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

TÓM TẮT LUẬN VĂN THẠC SĨ

HÀ NỘI - 2022

Luận văn được hoàn thành tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

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

Phản biện 1: PGS.TS. Đỗ Trung Tuấn.

Phản biện 2: TS. Ngô Quốc Dũng.

Luận văn này được bảo vệ trước Hội đồng chấm luận văn thạc sĩ tại Học viện Công nghệ Bưu chính Viễn thông

Vào lúc: 10h45 ngày 02 tháng 07 năm 2022

Có thể tìm hiểu luận văn tại:

- Thư viện của Học viện Công nghệ Bưu chính Viễn thông

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.

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.

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.

2. Tổng quan về vấn đề nghiên cứu

- **Về vấn đề lỗ hổng bảo mật của 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.

- **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 mã 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.

Hiện nay, đã có một số nghiên cứu và ứng dụng kỹ thuật Fuzzing trong kiểm thử phần mềm. 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.

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

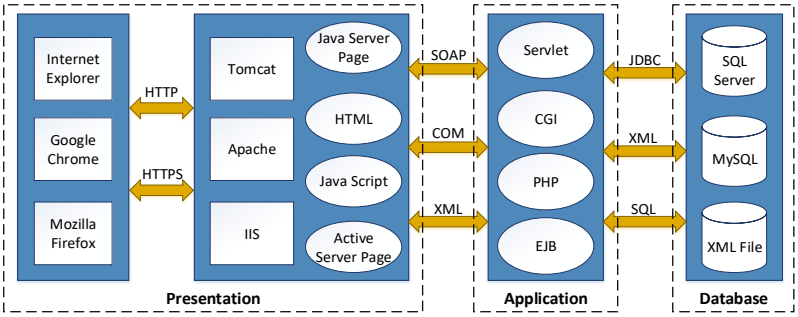
1.1.1. Các khái niệm cơ bản về hoạt động của website (ngân hàng)

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).

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.

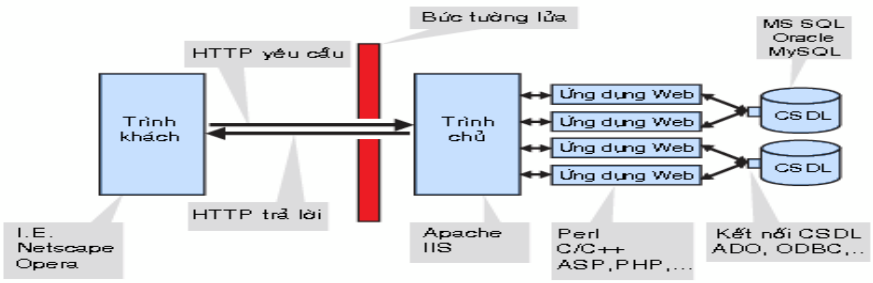
Một ứng dụng web thường có kiến trúc gồm:



Hình 1.1. Kiến trúc một ứng dụng web

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

➢ ***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.

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.

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.

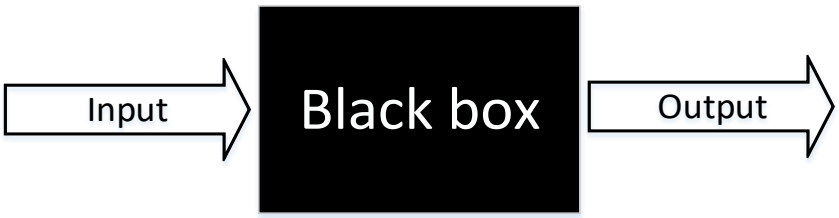
❖ **Lỗ hổng website**

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.

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

1.2.1. Kiểm thử hộp đen

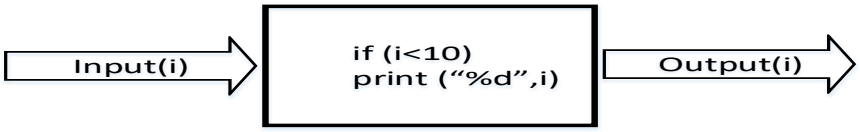
Là phương pháp kiểm thử được thực hiện mà không biết được cấu trúc và hành vi bên trong của phần mềm, là cách kiểm thử mà hệ thống được xem như một chiếc hộp đen, không cách nào nhìn thấy phía bên trong cái hộp [14].



Hình 1.3. Kiểm thử hộp đen

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.



Hình 1.4. Kiểm thử hộp trắng

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

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.

1.2.5. Fuzzing

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.

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

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 và 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.

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

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,…

1.3.3. Ưu nhược điểm của Fuzzing

Ưu điểm

- Kết quả sử dụng kiểm thử Fuzzing hiệu quả hơn khi sử dụng các phương pháp kiểm thử khác.

- Kiểm thử Fuzzing chỉ theo dõi các trường hợp mà kết quả trả về có sự bất thường hay hành vi không mong muốn.

- 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

- 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.

- 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

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.

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.

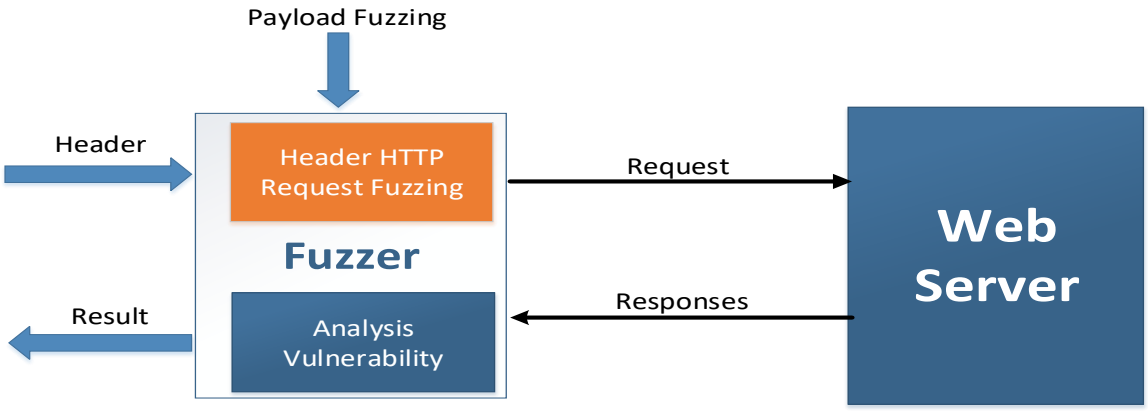
- Kỹ thuật Fuzzing, phần này đã trình bày khái quát về lịch sử, phân loại và ưu nhược điểm của kỹ thuật Fuzzing trong kiểm thử bảo mật.

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

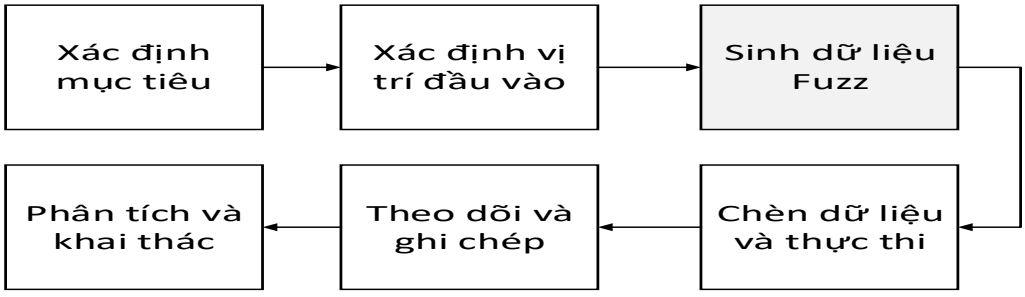
Quá trình kiểm thử chủ yếu được thực trên các điểm đầu vào của hệ thống, cụ thể các trường dữ liệu của request headers được gửi qua phương thức truyền dữ liệu của HTTP, chủ yếu là phương thức GET, POST. Fuzzer sẽ phải thực hiện việc thu thập toàn bộ các điểm đầu vào của hệ thống trước khi thực hiện fuzzing.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



Hình 2.1. Mô hình Fuzzing cho ứng dụng web

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

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

2.2.1. Thu thập các điểm đầu vào

2.2.1.1. Cơ chế trích xuất URL từ mã HTML

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).

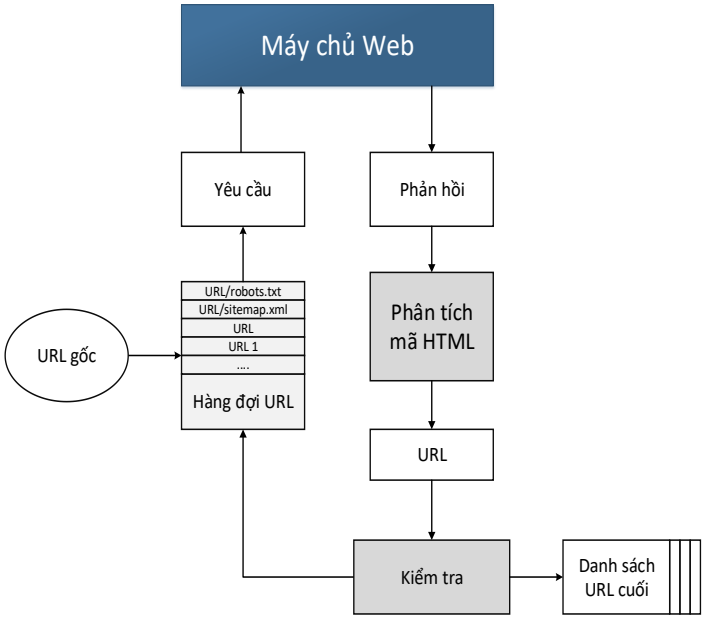
2.2.1.2. Phương pháp thu thập

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.

Quy trình thu thập điểm đầu vào

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.



Hình 2.4. Mô hình thu thập URL theo mã HTML

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
URLFuzzing	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
URLFuzzing	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

Nguyên tắc chèn dữ liệu vào data post:

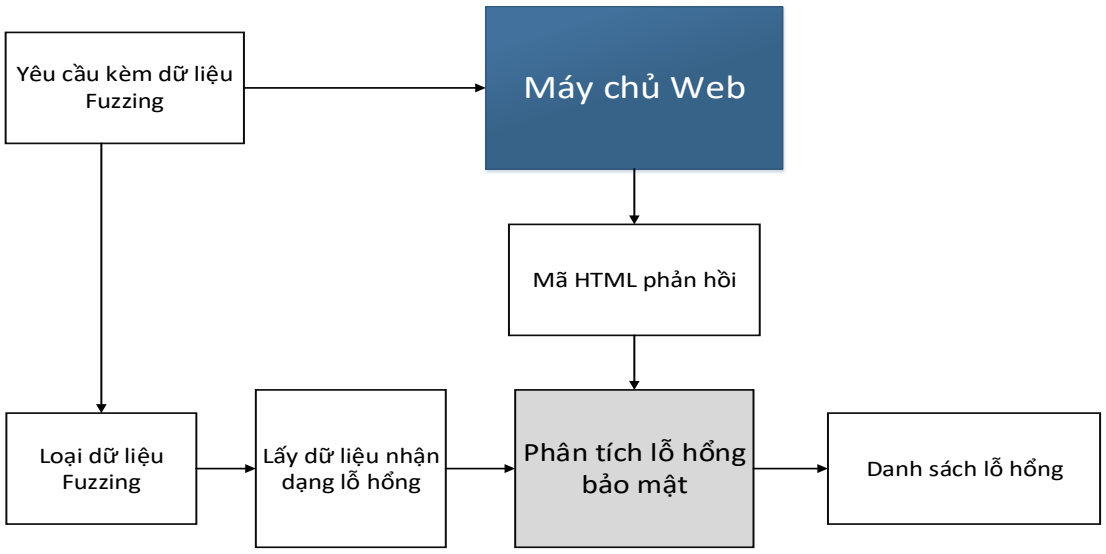
Bảng 2.4. Chèn dữ liệu fuzzing vào phương thức POST

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

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

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.

2.2.4. Phát hiện lỗ hổng dựa trên đặc trưng

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 xuất hiện của chữ ký	Cross Site Script	Chữ ký được đính kèm với dữ liệu fuzzing, 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.

2.2.5. Phát hiện lỗ hổng dựa trên cấu hình

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.

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?

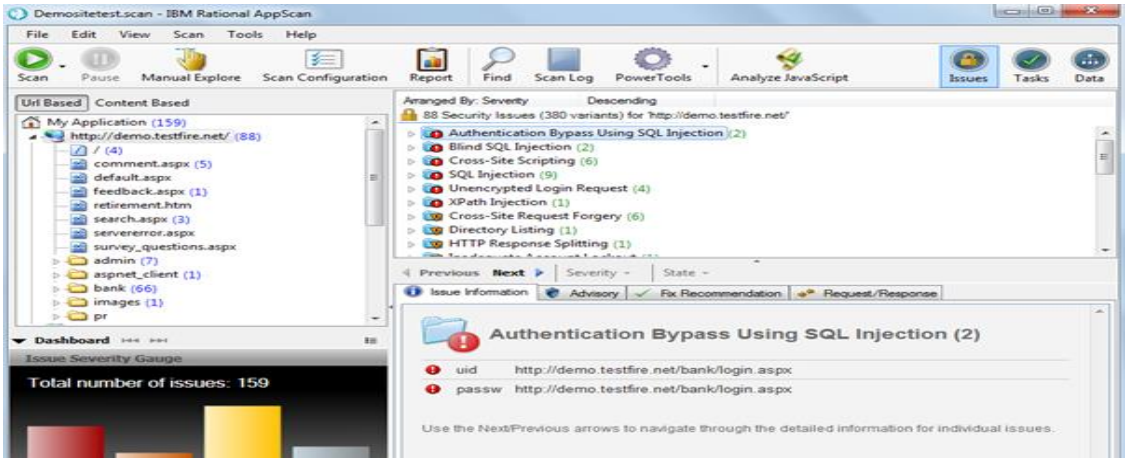
2.3.2. Một số tool tiêu biểu

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

Nikto 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.

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.

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, 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".

Kết luận chương 2

Trongchương 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

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

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.

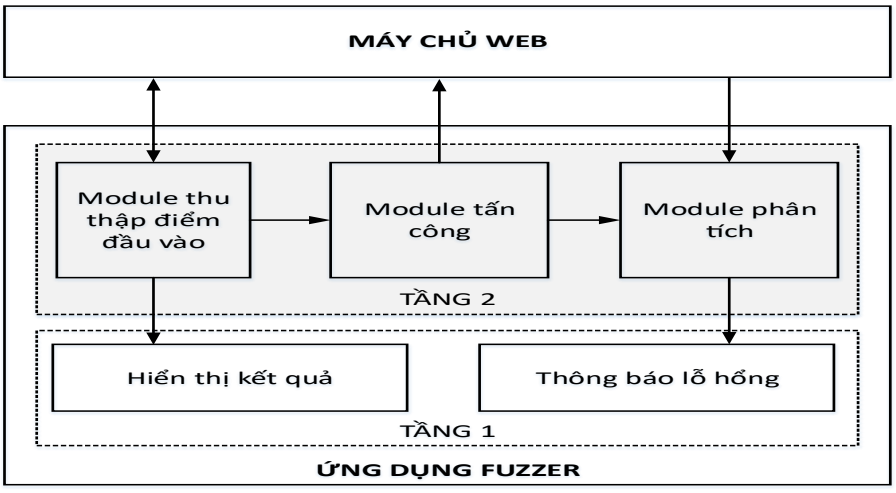
Vấn đề bảo mật Website ngân hàng BCEL nói riêng và ngân hàng ở Lào nói chung vẫn chưa được các cơ quan, doanh nghiệp ở Lào chú trọng đầu tư. Không phải tất cả các tổ chức, doanh nghiệp đều có thể trang bị đầy đủ cũng như có thể đảm bảo an toàn, bảo mật thông tin một cách toàn diện.

Việc kiểm thử 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.

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

3.2.1. Sơ đồ kiến trúc ứng dụng

❖ **Kiến trúc chương trình ứng dụng**



Hình 3.1. Kiến trúc phân tầng của ứng dụng

3.2.2. Xây dựng ứng dụng

❖ **Ngôn ngữ sử dụng**

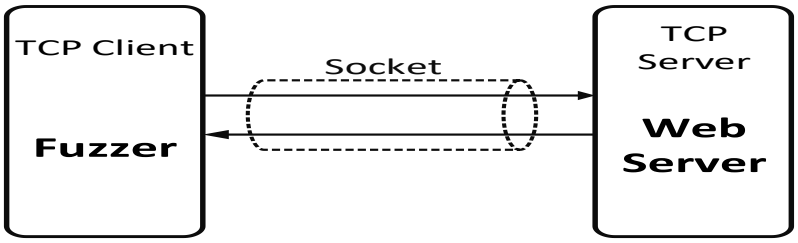
C# được sử dụng để xây dựng ứng dụng này với 3 lý do chính như sau:

- 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**



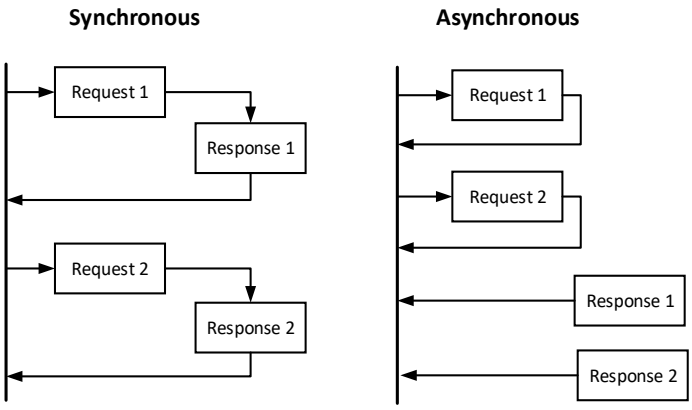
Hình 3.2. Giao tiếp giữa Fuzzer và Server

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**

Lập trình bất đồng bộ thường được sử dụng trong các xử lý tiềm ẩn độ trễ về mặt thời gian, như việc truy cập tới website, việc truy cập tài nguyên của một website thường có sự chậm trễ. Vì vậy, để tránh lãng phí thời gian và hiệu năng trong gửi yêu cầu và nhận phản hồi thì lập trình bất đồng bộ là một giải pháp tối ưu. Lập trình bất đồng bộ được mô tả như hình 3.3.



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.

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

❖ **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.

❖ **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 fuzzying.

```
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.

❖ **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.

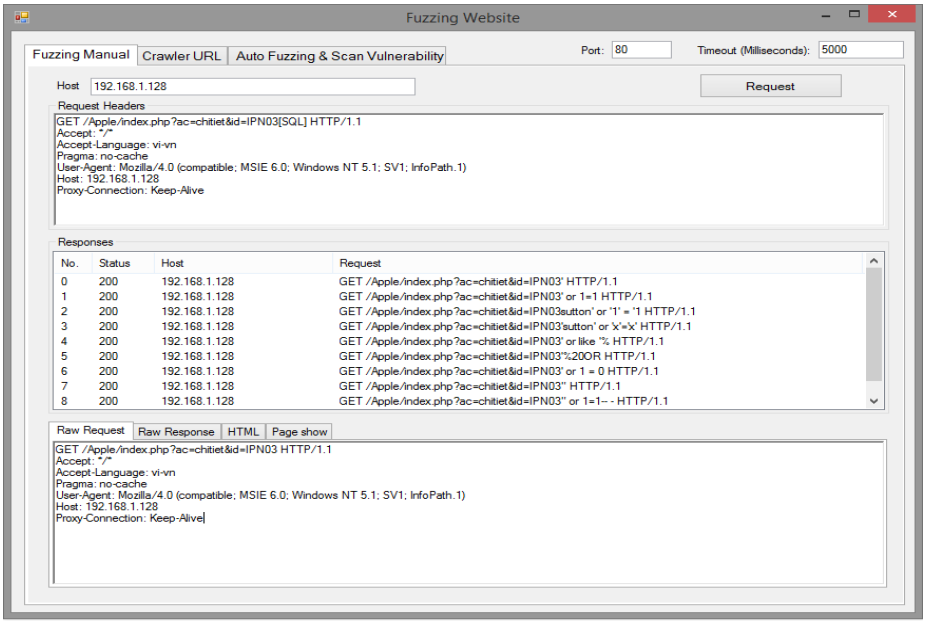
❖ ***Cài đặt ứng dụng thử nghiệm***

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***

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. Sau khi hoàn thành các thông số, nhấn Request để thực hiện chức năng này.

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



Hình 3.8. Giao diện Fuzzing thủ công

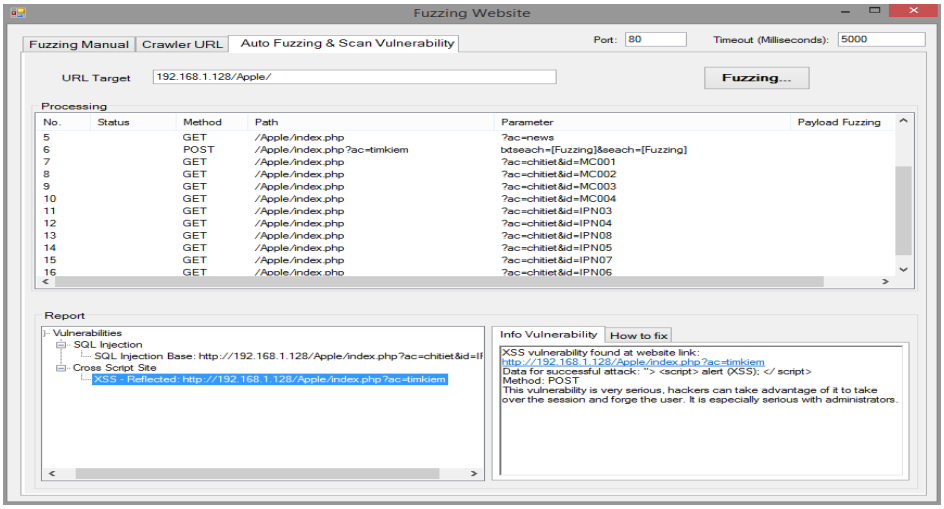
❖ ***Chức năng cào thông tin URL (Crawler URL)***

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,...

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

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.

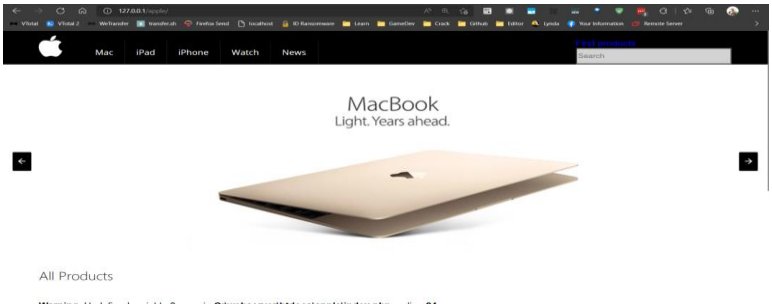
- Bộ dữ liệu Fuzzing chưa đa dạng để phát hiện được tất cả các loại lỗi.

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

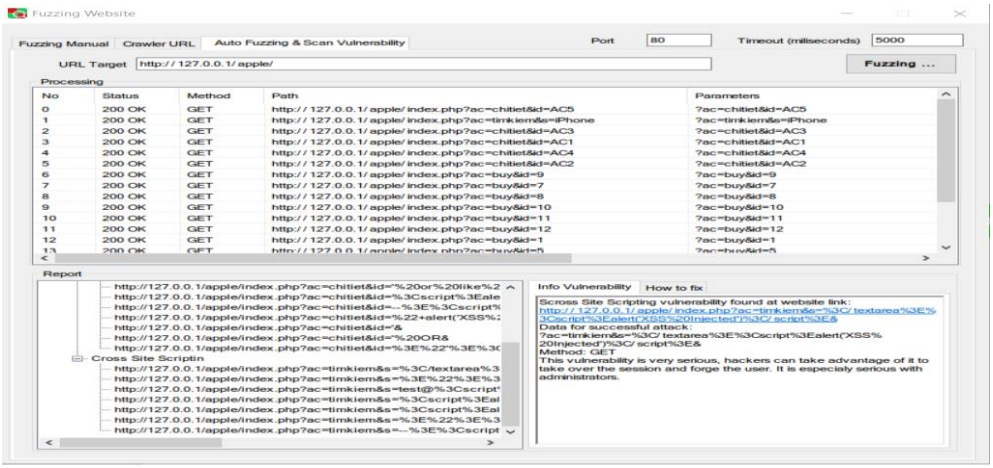
STT	Quá trình	Số lượng truy vấn	Thời gian thực thi	Kết quả
1	Crawling	19	~444ms	Thu được 19 URL
2	Filtering	17	~205ms	Lọc còn 17 URL
3	Fuzzing	17	~759ms	Phát hiện 3 lỗ hổng

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

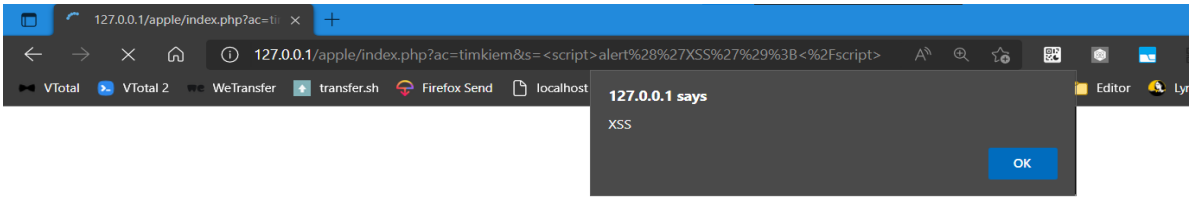
STT	Lỗ hổng	URL Cơ sở	Dữ liệu fuzz
1	SQL Injection	http://127.0.0.1/apple/index.php?ac=chitiet&id=ACx	'
2	Cross Script Site	http://127.0.0.1/apple/index.php?ac=timkiem&s=<query>	"><script> alert("XSS"); </script>

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/.