụ được insecure.org bình chọn một trong những công cụ quét lỗi bảo mật web tốt nhất trong 10 công cụ. Ngoài ra,

Nikto cho phép người sử dụng tùy biến viết các thành phần và nhúng kết với Nikto để thực thi. Hơn nữa, Nikto cũng hỗ trợ nhiều định dạng của những chương trình quét lỗi bảo mật khác như: Nmap, Nessus.

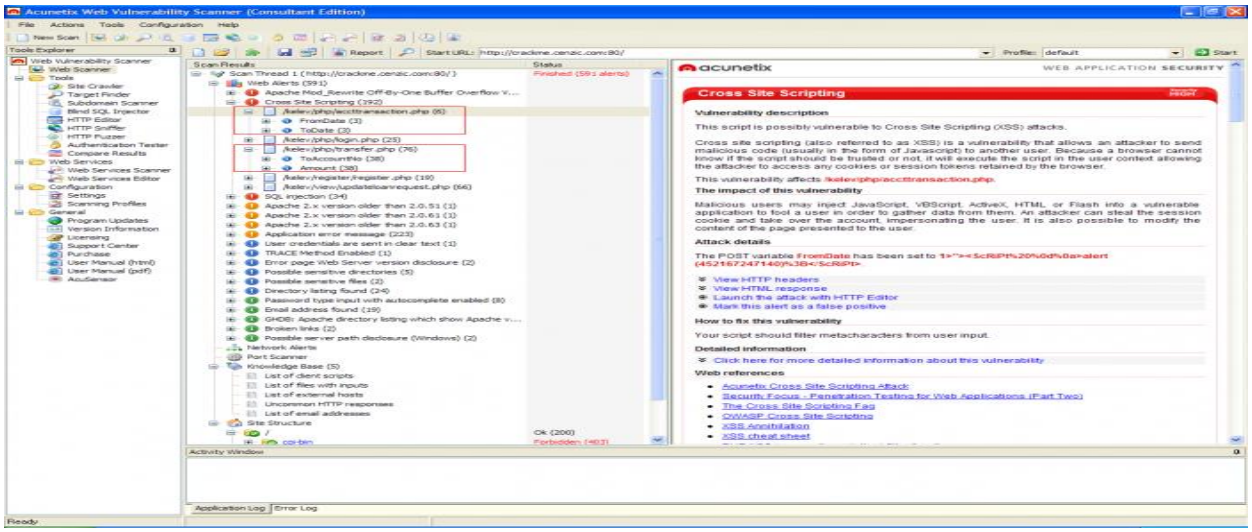
2.3.2.2. AppScan (http://www.ibm.com/software/awdtools/appscan/)

Một công cụ thương mại tập hợp lớn các lỗi bảo mật cho từng ứng dụng web riêng biệt.



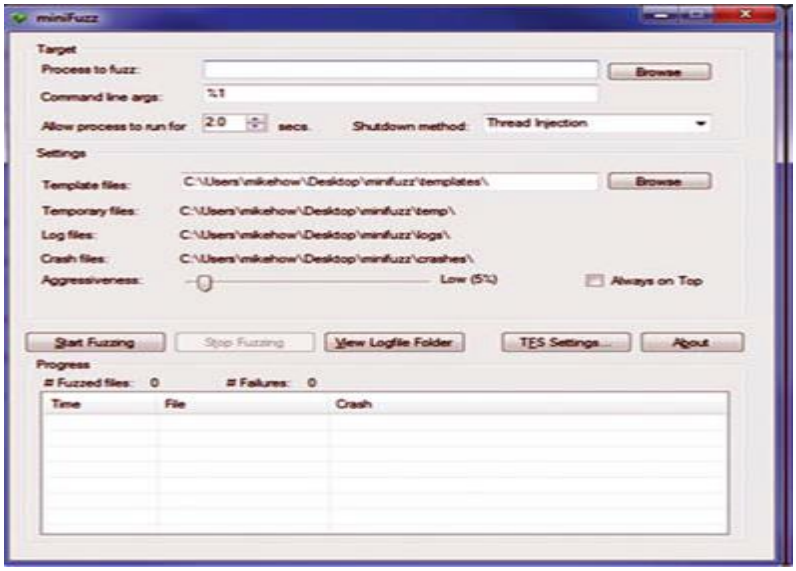
2.3.2.3. Acunetix Web Vulnerability Scanner (WVS)

Công cụ này kiểm tra tất cả các lỗ hổng web, bao gồm SQL Injection, Cross Site Scripting và nhiều lỗ hổng khác [22].Acunetix sử dụng kỹ thuật phân tích động với hướng tiếp cận dựa trên phỏng đoán, sử dụng thuật toán Fuzzing. Ban đầu Module Crawler phân tích toàn bộ trang web bằng cách làm theo tất cả các liên kết trên trang web. Sau đó, WVS sẽ vạch ra cấu trúc trang web và hiển thị thông tin chi tiết về mỗi tập tin. Sau quá trình thu thập thông tin, WVS tự động hiển thị một loạt các lỗ hổng có thể bị tấn công trên mỗi trang được tìm thấy. Khi các lỗ hổng được tìm thấy, Acunetix WVS báo cáo về lỗ hổng này. Dưới đây là một giao diện của phần mềm Acunetix được đánh giá thử nghiệm:



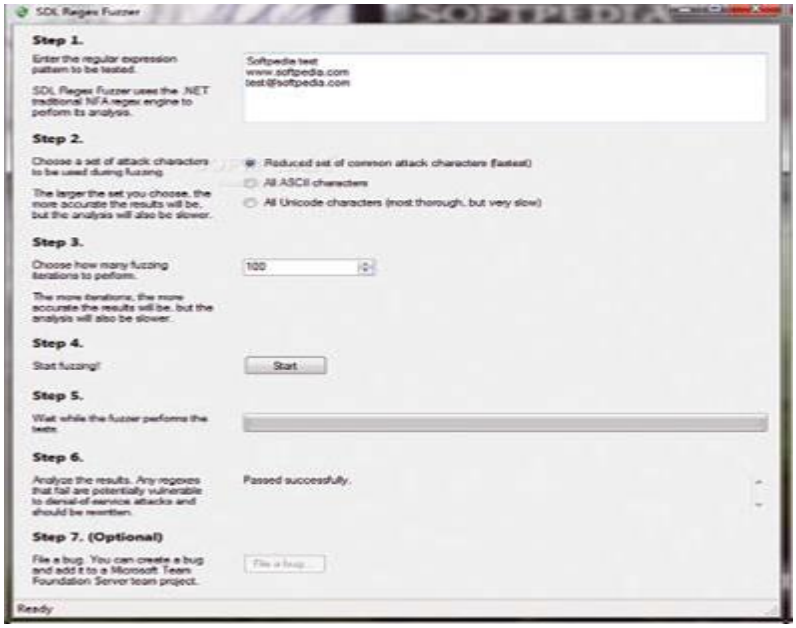
2.3.2.4. MiniFuzz File Fuzzer

Chiếu theo chính sách SDL, một trong những công đoạn bắt buộc khi kiểm tra (test) ứng dụng là fuzzing, tức là kiểm tra bằng các dữ liệu đầu vào ngẫu nhiên. Minifuzz File Fuzzer là công cụ kiểm tra mờ (fuzzy testing) cơ bản [23], được phát triển nhằm cho phép các chuyên gia không thuộc lĩnh vực an toàn, không có kiến thức về kiểm tra mờ cũng có thể thực hiện được việc kiểm tra này. Công cụ này đưa tới đầu vào của ứng dụng được kiểm tra các giá trị khác nhau nhằm tìm ra những ngoại lệ chưa được xử lý.



2.3.2.5. SDL Regex Fuzzer

Là công cụ kiểm tra mờ các biểu thức chính quy (Regex), SDL Regex Fuzzer [24] cho phép kiểm tra các biểu thức chính quy để phát hiện các lỗ hổng loại "từ chối dịch vụ". Đây là nhiệm vụ không đơn giản bởi các biểu thức chính quy thường chứa các mệnh đề có độ phức tạp hàm mũ về thời gian. Các biểu thức dạng này có thể bị lợi dụng để thực hiện tấn công từ chối dịch vụ.Công cụ SDL Regex Fuzzer cho phép tìm kiếm và chỉ ra sự có mặt của các biểu thức nguy hiểm này ngay trong quá trình phát triển ứng dụng.



Kết luận chương 2

Trong chương 2, với phạm vi nằm trong lĩnh vực kiểm thử bảo mật website, kỹ thuật Fuzzing đã được trình bày một cách chi tiết. Cùng với đó là các nguyên lý và phương pháp phát hiện lỗ hổng bảo mật web.

Các nội dung chính đã trình bày trong chương gồm:

- Trình bày tổng quan về mô hình và quy trình thực hiện Fuzzing. Cho thấy rằng, quy trình này cũng tương tự như quy trình kiểm thử phần mềm nhưng nó sử dụng các phương thức, đặc trưng phát hiện và bộ dữ liệu dành riêng cho kiểm thử website.

- Các phần thu thập điểm đầu vào, chèn dữ liệu fuzzing, phương pháp phát hiện lỗ hổng: đây là ba giai đoạn quan trọng nhất trong quy trình fuzzing kiểm thử bảo mật website. Thu thập điểm đầu vào và chèn dữ liệu là giai đoạn khởi đầu cho một cuộc tấn công kiểm thử, chúng thu thập những vị trí chèn dữ liệu và lần lượt thực hiện chèn các dữ liệu này. Phương pháp phát hiện là một phần trong giai đoạn phân tích lỗ hổng, đây là những căn cứ giúp cho Fuzzer có thể nhận dạng lỗ hổng tồn tại trong kết quả trả về của các lượt truy vấn.

Đây là nội dung nghiên cứu quan trọng cho việc đưa ra toàn bộ chi tiết kỹ thuật, quy trình thực hiện xây dựng một ứng dụng kiểm tra và phát hiện lỗ hổng bảo mật web.

Chương 3 tiếp theo sẽ trình bày việc ứng dụng kỹ thuật Fuzzing vào kiểm thử lỗ hổng bảo mật website cho một ngân hàng cụ thể.

CHƯƠNG 3. ỨNG DỤNG KIỂM THỬ LỖ HỔNG BẢO MẬT WEBSITE CHO NGÂN HÀNG NGOẠI THƯƠNG LÀO ĐẠI CHÚNG (BCEL)

3.1. Lý do chọn website ngân hàng BCEL

Ngân hàng BCELLà một trong 26 NHTM đang hoạt động tại nước CHDCND Lào, ngân hàng BCELđược thành lập từ năm 1975. Đây là một trong những ngân hàng được thành lập sớm nhất tại nước CHDCND Lào, đánh dấu sự ra đời và phát triển của lĩnh vực ngân hàng từ khi nước CHDCND Lào được thành lập. Giai đoạn đầu từ năm 1975 đến năm 1989 NHNT hoạt động với tư cách là một chi nhánh của NHNN. Trong giai đoạn này Ngân hàng BCEL là ngân hàng duy nhất được cấp phép kinh doanh quốc tế trong lĩnh vực ngân hàng. Một trong những đặc trưng của trong hoạt động của Ngân hàng BCEL là ngân hàng được giao nhiệm vụ quản lý các nguồn vốn hỗ trợ và vốn vay trực tiếp nước ngoài mà các quốc gia hoặc các tổ quốc tế dành cho nước CHDCND Lào.

Từ năm 1989, khi Đảng và Nhà nước Lào chủ trương tái cấu trúc lại nền kinh tế Lào cũng là lúc lĩnh vực ngân hàng có nhiều thay đổi mang tính bước ngoặt. Đối với Ngân hàng BCEL, năm 1989 đánh dấu thời kỳ bắt đầu hoạt động độc lập với tư cách là ngân hàng thương mại nhà nước theo các quy định của Luật Quản lý các ngân hàng thương mại (sau đó được thay thế bằng Luật Ngân hàng thương mại năm 2006) và chịu sự quản lý, giám sát trực tiếp của NHNN Lào.

Những năm 1997, các nền kinh tế trong khu vực châu Á - Thái Bình Dương, trong đó có nền kinh tế Lào chịu ảnh hưởng mạnh mẽ của cuộc khủng hoảng kinh tế - tài chính. Điều này cũng ảnh hưởng không nhỏ đến sự phát triển của lĩnh vực ngân hàng nói chung và hoạt động của NHNT riêng. Nhưng bằng sự nỗ lực của mình và sự hỗ trợ của Chính phủ Lào, NHNN Lào, Ngân hàng BCEL đã từng bước vượt qua cuộc khủng hoảng đó. Với sự lãnh đạo của NHNN Lào, Ngân hàng BCEL và những ngân hàng còn lại đã góp phần thực hiện tốt chính sách tiền tệ quốc gia, là kênh huy động vốn và cấp tín dụng chủ yếu cho nền kinh tế Lào. Để có được những thành tựu nói trên, Ngân hàng BCEL đã tiến hành cải tổ, cơ cấu lại tổ chức và hoạt động, ứng

dụng công nghệ hiện đại vào quản lý và kinh doanh phục vụ tốt nhất cho nhu cầu của khách hàng, đã giúp cho Ngân hàng BCEL vượt qua cuộc khủng hoảng trên một cách nhanh chóng.

Như vậy với hơn 47 năm phát triển của mình, lúc nào Ngân hàng BCEL cũng khẳng định và giữ vững vai trò là một trong những ngân hàng thương mại hàng đầu tại nước CHDCND Lào, đóng góp rất nhiều cho nền kinh tế Lào, dành được niềm tin và khẳng định được uy tín đối với các quốc gia, tổ chức quốc tế và các ngân hàng thương mại của các quốc gia trên thế giới.

Website ngân hàng BCEL chứa nhiều thông tin nhạy cảm của khách hàng, cần được bảo vệ chống các xâm nhập của tin tặc. Bản chất các Website khi phát triển luôn tồn tại các lỗ hổng bảo mật. Do vậy, việc rà soát và kiểm tra các lỗ hổng bảo mật trên các Website ngân hàng BCEL luôn được quan tâm.

Vấn đề bảo mật Website ngân hàng BCEL nói riêng và ngân hàng ở Lào nói chung vẫn chưa được các cơ quan, doanh nghiệp ở Lào chú trọng đầu tư. Không phải tất cả các tổ chức, doanh nghiệp đều có thể trang bị đầy đủ cũng như có thể đảm bảo an toàn, bảo mật thông tin một cách toàn diện. Việc kiểm thử lỗ hổng bảo mật Website ngân hàng BCEL nói riêng và ngân hàng ở Lào nói chung rất cần thiết vì sẽ góp phần phát hiện các lỗ hổng và giúp khắc phục, vá các lỗi bảo mật này. Nhiều kỹ thuật kiểm thử, song kỹ thuật kiểm thử hộp đen (Black-Box) cụ thể là kỹ thuật Fuzzing (thuộc loại kiểm thử Black-Box) là quan trọng nhất. Mặc dù không mới, song mới được quan tâm từ vài năm nay do các ưu việt so với các kỹ thuật truyền thống đã biết.

Việc kiểm thử phần mềm hiện nay đa phần được thực hiện một cách thủ công, không có hiệu quả cao trong việc phát hiện những lỗ hổng an ninh tiềm tàng. Kỹ thuật Fuzzing trong kiểm thử lỗ hổng bảo mật Website ngân hàng BCEL chính là một giải pháp cho vấn đề trên. Với khả năng tự động hóa cao cùng với cơ chế phát hiện lỗ hổng hiệu quả, công nghệ này được rất nhiều hãng quan tâm sử dụng. Tuy kỹ thuật fuzzing đã được áp dụng trong nhiều lĩnh vực, song việc áp dụng kỹ thuật

này vào kiểm thử lỗ hổng bảo mật website vẫn còn có những vấn đề cần nghiên cứu do sự phát triển của lĩnh vực an toàn thông tin.

3.2. Xây dựng ứng dụng kiểm thử fuzzing cho website ngân hàng BCEL

3.2.1. Sơ đồ kiến trúc ứng dụng

❖ **Kiến trúc chương trình ứng dụng**

Ứng dụng xây dựng theo một kiến trúc linh hoạt, bao gồm các tầng, trong mỗi tầng có các thành phần xử lý riêng biệt. Với giao diện đồ họa, người dùng có thể cấu hình đơn lẻ hay kết hợp cho một cuộc tấn công. Với kiến trúc phân tầng, chương trình được chia làm 2 tầng chính:

- Tầng 1: Tầng giao diện, có nhiệm vụ hiển thị kết quả xử lý lên giao diện cho người dùng xem kết quả:

+ Các URL thu thập được của một website.

+ Hiển thị các thông báo về lỗ hổng tồn tại trên website, đưa ra các khuyến nghị và biện pháp khắc phục của từng lỗ hổng.

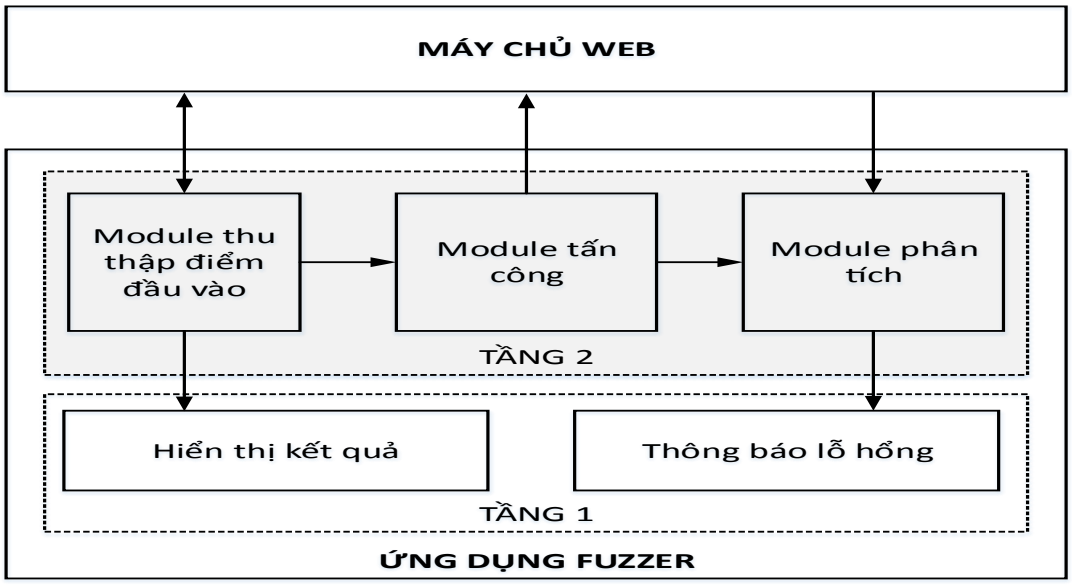
- Tầng 2: Tầng xử lý, đảm nhiệm xử lý toàn bộ hoạt động của ứng dụng. Tầng này được phân tách thành các thành phần đảm nhiệm từng xử lý riêng biệt, bao gồm: thành phần thu thập điểm đầu vào, thành phần tấn công, thành phần xử lý. Tầng này có nhiệm vụ:

+ Xử lý các thông tin trả lời từ máy chủ và thu thập toàn bộ URL và các điểm đầu vào.

+ Thực hiện cuộc tấn công thử nghiệm Fuzzing vào tất cả các điểm đầu vào thu thập được.

+ Phân tích các phản hồi trả về của cuộc tấn công Fuzzing, xác định sự tồn tại của lỗ hổng và đưa ra kết quả.

Kiến trúc phân tầng của ứng dụng kiểm tra lỗ hổng bảo mật website được mô tả như hình 3.1 dưới đây:



Hình 3.1. Kiến trúc phân tầng của ứng dụng

3.2.2 Xây dựng ứng dụng

❖ **Ngôn ngữ sử dụng**

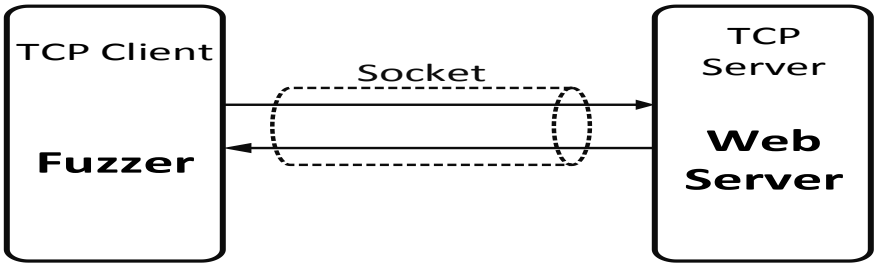
C# là một trong những ngôn ngữ lập trình được sử dụng rộng rãi nhất. Đi kèm với framework .NET nên C# được hỗ trợ nhiều tính năng, có thể tạo ra những chương trình hay hệ thống mạnh mẽ. C# được miêu tả là ngôn ngữ có được sự cân bằng giữa C++, Visual Basic, Delphi và Java.

Để xây dựng một công cụ kiểm tra lỗ hổng bảo mật website đơn giản, thân thiện với người dùng, chúng ta cần xây dựng nó với giao diện GUI. C# là một ngôn ngữ lập trình hướng đối tượng được phát triển bởi Microsoft, hoạt động dựa trên framework .NET của Windows. C# được sử dụng để xây dựng ứng dụng này với 3 lý do chính như sau:

- C# cho phép thiết kế với giao diện đồ họa GUI chuyên nghiệp.
- C# cung cấp một số lớp, thư viện hỗ trợ cho việc gửi và nhận gói tin mạng.
- Nó cung cấp cơ chế lập trình xử lý không đồng bộ phù hợp cho xây dựng ứng dụng cần trao đổi nhiều gói tin.

Nhược điểm tồn tại khi sử dụng C# đó là nó gắn liền với nền tảng Windows mà khó chuyển sang sử dụng tại các nền tảng khác.

❖ **Giao tiếp giữa ứng dụng và máy chủ web**



Hình 3.2. Giao tiếp giữa Fuzzer và Server

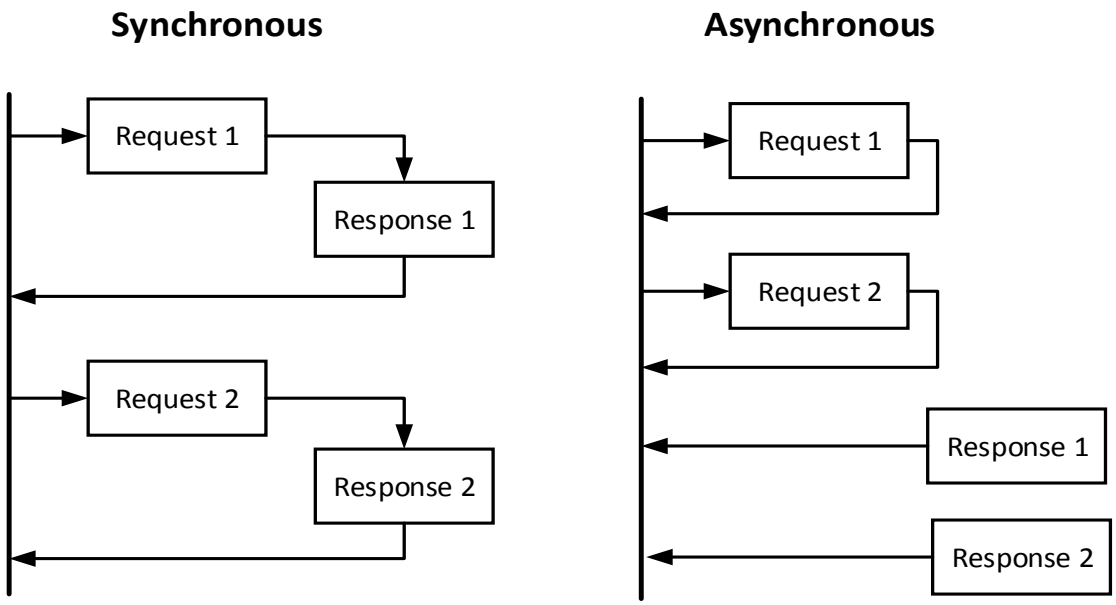
Giao tiếp giữa ứng dụng và máy chủ là giao tiếp giữa client và server. Trong đó client kết nối đến server theo kiểu stream socket. Giao tiếp giữa máy chủ web với chương trình Fuzzer.

C# cung cấp lớp WebClient bao gồm các chức năng xử lý yêu cầu và phản hồi HTTP.Nó đáp ứng đầy đủ các chức năng để xử lý các yêu cầu của một Fuzzer, C # cung cấp các lớp HTTPWebRequest và HTTPWebResponse.Các lớp này tiện lợi và có nhiều chức năng tiên tiến như khả năng sử dụng proxy.Nhưng nó lại không phù hợp để xây dựng một Fuzzer với số lượng request rất lớn.Thay vào đó, lớp TcpClient được thiết kế cho bất kỳ gói tin TCP nào.Các lớp và hàm tiêu chuẩn có sẵn không phù hợp với nhu cầu xử lý dữ liệu cần thiết.Để xây dựng một Fuzzer, điều cần thiết phải kiểm soát hoàn toàn yêu cầu HTTP thô, mà điều đó nhiều lớp có sẵn lại không cung cấp với mức độ chi tiết.

❖ **Xử lý bất đồng bộ**

Một giao dịch sử dụng socket đồng bộ bị chặn hay có độ trễ lớn, điều này có nghĩa là yêu cầu hoặc phản hồi đang gặp phải sự chậm trễ trong giao dịch. Như vậy máy chủ hay máy khách sẽ dừng lại và chờ đợi cho giao dịch đó hoàn thành trước khi thực hiện một giao dịch tiếp theo. Với một ứng dụng Fuzzer, quá trình fuzzing được thực hiện dựa trên bộ dữ liệu bất thường, như vậy một số yêu cầu gửi đi có khả năng cao xảy ra sẽ bị chặn hay xảy ra sự cố.Một fuzzer không thể đứng im chờ đợi một giao dịch có thể không bao giờ thành công, điều đó làm cho Fuzzer hoạt động không hiệu quả.Socket với không đồng bộ khởi động các luồng riêng biệt để

xử lý và sử dụng chức năng gọi lại để báo hiệu một giao dịch hoàn thành.Điều này cho phép các sự kiện khác được xử lý mà không bị gián đoạn.



Hình 3.3. Xử lý đồng bộ và bất đồng bộ

❖ **Lập trình xử lý bất đồng bộ**

Socket xử lý bất động bộ được mô tả dưới đoạn mã sau:

```
TcpClient client;
NetworkStream stream;
ClientState cs;

client = new TcpClient();
client.Connect(reqHost, Convert.ToInt32(tbxPort.Text));
stream = client.GetStream();
cs = new ClientState(stream, reqBytes);

IAsyncResult result = stream.BeginWrite(cs.ByteBuffer, 0,
cs.ByteBuffer.Length, new AsyncCallback(OnWriteComplete), cs);

result.AsyncWaitHandle.WaitOne();
```

Sau khi tạo một Client TCP và NetworkSteam, chúng gọi phương thức BeginWrite() với 5 đối số như sau:

- Byte[] array: Một bộ đệm chứa luồng dữ liệu muốn gửi.
- Int offset: Vị trí trong bộ đệm để bắt đầu gửi dữ liệu.
- Int numBytes: Số byte đối đa để ghi gửi dữ liệu.
- AsyncCallback userCallback: Phương thức gọi lại, sẽ được gọi khi kết nối hoàn tất.
- Object stateObject: Một đối tượng để phân biệt yêu cầu viết không đồng bộ này từ các yêu cầu khác.

AsyncWaitHandle.WaitOne() làm cho sự kiện lắng nghe bị khóa cho tới khi yêu cầu được gửi thành công. Tại thời điểm đó, chức năng gọi lại sẽ được thực hiện bằng mã lệnh sau:

```
ClientState cs = (ClientState)ar.AsyncState;
cs.NetStream.EndWrite();
```

Khi chúng tôi viết xong yêu cầu của chúng tôi đến luồng mạng, chúng tôi sẽ có thể nhận được kết quả từ máy chủ:

```
result = stream.BeginRead(cs.ByteBuffer, cs.TotalBytes,
cs.ByteBuffer.Length - cs.TotalBytes, new AsyncCallback(OnReadComplete), cs);
```

Tại thời điểm này, chúng ta có thể gọi socket không đồng bộ một lần nữa nhưng tốt hơn hết là sử dụng khoảng thời gian nên sử dụng để nhận phản hồi. Chúng ta gọi phương thức BeginRead(), lấy các đối số giống như phương thức BeginWrite(), sử dụng hàm OnReadComplete() như là phương thức gọi lại của chúng ta:

```
public void OnReadComplete(IAsyncResult ar)
{
cs.NetStream.BeginRead(cs.ByteBuffer, cs.TotalBytes, cs.ByteBuffer.Length
- cs.TotalBytes, new AsyncCallback(OnReadComplete), cs);
}
```

Chúng ta có thể bắt đầu OnReadComplete() bằng cách tạo một bộ đếm thời gian (readTimeout) sẽ gọi ReadDone.Set() nếu đạt đến khoảng thời gian chờ do người dùng xác định. Điều này cho phép chúng ta đảm bảo rằng một giao dịch không hoạt động vô hạn nếu việc nhận phản hồi không được hoàn thành. Nó cung cấp cho người dùng một phương tiện để kiểm soát độ dài thời gian trễ.

3.2.3. Xây dựng các thành phần chính của ứng dụng kiểm thử

Dựa trên kiến trúc của chương trình, ứng dụng bao gồm 3 phần chính. Đầu tiên là thành phần thu thập điểm đầu vào, nó sẽ thu thập toàn bộ các liên kết trong website. Sau đó, thành phần tấn công thực hiện các cuộc tấn công vào mục tiêu này. Cuối cùng, thành phần phân tích thực hiện kiểm tra kết quả trả về bởi các ứng dụng web để xác định lỗ hổng tồn tại:

❖ **Thành phần thu thập điểm đầu vào:**

```
public void AFCrawling()
{
    string defaultreqCrawl = "GET [path] HTTP/1.1\r\nAccept: */*\r\nAccept-Language: vi-vn\r\nPragma: no-cache"

    string path = "", method = "GET";
    string request = "";
    string pthside = "";
    string host = "";
    int lstcrawledcount = 0;
    string urlroot = tbxURLtarget.Text;
    int arrayEnd, arrayCount = 0;
    ArrayList crawlArray = new ArrayList();
    List<URL> lstcrawlArray = new List<URL>();
    List<URL> lstformPOST = new List<URL>();
    //Thay host
    host = gethost(urlroot);
    string request1 = defaultreqCrawl.Replace("localhost", host);

    path = getpth(urlroot);

    if (path == "/")
    {
        crawlArray.Add(getfile(urlroot) + getparameter(urlroot));
        crawlArray.Add("robots.txt");
        crawlArray.Add("sitemap.xml");
```

Hình 3.4. Thành phần thu thập điểm đầu vào

Để thực hiện một phiên làm việc với thành phần thu thập điểm đầu vào, ứng dụng cần được bắt đầu với một địa chỉ website gốc. Nó được bắt đầu như là một điểm khởi đầu, trình thu thập điểm đầu vào lặp lại quá trình thực hiện thu thập tất cả các liên kết và các biểu mẫu web trong suốt quá trình xử lý. Để giảm số lượng thực hiện gửi yêu cầu, thành phần thu thập điểm đầu vào lọc và loại bỏ các liên kết không thuộc tên miền gốc mà người dùng nhập, kể cả tên miền phụ.

Cũng giống như những trình thu thập dữ liệu, chúng cũng có những tùy chọn cấu hình cho quá trình thực hiện. Thành phần thu thập điểm đầu vào được xây dựng tách biệt trong phần thu thập thủ công và trong cả quá trình tự động quét lỗ hổng.

Hình 3.4 mô tả một phần đoạn mã hàm thực hiện chức năng Crawling bằng ngôn ngữ C#:

❖ **Thành phần tấn công (thành phần gửi mẫu kiểm thử):**

Thành phần tấn công thực chất là thành phần tạo các mẫu yêu cầu kiểm thử gửi tới máy chủ Website theo kỹ thuật fuzzing.

```
private void btnStartAttack_Click(object sender, EventArgs e)
{
    int sqlicount = 0, xsscount = 0, ficount = 0;
    //CRAWLING
    btnStartAttack.Text = "Crawling...";
    AFCrawling();

    //FILTER AND REPLACE DATA FUZZING
    btnStartAttack.Text = "Filtering...";
    int AFurlfuzzcount = AFurlfuzzing.Count;

    for (int i = 0; i < AFurlfuzzcount; i++)
    {
        for (int j = i + 1; j < AFurlfuzzcount; j++)
        {
            if (AFreplaceprt(AFurlfuzzing[i].Data.ToString()) == AFreplaceprt(AFurlfuzzing[j].Data.ToString()))
            {
                AFurlfuzzing.RemoveAt(j);
                j--;
                if (AFurlfuzzing[i].Data.ToString().Contains("[Fuzzing]"))
                {
```

Hình 3.5. Thành phần tấn công

Sau quá trình thu thập các điểm đầu vào được hoàn thành, ứng dụng bắt đầu xử lý danh sách các mục tiêu tấn công này. Thành phần tấn công thực hiện quét từng mục tiêu với mỗi biểu mẫu có trên trang web. Với mỗi mục tiêu biểu mẫu web hay liên kết được trích, đi cùng với phương thức là GET hay POST, các trường thông số của một gói tin HTTP sẽ được sử dụng để gửi nội dung yêu cầu fuzzing. Sau đó, tùy thuộc vào cuộc tấn công thực tế mà giá trị trên các trường được thay đổi cho phù hợp. Cuối cùng yêu cầu sẽ được gửi lên máy chủ xác định bằng phương thức GET hay POST yêu cầu.

❖ **Thành phần phân tích:**

Sau một cuộc tấn công vào các mục tiêu của một website, các phản hồi gửi

trả về cho ứng dụng. Công việc lúc này thuộc về thành phần phân tích, nó thực hiện phân tích và giải thích các phản ứng từ máy chủ. Dựa trên các tiêu chuẩn tấn công cụ thể, từ khóa để tìm kiếm các biểu hiện của lỗ hổng mà cuộc tấn công đó đang thực hiện và tính toán đưa ra quyết định cuộc tấn công đó đã thành công, website có tồn tại lỗ hổng.

```
public bool parseSQLi(string response)
{
    bool rtnSQLi = false;
    Read SQLText = null;
    ArrayList SQLArray = null;
    int arrayEnd, arrayCount = 0;

    SQLText = new Read("detectSQLi.txt");
    SQLArray = SQLText.readFile();
    arrayEnd = SQLArray.Count;
    while (arrayCount < arrayEnd)
    {
        if (response.Contains(SQLArray[arrayCount].ToString()))
        {
            rtnSQLi = true;
            break;
        }
        arrayCount++;
    }
```

Hình 3.6. Thành phần phân tích

3.3. Xây dựng kịch bản thử nghiệm kiểm thử lỗ hổng bảo mật website ngân hàng BCEL.

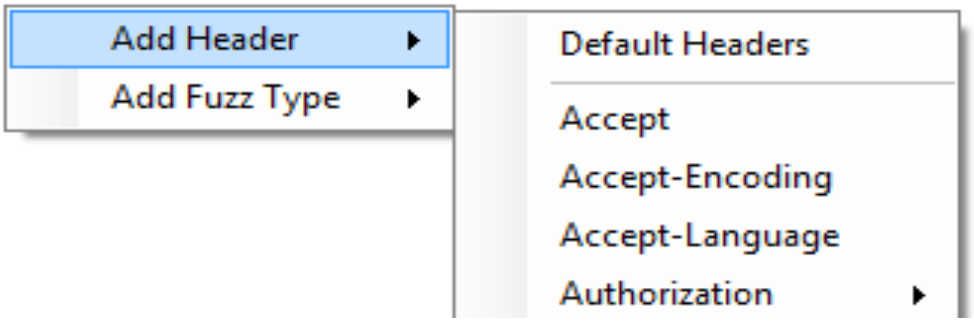
❖ ***Cài đặt ứng dụng thử nghiệm***

Ứng dụng kiểm tra lỗ hổng website được xây dựng với các phần chính là: Fuzzing Manual - thực hiện fuzzing thủ công, Crawler URL - thu thập URL website, Auto Fuzzing & Scan Vulnerability - tự động quét và phân tích lỗ hổng website. Người dùng cài đặt các thông số chung như Port để kết nối, Timeout cho thời gian chờ đợi phản hồi của yêu cầu.

❖ ***Chức năng đặt thủ công Fuzzing***

Cho phép người sử dụng có thể thực hiện fuzzing thủ công với bộ dữ liệu Fuzz có sẵn trên hệ thống. Nội dung Request Headers được người dùng tùy chỉnh theo ý muốn.

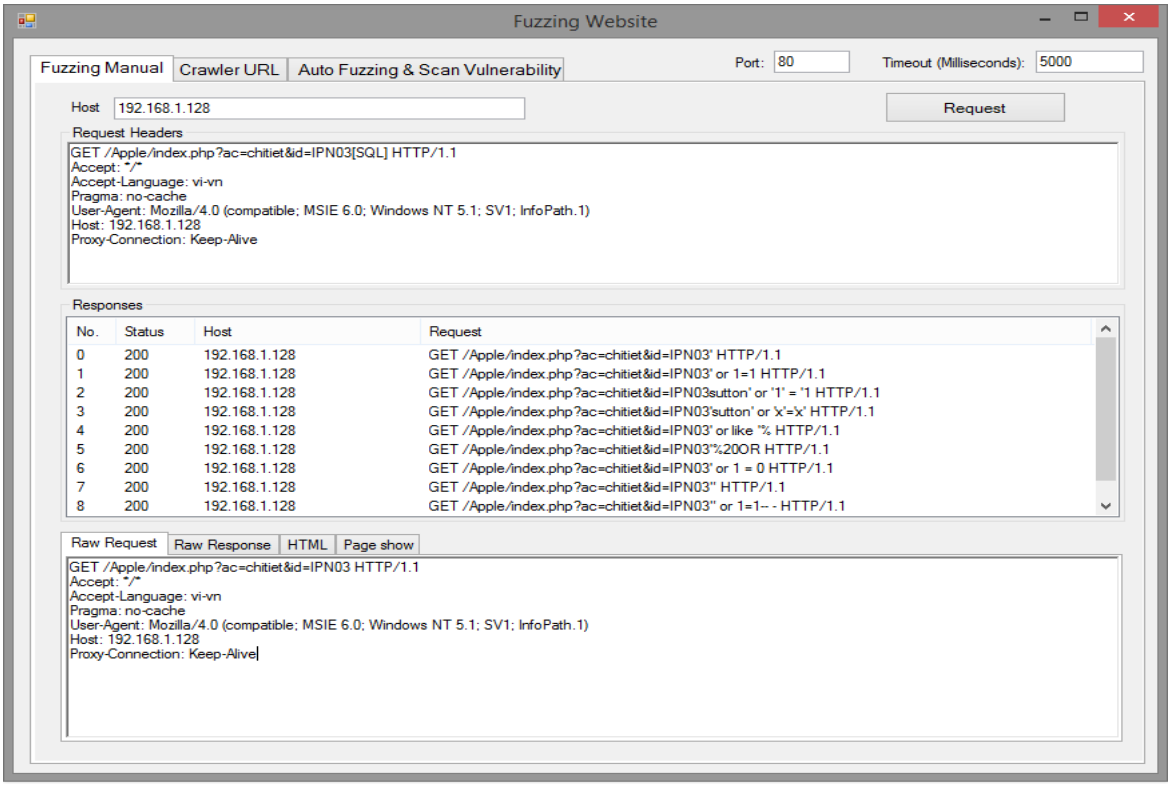
Người dùng tự cài đặt các thông số cho quá trình này bằng cách nhấp chuột phải vào Request Header tại vị trí muốn chèn thông số, một hộp thoại bao gồm các lựa chọn về Add Header, Add Fuzz Type như hình 3.7. Sau khi hoàn thành các thông số, nhấn Request để thực hiện chức năng này.



Hình 3.7. Danh sách các thông số tùy chọn

Kết quả trả về là danh sách các phản hồi từ máy chủ trả về nằm trong Responses, người dùng có thể nhấp chọn để xem chi tiết các truy vấn được hiển thị tại các hộp thoại dưới cùng.

Hình 3.8 mô tả giao diện của Fuzzing Manual cho quá trình quét thủ công:



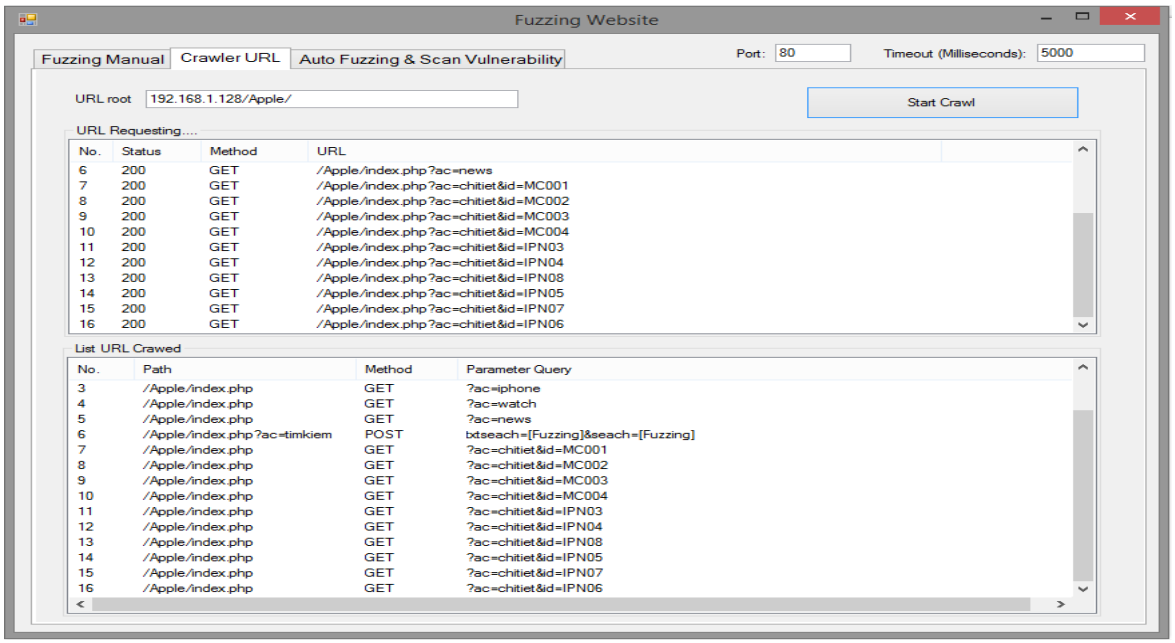
Hình 3.8. Giao diện Fuzzing thủ công

❖ ***Chức năng cào thông tin URL (Crawler URL)***

Chức năng này để người dùng thực hiện chức năng crawl tách biệt khỏi quá trình Fuzzing, thu thập toàn bộ các liên kết khác nhau của một website.

Để sử dụng chức năng này, người dùng chỉ cần nhấn chọn Crawler URL và nhập địa chỉ website mong muốn vào URL root. Sau đó nhấn nút Start Crawl để bắt đầu quá trình thu thập. Quá trình này diễn ra trong thời gian khá dài, tùy độ phức tạp của website.

Kết quả trả về là danh sách các đường dẫn khác nhau của website mà người dùng nhập. Nó được tách thành các thành phần khác nhau: Path, Method, Parameter Query,...



Hình 3.9. Giao diện Crawler URL

❖ ***Chức năng đặt Fuzzy tự động và quét điểm yếu Website***

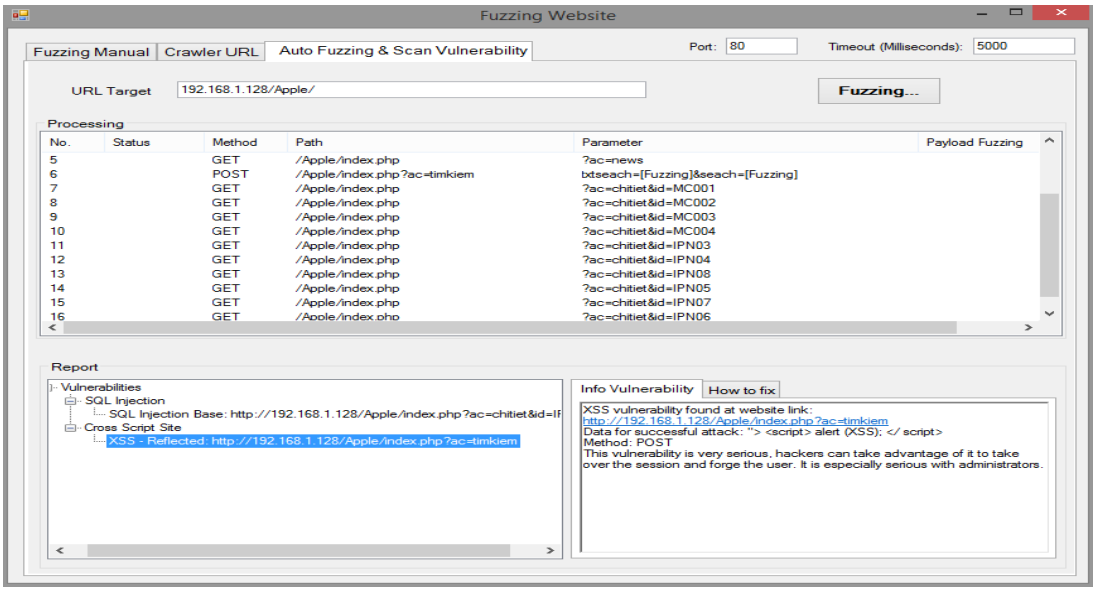
Đặt Fuzzy tự động (Auto Fuzzing) và quét điểm yếu (Scan Vulnerability) bao gồm 2 thành phần chính là Crawling và Fuzzing.Hai thành phần này thực hiện các chức năng là thu thập điểm đầu vào, thực thi tấn công và phân tích lỗ hổng. Chức năng này được mô tả như hình 3.10.

Với chức năng này, quá trình thực thi diễn ra sẽ hoàn toàn tự động.Người dùng chỉ cần nhập địa chỉ website cần rà quét lỗ hổng, sau đó nhấp vào nút Start để

thực hiện quét. Quá trình quét lỗ hổng diễn ra qua các giai đoạn Crawling, Filtering, Fuzzing và khi hoàn thành sẽ hiển thị Done.

Kết quả trả về là danh sách các lỗ hổng tồn tại trên website đó. Danh sách các lỗ hổng này được thể hiện qua cấu trúc thư mục cây, với mỗi nút chính là một loại lỗ hổng và mỗi nút con là một lỗ hổng.

Người dùng có thể nhấn chọn từng lỗ hổng để xem chi tiết về lỗ hổng và cách khắc phục lỗ hổng đó.Các thông tin chi tiết về từng lỗ hổng được mô tả trong Info Vulnerablity và cách khắc phục lỗ hổng này được mô tả trong phần How to fix.



Hình 3.10. Giao diện Auto Fuzzing & Scan Vulnerability

- Thực thi với tất cả các loại fuzzing mà chưa kiểm soát được điểm đầu vào nào phù hợp với loại tấn công nào.

- Bộ dữ liệu Fuzzing chưa đa dạng để phát hiện được tất cả các loại lỗi.

❖ ***Xây dựng kịch bản thử nghiệm cho ứng dụng***

Kịch bản:

1. Bước 1: Thiết lập một website thử nghiệm (giả lập cho website ngân hàng BCEL).

   Do điều kiện không thực hiện được trực tiếp trên website ngân hàng BCEL nên luận văn đặt thử một website mô phỏng website thương mại với địa chỉ: *http://127.0.0.1 /Apple/*

Kết quả thử nghiệm nhằm mục đích thể hiện khả năng thực hiện của ứng dụng cho bất kỳ một website nào. Do đó, hoàn toàn có thể áp dụng cho website của ngân hàng BCEL trong thực tế.

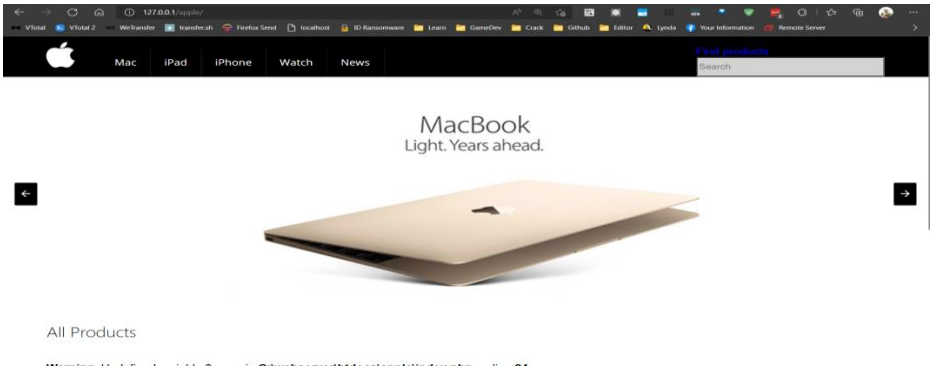
2. Bước 2: Website chứa sẵn một số lỗ hổng bảo mật phục vụ cho kiểm thử. Tùy theo có thể thiết lập tùy ý các lỗ hổng bảo mật.
3. Bước 3: Thực hiện chạy ứng dụng kiểm thử Fuzzing để thu kết quả phản hồi từ Website.
4. Bước 4: Quan sát kết quả phân tích của ứng dụng, thống kê, đánh giá kết quả.

3.4. Thực hiện kiểm thử, đánh giá kết quả

Thiết lập và thực thi kịch bản thử nghiệm

Dữ liệu đầu vào là một website có địa chỉ: *http://127.0.0.1 /Apple/*

Thông tin về máy chủ web: Windows (Sử dụng gói XAMPP),Apache2.4.xx, MariaDB 10.4.xx, PHP 7.4.xx.



Hình 3.11. Website kiểm thử nghiệm

Kết quả kiểm thử quan sát, thống kê được

Quá trình thực hiện chức năng Auto Fuzzing & Scan Vulnerability trong từng giai đoạn như sau:

Bảng 3.1. Kết quả quá trình thu thập

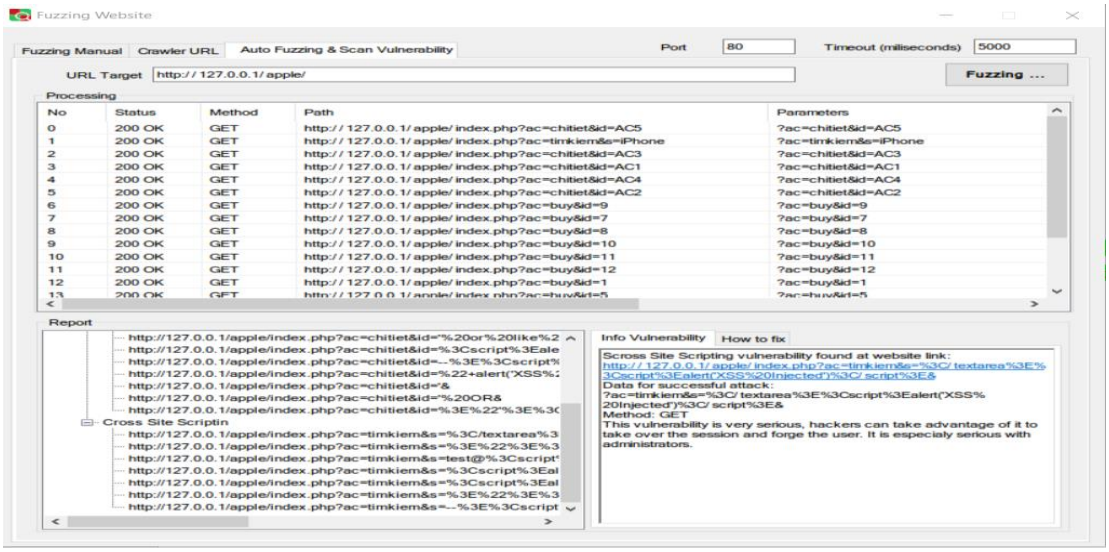
STT	Quá trình	Số lượng truy vấn	Thời gian thực thi	Kết quả
1	Crawling	19	~444ms	Thu được 19 URL
2	Filtering	17	~205ms	Lọc còn 17 URL
3	Fuzzing	17	~759ms	Phát hiện 3 lỗ hổng

Kết quả sau quá trình thực hiện Fuzzing phát hiện được 21 lỗ hổng: 14 lỗ hổng SQL Injection và 7 lỗ hổng XSS. Chi tiết các lỗ hổng được mô tả chi tiết trong bảng sau:

Bảng 3.2. Danh sách các lỗ hổng phát hiện

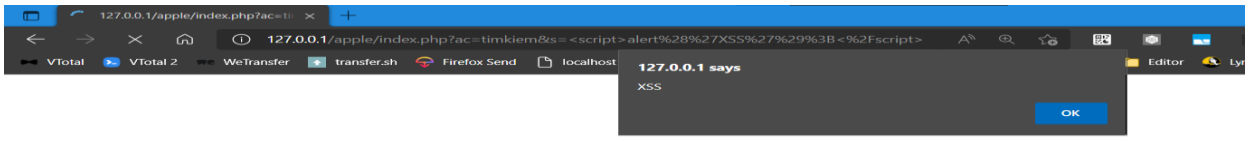
STT	Lỗ hổng	URL Cơ sở	Dữ liệu fuzz
1	SQL Injection	http://127.0.0.1/apple/index.php?ac=chitiet&id=ACx	'
2	Cross Script Site	http://127.0.0.1/apple/index.php?ac=timkiem&s=<query>	"><script> alert("XSS"); </script>

Tổng thời gian quá trình thực hiện khoảng 1.5 giây.



Hình 3.12. Danh sách các lỗ hổng website thử nghiệm

Kiểm tra lại các lỗ hổng vừa phát hiện được, ta thấy rằng các lỗ hổng này hoàn toàn tồn tại. Đoạn mã "><script>alert("XSS"); </script> được thực thi sau khi chèn vào nội dung tìm kiếm của website:



Hình 3.13. Lỗ hổng XSS được phát hiện

Kiểm tra lỗ hổng SQL Injection với tham số ‘, kết quả thu được:

Hình 3.14. Lỗ hổng SQL Injection

Đánh giá

Ưu điểm

- Phần mềm sau khi xây dựng và thực thi đã kiểm tra và phát hiện được một số lỗ hổng nghiêm trọng của website.

- Tốc độ thu thập điểm đầu vào và thực thi tấn công trên website thử nghiệm nhanh.

- Phát hiện các loại lỗ hổng có độ chính xác cao.

- Cho phép người dùng có thể thực hiện từng công đoạn tấn công và phát hiện lỗ hổng.

Nhược điểm

- Quá trình crawling tại một số website trực tuyến còn chậm so với một số phần mềm chuyên nghiệp.

- Quá trình lọc điểm đầu vào tương tự còn chưa chính xác, với một số website có thiết kế đặc biệt thì khả năng bỏ sót là lớn.

- Chương trình mới chỉ hỗ trợ kiểm thử trên các đường dẫn phương thức GET, chưa hỗ trợ tất cả các phương thức của HTTP.

- Các mẫu kiểm thử còn hạn chế, chưa áp dụng được các mẫu phức tạp như Blind SQL Injection.

- Thực thi với tất cả các loại fuzzing mà chưa kiểm soát được điểm đầu vào nào phù hợp với loại tấn công nào.

- Bộ dữ liệu Fuzzing chưa đa dạng để phát hiện được tất cả các loại lỗi.

KẾT LUẬN

Ngày nay, bảo mật Website ngân hàng đang ngày càng mở rộng và phát triển mạnh mẽ, vì vậy vấn đề kiểm thử lỗ hổng bảo mật Website ngân hàng cũng ngày càng được quan tâm và trú trọng. Nó trở thành yếu tố quyết định sinh tồn của một website hay hơn nữa là của cả một tổ chức, doanh nghiệp đứng sau nó.

Kiểm thử lỗ hổng bảo mật Website ngân hàng 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 website. Việc lựa chọn phương pháp kiểm thử là kỹ thuật Fuzzing giúp cho việc kiểm thử lỗ hổng bảo mật Website ngân hàng trở nên hiệu quả, giảm chi phí và thời gian.

Sau khoảng thời gian nghiên cứu và thực hiện luận văn, theo yêu cầu ban đầu đặt ra là nghiên cứu kỹ thuật Fuzzing và áp dụng trong kiểm thử lỗ hổng bảo mật Website ngân hàng, luận văn đã đạt được những kết quả như sau:

- Có kết quả nghiên cứu cơ sở lý thuyết về website, cách thức hoạt động, phân loại lỗ hổng bảo mật website 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 web trong ngôn ngữ máy.

- 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 lỗ hổng bảo mật Website.

- Nghiên cứu áp dụng kỹ thuật Fuzzing vào ứng dụng kiểm thử lỗ hổng bảo mật Website ngân hàng BCEL.

Một số hướng phát triển của luận văn trong quá trình thực hiện tiếp theo có thể là:

- 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 website.

- Phát triển, nâng cấp và mở rộng các trường hợp xử lý cho việc thực hiện kiểm thử trên mô hình bài toán website rộng hơn, phức tạp hơn. Phát triển sâu hơn để bảo mật ở mức hệ thống mạng và dịch vụ.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Vũ Thị Hương Giảng, Phan Văn Huy, Vũ Văn Trung, "*Giải pháp kiểm tra đồng thời mức độ an toàn và khả năng tiếp cận của trang web*", Tạp chí Khoa học Công nghệ An toàn thông tin, Số 2. CS (03) 2016, tr. 50-56.

[2] Nguyễn Ngọc Quân (2014), "*Lỗ hổng Cross Site Scripting (XSS) và biện pháp khắc phục*", Tạp chí, Học viện Công nghệ Bưu chính Viễn thông, Số 3. CS (04) 2014, tr. 301-307..

Tiếng Anh

[3] V.J. Manes, H.S. Han, C. Han, "The Art, Science, and Engineering of Fuzzing: A Survey", IEEE Transactions on Software Engineering, (99), Oct. 2019.

[4] H. Liang, X. Pei, X. Jia, W.Shen, "Fuzzing: State of the Art", IEEE Transactions on Reliability, Vol. 67, No.3, Sept. 2018, pp. 1199-1218.

[5] L. McDonal, M.I. Haq, A. Barkworth, " Survey of Software Fuzzing Techniques", Dec. 2021.https://www.researchgate.net/ publication/ 356 980212

[6] Danyang Zhao, "Fuzzing Technique in Web Applications and Beyond", Jounal of Physics, MCTE 2020, IOP Publishing, 1678 (2020) 012109, pp. 1-8, 2020.

[7] Glenford J. Myers, "The Art of software testing",*3 rd Editions, ISBN: 978-1-119-20248-6, Wiley Publishing, Canada, Sept. 2015.*

[8] IEEE 610.12:1990, "Standard Glossary of Software Engineering Terminology", *IEEE Standards Board, United States of America.*

[9] Justin Clarke, "SQL Injection Attacks and Defense", *Gotham Digital Science, ISBN-13: 978-1597499637, 2nd Editions, SynGress Publisher, UK, Elsevier Inc. 2012.*

[10] OWASP, "The ten most critical web application security risks", *OWASP, USA.*

[11] OWASP, “Testing Guide 4.0”, *OWASP,USA.*

[12] The Internet Society, “Request for Comments (RFC) 2616”, *Internet Engineering Task Force - IETF, USA.*

Website

[13] http://securitydaily.net/cac-phuong-phap-kiem-tra-ung-dung-web/

[14] https://books.google.com.vn/books?id=smEMCAAAQBAJ&printsec=frontcover&hl=vi&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[15] http://kcntt.duytan.edu.vn/Home/ArticleDetail/vn/128/2461/bai-01-so-luoc-ve-fuzzing-testing

[16] https://vi.wikipedia.org/wiki

[17] http://vietjack.com/http/http_status_codes.jsp

[18] https://itsecuritykma.blogspot.com/2014/01/tim-hieu-web-application-1.html

[19] https://viblo.asia/tran.thi.huong.trang/posts/RQqKLM64Z7z

[20] Nikto (http://cirt.net/nikto2)

[21] AppScan (http://www.ibm.com/software/awdtools/appscan/)

[22] https://www.acunetix.com/

[23] *MiniFuzz File Fuzzerhttps://www.security-database.com/tools watch / MiniFuzz-File-Fuzzer-v0-1.html*

[24] https://sdl-regex-fuzzer.software.informer.com/.

BẢN CAM ĐOAN

Tôi cảm đoan đã thực hiện việc kiểm tra mức độ tương đồng nội dung luận văn qua phần mềm Kiểm tra tài liệu một cách trung thực và đạt kết quả mực độ tương đồng 11% toàn bộ nội dung luận văn. Bản luận văn kiểm tra qua phần mềm là bản cứng luận văn đã nộp để bảo vệ trước hội đồng. Nếu sai tôi xin chịu các hình thức kỷ luật theo quy định hiện hành của Học viện.

Hà Nội, ngày 07 tháng 07 năm 2022

Tác giả luận văn

VIENGOUDOM XAYAVONG

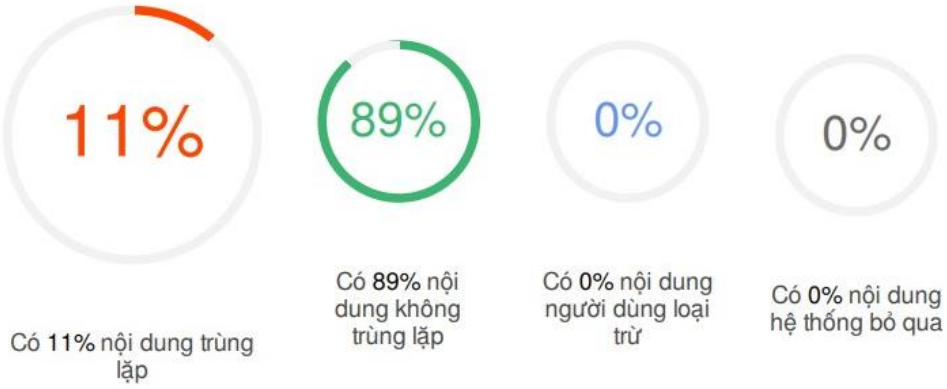
KiểmTraTàiLiệu

BÁO CÁO KIỂM TRA TRÙNG LẶP

Thông tin tài liệu

Tên tài liệu:	NGHIÊN CỨU KỸ THUẬT FUZZING TRONG KIỂM THỬ LỖ HỔNG BẢO MẬT WEBSITE NGÂN HÀNG-VIENGOUDOM XAYAVONG
Tác giả:	VIENGOUDOM XAYAVONG
Điểm trùng lặp:	11
Thời gian tải lên:	21:54 15/05/2022
Thời gian sinh báo cáo:	09:09 16/05/2022
Các trang kiểm tra:	79/79 trang

Kết quả kiểm tra trùng lặp



Nguồn trùng lặp tiêu biểu

123docz.net tailieu.vn

Giảng viên hướng dẫn Học viên