HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

THÁI THỊ MỸ HẠNH

NGHIÊN CỨU CÁC KỸ THUẬT KIỂM THỬ BẢO MẬT ỨNG DỤNG WEB

LUẬN VĂN THẠC SỸ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI - 2020

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

THÁI THỊ MỸ HẠNH

NGHIÊN CỨU CÁC KỸ THUẬT KIỂM THỬ BẢO MẬT ỨNG DỤNG WEB

CHUYÊN NGÀNH : HỆ THỐNG THÔNG TIN

MÃ SỐ : 8.48.01.04

LUẬN VĂN THẠC SỸ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS TS. LÊ HỮU LẬP

HÀ NỘI – 2020

LỜI CAM ĐOAN

Luận văn này là thành quả của quá trình học tập nghiên cứu của tôi cùng sự giúp đỡ, khuyến khích của các Quý thầy cô, đặc biệt là PGS.TS Lê Hữu Lập sau hai năm theo học chương trình đào tạo Thạc sỹ, chuyên ngành Hệ thống thông tin của Học viện Công nghệ Bưu chính Viễn thông.

Tôi xin cam đoan đây là công trình nghiên cứu của riêng. Nội dung của luận văn có tham khảo và sử dụng một số thông tin, tài liệu từ các nguồn sách, tạp chí được liệt kê trong danh mục các tài liệu tham khảo và được trích dẫn hợp pháp.

Tác giả

(Ký và ghi rõ họ tên)

Thái Thị Mỹ Hạnh

LỜI CẢM ƠN

Sau một thời gian dài học tập và nghiên cứu, cuối cùng tôi cũng đã hoàn thành luận văn tốt nghiệp này, đây là dịp tốt nhất để tôi có thể bày tỏ lòng biết ơn sâu sắc đến mọi người.

Tôi xin gửi lời cảm ơn sâu sắc đến thầy Lê Hữu Lập, đã tận hình hướng dẫn, định hướng cho tôi trong suốt thời gian thực hiện đề tài. Thầy đã cho tôi những lời khuyên quý báu để hoàn thành tốt luận văn.

Tôi xin cảm ơn Khoa Sau Đại học, Khoa Công Nghệ Thông Tin – Học viện Công nghệ Bưu chính Viễn thông, cảm ơn các thầy cô trong khoa đã tận tình giảng dạy, truyền đạt cho tôi những kiến thức quý báu trong những năm học vừa qua, giúp cho tôi có một nền tảng kiến thức vững chắc để thực hiện luận văn cũng như nghiên cứu học tập sau này.

Cuối cùng, tôi xin gửi lời cảm ơn đến tất cả bạn bè, anh, chị, đồng nghiệp, những người đã giúp đỡ, khích lệ cũng như phê bình, góp ý, giúp tôi hoàn thành luận văn một cách tốt nhất.

Hà Nội, tháng 12 năm 2019

Thái Thị Mỹ Hạnh

MỤC LỤC

DANH MỤC KÝ HIỆU, CHỮ VIẾT TẮT

STT	Từ viết tắt	Tiếng Việt	Tiếng Anh
1	API	Giao diện lập trình ứng dụng	Application Programming Interface
2	CSDL	Cơ sở dữ liệu	Database
3	CSRF	Giả mạo yêu cầu giữa các trang web	Cross Site Request Forgery
4	DAST	Kiểm tra bảo mật ứng dụng động	Dynamic Application Security Testing
5	DOM	Mô hình Đối tượng Tài liệu	Document Object Model
6	DDos	Tấn công từ chối dịch vụ phân tán	Distributed Denial of Service
7	HTTP	Giao thức truyền tải siêu văn bản	Hypertext Transfer Protocol
8	ID	Nhận dạng	Identification
9	ISECOM	Viện bảo mật và phương pháp mở	Institute for Security and Open Methodologies
10	OWASP	Dự án mở về bảo mật ứng dụng web	Open Web Application Security Project.

11	PHP	Ngôn ngữ lập trình kịch bản	Hypertext Preprocessor
12	SAST	Kiểm tra bảo mật ứng dụng tĩnh	Static Application Security Testing
13	SQL	Ngôn ngữ truy vấn dữ liệu	Structured Query Language
14	XML	Ngôn ngữ đánh dấu mở rộng	Extensible Markup Language
15	XSS	Lỗ hổng Cross-site scripting	Cross Site Scripting
16	URL	Định vị Tài nguyên thống nhất	Uniform Resource Locator

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH VẼ, SƠ ĐỒ

MỞ ĐẦU

Ngày nay với cơ sở hạ tầng về mạng Internet phát triển rất mạnh mẽ, cùng với đó là sự ra đời của hàng hoạt các ứng dụng Web để đáp ứng nhu cầu của người sử trong mọi lĩnh vực của cuộc sống. Các ứng dụng web cho các dịch vụ khác nhau đã nhận được sự tin tưởng của khách hàng qua một thời gian dài. Hàng triệu triệu dữ liệu được tải và chia sẻ giữa các nền tảng khi mọi người cho rằng các giao dịch được giám sát an toàn. Tuy nhiên, khi các cuộc tấn công trên mạng tiếp tục gây ra sự hoang mang, nguy cơ bảo mật ứng dụng và dữ liệu của chúng ta trong lĩnh vực kỹ thuật số ngày càng tăng. Với sự nổi lên của các công nghệ mới là thách thức trong việc cung cấp một môi trường an toàn. Kiểm tra tính bảo mật của các ứng dụng web là rất quan trọng. Nó là một kỹ thuật kiểm thử phần mềm nhằm mục đích xác minh rằng chức năng của phần mềm có khả năng chống lại các cuộc tấn công và dữ liệu được phần mềm xử lý được bảo vệ. Để thiết lập các yêu cầu chung mà phần mềm phải đáp ứng, có các tiêu chuẩn bảo mật phần mềm. Luận văn này nhằm mục đích mô tả và áp dụng một quy trình cần thiết để xác minh tính bảo mật của một ứng dụng web.

Đề tài "**Nghiên cứu các kỹ thuật kiểm thử bảo mật ứng dụng web**" nhằm nghiên cứu và thử nghiệm các kỹ thuật nhằm phát hiện để ngăn chặn kịp thời các nguy cơ về an ninh, bảo mật ứng dụng web. Bảo mật trang web đòi hỏi sự cảnh giác trong tất cả các khía cạnh của thiết kế và quá trình sử dụng trang web. Luận văn giúp hiểu các mối đe dọa đến từ đâu và cách để bảo vệ ứng dụng web của mình trước các cuộc tấn công phổ biến nhất.

1. Mục đích nghiên cứu

Tìm hiểu các vấn đề về bảo mật, các phương pháp và các công cụ kiểm thử bảo mật website lựa chọn công cụ kiểm thử và thực hiện thử nghiệm kiểm thử bảo mật website qua đó đánh giá kết quả

- Tìm hiểu tổng quan về bảo mật

- Nghiên cứu các loại kiểm thử bảo mật, quy trình kiểm thử bảo mật
- Nghiên cứu các kỹ thuật và các công cụ tương ứng trong kiểm thử bảo mật website
- Phân tích đánh giá, lựa chọn công cụ và thực hiện kiểm thử bảo mật website
- Đánh giá kết quả đạt được.

2. Đối tượng và phạm vi nghiên cứu

Trong khuôn khổ luận văn thuộc loại nghiên cứu và ứng dụng, em chỉ giới hạn nghiên cứu các vấn đề sau:

- Nghiên cứu các kỹ thuật và các công cụ tương ứng trong kiểm thử bảo mật website
- Phân tích đánh giá, lựa chọn công cụ và thực hiện kiểm thử bảo mật website https://duticrm.info/ và https://hack-yourself-first.com/

3. Bố cục luận văn

Bố cục luận văn bao gồm phần mở đầu, phần kết luận và các chương nội dung được tổ chức như sau:

- **Chương 1**: Tổng quan về bảo mật website và các ứng dụng trên web. Chương này đưa ra các khái niệm tổng quan về bảo mật website, các ứng dụng trên web và tình trạng bảo mật website hiện nay.
- **Chương 2**: Các kỹ thuật kiểm thử bảo mật ứng dụng web. Chương hai tập trung vào việc đưa ra các khái niệm tổng quan về kiểm thử bảo mật website; các phương pháp, quy trình, kỹ thuật kiểm thử bảo mật ứng dụng web.
- **Chương 3**: Các công cụ kiểm thử và thực nghiệm kiểm thử bảo mật ứng dụng web.. Nội dung chương 3 trình bày về công cụ kiểm thử Zed Attack Proxy. Sau khi phân tích công cụ đã trình bày, thực hiện kiểm thử bảo mật cho website https://duticrm.info và và https://hack-yourself-first.com/.

Mặc dù có nhiều cố gắng nhưng do thời gian hạn chế nên luận văn không tránh khỏi những khiếm khuyết. Kính mong các thầy cô và đồng nghiệp thông cảm và cho các ý kiến góp ý.

Xin trân trọng cảm ơn!

Tác giả

Thái Thị Mỹ Hạnh

Chương 1. TỔNG QUAN VỀ BẢO MẬT VÀ CÁC LỖ HỔNG BẢO MẬT TRÊN WEB

Trong chương này, luận văn sẽ giới thiệu một cách tổng quan về bài toán bảo mật website, sự cần thiết bảo mật website, các lỗ hổng bảo mật và tình trạng tấn công các trang web hiện nay.

1.1 Tổng quan về bảo mật [1], [2]

1.1.1 Bảo mật website

Bảo mật là bảo vệ an toàn thông tin trước những "tay" chuyên rình mò thông tin của người khác (Hình 1.1).

Hình 1.1. Bảo mật website

Với hệ thống thông tin của doanh nghiệp, thông tin chính là tiền. Với website của doanh nghiệp, việc có thể thay đổi thông tin trên đó sẽ ảnh hưởng xấu đến thương hiệu. Trang web là thương hiệu của một đơn vị hay một cá nhân, mặt tiền của một cửa hàng và thường là lần tiếp xúc đầu tiên với khách hàng. Nếu nó không an toàn và bảo mật, những mối quan hệ kinh doanh quan trọng đó có thể bị ảnh hưởng.

Các mối đe dọa có thể xuất hiện dưới nhiều hình thức - lây nhiễm một trang web có phần mềm độc hại để phát tán phần mềm độc hại đó cho khách truy cập trang

web, đánh cắp thông tin khách hàng, như tên và địa chỉ email, đánh cắp thẻ tín dụng và thông tin giao dịch khác, thêm trang web vào botnet của các trang web bị nhiễm và thậm chí chiếm quyền điều khiển hoặc đánh sập trang web. Nếu ứng dụng thương mại điện tử, thiệt hại sẽ rất lớn nếu website bị tấn công, khách hàng sẽ không bao giờ tin tưởng để cung cấp những thông tin cá nhân trên website nữa.

Bảo mật là sự thỏa hiệp giữa việc đảm bảo an toàn và chức năng hay khả năng sử dụng. Nếu bảo mật của hệ thống quá chặt chẽ, nó sẽ trở nên rất khó sử dụng hoặc khó hoạt động một cách hiệu quả. Nếu bảo mật quá đơn giản, hệ thống dễ bị tấn công và xâm nhập.

Tùy thuộc vào các yêu cầu của mỗi hệ thống mà có những mục đích về bảo mật khác nhau, nhưng chúng đều có điểm chung là: Đảm bảo sự an toàn dữ liệu cho hệ thống và bảo vệ các tài nguyên trên mạng trước sự tấn công nhằm phá vỡ hệ thống hoặc sử dụng trái phép các tài nguyên của một số người có chủ ý xấu.

Bảo mật website có thể hiểu là một loại chức năng hoặc thao tác quan trọng, cần thiết trong quá trình sử dụng và vận hành website, bảo vệ trang web khỏi sự truy cập, sử dụng, sửa đổi, phá hủy hoặc phá vỡ trái phép. Bảo mật website có nghĩa là bảo vệ trang web của bạn và dữ liệu khỏi các mối đe dọa ác ý, tin tặc, tấn công mạng, tấn công DDOS và phần mềm độc hại. Mục đích của kiểm tra bảo mật là khám phá các lỗ hổng của ứng dụng web để các nhà phát triển có thể loại bỏ các lỗ hổng này khỏi ứng dụng và làm cho ứng dụng web và dữ liệu an toàn khỏi mọi hành động trái phép.

Để trang web có thể hoạt động tốt, các nhà quản trị website cần thực hiện các thao tác bảo mật thường xuyên, kịp thời. Bảo mật trang web muốn hiệu quả đòi hỏi sự nỗ lực trên toàn bộ trang web: trong ứng dụng web, cấu hình của máy chủ web, chính sách tạo mật khẩu từ phía máy khách.

Quá trình phân tích bảo mật nên được chạy song song với phát triển ứng dụng Web. Nhóm lập trình viên và nhà phát triển chịu trách nhiệm phát triển mã cũng chịu

trách nhiệm thực hiện các chiến lược khác nhau, phân tích sau rủi ro, giảm thiểu và giám sát.

1.1.2 Bảo mật ứng dụng web

Bảo mật ứng dụng web là một nhánh của bảo mật thông tin liên quan cụ thể đến bảo mật của các trang web, ứng dụng web và dịch vụ web. Ở cấp độ cao, bảo mật ứng dụng web dựa trên các nguyên tắc bảo mật ứng dụng nhưng áp dụng chúng cụ thể cho các hệ thống Internet và web.

Bảo mật ứng dụng web là quá trình bảo mật dữ liệu bí mật được lưu trữ trực tuyến khỏi sự truy cập và sửa đổi trái phép. Điều này được thực hiện bằng cách thực thi các biện pháp chính sách nghiêm ngặt. Các mối đe dọa bảo mật có thể ảnh hưởng đến dữ liệu được lưu trữ với mục đích xấu là cố gắng truy cập vào thông tin nhạy cảm.

Bảo mật ứng dụng web nhằm giải quyết và đáp ứng ba điều kiện bảo mật, còn được gọi là các nguyên tắc bảo mật:

- Bí mật: Hạn chế quyền được phép truy nhập và công bố thông tin, bao gồm cả bảo vệ quyền riêng tư cá nhân và quyền sở hữu thông tin.
- Tính toàn vẹn: Xác định rằng dữ liệu trong ứng dụng Web là nhất quán và không bị sửa đổi bởi người dùng trái phép.
- Tính khả dụng: bảo đảm Ứng dụng web có thể truy nhập và sử dụng được kịp thời, tin cậy. Tính khả dụng (utility) khác với tính sẵn sàng (availability) ở chỗ ứng dụng web có thể sẵn sàng song vẫn vô dụng (không dùng được) nếu thiếu mã khóa để giải mã chẳng hạn.

1.2 Các lỗ hổng bảo mật [3], [4], [8]

Lỗ hổng bảo mật là tập hợp những điều kiện mà cho phép một kẻ xấu tấn công làm vi phạm những chính sách bảo mật một cách tương minh hoặc ngầm. Đó là những điểm yếu nằm trong thiết kế và cấu hình của hệ thống, lỗi của lập trình viên hoặc sơ

suất trong quá trình vận hành. “90% lỗ hổng bảo mật bắt nguồn từ ứng dụng web, 90% nhà quản trị chưa có cái nhìn tổng quan về bảo mật Web App”[3]. Đây là lý do dẫn tới số lượng các cuộc tấn công trên mạng ngày càng nhiều.

Một số nguyên nhân sâu xa của lỗ hổng bảo mật như sau:

- *Phức tạp*: Các lỗ hổng bảo mật tăng tỷ lệ thuận với sự phức tạp của một hệ thống. Sự phức tạp về phần mềm, phần cứng, thông tin, doanh nghiệp và quy trình giới thiệu nhiều lỗ hổng bảo mật hơn.
- *Kết nối*: Mỗi kết nối không bảo đảm là một con đường tiềm năng để các hacker khai thác thông tin
- *Lỗi thiết kế*: Không nên có bất kỳ lỗi thiết kế nào trong phần mềm và phần cứng. Những lỗi này có thể khiến doanh nghiệp gặp rủi ro đáng kể.
- *Cấu hình*: Cấu hình hệ thống kém làm tăng lỗ hổng bảo mật.
- *Đầu vào của người dùng*: Dữ liệu nhận được thông qua SQL injections, tràn bộ đệm, v.v., có thể được thiết kế để tấn công hệ thống nhận.
- *Sự quản lý*: Quản lý nên có một kế hoạch quản lý rủi ro thích hợp để tránh các lỗ hổng bảo mật trong hệ thống.
- *Mật khẩu*: Mật khẩu để tránh truy cập trái phép và bảo mật dữ liệu cá nhân của bạn. Mật khẩu không được bảo mật (chia sẻ với người khác, viết chúng xuống một nơi nào đó, cài đặt dễ đoán) cho phép tin tặc đoán mật khẩu của bạn một cách dễ dàng.
- *Lỗi của con người*: Các lỗi của con người như xử lý tài liệu không đúng cách, lỗi mã hóa, đưa ra mật khẩu cho các trang web lừa đảo là một nguồn lỗ hổng bảo mật đáng kể. Lỗi của con người có thể được ngăn chặn bằng cách đào tạo thích hợp cho nhân viên.
- *Giao tiếp*: Các kênh truyền thông như điện thoại, điện thoại di động, internet cung cấp phạm vi cho các lỗ hổng bảo mật.

OWASP Top ten là một tiêu chuẩn toàn cầu để phục vụ việc kiểm thử xâm nhập được dễ dàng hơn. Tiêu chuẩn này được đề xuất bởi một tổ chức phi lợi nhuận:

Open Web Application Security Project (OWASP- Dự án mở về bảo mật ứng dụng web). Dưới đây là TOP 10 lỗ hổng bảo mật web phổ biến nhất theo tiêu chuẩn OWASP năm 2017 [8]:

1.2.1 A1: Injection

- Tổng quan

Injection là một kĩ thuật cho phép những kẻ tấn công lợi dụng lỗ hổng trong việc kiểm tra dữ liệu nhập trong các ứng dụng web và các thông báo lỗi của hệ quản trị cơ sở dữ liệu để "tiêm vào" (inject) và thi hành các câu lệnh bất hợp pháp (không được người phát triển ứng dụng lường trước). Hậu quả của nó có thể rất lớn vì nó cho phép những kẻ tấn công có thể thực hiện các thao tác: vượt qua khâu xác thực, đánh cắp thông tin trong cơ sở dữ liệu, chèn, xóa, sửa đổi dữ liệu bên trong cơ sở dữ liệu, chiếm quyền điều khiển của hệ thống. Các lỗ hổng đặc trưng: SQL injection; OS injection; LDAP injection; NoSQL injection…Ví dụ điển hình của Injection:

- Kịch bản # 1: Một ứng dụng sử dụng dữ liệu không tin cậy khi xây dựng lệnh gọi SQL dễ bị tấn công sau:

```
String query = "SELECT * FROM accounts WHERE custID='" +
request.getParameter("id") + "'";
```

- Kịch bản # 2: Tương tự, một ứng dụng nếu tin tưởng quá vào các khung sẵn có, có thể dẫn đến các truy vấn vẫn dễ bị tấn công, (ví dụ: Ngôn ngữ truy vấn Hibernate (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID='" +
request.getParameter("id") + "'");
```

Trong cả hai trường hợp, kẻ tấn công đều sửa đổi giá trị tham số ‘id, trong trình duyệt của họ để gửi: hoặc’ 1, = 1 Ví dụ:

```
http://example.com/app/accountView?id=' or '1'='1
```

Điều này thay đổi ý nghĩa của cả hai truy vấn để trả về tất cả các bản ghi từ bảng “Account”. Các cuộc tấn công nguy hiểm hơn có thể sửa đổi hoặc xóa dữ liệu, hoặc thậm chí gọi các thủ tục được lưu trữ.

- Phòng chống

Lỗ hổng Injection xảy ra do các biến được nhập vào từ người dùng không được kiểm soát chặt chẽ trước khi xây dựng câu truy vấn tới CSDL. Đó chính là nguyên nhân chung nhất của các lỗ hổng dạng Injection.

Lỗ hổng Injection xảy ra khi có kết hợp cả 2 điều kiện:

- Có sự truy vấn tới CSDL
- Câu truy vấn chưa được kiểm soát sự an toàn

Có thể khẳng định rằng, một biến được nhập vào từ người dùng, qua nhiều bước xử lý trung gian xây dựng câu truy vấn tới cơ sở dữ liệu mà không có bất cứ bước kiểm tra sự an toàn nào thì chắc chắn sẽ mắc các lỗ hổng Injection. Đây cũng chính là điểm mấu chốt để nhận diện và phòng chống các lỗ hổng Injection. Phương pháp hữu hiệu nhất để chống tấn công Injection là kiểm soát thật chặt chẽ các giá trị nhập được nhập vào từ người dùng.

1.2.2 A2: Lỗi xác thực (Broken Authentication)

Đây là nhóm các vấn đề có thể xảy ra trong quá trình xác thực, quản lý phiên đăng nhập.

- Những điểm yếu gây ra lỗi xác thực như:

- Ứng dụng cho phép một công cụ tự động gửi nhiều yêu cầu để đăng nhập, sau đó những kẻ tấn công sẽ dùng công cụ để quét tên người dùng và mật khẩu để tìm ra cặp Tên người dùng/mật khẩu có trong ứng dụng.
- Ứng dụng cho phép sử dụng những mật khẩu yếu hoặc vô ý chưa xóa những cặp Tên người dùng/mật khẩu mặc định.
- Tính năng quên mật khẩu nhưng thiếu an toàn với những câu hỏi dạng kiến thức.
- Mật khẩu của người dùng không được mã hóa hoặc mã hóa đơn giản.

- Hiển thị mã phiên đăng nhập trong URL.
- Không tạo lại phiên đăng nhập sau khi đăng nhập thành công.

- Cách phòng tránh:
  - Nếu có thể nên để xác thực hai yếu tố.
  - Bảo vệ chống lại đăng nhập: Thực thi việc vô hiệu hóa tài khoản sau một số lần đăng nhập không hợp lệ được thiết lập.
  - Kiểm tra các mật khẩu yếu, tăng chiều dài của mật khẩu hoặc có thể xem xét không sử dụng top 1000 mật khẩu phổ biến
  - Thay đổi mật khẩu mặc định của các dịch vụ
  - Giới hạn và tăng thời gian chờ vài lần cố gắng đăng nhập.
  - Tạo lại ID phiên: ID phiên sẽ được tạo lại sau khi đăng nhập thành công.

1.2.3 A3: Rò rỉ dữ liệu nhạy cảm (Sensitive Data Exposure)

Những ứng dụng web không quản lý, bảo vệ thông tin nhạy cảm một cách đúng đắn và để rò rỉ những thông tin như tài chính, sức khoẻ của khách hàng, thông tin cá nhân, thông tin tài khoản,...Hoặc đôi khi việc chia sẻ những tài liệu cá nhân cho bên thứ 3 sẽ thuộc loại quy phạm quy định pháp luật tuỳ vào quy định của từng quốc gia. Mức độ ảnh hưởng: phụ thuộc vào thông tin bị lộ ra ngoài.

- Cách phòng tránh:
  - Tắt bộ nhớ đệm của các phản hồi với dữ liệu nhạy cảm. Tin tặc có thể nhận được các bản sao được lưu trong bộ nhớ cache và lấy cắp thông tin từ chúng.
  - Không lưu trữ dữ liệu nhạy cảm nếu không cần thiết.
  - Đảm bảo mã hoá tất cả các dữ liệu nhạy cảm.
  - Sử dụng những phương thức bảo mật trong khi truyền/nhận dữ liệu, TLS, HTTPS.
  - Kiểm tra việc chia sẻ dữ liệu nhạy cảm cho bên thứ ba.

1.2.4 A4: Tấn công thực thể bên ngoài XML (XML External Entities- XEE)

XML là một ngôn ngữ đánh dấu mở rộng, được ứng dụng rất rộng rãi. Nó sử dụng để trao đổi dữ liệu giữa các ứng dụng. Hiện nay có rất nhiều loại tài liệu sử dụng định dạng XML như rtf, pdf, tệp hình ảnh (svg) hay các file cấu hình.

Nhiều vấn đề XXE công khai đã được phát hiện, bao gồm cả tấn công các thiết bị nhúng. XXE xảy ra ở rất nhiều nơi bất ngờ, bao gồm cả các phụ thuộc lồng nhau sâu sắc. Cách dễ nhất là tải lên tệp XML độc hại, nếu được chấp nhận, một số kịch bản tấn công như sau:

- Kịch bản # 1: Kẻ tấn công cố gắng trích xuất dữ liệu từ máy chủ:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```

- Kịch bản # 2: Kẻ tấn công thăm dò mạng riêng của máy chủ bằng cách thay đổi dòng ENTITY ở trên thành:

```
<!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>
```

- Kịch bản # 3: Kẻ tấn công thực hiện một cuộc tấn công từ chối dịch vụ bằng cách bao gồm một tệp có khả năng vô tận:

```
<!ENTITY xxe SYSTEM "file:///dev/random" >]>
```

- Cách phòng tránh: Đào tạo các lập trình viên là điều cần thiết để xác định và giảm thiểu XXE. Bên cạnh đó, việc ngăn chặn XXE yêu cầu:
  - Bất cứ khi nào có thể, hãy sử dụng các định dạng dữ liệu ít phức tạp hơn như JSON và tránh tuần tự hóa dữ liệu nhạy cảm.
  - Vá hoặc nâng cấp tất cả các bộ xử lý và thư viện XML được sử dụng bởi ứng dụng hoặc trên hệ điều hành cơ bản.
  - Vô hiệu hóa thực thể bên ngoài XML và xử lý DTD trong tất cả các trình phân tích cú pháp XML trong ứng dụng.

1.2.5 A5: Kiểm soát truy cập bị hỏng (Broken Access Control)

Lỗi liên quan đến việc kẻ tấn công có thể sửa chữa để chiếm quyền của người khác. Lỗi này tập trung vào những chức năng liên quan đến việc quản lý quyền hạn (như Quản trị viên hay người dùng)

Việc hạn chế những thứ mà người dùng đã đăng nhập mới được xem hoặc được làm trong ứng dụng nếu hoàn thành không chính xác sẽ bị những kẻ tấn công lợi dụng mà không cần tới việc đăng nhập.

- Cách phòng tránh
  - Ngoại trừ những tài liệu công cộng, việc phân quyền nên để "Từ chối" sẽ là mặc định
  - Nhật ký lỗi kiểm soát truy cập, quản trị viên cảnh báo khi thích hợp (ví dụ: lỗi lặp lại).
  - Nên có cơ chế giới hạn tốc độ truy cập và bộ điều khiển để giảm thiểu tác hại từ công cụ tấn công tự động.

1.2.6 A6: Lỗi cấu hình (Security Misconfiguration)

- Khi cấu hình chưa đủ hoặc chưa đúng sẽ dẫn đến hậu quả liên quan đến bảo mật. Kẻ tấn công sẽ khai thác những lỗ hổng đó trong phần cấu hình để tấn công vào hệ thống:
  - Không cập nhật những lỗ hổng liên quan đến bảo mật trên server dẫn đến tồn tại những lỗ hổng (Ví dụ một ứng dụng được phát hành từ tháng 1. Trong khoảng thời gian từ tháng 1 đến tháng 9, nhà phát hành có thể đưa ra nhiều bản vá những lỗ hổng bảo mật nhưng chưa cập nhật trên server nên sẽ bị kẻ tấn công khai thác).
  - Thư mục, tệp được cấu hình chưa đúng, chưa hợp lý. Ví dụ như một thư mục cá nhân nhưng cấu hình theo chế độ công khai.
  - Bật những dịch vụ không cần thiết, những dịch vụ này không sử dụng đến nên sẽ quản lý không tốt. Kẻ tấn công có thể tấn công vào những dịch vụ đó thay vì trực tiếp vào những dịch vụ đang chạy.

- Sử dụng những tài khoản mặc định khi cấu hình.
- Lộ trang quản lý trên URL. Ví dụ như …/admin; …./administrator,…/quantri
- Cấu hình SSL (https)
- Cấu hình quyền truy cập của các loại tài khoản khác nhau

- Cách phòng tránh
  - Nên xem xét dọn rác những tính năng, dịch vụ không cần thiết.
  - Kiểm tra và dọn dẹp những tài khoản mặc định.
  - Theo dõi các bảng cập nhật của từng dịch vụ.

1.2.7 A7: Lỗ hổng Cross Site Scripting- XSS

- Tổng quan

XSS là một thuật ngữ được sử dụng để mô tả một lớp các cuộc tấn công cho phép kẻ tấn công chèn các tập lệnh phía máy khách thông qua trang web vào trình duyệt của những người dùng khác. Kẻ tấn công chèn các đoạn mã JavaScript vào ứng dụng web. Khi đầu vào này không được lọc, chúng sẽ được thực thi mã độc trên trình duyệt của người dùng.

- Những lý do lỗi XSS nguy hiểm:
  - Các lỗ hổng XSS trong lịch sử phổ biến hơn bất kỳ mối đe dọa bảo mật nào khác trên ứng dụng web hiện tại.
  - Lỗi này khó để nhận diện và khắc phục. Do trên trang web có rất nhiều tham số, để quản lý hết tất cả các tham số rất khó. Kẻ tấn công chỉ cần khai thác thành công một trong những tham số đó thì có thể chèn mã độc vào tham số đó.
  - Có thể đánh cắp thông tin, ID phiên làm việc, lấy được cookie của người dùng trên hệ thống hoặc lừa người dùng đến các trang web độc hại. Khi kẻ tấn công có cookie, họ có thể đăng nhập vào một trang web như thể họ là

người dùng và làm bất cứ điều gì người dùng có thể, chẳng hạn như truy cập chi tiết thẻ tín dụng của họ, xem chi tiết liên hệ hoặc thay đổi mật khẩu.

- Phòng chống

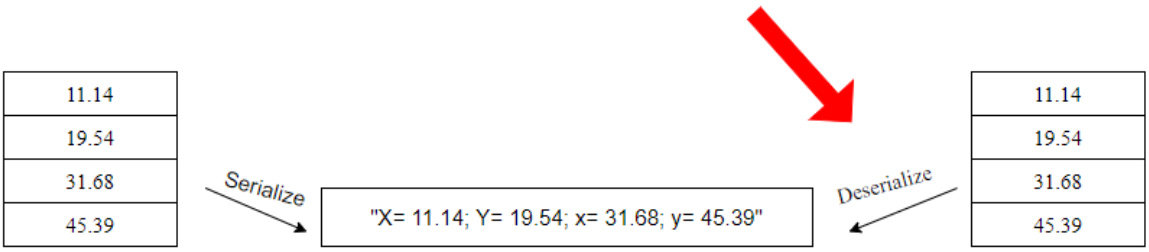
XSS là một lỗ hổng rất phổ biến và rất nguy hiểm đối với người dùng hệ thống. Đối với các dữ liệu được nhận từ người dùng, khi thực hiện việc hiển thị ta cần mã hóa tất cả các giá trị được in ra. Khi đó các đoạn mã độc sẽ không thể thực thi được. Trong các ngôn ngữ lập trình đều có các hàm hỗ trợ việc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

- Hàm htmlentities() – trong ngôn ngữ PHP
- Hàm htmlEncode() – trong ngôn ngữ C#

Trong jsp cung cấp cú pháp: ${specialCharString} để thực hiện mã hóa html tag.

1.2.8 A8: Chuyển đổi cấu trúc dữ liệu không an toàn(Insecure Deserialization)

Quá trình khôi phục lại dữ liệu, thông tin như ban đầu không an toàn, tạo lỗ hổng cho những kẻ tấn công khai thác.



Hình 1.2: Khôi phục lại dữ liệu không an toàn

Ví dụ khi đăng nhập vào tài khoản, tài khoản đó tạo ra một phiên làm việc mới, một cookie mới có chứa đối tượng (object) như sau:

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user";i:3;s:32:"b6a8b3bea87fe0e0502
2f8f3c88bc960";}
```

Đối tượng (object) này được lưu vào trong trình duyệt. Khi trình duyệt đọc sẽ thấy các thông tin như ID người dùng, vai trò, hàm băm mật khẩu và trạng thái khác…Kẻ tấn công thay đổi đối tượng từ “user” thành “admin” để cung cấp cho mình quyền quản trị viên:

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin";i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

1.2.9 A9: Sử dụng các thành phần có lỗ hổng (Using Components with Know Vulnerabilities)

Ứng dụng thường kết hợp với các thư viện khác, hoặc những ứng dụng mã nguồn mở thì cài thêm plug-in,...Việc dùng những thành phần, thư viện, frameworks chứa sẵn những lỗ hổng sẽ làm cho ứng dụng của bạn dễ bị khai thác hơn. Việc tận dụng những ứng dụng đã có và cộng với một khối lượng code-base của nó khá lớn dễ dẫn đến bạn không hiểu và mất kiểm soát hay tệ hơn là có cả nguy cơ bảo mật bên trong những thư việc này.

- Cách phòng tránh
  - Xác nhận những thành phần, thư viện đang dùng có những cái nào bị cũ hay bị lỗi liên quan đến bảo mật hay không.
  - Nên rà soát và xóa những thư viện, tính năng không cần thiết.
  - Theo dõi những lỗi bảo mật để kịp thời cập nhật những thành phần trong ứng dụng.

1.2.10 A10: Không ghi nhật ký và giám sát đầy đủ (Insufficient Logging & Monitoring)

Trang web dễ bị lỗi nếu không có tính năng ghi nhật ký sự kiện lại. Những kẻ tấn công dựa vào việc thiếu giám sát và phản ứng kịp thời để đạt được mục đích mà không bị phát hiện.

Ví dụ kịch bản tấn công: Một phần mềm diễn đàn dự án nguồn mở được điều hành bởi một nhóm nhỏ đã bị tấn công bằng cách sử dụng lỗ hổng trong phần mềm. Những kẻ tấn công đã tìm cách xóa sạch kho lưu trữ mã nguồn nội bộ có chứa phiên bản tiếp theo và tất cả các nội dung của diễn đàn. Mặc dù mã nguồn có thể được phục hồi, nhưng việc thiếu giám sát, ghi nhật ký hoặc cảnh báo dẫn đến ảnh hưởng nghiêm trọng dẫn đến việc có thể không thể hoạt động diễn đàn nữa.

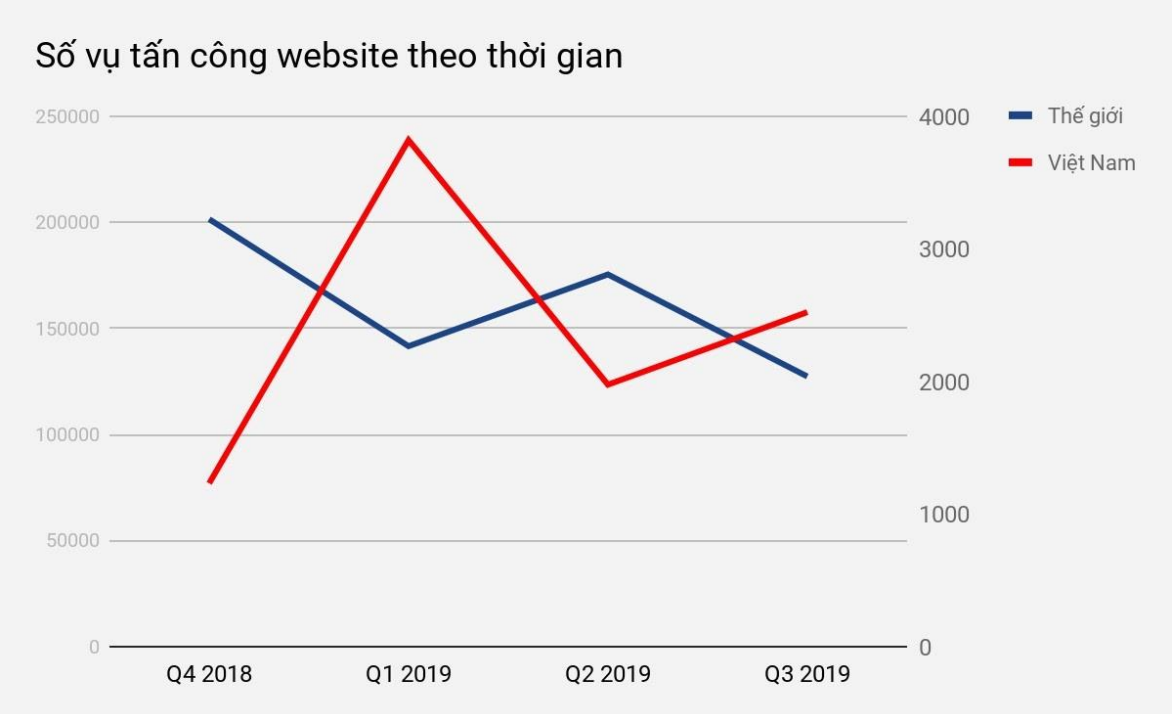
- Cách đề phòng
  - Đảm bảo tất cả các lỗi đăng nhập, kiểm soát truy cập và lỗi xác thực đầu vào phía máy chủ có thể được ghi lại với bối cảnh người dùng đủ để xác định các tài khoản đáng ngờ hoặc độc hại và giữ đủ thời gian để cho phép phân tích lỗ hổng
  - Thiết lập giám sát và cảnh báo hiệu quả sao cho các hoạt động đáng ngờ được phát hiện và phản hồi kịp thời.
  - Đảm bảo rằng các bản ghi nhật ký được tạo ở định dạng có thể dễ dàng sử dụng bởi các giải pháp quản lý nhật ký tập trung
  - Thiết lập một kế hoạch ứng phó và khắc phục sự cố

1.3 Tình trạng tấn công các trang web hiện nay [17]

Ngày nay với sự phát triển của phương tiện truyền thông xã hội, hệ thống Internet banking, các trang mua sắm trực tuyến, ví điện tử,... ngày càng có nhiều thông tin nhạy cảm được lưu trữ và trao đổi giữa các hệ thống khác nhau. Dữ liệu này bao gồm số tài khoản ngân hàng, số thẻ tín dụng, mật khẩu, và các thông tin tài chính, thông tin cá nhân khác. Giá trị của thông tin cao, và nhu cầu bảo vệ thông tin cũng vậy. Đồng thời, các công ty khởi nghiệp được hình thành với tốc độ ngày càng tăng, và họ đổ xô vào việc xuất bản phần mềm mới ra thị trường. Các công ty nhỏ, từ một đến một vài nhà phát triển quy mô nhỏ, thường không có đủ tài nguyên nên các khía cạnh bảo mật ứng dụng mới có nguy cơ có thể bị bỏ qua.

Khi những kẻ tấn công ngày càng nhắm mục tiêu vào các ứng dụng web, chúng có thể tinh chỉnh và thử nghiệm các phương pháp của chúng, tăng độ tinh vi của chúng. Ngay cả khi một công ty tuân theo các thực tiễn tốt nhất để bảo vệ bản thân trước các cuộc tấn công ứng dụng web phổ biến nhưng điều này có thể không đủ. Đột nhập vào các ứng dụng web có thể sinh lợi cho bọn tội phạm. Họ có động lực sử dụng các phương pháp và công cụ tấn công mới nhất và tốt nhất, và chúng có thể có các nguồn lực của tội phạm có tổ chức đằng sau chúng. Nhiều tội phạm mạng ngày nay thậm chí còn được tài trợ bởi các nhà nước, các quốc gia duy trì đội hack nội bộ của riêng họ và thiết bị của riêng họ. Những người được giao nhiệm vụ phát triển phần mềm độc hại đang hoạt động, để thực hiện nhiều điều chỉnh hơn, thông qua việc sử dụng các máy chủ chỉ huy và kiểm soát.

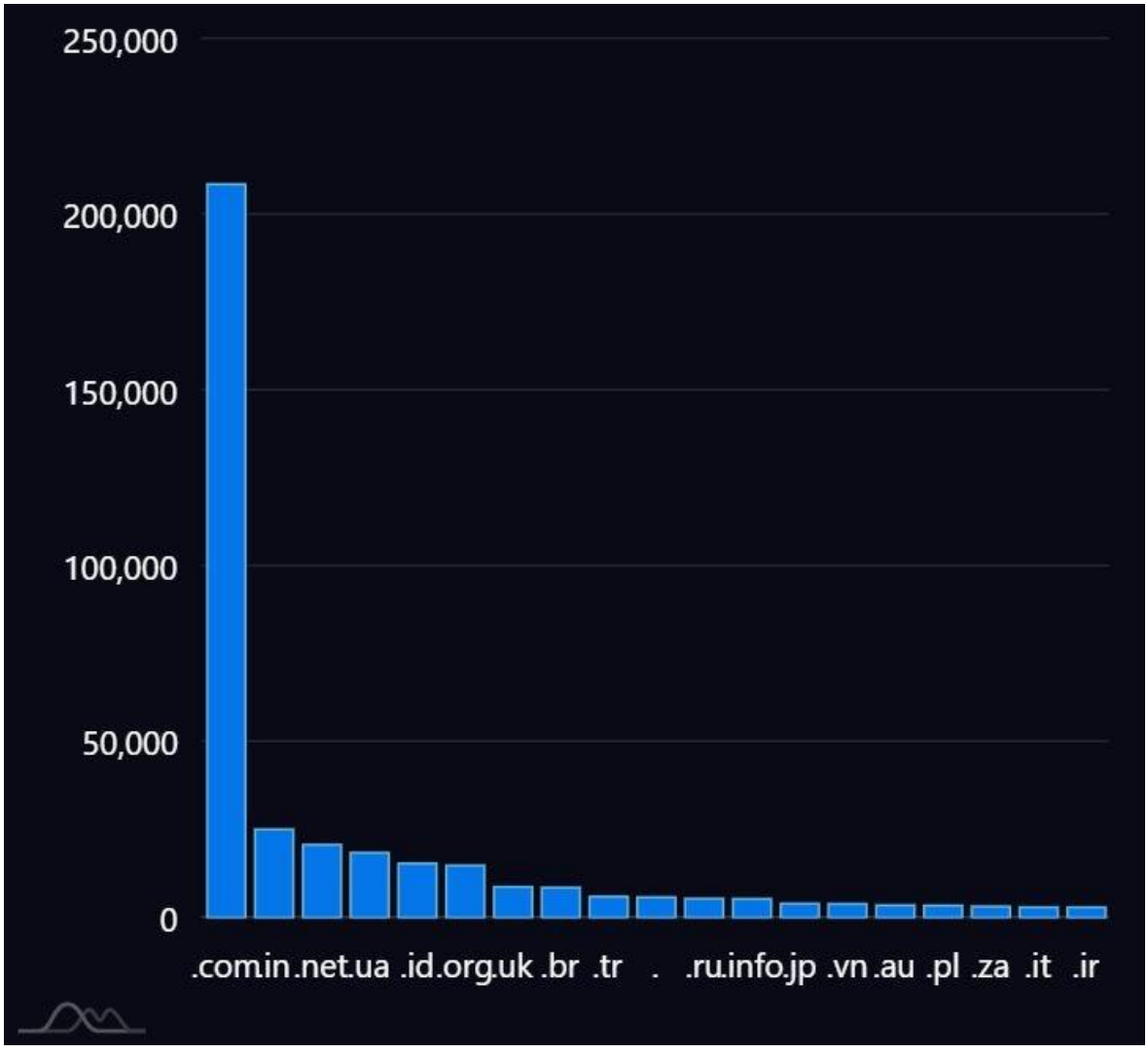
Theo Báo cáo an ninh website quý 3 năm 2019 của CTCP An ninh mạng CyStack Việt Nam [17], trong 4 quý gần nhất, số website bị xâm phạm trên toàn tế giới có xu hướng tăng cao vào cuối năm. Xu hướng này ngược lại tại Việt Nam, số vụ tấn công tăng mạnh vào đầu năm (Quý 1). (Hình 1.3)



Hình 1.3 Biểu đồ đường số vụ tấn công website theo thời gian thực

Trong quý 3 năm 2019 vừa qua, hệ thống CyStack Attack Map đã ghi nhận 127.367 website bị tấn công và chiếm quyền điều khiển. Như vậy, số website này đã giảm 27% so với con số 175.451 website bị tấn công trong quý 2. Cụ thể trong quý 3, số lượng website tháng 7, 8 và 9 lần lượt là 38.385, 44.848 và 44.134, giảm hơn so với mức trung bình 42.483 website/tháng của quý trước.

Tên miền .com phổ biến vẫn là đối tượng được nhắm đến nhiều nhất bởi các hacker, sau đó là tên miền .net với 5,99%. Ngoài ra còn có tên miền đặc trưng của các quốc gia như: .in (Ấn Độ), .ua (Australia), .id (Indonesia), .br(Brazil), .ru (Nga), .vn (Việt Nam), … (Hình 1.4).



Hình 1.4. Biểu đồ thống kê các vụ tấn công website theo tên miền

Trong 9 tháng đầu năm 2019, Việt Nam đứng thứ 11 trong số các quốc gia bị hack nhiều nhất thế giới và đứng số 3 trong Đông Nam Á (sau Indonesia và Singapore) với tổng số website bị xâm phạm là 8,406. Theo các chuyên gia, tình trạng này diễn ra do công tác bảo đảm an toàn an ninh mạng tại Việt Nam còn chưa được chú trọng. Các doanh nghiệp vẫn chưa ý thức trang bị giải pháp an ninh mạng tổng thể và đồng bộ cho toàn bộ các máy tính trong mạng nội bộ. Chỉ đợi đến khi trang web bị tấn công rồi mới tìm cách khắc phục sự cố.

Trong các tên miền bị tấn công tại Việt Nam, tên miền .com chiếm tỉ lệ lớn nhất với 55%, theo sau bởi các tên miền .vn, .edu, .gov và .org với lần lượt 37%, 4%, 2% và 1%. Có thể thấy đối tượng chủ yếu của tin tặc vẫn là các trang web thương mại và bán lẻ, lĩnh vực được quan tâm tiếp theo là giáo dục và cuối cùng là các trang web thuộc chính phủ cũng như các tổ chức phi chính phủ.

Trước những diễn biến khó lường của các cuộc tấn công mạng cùng tốc độ phát triển không ngừng nghỉ của Internet mỗi doanh nghiệp tổ chức cần phải có những kế hoạch cụ thể để triển khai hệ thống bảo mật cũng như hệ thống ứng cứu sự cố của mình nhằm giảm thiểu rủi ro thiệt hại do mất an toàn thông tin gây nên.

1.4 Kết chương

Có thể nói vấn đề bảo mật website và các ứng dụng là hết sức hệ trọng và cần thiết đối với mỗi đơn vị doanh nghiệp và cá nhân khi thiết lập website của mình. Có rất nhiều lỗ hổng bảo mật website để nảy sinh ra các nguy cơ tấn công của tội phạm ngày càng gia tăng.

Trong chương 2, luận văn sẽ đi sâu trình bày về các kỹ thuật kiểm tra thử nghiệm về độ bảo mật của các website và ứng dụng trên nó.

Chương 2. CÁC KỸ THUẬT KIỂM THỬ BẢO MẬT ỨNG DỤNG WEB

Trong chương này luận văn tập trung trình bày về các kỹ thuật kiểm thử bảo mật ứng dụng web.

2.1 Kiểm thử bảo mật ứng dụng web [2], [7]

Trong những năm gần đây, các ứng dụng web có xu hướng trở thành phổ biến. Một số lượng lớn các giao dịch điện tử bao gồm thương mại điện tử, ngân hàng điện tử, học tập điện tử, các hoạt động của chính phủ ... có thể được tiến hành trực tuyến bất cứ lúc nào và bất cứ nơi đâu. Do vậy việc đảm bảo an toàn thông tin trên môi trường internet là một vấn đề vô cùng quan trọng.

2.1.1 Ứng dụng web

Ứng dụng là một loại chương trình có khả năng làm cho máy tính thực hiện trực tiếp một công việc nào đó người dùng muốn thực hiện.

Ban đầu, các website chỉ bao gồm text, hình ảnh và video, liên kết với nhau thông qua các link. Tác dụng của website là lưu trữ và hiển thị thông tin. Người dùng chỉ có thể đọc, xem, click các link để di chuyển giữa các trang. Về sau, với sự ra đời của các ngôn ngữ server: CGI, Perl, PHP, … các website đã trở nên "động" hơn, 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 web app (ứng dụng web) ra đời. Với sự tiến bộ trong công nghệ web và chuyển từ ứng dụng máy tính để bàn truyền thống sang các giải pháp dựa trên web, sự phổ biến của các ứng dụng dựa trên web đã tăng lên rất nhiều. Ngày nay, các ứng dụng dựa trên web được sử dụng trong các môi trường quan trọng về bảo mật, chẳng hạn như các hệ thống y tế, tài chính và quân sự. Ứng dụng web được dùng để hiện thực bán hàng trực tuyến, đấu giá trực tuyến, wiki, diễn đàn thảo luận, hệ quản trị nội dung, phần mềm quản lý nguồn nhân lực và nhiều chức năng khác…

Ứng dụng web là chương trình máy tính sử dụng trình duyệt web và công nghệ web để thực hiện các tác vụ qua Internet. Thông qua ứng dụng web, người dùng có thể thực hiện các công việc như: tính toán, chia sẻ hình ảnh, video, mua sắm …

2.1.2 Kiểm thử bảo mật ứng dụng web

Kiểm thử bảo mật là một quá trình để xác định xem hệ thống có bảo vệ dữ liệu và duy trì chức năng như dự định hay không, kiểm tra xem dữ liệu bí mật có được giữ bí mật hay không và người dùng chỉ có thể thực hiện những nhiệm vụ mà họ được phép thực hiện (Ví dụ: người dùng sẽ không thể thay đổi chức năng của ứng dụng web theo cách không có chủ ý, ...).

Kiểm thử bảo mật nhằm tìm ra tất cả các lỗ hổng và điểm yếu có thể có của hệ thống trong giai đoạn khởi đầu để tránh hiệu suất hệ thống không nhất quán, sự cố bất ngờ, mất thông tin, mất doanh thu, mất niềm tin của khách hàng. Trong thực tế, kiểm thử bảo mật cũng tương tự như kiểm thử chức năng. Việc xác định rủi ro, xác định hành vi dự kiến sẽ là gì và sau đó thực hiện một số thử nghiệm để giảm thiểu rủi ro đó bằng cách chứng minh rằng điều không mong muốn không xảy ra. Ví dụ: giả sử hệ thống đang thử nghiệm là một ứng dụng web trực tuyến, được hỗ trợ bởi cơ sở dữ liệu. Một rủi ro có thể là kẻ tấn công ở đâu đó trên Internet có thể sử dụng giao diện người dùng và có quyền truy cập vào dữ liệu nhạy cảm được lưu trữ ở trong hệ thống.

Khi kiểm tra một chức năng tức là đang cố chứng minh rằng một tính năng hoạt động cho người dùng cuối - thực hiện những gì như mong đợi và không cản trở việc hoàn thành nhiệm vụ. Người kiểm thử có thể sẽ ưu tiên tập trung vào các tính năng được sử dụng thường xuyên hơn, được sử dụng bởi nhiều người dùng hơn, được coi là quan trọng nhất,... Là người kiểm thử bảo mật, người dùng cuối bây giờ là kẻ tấn công đang cố gắng phá vỡ ứng dụng. Mục tiêu thử nghiệm là chứng minh rằng một kịch bản tấn công cụ thể không thành công với bất kỳ kịch bản tấn công nào.

Các tổ chức phải tiến hành kiểm thử bảo mật thường xuyên vì những lý do:

- Để ngăn chặn vi phạm dữ liệu
- Để kiểm tra kiểm soát an ninh
- Để đáp ứng yêu cầu tuân thủ
- Để đảm bảo an toàn cho các ứng dụng mới
- Để tìm lỗ hổng bảo mật trong một hệ thống
- Để bảo mật dữ liệu người dùng
- Để xác định các lỗi mới trong một hệ thống hiện có sau khi triển khai hoặc sau những thay đổi lớn được thực hiện trong hệ thống
- Để ngăn chặn các cuộc tấn công mũ đen và bảo vệ dữ liệu người dùng
- Để kiểm soát tổn thất doanh thu
- Để cải thiện các tiêu chuẩn bảo mật hiện có

OWASP nhấn mạnh rằng "Bảo mật là một quá trình chứ không phải là một sản phẩm".

Kiểm thử bảo mật ứng dụng web là một quy trình tổng quan bao gồm vô số các quy trình cho phép kiểm tra bảo mật của ứng dụng Web, là quá trình kiểm tra, phân tích và báo cáo về mức độ bảo mật của ứng dụng Web. Đó là một quy trình có hệ thống bắt đầu từ việc xác định và phân tích toàn bộ ứng dụng, sau đó là lập kế hoạch cho các thử nghiệm. Mục tiêu chính của kiểm thử bảo mật ứng dụng Web là xác định bất kỳ lỗ hổng hoặc mối đe dọa nào có thể gây nguy hiểm cho tính bảo mật hoặc tính toàn vẹn của ứng dụng Web.

Thông thường, kiểm tra bảo mật ứng dụng Web được thực hiện sau khi ứng dụng Web được phát triển. Ứng dụng Web trải qua quá trình kiểm tra nghiêm ngặt bao gồm một loạt các cuộc tấn công độc hại giả lập để xem ứng dụng Web thực hiện/ phản hồi như thế nào, có tốt không. Sau quá trình kiểm tra bảo mật tổng thể sẽ có một báo cáo bao gồm các lỗ hổng được xác định, các mối đe dọa và khuyến nghị có thể có để khắc phục các thiếu sót về bảo mật.

Theo ISECOM (Open Source Security Testing) có 7 hình thức Kiểm thử bảo mật:

- Rà soát các lỗ hổng tiềm ẩn – Vulnerable Scanning: thực hiện thông qua các phần mềm để tự động scan một hệ thống nhằm phát hiện ra các lỗ hổng dựa trên các signatures đã biết.
- Rà soát các điểm yếu của hệ thống – Security Scanning: bao gồm việc xác định các điểm yếu của mạng và hệ thống, sau đó cung cấp các giải pháp nhằm giảm thiểu các rủi ro này. Có thể thực hiện bằng thủ công hoặc tự động.
- Đánh giá bảo mật bằng cách tấn công vào hệ thống – Penetration testing: Đây là loại kiểm thử mô phỏng cuộc tấn công từ phía một hacker thiếu thiện ý. Kiểm thử bao gồm việc phân tích một hệ thống cụ thể, tìm ra các lỗ hổng tiềm ẩn bằng cách tấn công từ bên ngoài.
- Đánh giá rủi ro – Risk Assessment: Kiểm thử này liên quan đến phân tích các rủi ro bảo mật nhận thấy được. Các rủi ro được phân loại là Low, Medium, High. Loại kiểm thử này đưa ra các khuyến nghị nhằm giảm thiểu các rủi ro.
- Kiểm toán an ninh – Security Auditing: Kiểm tra bảo mật nội bộ ứng dụng và OS.
- Tấn công vào hệ thống tìm các điểm yếu bảo mật – Ethical hacking: Các hacker thiện ý thực hiện phương pháp tương tự như những kẻ tấn công "thiếu thiện ý", với mục tiêu tìm kiếm các điểm yếu bảo mật và xác định cách thức để thâm nhập vào mục tiêu, nhằm đánh giá mức độ thiệt hại do các lổ hỗng này gây ra, từ đó đưa ra cảnh báo cùng những phương án gia cố, kiện toàn bảo mật thích hợp.
- Đánh giá tư thế- Posture assessment: Kết hợp Rà soát các điểm yếu của hệ thống, Tấn công vào hệ thống tìm các điểm yếu bảo mật và Đánh giá rủi ro để đánh giá bảo mật tổng thể một tổ chức.

2.2 Phân loại kiểm thử bảo mật

2.2.1 Kiểm thử yêu cầu và thiết kế

Bất kỳ hệ thống nào cũng được xây dựng từ một tập hợp các yêu cầu. Đôi khi những yêu cầu này được viết một cách rõ ràng, nhưng thường chúng là những phát biểu mập mờ không được định nghĩa rõ ràng. Ví dụ, có thể có phát biểu "Ứng dụng phải an toàn". Nhưng "an toàn" nghĩa là gì và nên phải dành bao nhiêu công sức và thời gian để làm cho sản phẩm an toàn. Kiểm thử bảo mật ở giai đoạn yêu cầu thiết kế cho ứng dụng web chính là việc xem xét các yêu cầu thiết có đã mô tả rõ các tiêu chí cụ thể về yêu cầu bảo mật cho ứng dụng web hay chưa.

2.2.2 Kiểm thử mã nguồn

Phương pháp kiểm tra độ bảo mật của ứng dụng thông qua mã nguồn của ứng dụng. Phương pháp kiểm thử này chủ yếu dùng để xác định sự an toàn của thuật tốn được dùng trong ứng dụng, xác độ nguy cơ rò rỉ thông tin, nguy cơ bị tấn công chiếm quyền kiểm sốt thông qua mã nguồn.

Kiểm thử mã nguồn là một kỹ thuật kiểm thử bảo mật hộp trắng bao gồm xem xét mã nguồn ứng dụng cho các lỗ hổng bảo mật có thể có.

2.2.3 Kiểm thử các thiết lập của trình duyệt

Các thiết lập của trình duyệt có thể được cài đặt trong các trình duyệt như Mozilla FireFox và Microsoft Internet Explorer cho phép giới hạn truy cập đến các nội dung Internet có thể gây hại. Người sử dụng sẽ thường có các chỉnh sửa các thiết lập này. Hơn nữa, có một sự thay đổi lớn phía người sử dụng về khả năng làm chủ các thiết lập này. Những người sử dụng Web ngày càng được đào tạo nhiều hơn cách sử dụng các thiết lập để bảo vệ chính họ. Với tư cách là một đội phát triển Website hay ứng dụng Web, chúng ta không thể bắt buộc người sử dụng chấp nhận các thiết lập mặc định. Vì vậy, chúng ta cần phải kiểm thử nhiều sự kết hợp của các thiết lập.

2.2.4 Kiểm thử tường lửa

Kiểm thử tường lửa nhằm nhận biết các hiệu ứng về chức năng được tạo ra bởi sự chuyển dữ liệu qua các mạng khác nhau. Cần nhắc lại rằng nhóm kiểm thử phần mềm không chịu trách nhiệm kiểm thử sự hiệu quả của các tường lửa và sự cấu hình chúng.

2.3 Quy trình kiểm thử bảo mật [3]

Một trong những quy tắc bảo mật quan trọng nhất là biết rõ về môi trường của bạn. Giai đoạn này sẽ xác định phạm vi (sẽ kiểm tra hệ thống nào; kế hoạch và mục tiêu cần đạt được với kiểm tra thâm nhập) và các tài nguyên và công cụ (máy quét lỗ hổng hoặc công cụ kiểm tra thâm nhập) để sử dụng để thực hiện kiểm tra.

Xây dựng một kế hoạch và chiến lược phải luôn là bước đầu tiên của kiểm thử bảo mật. Tester phải hiểu mục đích kinh doanh, số người dùng truy cập vào ứng dụng và luồng hoạt động của ứng dụng... để xác định cách kiểm tra cụ thể cho từng kịch bản.

Trước khi thực hiện bất kỳ dự án nào, tốt nhất nên tổ chức một cuộc họp với đội phát triển phần mềm để hiểu được luồng hoạt động của ứng dụng. Điều này giúp người kiểm thử xác định các lỗ hổng logic, chẳng hạn như bỏ qua giấy phép, mà các công cụ tự động không thể xác định được. Doanh nghiệp nên có một con số ballpark - con số gần đúng số người dùng sẽ truy cập vào ứng dụng. Nắm được số lượng người dùng tối đa giúp người kiểm thử tạo ra số lượng người dùng ảo để xác định các cuộc tấn công từ chối dịch vụ có thể xảy ra. Bước này rất quan trọng, khi đã xác định được nhiều thông tin liên quan tới ứng dụng sẽ tạo điều kiện thuận lợi cho việc tấn công và khai thác.

2.3.1 Giai đoạn khám phá

Đây là giai đoạn thu thập các thông tin về các hệ thống nằm trong phạm vi thử nghiệm bảo mật. Thu thập càng nhiều thông tin về website mục tiêu càng tốt về hệ thống, thông tin về tài khoản gồm tên và mật khẩu (nếu tìm được).

Mô hình các mối đe dọa cấp cao đối với ứng dụng cho phép kiểm tra đánh giá các rủi ro có thể và các kịch bản liên quan đến nó. Mô hình mối đe dọa xác định các khu vực yếu của ứng dụng, giúp sửa các phương án kiểm thử.

2.3.2 Đánh giá lỗ hổng

Sau khi bản thiết kế hoàn thiện, phần kỹ thuật bắt đầu, nơi các thành phần được xác định để phát triển. Nó có thể là ngôn ngữ mã hóa, nền tảng, công nghệ stack,… Mỗi thành phần đi kèm với tập hợp các điểm yếu và điểm mạnh của nó, do đó điều quan trọng là xác định các lỗ hổng trước giai đoạn code. Điều này giúp xác định các lựa chọn an toàn hơn và giảm đáng kể chi phí để sửa chúng.

Có một số khu vực chung cho lỗ hổng an ninh:

- Hidden field: Lỗ hổng này chủ yếu được khai thác cho các trang web thương mại điện tử. Ứng dụng ẩn các trường ẩn trong các trang web và do tiêu chuẩn mã hoá kém, các trường ẩn này thường chứa thông tin bí mật, chẳng hạn như giá sản phẩm.
- Cross-site scripting: Đây là một trong những lỗ hổng phổ biến nhất. Nó cho phép tin tặc ăn cắp session, xóa trang, nhúng nội dung hoặc chuyển hướng người dùng đến các trang web độc hại.
- Thẩm định yêu cầu cross-site: Nhiều lập trình viên bỏ qua tầm quan trọng của các token và xác nhận lại ngẫu nhiên trên trang dữ liệu quan trọng. Nếu không có nó, kẻ tấn công có thể thực hiện hành động bởi người dùng mạo danh, chẳng hạn như thêm hoặc xóa tài khoản hoặc sửa đổi thông tin người dùng.

Kiểm thử bảo mật có thể được chia thành các hình thức như kiểm thử bảo mật thủ công, kiểm thử bảo mật tự động và có thể kết hợp cả kiểm thử bảo mật thủ công & tự động. Cả kiểm thử bảo mật thủ công và kiểm thử bảo mật tự động đều được thực hiện cho cùng một mục đích. Sự khác biệt duy nhất giữa chúng là cách chúng được tiến hành. Như tên gọi cho thấy, kiểm thử bảo mật thủ công được thực hiện bởi con

người (chuyên gia của lĩnh vực này) kiểm thử bảo mật tự động được thực hiện bởi máy móc.

Bằng cách sử dụng các công cụ kiểm thử bảo mật tự động, không thể tìm thấy tất cả các lỗ hổng. Một số lỗ hổng có thể được xác định bằng cách sử dụng quét thủ công. Vì vậy, những người kiểm thử bảo mật có kinh nghiệm sử dụng kinh nghiệm và kỹ năng của họ để tấn công một hệ thống bằng cách sử dụng các phương pháp kiểm thử bảo mật thủ công.

a. Kiểm thử thủ công

Các kỹ sư kiểm thử thủ công thực hiện các phương pháp sau:

- Thu thập dữ liệu: Thu thập dữ liệu đóng vai trò chính để kiểm thử. Người ta có thể thu thập dữ liệu theo cách thủ công hoặc có thể sử dụng các dịch vụ công cụ (như kỹ thuật phân tích mã nguồn trang web,...) có sẵn miễn phí trực tuyến. Các công cụ này giúp thu thập thông tin như tên bảng, phiên bản hệ thống cơ sở dữ liệu, cơ sở dữ liệu, phần mềm, phần cứng hoặc thậm chí về các plugin của bên thứ ba khác,...
- Đánh giá tính dễ bị tổn thương: Dữ liệu được thu thập được người kiểm thử sử dụng để xác định các điểm yếu về bảo mật và thực hiện các bước phòng ngừa tương ứng.
- Triển khai thực tế: Đây là phương pháp điển hình mà để người kiểm thử sử dụng để khởi động một cuộc tấn công vào hệ thống mục tiêu, làm giảm nguy cơ bị tấn công.
- Chuẩn bị báo cáo: Sau khi thâm nhập xong, người kiểm thử chuẩn bị báo cáo mô tả mọi thứ về hệ thống. Sau đó, báo cáo này sẽ được phân tích để thực hiện các bước khắc phục để bảo vệ hệ thống đang hướng tới.

Kiểm thử thâm nhập thủ công thường được phân loại thành hai cách sau:

- Kiểm thử bảo mật tập trung: Đây là phương pháp tập trung nhiều để kiểm tra các lỗ hổng và rủi ro cụ thể. Kiểm tra bảo mật tự động không thể thực

hiện kiểm tra này; nó chỉ được thực hiện bởi các chuyên gia kiểm thử thực hiện kiểm tra các lỗ hổng ứng dụng cụ thể trong các miền nhất định.

- Kiểm thử bảo mật toàn diện: Kiểm thử toàn bộ các hệ thống được kết nối với nhau để xác định tất cả các loại rủi ro và lỗ hổng. Tuy nhiên, thử nghiệm này mang tính tình huống hơn, chẳng hạn như kiểm tra xem nhiều lỗi có nguy cơ thấp hơn có thể mang lại kịch bản tấn công dễ bị tổn thương hơn không, v.v.

Quá trình tìm kiếm lỗi bảo mật trong mã nguồn của ứng dụng bằng phương pháp thủ công thì phải đòi hỏi người kiểm thử phải có một phương pháp kiểm thử và ra soát hợp lý. Bởi vì khối lượng tập tin cũng như nội dung trong các ứng dụng web là rất lớn, nếu như không có một phương pháp rà soát và đánh giá hợp lý thì sẽ tiêu tốn rất nhiều thời gian để phát hiện lỗi.

a. Kiểm thử tự động

Phương pháp kiểm thử tự động là quá trình các công cụ sẽ thực hiện tự động quét thư mục, tập tin của ứng dụng web và tự động xác định các điểm mà cần kiểm tra dữ liệu. Trên cơ sở đã xác định các điểm kiểm tra công cụ sẽ thực hiện đệ trình các tập dữ liệu được định nghĩa sẵn và chờ sự phản hồi từ phía ứng dụng web để kiểm tra xem liệu ứng dụng đó có bị các lỗi bảo mật hay không.

Kiểm thử bảo mật tự động nhanh hơn, hiệu quả, dễ dàng và đáng tin cậy để kiểm tra lỗ hổng và rủi ro của website một cách tự động. Công nghệ này không yêu cầu bất kỳ kỹ sư chuyên gia nào, nó có thể được vận hành bởi người có ít kiến thức nhất về lĩnh vực này. Các công cụ quét bảo mật tự động rất tốt trong việc tìm kiếm các lỗ hổng phổ biến một cách nhanh chóng và có hệ thống.

Bảng 2.1: So sánh giữa kiểm thử bảo mật thủ công và tự động

Kiểm thử bảo mật thủ công	*Kiểm thử bảo mật tự động*
- Các trường hợp kiểm thử được thực hiện bởi kiểm thử viên	- Kiểm thử tự động sử dụng các công cụ tự động để thực hiện các trường hợp kiểm thử.
- Đòi hỏi chuyên gia giàu kinh nghiệm	- Có thể được vận hành bởi người có ít kinh nghiệm
- Cần các công cụ khác nhau để thử nghiệm	- Chỉ cần một công cụ được tích hợp với nhiều nguồn
- Kiểm thử thủ công có thể bị nhàm chán và dễ bị lỗi.	- Kiểm thử tự động là một phương pháp đáng tin cậy, vì được thực hiện bởi các công cụ và scripts nên chính xác và không gây nhàm chán
- Kiểm thử ngẫu nhiên có thể thực hiện trong Kiểm thử thủ công	- Tự động hóa không cho phép kiểm thử ngẫu nhiên
- Đầu tư ban đầu trong kiểm thử thủ công là tương đối thấp hơn kiểm thử tự động.	- Đầu tư ban đầu trong kiểm thử tự động cao hơn.

2.3.3 Giai đoạn khai thác

Trong giai đoạn này, cố gắng khai thác các lỗ hổng được xác định trong giai đoạn trước (tức là giai đoạn khám phá) để có quyền truy cập vào hệ thống đích.

Với việc biết được chính xác port nào đang mở, dịch vụ nào đang chạy, điểm yếu hay lỗ hổng nào mà các dịch vụ này đang dính, chúng ta có thể tiến hành khai thác, đây chính là bước gần giống với một cuộc tấn công thực sự nhất. Tuy nhiên khác với các tin tặc, mục đích của người kiểm thử xâm nhập chỉ là xác thực lỗ hổng

có tồn tại hay không chứ không nhằm mục đích gây ảnh hưởng đến doanh nghiệp như lấy các thông tin nhạy cảm hay thậm chí là chiếm quyền điều khiển hệ thống.

2.3.4 Giai đoạn báo cáo

Đây là giai đoạn cuối cùng của quá trình kiểm thử. Giai đoạn này bao gồm việc tạo báo cáo đánh giá chi tiết về những lỗ hổng, nguy cơ mất an toàn thông tin và trình bày với khách hàng để thẩm định. Sau đó, cả hai bên sẽ cùng tiến hành thảo luận các giải pháp khắc phục các lỗ hổng đã phát hiện.

Mẫu báo cáo sẽ thường bao gồm các mục cơ bản sau:

- Tên lỗ hổng
- Mô tả nguy cơ của lỗ hổng: mức độ nguy hiểm, hậu quả có thể xảy ra nếu lỗ hổng bị khai thác, các thông tin liên quan khác
- Minh chứng về lỗ hổng
- Khuyến cáo khắc phục (có thể có hoặc không)

2.4 Các kỹ thuật kiểm thử bảo mật [5], [6], [8], [9]

Kiểm thử bảo mật có thể được chia thành hai loại: kiểm tra hộp trắng, hộp đen. Các phương pháp được sử dụng thường phụ thuộc vào tình hình và nhu cầu thử nghiệm.

Kỹ thuật kiểm thử hộp trắng cho phép kiểm thử đến mức chi tiết từng mã lệnh, từ đó có thể tìm kiếm lỗi ngay trong mã lệnh dẫn đến việc sửa lỗi có thể dễ dàng và không mất nhiều thời gian, chi phí sửa lỗi. Nhưng nhược điểm là đòi hỏi người kiểm thử phải có kiến thức về lập trình, và mất thời gian để kiểm thử nếu chương trình có lượng mã lệnh lớn. Và không phải tất cả các lỗi, lỗ hổng bảo mật của website có thể tìm được qua phương pháp này.

Kỹ thuật kiểm thử hộp đen thích hợp kiểm tra phân đoạn lớn các mã lệnh, các chức năng lớn. Người kiểm thử không cần hiểu biết về mã lệnh viết trong chương

trình mà dựa và hiểu biết yêu cầu của chương trình, kinh nghiệm và kỹ năng để đưa ra các bộ đầu vào và thực hiện kiểm thử.

2.4.1 Kiểm thử hộp đen

2.4.1.1. *Khái niệm [6], [12]*

Trong kỹ thuật Kiểm thử bảo mật hộp đen, người kiểm thử đánh giá hệ thống mục tiêu mà không có bất kỳ kiến thức nào về chi tiết hệ thống. Họ sẽ không xem xét bất kỳ đoạn code nào và không thu thập thông tin về hệ thống mục tiêu từ chủ sở hữu của hệ thống. Các kỹ thuật kiểm thử bảo mật hộp đen liên quan đến việc cố gắng phân tích và tìm lỗ hổng trong việc chạy phần mềm bằng cách thao tác đầu vào mà không có bất kỳ kiến thức nào về mã nguồn. Vì vậy, người kiểm thử sẽ phát một cuộc tấn công toàn diện chống lại hệ thống để tìm ra điểm yếu hoặc lỗ hổng trong hệ thống.

Hình 2.1: Kiểm thử hộp đen

2.4.1.2. *Kiểm tra bảo mật ứng dụng động DAST [11], [12]*

Kiểm tra bảo mật ứng dụng động (Dynamic Application Security Testing - DAST): Cách tiếp cận DAST liên quan đến việc tìm kiếm các lỗ hổng trong ứng dụng web mà kẻ tấn công có thể cố gắng khai thác.

Các công cụ DAST được coi như các công cụ kiểm thử mũ đen hoặc hộp đen, nơi mà những người kiểm thử không có hiểu biết về hệ thống. Họ dò tìm những lỗ hổng an ninh của một ứng dụng trong trạng thái hoạt động của nó. Những công cụ DAST chạy trên code đang hoạt động để dò tìm các vấn đề với các giao tiếp, những

yêu cầu, những phản hồi, những script (ví dụ javascript), lỗ hổng dữ liệu, sự xác thực, …DAST phát hiện các lỗ hổng bằng cách thực hiện các cuộc tấn công thực sự.

Phương pháp thử nghiệm này hoạt động để tìm ra lỗ hổng nào mà kẻ tấn công có thể nhắm mục tiêu và làm thế nào chúng có thể xâm nhập vào hệ thống từ bên ngoài. DAST thử nghiệm ứng dụng bên ngoài trong khi chạy ở chế độ thử nghiệm hoặc trong môi trường sản xuất. Nó giúp theo dõi nhanh chóng, linh hoạt, và khả năng mở rộng của ứng dụng để tích hợp liền mạch với chiến lược bảo mật của công ty. Các công cụ kiểm tra bảo mật ứng dụng động không yêu cầu quyền truy cập vào mã nguồn gốc của ứng dụng, do đó, kiểm tra với DAST có thể được thực hiện nhanh chóng và thường xuyên.

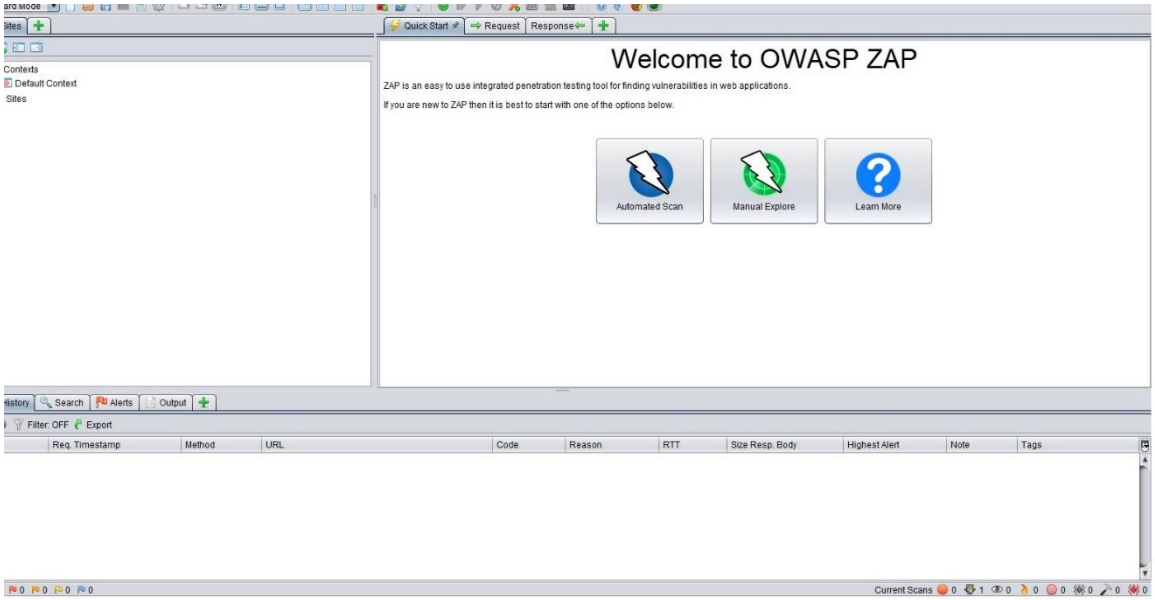
Các công cụ DAST cho phép quét tinh vi, phát hiện các lỗ hổng với các tương tác tối thiểu của người dùng một khi được định cấu hình với tên máy chủ, thu thập thông số và thông tin xác thực. Các công cụ này sẽ cố gắng phát hiện các lỗ hổng trong chuỗi truy vấn, tiêu đề, đoạn, hành động (GET/POST/PUT) và tiêm DOM.

- ***Điểm mạnh***
  - Các công cụ quét sử dụng kỹ thuật DAST không quan tâm đến mã nguồn của ứng dụng do đó nó có thể hoạt động được với bất kỳ ngôn ngữ lập trình và framework nào.
- ***Điểm yếu***
  - Trong khi quét bằng công cụ DAST, dữ liệu có thể bị ghi đè hoặc tải trọng độc hại được đưa vào trang web thực hiện kiểm thử. Các trang web nên được quét trong một môi trường giống như thực tế nhưng không phải là thực tế (Ví dụ môi trường Pre-production) để đảm bảo kết quả chính xác trong khi bảo vệ dữ liệu ở môi trường thực tế.
- Một số công cụ DAST mã nguồn mở:

a. Zed Attack Proxy [7], [12]

Zed Attack Proxy (ZAP) là một công cụ nguồn mở được cung cấp bởi OWASP để thực hiện kiểm tra bảo mật. Nó giúp tìm ra các lỗ hổng bảo mật trong các ứng

dụng. Đây là một trong những công cụ bảo mật miễn phí phổ biến nhất thế giới và được các tình nguyện viên duy trì tích cực. Đây là một công cụ kiểm tra thâm nhập tích hợp dễ dàng để tìm một số lỗ hổng bảo mật trong một ứng dụng web trong khi đang phát triển và thử nghiệm một ứng dụng.



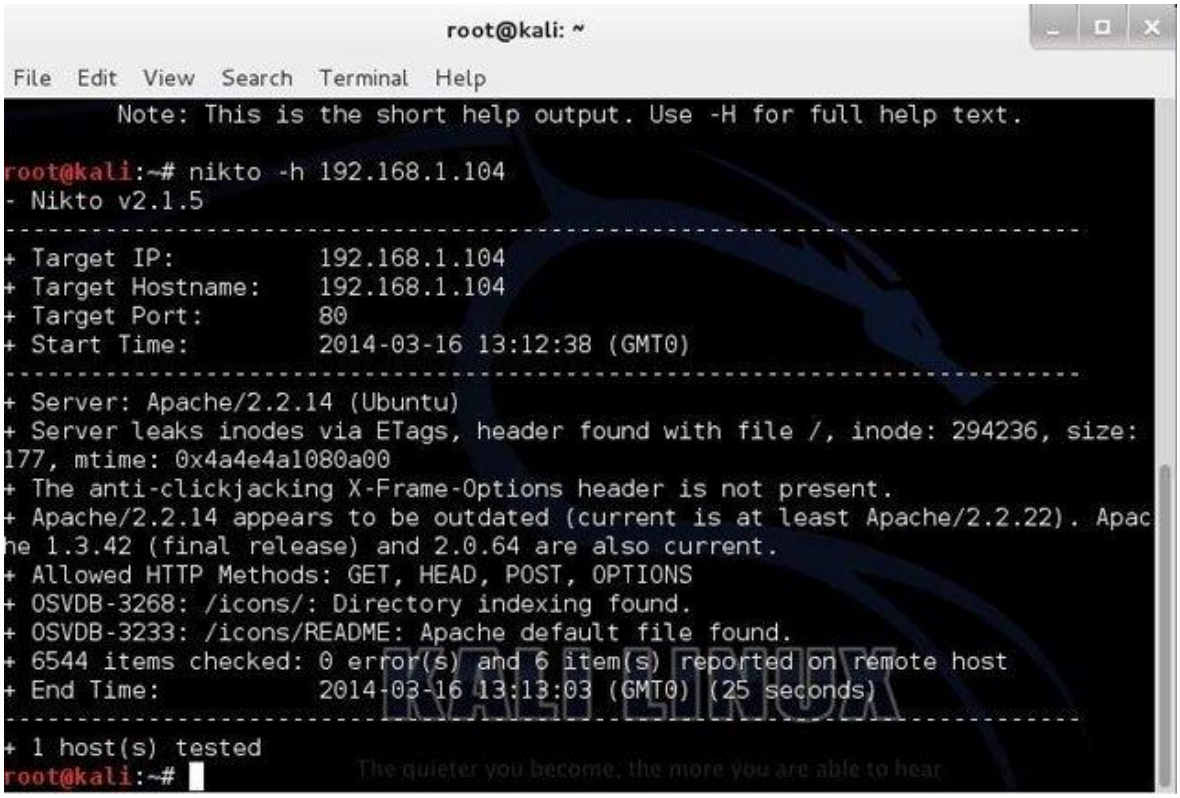
Hình 2.2: Màn hình giao diện công cụ ZAP

ZAP sẽ tiến hành thu thập dữ liệu ứng dụng web bằng trình thu thập dữ liệu của nó và quét thụ động từng trang mà nó tìm thấy. Sau đó, ZAP sẽ sử dụng trình quét để tấn công tất cả các trang, chức năng và tham số được phát hiện.

b. Nikto [12], [16]

Nitko là một trình quét máy chủ web nguồn mở thực hiện quét các máy chủ web để tìm các tệp/ chương trình nguy hiểm tiềm tàng, các phiên bản lỗi thời và các sự cố cụ thể của phiên bản khác. Nó cũng quét các cấu hình máy chủ như tùy chọn máy chủ HTTP và sẽ cố gắng xác định các máy chủ và phần mềm web đã cài đặt, giúp kiểm tra nhanh các vấn đề mà web server đối tượng đang gặp phải như: các cấu hình phía dịch vụ máy chủ hoặc phần mềm sai sót, chương trình hay file mặc định được tìm thấy, các chương trình hay file không an toàn được tìm thấy, những lỗ hổng cơ bản ở ứng dụng web.

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. Một điều bất tiện là Nikto không có giao diện đồ họa nên đòi hỏi người sử dụng phải có kiến thức lập trình và sử dụng các lệnh command.

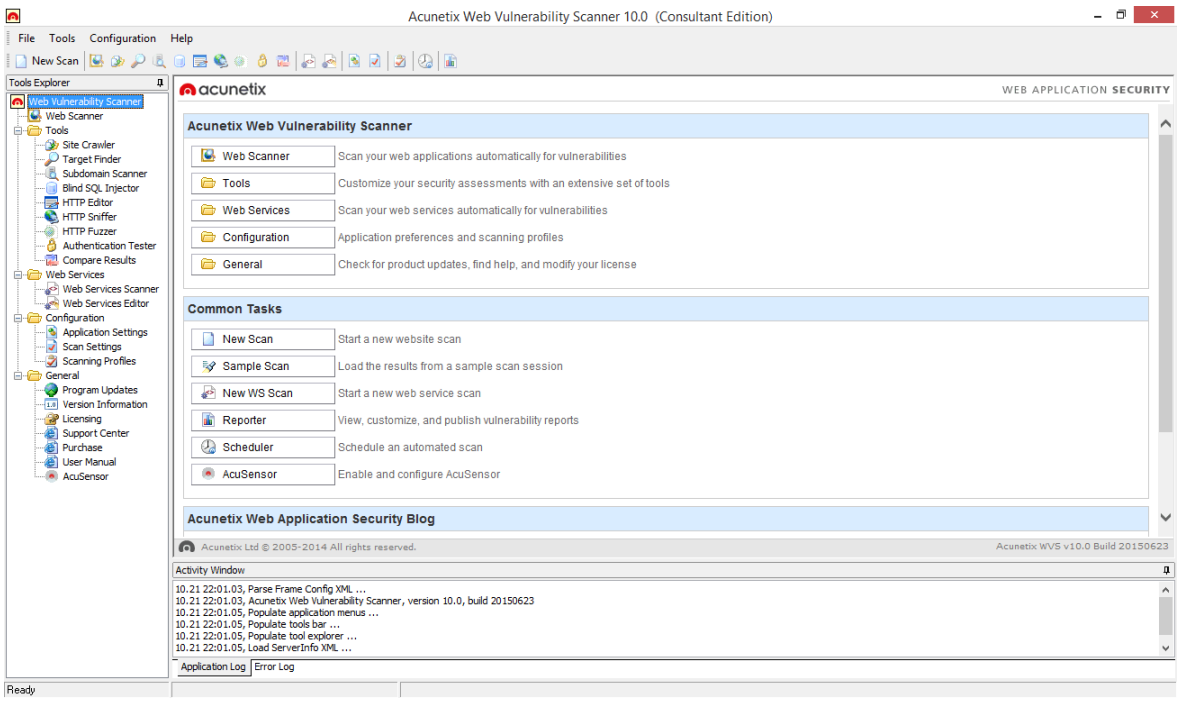


Hình 2.3: Giao diện Nikto

c. Acunetix Web Vulnerability Scanner (WVS) [12], [13]

Acunetix WVS (Web Vulnerability Scanner) là chương trình tự động kiểm tra các ứng dụng Web để tìm kiếm các lỗ hổng bảo mật như SQL Injection, hay Cross-Site Scripting,… và tìm kiếm những chính sách đối với mật khẩu đăng nhập cũng như các phương thức xác thực vào Web Site. Acunetix WVS là một công cụ quét lỗi cho ứng dụng Web dựa trên một cơ sở dữ liệu rộng lớn được cập nhật thường xuyên, với các thuật toán Heuristic đáp ứng được các cơ chế hoạt động phức tạp của môi trường Web. Acunetix WVS có thể tự động kiểm tra các lổ hỗng thông dụng và các

mối nhạy cảm khác của những website có thể truy cập bằng trình duyệt, hay những ứng dụng được xây dựng trên các kỹ thuật tiên tiến như AJAX..

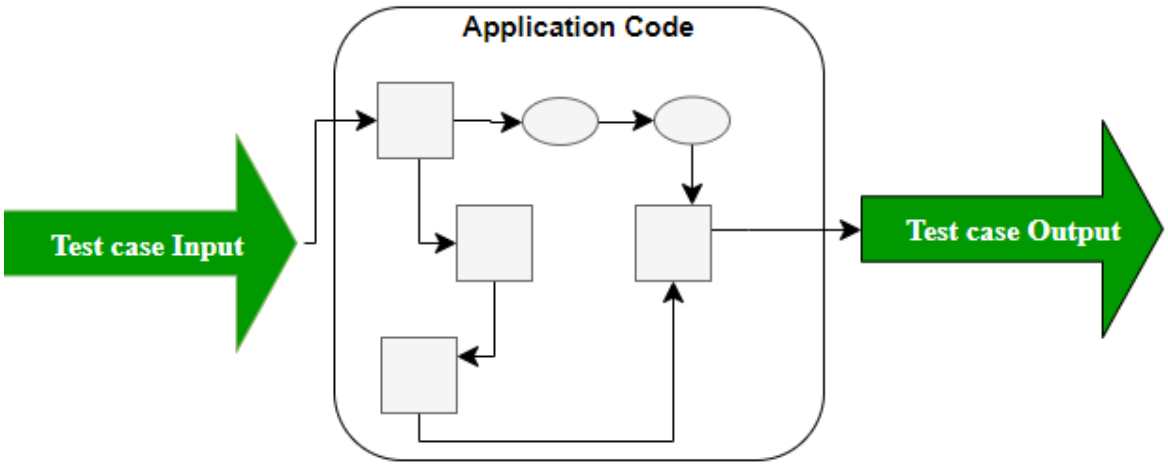


Hình 2.4: Màn hình giao diện công cụ Acunetix WVS

2.4.2 Kiểm thử hộp trắng

2.4.2.1. Khái niệm [5], [12]

Trong kỹ thuật Kiểm thử bảo mật hộp trắng, người kiểm thử truy cập hệ thống đích với các chi tiết đầy đủ về hệ thống. Vì đã có chi tiết đầy đủ về hệ thống nên kiểm thử hộp trắng có thể được thực hiện nhanh hơn nhiều so với kiểm nghiệm hộp đen. Các kỹ thuật kiểm tra bảo mật hộp trắng nhằm phân tích mã nguồn và kiến trúc ứng dụng từ quan điểm đảm bảo an ninh.



Hình 2.5: Kiểm thử hộp trắng

2.4.2.2. Kiểm tra bảo mật ứng dụng tĩnh SAST [11]

Kiểm tra bảo mật ứng dụng tĩnh (Static Application Security Testing- SAST): được thiết kế để phân tích mã nguồn và/ hoặc các phiên bản mã được biên dịch để giúp tìm ra các lỗi bảo mật.

Đối lập với DAST, những công cụ sử dụng kỹ thuật SAST có thể được xem như quá trình kiểm thử mũ trắng hay kiểm thử hộp trắng, nơi mà người kiểm thử biết thông tin về hệ thống và phần mềm được kiểm thử, bao gồm cả kiến trúc hệ thống, mã nguồn, … Những công cụ SAST kiểm tra mã nguồn để dò tìm và báo cáo những điểm yếu có thể dẫn đến những lỗ hổng an ninh.

SAST có cách tiếp cận từ trong ra ngoài nhiều hơn, nghĩa là không giống như DAST, nó tìm kiếm các lỗ hổng trong mã nguồn của ứng dụng web. Kể từ khi nó đòi hỏi quyền truy cập vào mã nguồn của ứng dụng, SAST có thể cung cấp một bản chụp trong thời gian thực của an ninh ứng dụng web. SAST bao gồm kiểm tra nội bộ của ứng dụng, nơi người kiểm thử hoặc công cụ kiểm tra ứng dụng với quyền truy cập không giới hạn đến mã nguồn hoặc mã nhị phân. Nó có thể được thực hiện cả bằng tay và tự động và kiểm tra các ứng dụng cho các lỗ hổng phức tạp có thể không bị phát hiện.

- ***Điểm mạnh:***

- Có thể chạy trên nhiều phần mềm và có thể chạy liên tục (như với các bản dựng hàng đêm hoặc tích hợp liên tục)
- Có thể tự động tìm thấy nhiều lỗi với độ tin cậy cao, chẳng hạn như tràn bộ đệm, Lỗi SQL injection, v.v.
- Đưa ra kết quả báo cáo rõ ràng, dễ hiểu cho đội ngũ lập trình để dễ dàng tìm thấy lỗ hổng bảo mật

- ***Điểm yếu***

Nhiều loại lỗ hổng bảo mật rất khó tìm thấy tự động, chẳng hạn như các vấn đề xác thực, các vấn đề kiểm soát truy cập, sử dụng mật mã không an toàn, v.v.

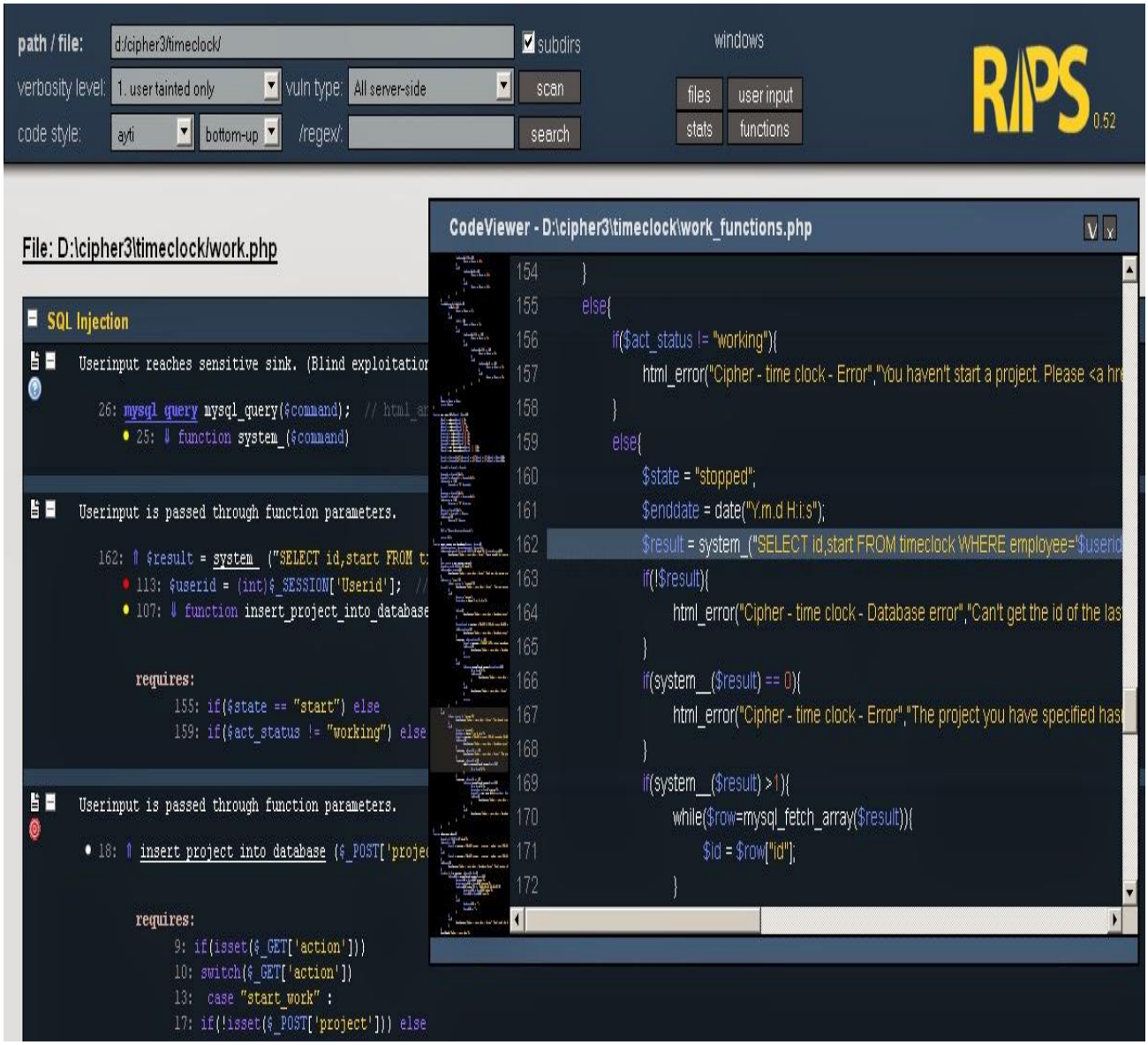
- Số lượng lỗi không chính xác cao
- Thường không thể tìm thấy các vấn đề cấu hình, vì chúng không được thể hiện trong mã nguồn.
- Khó 'chứng minh' rằng một vấn đề bảo mật được xác định là một lỗ hổng thực sự.
- Nhiều trong số các công cụ này gặp khó khăn khi phân tích mã không thể biên dịch được. Do không có thư viện phù hợp, không có đủ các trình biên dịch, không hiểu hết tất cả các mã…

- Một số công cụ SAST:

a. RIPS [14]

RIPS - RIPS Open Source là một bộ phân tích mã nguồn tĩnh cho các lỗ hổng trong các ứng dụng web PHP.

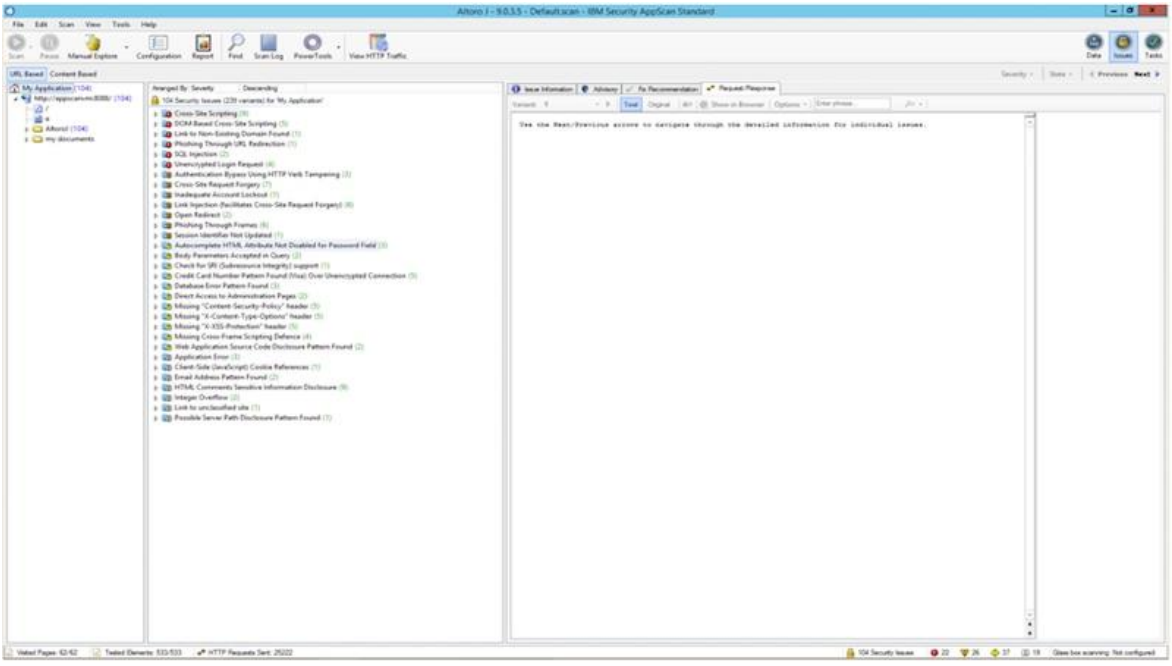
RIPS là một công cụ được viết bằng PHP để tìm kiếm lỗ hổng trong ứng dụng PHP sử dụng việc phân tích code tĩnh. Bằng việc chia nhỏ và phân tích tất cả các file code, RIPS có khả năng để chuyển PHP source code vào một mô hình chương trình để phát hiện các hàm tiềm ẩn khả năng bị hỏng mà có thể bị bị phá hỏng bởi đầu vào của người dùng trong quá trình thiết kế. Ngoài việc cho phép tìm kiếm lỗi tự động, RIPS còn cung cấp khản năng phân tích code bằng tay.



Hình 2.6: Màn hình giao diện công cụ RIPS

b. AppScan Source [15]

IBM AppScan Source được phát hành bởi IBM – tập đoàn về máy tính có tuổi đời lớn nhất thế giới. AppScan Source có 2 phiên bản: Standard (dành cho doanh nghiệp vừa và nhỏ) & Enterprise (dành cho các tập đoàn lớn).



Hình 2.7: Giao diện công cụ AppScan Source

- Các chức năng chính:
  - Cung cấp kiến thức về bảo mật ứng dụng web
  - Scan ứng dụng web & mobile app để tìm lỗ hổng
  - Đề xuất phương án khắc phục
  - Xuất báo cáo riêng theo đặc trưng từng ngành
- Ưu điểm:
  - Dùng thử miễn phí
  - Giao diện tối giản, trình bày khoa học
  - Cung cấp kiến thức cần thiết cho người dùng (miễn phí)
  - Phát hiện được nhiều loại lỗ hổng khác nhau như: Cross-site scripting, SQL Injection, Command Injection, Path Traversal, etc.
  - Xuất báo cáo đặc trưng theo từng ngành cụ thể
  - Hỗ trợ 24/7
- Nhược điểm:
  - Thủ tục tải và sử dụng phần mềm tương đối phức tạp và tốn thời gian, do luật liên bang về xuất khẩu phần mềm ngặt nghèo của Mỹ.

- Không hỗ trợ tiếng Việt. Dù dịch vụ khách hàng của IBM phục vụ 24/7 nhưng rào cản ngôn ngữ vẫn gây trở ngại phần nào cho người dùng Việt.
- Chỉ hỗ trợ HĐH Windows.
- Giá cao. Khởi điểm từ 11,000 USD/năm đối với bản Standard và 33,400 USD/năm đối với phiên bản Enterprise.

2.5 Đánh giá các kỹ thuật kiểm thử bảo mật

Để có thể kiểm thử bảo mật website hiệu quả, nhanh và chính xác thì việc lựa chọn công cụ hỗ trợ kiểm thử đóng một vai trò vô cùng quan trọng. Với các công cụ tương ứng với các phương pháp kiểm thử ở trên thì:

Các công cụ áp dụng kỹ thuật hộp trắng giúp cho người kiểm thử có thể phân tích được các lỗi tiềm ẩn, rủi ro của code từ đó giúp các lập trình viên dễ dàng tìm và sửa lỗi. Nhưng việc tìm lỗi bằng kỹ thuật kiểm thử hộp trắng là chưa đầy đủ. Để có một website hoàn chỉnh đến tay người dùng thì ngoài việc coding còn phải tích hợp với các hệ thống cơ sở dữ liệu, cơ sở hạ tầng, máy chủ cũng như các thiết lập an toàn khác cho website. Do đó cần thực hiện thêm các kỹ thuật kiểm thử hộp đen bằng tay hoặc thông qua công cụ. Với việc kiểm thử bảo mật thì sử dụng công cụ kiểm thử sẽ giúp cho quá trình kiểm thử nhanh, chính xác hơn từ đó tiết kiệm thời gian và công sức của con người. Nhưng nếu áp dụng toàn bộ kỹ thuật kiểm thử hộp đen tức là đẩy cả giá trị hợp lệ, không hợp lệ vào thực hiện kiểm thử bảo mật sẽ mất thời gian (vì thực tế các giá trị hợp lệ đã được kiểm tra theo chức năng). Do đó sẽ ưu tiên lựa chọn công cụ phát triển dựa theo kỹ thuật kiểm thử hộp đen nhưng các giá trị đầu vào là không hợp lệ (phương pháp Fuzzing) để tìm các lỗi liên quan đến bảo mật của website.

Cần chọn một thử nghiệm thâm nhập dựa trên các điểm sau.

- Dễ triển khai, cấu hình và sử dụng
- Quét lỗ hổng một cách dễ dàng
- Có thể phân loại các lỗ hổng dựa trên mức độ nghiêm trọng

- Có khả năng tự xác minh các lỗ hổng
- Tạo ra các báo cáo và nhật ký chi tiết về các lỗ hổng
- Có hiệu quả về mặt ngân sách
- Hỗ trợ tốt, có tài liệu kỹ thuật tham khảo

Trên cơ sở phân tích về một số kỹ thuật và một số thuật toán được sử dụng trong các công cụ phân tích, dò quét lỗ hổng cho website, bằng việc so sánh ưu, nhược điểm của các công cụ này, tác giả xin đề xuất chọn sử dụng công cụ Zed Attack Proxy (ZAP) để thử nghiệm dò quét và đánh giá mức độ an toàn bảo mật của các website trong thực tế. Ngoài ra, ZAP là một công cụ kiểm thử hộp đen sẽ tránh yêu cầu nhiều kiến thức chuyên sâu về lập trình cũng là một ưu điểm.

2.6 Kết chương

Kiểm thử bảo mật được phân chia theo nhiều lĩnh vực: thiết kế, mã nguồn, các thiết lập của trình duyệt, tường lửa. Quy trình kiểm thử bảo mật gồm bốn giai đoạn: Khám phá, Đánh giá lỗ hổng, Khai thác và Báo cáo. Có các kỹ thuật kiểm thử khác nhau, tuy nhiên kiểm thử hộp đen sẽ dò xét được lỗ hổng bảo mật đầy đủ.

Trong chương 3, luận văn sẽ trình bày sâu về công cụ kiểm thử hộp đen là Zed Attack Proxy đồng thời thực nghiệm kiểm thử trên các trang web cụ thể.

Chương 3. CÁC CÔNG CỤ KIỂM THỬ VÀ THỰC NGHIỆM KIỂM THỬ BẢO MẬT ỨNG DỤNG WEB

Trong chương này, luận văn sẽ tập trung vào việc trình bày công cụ kiểm thử Zed Attack Proxy (ZAP), thực hiện kiểm thử và đánh giá kết quả thử nghiệm, bộ dữ liệu thử nghiệm, kết quả thử nghiệm cũng như đưa ra kết luận đánh giá về công cụ thử nghiệm.

3.1 Giới thiệu công cụ kiểm thử Zed Attack Proxy [7]

3.1.1 Tổng quan

Zed Attack Proxy thường được gọi là ZAP là một công cụ kiểm tra bảo mật nguồn mở cho một ứng dụng web được phát triển bởi OWASP, cung cấp nhiều tùy chọn để thực hiện kiểm tra thâm nhập bảo mật tự động hoặc thủ công cho các ứng dụng web. ZAP có cấu hình cao và có thể mở rộng bằng cách sử dụng các plugin và nó được duy trì bởi hàng trăm tình nguyện viên quốc tế. ZAP được thiết kế để được sử dụng bởi những người có nhiều kinh nghiệm bảo mật và như vậy là lý tưởng cho các nhà phát triển và người thử nghiệm chức năng, những người mới thử nghiệm thâm nhập cũng như các chuyên gia bảo mật có kinh nghiệm. Nó đi kèm với một giao diện thân thiện giúp người mới cũng như các chuyên gia. Đồng thời ZAP cung cấp quyền truy cập dòng lệnh cho người dùng nâng cao.

Điểm nổi bật của công cụ ZAP:

- ZAP thu thập thông tin và phân tích các trang web bao gồm cả nội dung AJAX
- Có thể phân loại các lỗ hổng dựa trên mức độ nghiêm trọng
- Có khả năng tự xác minh các lỗ hổng
- Tạo ra các báo cáo và nhật ký chi tiết về các lỗ hổng
- Dễ sử dụng, dễ dàng để cài đặt
- Hỗ trợ kiểm thử bằng hình thức thủ công và tự động

3.1.2 Mô hình hoạt động

ZAP sẽ tiến hành thu thập dữ liệu ứng dụng web bằng trình thu thập dữ liệu của nó và quét thụ động từng trang mà nó tìm thấy. Sau đó, ZAP sẽ sử dụng trình quét để tấn công tất cả các trang, chức năng và tham số được phát hiện.

Các bước thực hiện gồm:

1. Khởi chạy công cụ quét web

2. Nhập URL của ứng dụng web sẽ được kiểm tra

3. Nhấp vào nút quét và đợi quá trình quét hoàn tất

4. Nếu quá trình quét được thực hiện thành công, một báo cáo sẽ được hiển thị cùng với kết quả.

Lưu ý: Chỉ nên sử dụng ZAP để tấn công một ứng dụng mà có quyền kiểm tra bằng một cuộc tấn công đang hoạt động. Vì đây là mô phỏng hoạt động như một cuộc tấn công thực sự, thiệt hại thực tế có thể được thực hiện đối với chức năng, dữ liệu của trang web,... Nếu lo lắng về việc sử dụng ZAP, có thể ngăn nó gây hại (mặc dù chức năng của ZAP sẽ giảm đáng kể) bằng cách chuyển sang chế độ an toàn.

3.2 Thực nghiệm kiểm thử bảo mật dựa Web dựa trên ZAP

3.2.1 Giai đoạn lập kế hoạch và khám phá

a. Chuẩn bị các thông tin về ứng dụng web cần kiểm thử và công cụ kiểm thử

- Thực hiện kiểm thử trên các website:

STT	**Server**	**IP / Domain**	**Account**
1	https://duticrm.info/	45.76.50.205	Username: myhanh Password: 123456

2	https://hack-yourself-first.com/		

- Sử dụng công cụ kiểm thử Zed Attack Proxy để thực hiện quét các lỗ hổng bảo mật.

Download ZAP tại link:

https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

Name	**Version**	**Operating system**
OWASP ZAP (Zed Attack Proxy)	2.8.0	Window, Linux, Mac OS/X

b. Thực nghiệm

Cách dễ nhất để bắt đầu sử dụng ZAP là thông qua tab Quick Start. Quick Start là một tiện ích bổ sung ZAP được bao gồm tự động khi bạn cài đặt ZAP.

Để chạy Quick Start, thực hiện các bước:

1. Khởi động ZAP và bấm vào tab **Quick Start** của Cửa sổ Workspace.

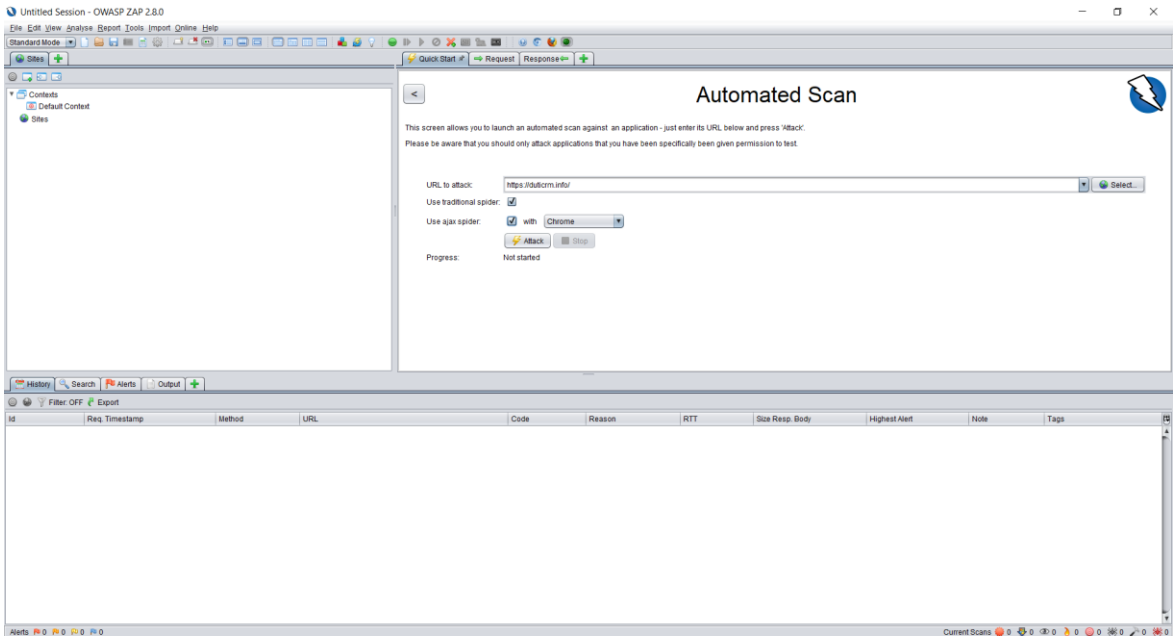
2. Nhấp vào nút **Automated Scan.**

3. Trong hộp văn bản URL để tấn công, hãy nhập URL đầy đủ của ứng dụng web bạn muốn tấn công. Ở đây với Ví dụ 1, nhập: https://duticrm.info/

Ngoài ra Trang web: https://duticrm.info/ có sử dụng ajax (có các tính năng như xử lý, tính toán, đăng ký, submit mà không load lại trang) nên sẽ lựa chọn thêm vào: **Use ajax spider.** Mục đích sử dụng giống như spider là để tìm kiếm link bằng cách giả lập thao tác trên màn hình. Từ đó sẽ tìm được link chuyển tiếp mà n mà spider thông thường không thể làm được.

4. Nhấp vào nút **Attack**

ZAP sẽ tiến hành thu thập dữ liệu ứng dụng web bằng trình thu thập dữ liệu của nó và quét chủ động từng trang mà nó tìm thấy. Sau đó, ZAP sẽ sử dụng trình quét để tấn công tất cả các trang, chức năng và tham số được phát hiện.

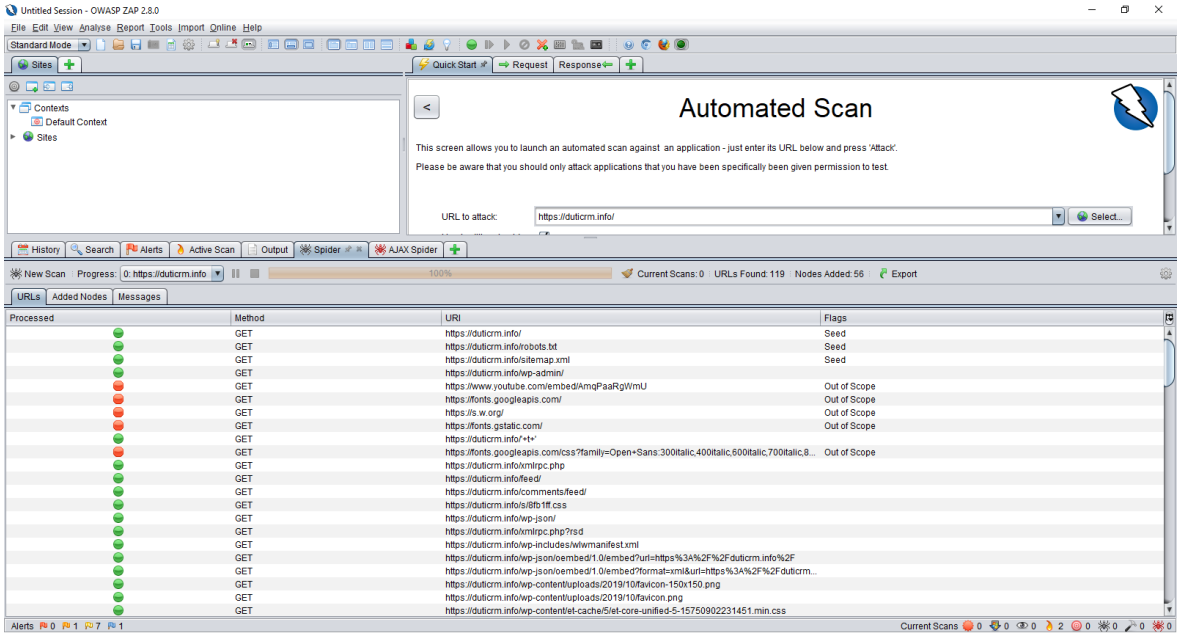


Hình 3.1. Giao diện chính của công cụ OWASP ZAP

Quy trình tấn công phổ biến của OWASP ZAP được chia thành hai bước cơ bản: thu thập dữ liệu của ứng dụng web và cố gắng xâm nhập tấn công trang web. OWASP ZAP cung cấp hai mô-đun Spider và Attack để thực hiện các bước này:

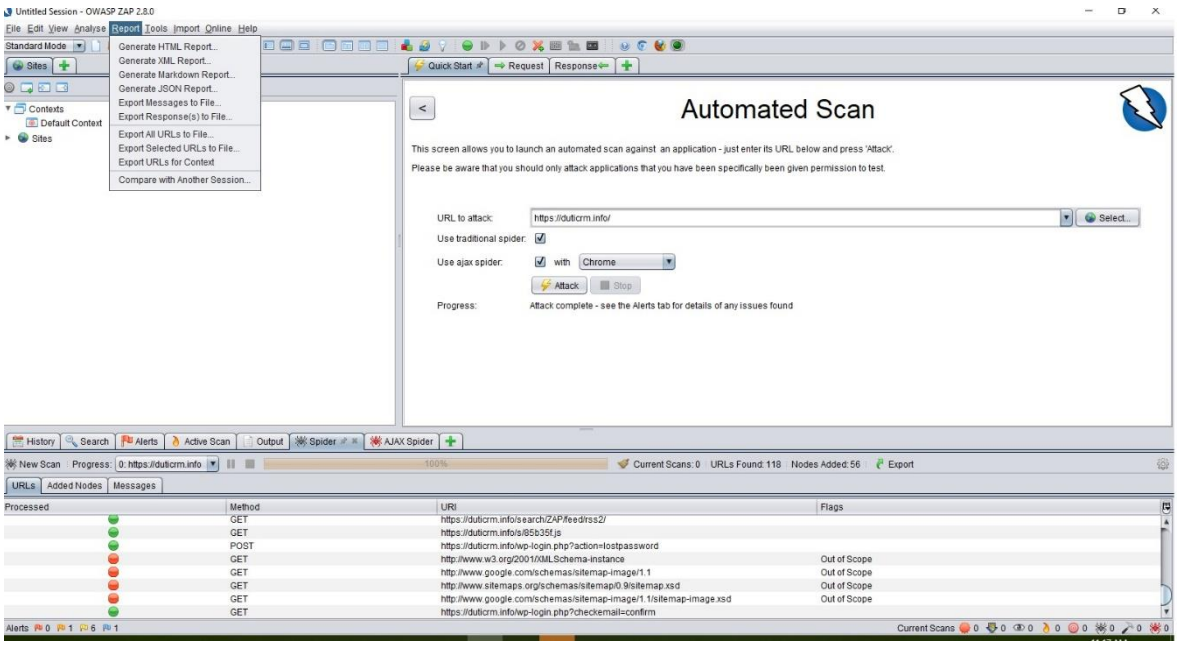
- **Thu thập dữ liệu ứng dụng web bằng cách sử dụng OWASP ZAP Spider**

Mô-đun Spider của ZAP được sử dụng để quét nội dung của ứng dụng web bằng cách theo các liên kết và kiểm tra các mẫu URL phổ biến để trả về danh sách các URL có sẵn mà ứng dụng có. Bước này là bắt buộc vì mô-đun tấn công sẽ được khởi chạy sau này cần phải có danh sách URL đã biết để khởi chạy các cuộc tấn công. Bước thu thập thông tin được khởi chạy với các tùy chọn mặc định bằng cách sử dụng tùy chọn Attack->Spider. Sau khi khởi chạy Spider, ZAP sẽ biên dịch danh sách các URL đã được tìm thấy trong ứng dụng.



Hình 3.2. Giao diện mô-đun Spider

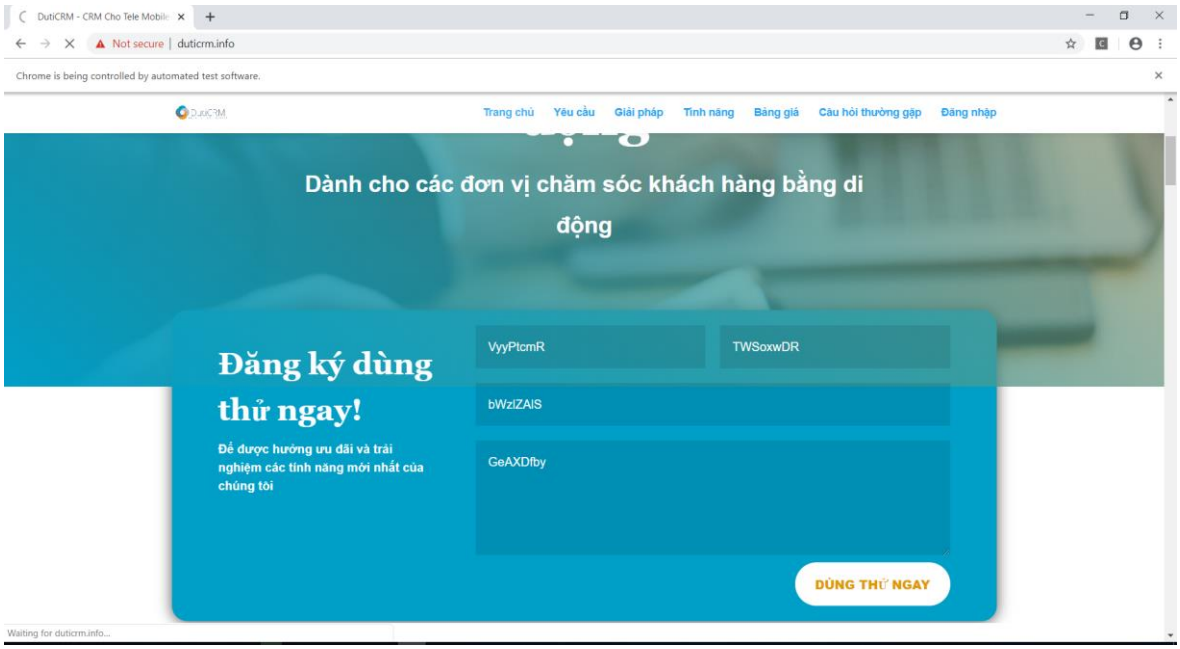
Có thể xuất danh sách các URL được tìm thấy bằng cách chọn: Report => Export all URL to file…(Danh sách các URL được thống kê ở Phụ lục)



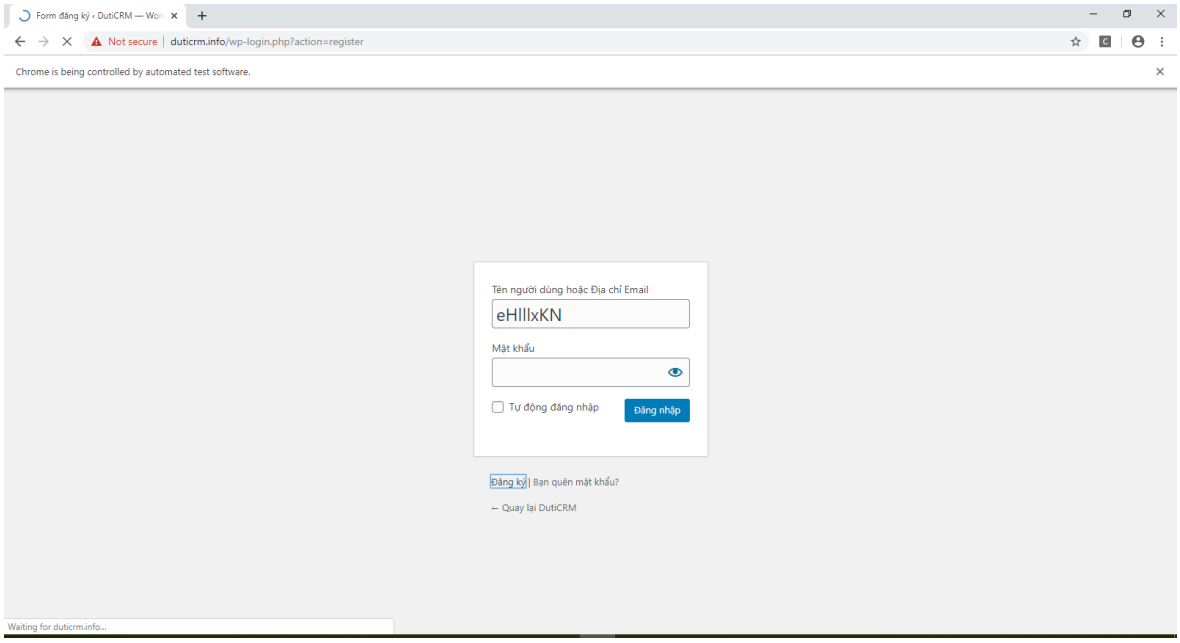
Hình 3.3: Cửa sổ lựa chọn Xuất báo cáo

- **Thu thập dữ liệu trên ứng dụng web ở các tính năng liên quan đến AJAX bằng AJAX Spider**

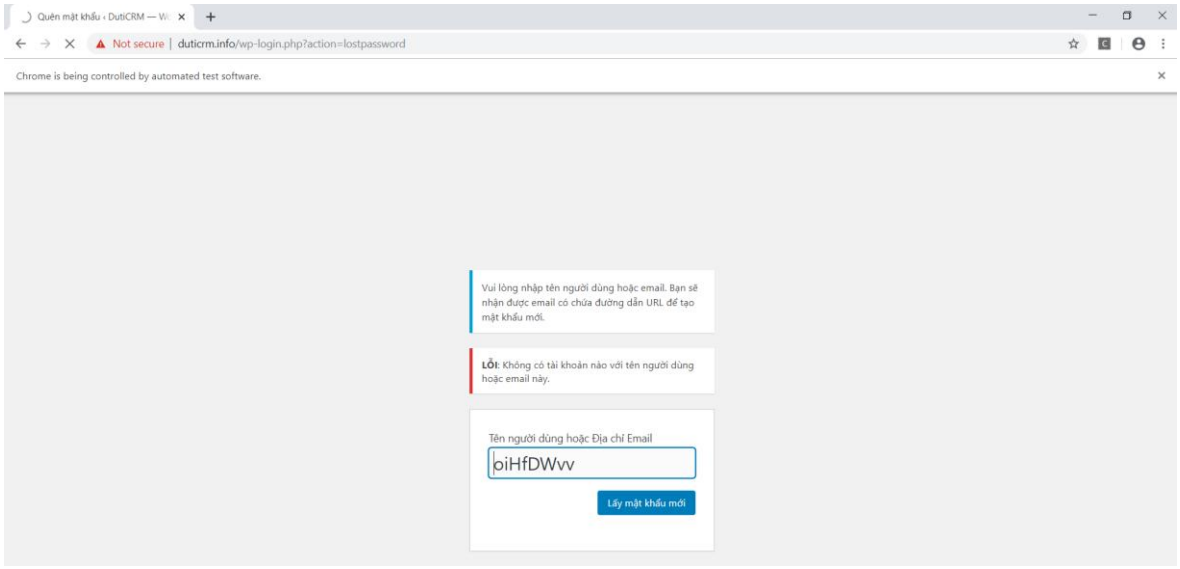
Lúc này phần mềm sẽ tự động giả lập thao tác trên màn hình như sau:



Hình 3.4: Thao tác tự động tại form Đăng ký dùng thử

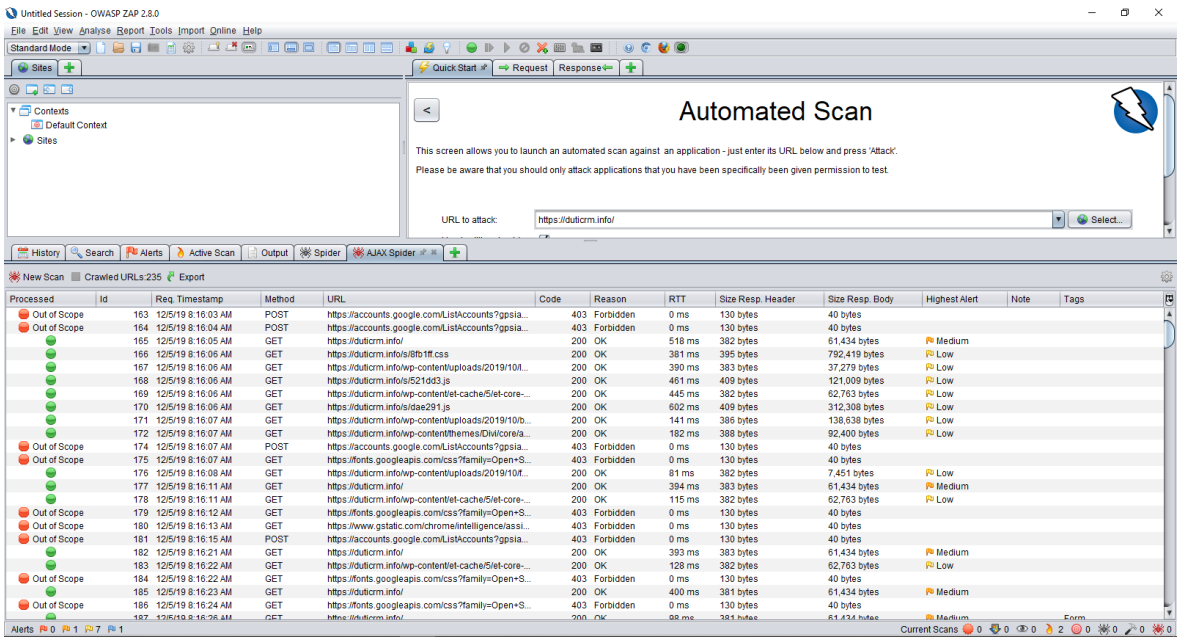


Hình 3.5: Thao tác tự động tại form Đăng nhập



Hình 3.6: Thao tác tự động tại form Quên mật khẩu

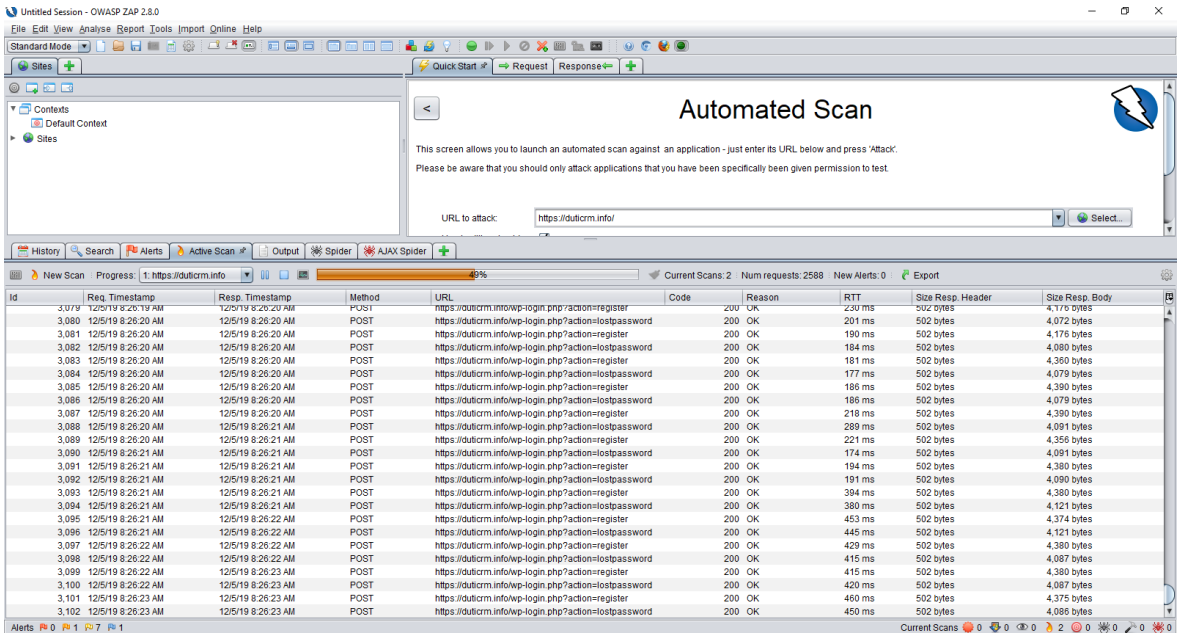
Sau đó danh sách các URL liên quan được tìm thấy trong ứng dụng được liệt kê ở mô đun AJAX Spider:



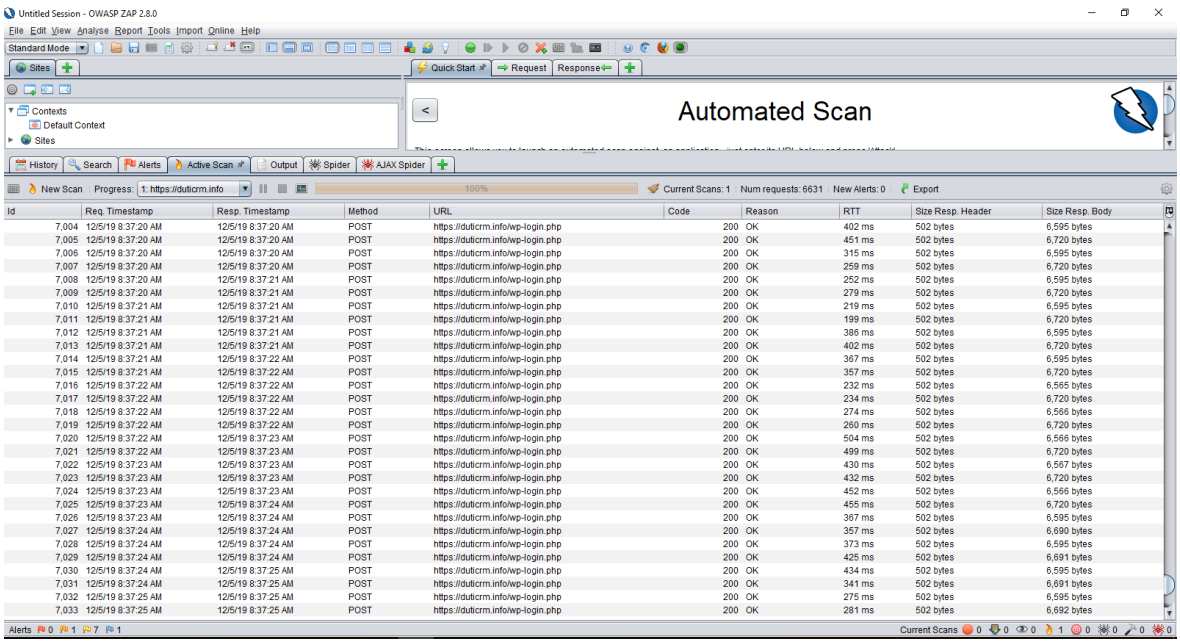
Hình 3.7: Mô đun AJAX Spider

- **Tấn công ứng dụng web bằng mô-đun Active Scan**

Mô-đun Active Scan của OWASP ZAP cho phép thực hiện các cuộc tấn công hoạt động thực sự để tìm và kiểm tra sự tồn tại của các lỗ hổng trong ứng dụng web cần kiểm thử:

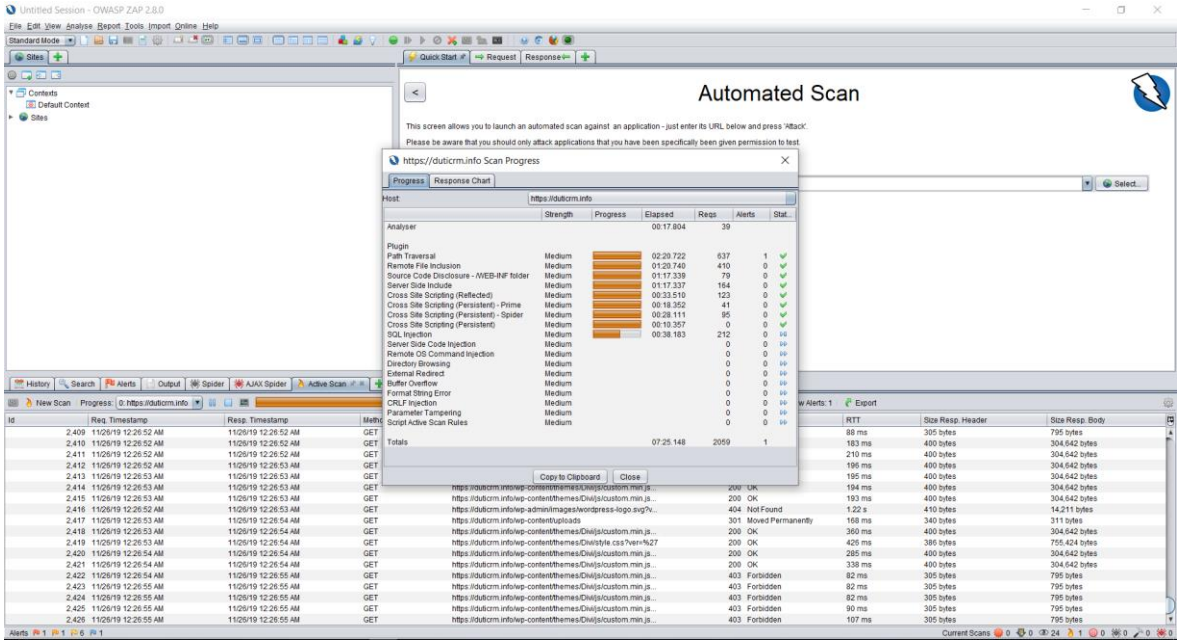


Hình 3.8: Giao diện mô-đun Active Scan đang quét các lỗ hổng bảo mật



Hình 3.9: Giao diện mô-đun Active Scan quét thành công các lỗ hổng bảo mật

Tiến trình của quá trình quét hoạt động có thể được theo dõi từ cửa sổ Progress Information.

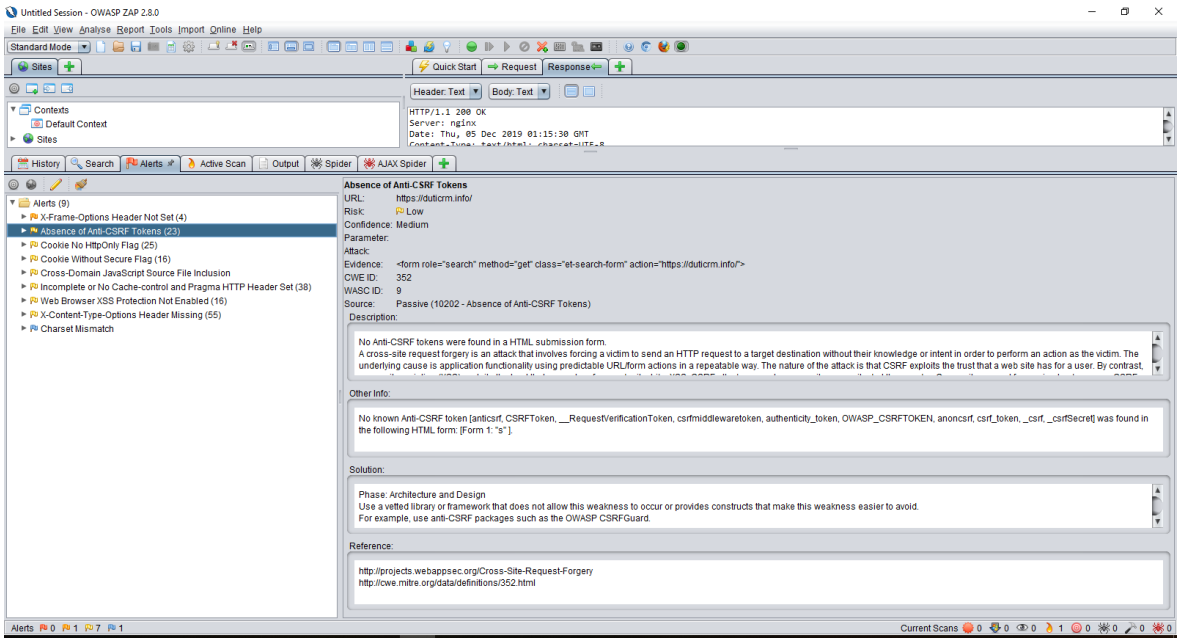


Hình 3.10: Cửa sổ Progress Information

Các lỗ hổng bảo mật sẽ được thống kê chi tiết ở tab Alerts, những Request có thể bị lỗi sẽ được liệt kê dưới chi tiết. Sau đó Đọc các thông tin lỗi, thực hiện tiến hành tái tạo lỗi. Có 4 màu thể hiện mức độ nguy hiểm từ cao đến thấp là Đỏ (Cao) ➞ Cam (Trung bình) ➞ Vàng (Thấp)➞ Xanh (Rất thấp):

- Cao: Kẻ tấn công có thể xâm nhập và thỏa hiệp hệ thống và /hoặc có quyền truy cập vào thông tin hệ thống có độ nhạy cảm cao. Điều này dẫn đến việc nó có thể trộm cắp hoặc mất dữ liệu riêng tư và nhạy cảm.
- Trung bình: Kẻ tấn công có thể có quyền truy cập vào thông tin hệ thống và dẫn đến các cuộc tấn công khác và có thể là thỏa hiệp toàn hệ thống. Điều này sẽ có thể dẫn đến hành vi trộm cắp hoặc mất dữ liệu cá nhân và nhạy cảm.

- Thấp: Kẻ tấn công có thể được truy cập vào hệ thống thông tin để hỗ trợ các cuộc tấn công cụ thể hơn dẫn đến hành vi trộm cắp hoặc mất dữ liệu cá nhân và nhạy cảm.
- Rất thấp: Tất cả các mục ở cấp độ này chỉ đơn giản là cung cấp thông tin bổ sung cho thông tin đã có sẵn về hệ thống được thử nghiệm. Nó không ngụ ý rằng hệ thống có dễ bị tổn thương hay không.



Hình 3.11: Thống kê lỗ hổng bảo mật

Khi click vào mỗi lỗ hổng, nhìn vào cột bên cạnh ta có thể thấy được thông tin về lỗ hổng cũng như giải pháp sửa chữa lỗ hổng được đưa ra.

3.2.2 Đánh giá lỗ hổng

Ví dụ như trên tab Alert ta thấy có lỗi: Absence of Anti-CSRF Tokens, tiến hành phân tích, đánh giá lỗi này ta có các thông tin như sau:

- CSRF (Cross Site Request Forgery) là kĩ thuật tấn công bằng cách sử dụng quyền chứng thực của người sử dụng đối với 1 website khác. CSRF là kiểu tấn công hacker lợi dụng phiên làm việc của mình trên trình duyệt đó để thực hiện một hành động nào đó mà mình không biết. Thường kết hợp với

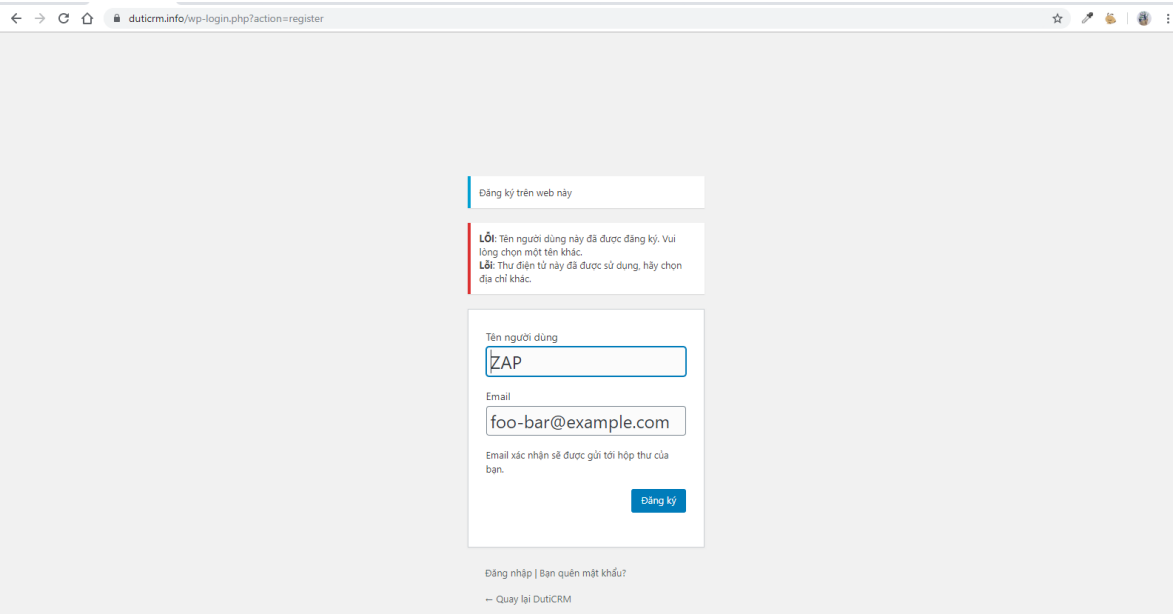
hình thức social engineering (dẫn dụ người dùng click một đường link nào đó hoặc load lại trang… mà user không rõ). Hacker tạo một trang web giả mạo giống trang web mình vừa truy cập, chứa những thông tin giống form thật. Nhưng khi người dùng thực hiện các giao dịch ở đây sẽ bị mất thông tin.

- Giả mạo yêu cầu trên nhiều trang web là một cuộc tấn công liên quan đến việc buộc nạn nhân gửi yêu cầu HTTP đến đích mà không có kiến thức hoặc ý định của họ để thực hiện hành động với tư cách là nạn nhân. Nguyên nhân cơ bản là chức năng ứng dụng sử dụng các hành động URL/ biểu mẫu có thể dự đoán theo cách lặp lại. Bản chất của cuộc tấn công là CSRF khai thác niềm tin mà một trang web dành cho người dùng.
- Một trong những kỹ thuật để phòng chống CSRF là sử dụng CSRF Token. Tạo ra một token tương ứng với mỗi form, token này sẽ là duy nhất đối với mỗi form và thường thì hàm tạo ra token này sẽ nhận đối số là"SESSION" hoặc được lưu thông tin trong SESSION. Khi nhận lệnh HTTP POST về, hệ thống sẽ thực hiện so khớp giá trị token này để quyết định có thực hiện hay không.

3.2.3 Giai đoạn khai thác

Giai đoạn này ta sẽ sử dụng những thông tin thu thập được ở trên cùng với kinh nghiệm của mình để tiến hành dò tìm và xác thực các lỗ hổng tồn tại trên hệ thống website. Tiếp lỗ hổng bên trên: Absence of Anti-CSRF Tokens, tiến hành tái tạo lại thủ công lỗ hổng trên ta được:

Tại trang đăng ký người dùng: nhập Tên người dùng = ZAP; Email = foo-bar@example.com, hiển thị thông báo lỗi như sau:



Hình 3.12: Giao diện Đăng ký người dung

Về kỹ thuật Anti-CSRF token: khi một người dùng gửi một form hoặc tạo các request được chứng thực yêu cầu một cookie, một mã token anti-CSRF nên được bao gồm trong request. Ứng dụng web sẽ xác nhận các token đã tồn tại và chính xác trước khi xử lý yêu cầu. Nếu token không có hoặc không đúng thì request sẽ bị từ chối. Nhìn từ phần Request thể hiện ở tool ZAP, khi gửi thông tin yêu cầu đăng nhập, không tìm thấy dòng nào chứa Anti-CSRF token:

```
1.  POST https://duticrm.info/wp-login.php?action=register HTTP/1.1
2.  User-
    Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:39.0) Gecko/20100101 Firefox/39.0

3.  Pragma: no-cache
4.  Cache-Control: no-cache
5.  Content-Type: application/x-www-form-urlencoded
6.  Content-Length: 93
7.  Referer: https://duticrm.info/wp-login.php?action=register
8.  Host: duticrm.info
9.  Cookie: wordpress_test_cookie=WP+Cookie+check
10. user_login=ZAP&user_email=foo-bar%40example.com&redirect_to=&wp-
    submit=%C4%90%C4%83ng+k%C3%BD
```

3.2.4 Giai đoạn báo cáo

Theo như thống kê các cảnh báo lỗ hổng từ Mô-đun Alert, ta có bảng Tổng hợp báo cáo như sau: Có 179 lỗ hổng được tìm thấy với 3 mức độ nguy hại: Medium (Trung bình), Low (Thấp), Informational (Rất thấp):

Bảng 3.1: Tổng hợp lỗ hổng bảo mật web: duticrm.info

No	Type	Number issues	Severity
1	X-Frame-Options Header Not Set	4	Medium
2	Absence of Anti-CSRF Tokens	23	Low
3	Cookie No HttpOnly Flag	25	Low
4	Cookie Without Secure Flag	16	Low
5	Incomplete or No Cache-control and Pragma HTTP Header Set	38	Low
6	Cross-Domain JavaScript Source File Inclusion	1	Low
7	Web Browser XSS Protection Not Enabled	16	Low
8	X-Content-Type-Options Header Missing	55	Low
9	Charset Mismatch	1	Informational

Từ báo cáo tổng quát, lập báo cáo chi tiết từng loại Lỗ hổng:

Với lỗ hổng Absence of Anti-CSRF Tokens Phân tích ở trên, ta có mẫu báo cáo như sau. Các lỗ hổng khác, báo cáo cũng được tiến hành tương tự.

◉ Title: Absence of Anti-CSRF Tokens
◉ Details: No Anti-CSRF tokens were found in a HTML submission form.
◉ Severity: Low
◉ Steps:

1. Go to: https://duticrm.info/wp-login.php?action=register

2. Enter the script in Username/Email textbox: user_login=ZAP&user_email=foo-bar@example.com

◉ Request:

```
POST https://duticrm.info/wp-login.php?action=register HTTP/1.1
User-
Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:39.0) Gecko/20100101 Firefox/39.0

Pragma: no-cache
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
Content-Length: 93
Referer: https://duticrm.info/wp-login.php?action=register
Host: duticrm.info
Cookie: wordpress_test_cookie=WP+Cookie+check
user_login=ZAP&user_email=foo-bar%40example.com&redirect_to=&wp-
submit=%C4%90%C4%83ng+k%C3%BD
```

◉ Response:

```
<!DOCTYPE html>
    <!--[if IE 8]>
        <html xmlns="http://www.w3.org/1999/xhtml" class="ie8" lang="vi-VN">
    <![endif]-->
    <!--[if !(IE 8) ]><!-->
        <html xmlns="http://www.w3.org/1999/xhtml" lang="vi-VN">
    <!--<![endif]-->
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Form đăng ký ‹ DutiCRM — WordPress</title>
    <link rel='dns-prefetch' href='//s.w.org' />
<link rel='stylesheet' id='dashicons-css'  href='https://duticrm.info/wp-
includes/css/dashicons.min.css?ver=5.3' type='text/css' media='all' />
<link rel='stylesheet' id='buttons-css'  href='https://duticrm.info/wp-
includes/css/buttons.min.css?ver=5.3' type='text/css' media='all' />
<link rel='stylesheet' id='forms-css'  href='https://duticrm.info/wp-
admin/css/forms.min.css?ver=5.3' type='text/css' media='all' />
<link rel='stylesheet' id='l10n-css'  href='https://duticrm.info/wp-
admin/css/l10n.min.css?ver=5.3' type='text/css' media='all' />
<link rel='stylesheet' id='login-css'  href='https://duticrm.info/wp-
admin/css/login.min.css?ver=5.3' type='text/css' media='all' />
    <meta name='robots' content='noindex,noarchive' />
    <meta name='referrer' content='strict-origin-when-cross-origin' />
        <meta name="viewport" content="width=device-width" />
    <link rel="icon" href="https://duticrm.info/wp-
content/uploads/2019/10/favicon-150x150.png" sizes="32x32" />
<link rel="icon" href="https://duticrm.info/wp-
content/uploads/2019/10/favicon.png" sizes="192x192" />
<link rel="apple-touch-icon-precomposed" href="https://duticrm.info/wp-
content/uploads/2019/10/favicon.png" />
<meta name="msapplication-TileImage" content="https://duticrm.info/wp-
content/uploads/2019/10/favicon.png" />
    </head>
    <body class="login no-js login-action-register wp-core-ui  locale-vi">
    <script type="text/javascript">
        document.body.className = document.body.className.replace('no-
js','js');
    </script>
        <div id="login">
```

```
30.         <h1><a href="https://vi.wordpress.org/">Xây dựng bằng WordPress</a></h1>
31.     <p class="message register">Đăng ký trên web này</p>
32. <div id="login_error">    <strong>LỖI</strong>: Tên người dùng này đã được đăng k
    ý. Vui lòng chọn một tên khác.<br />
33.     <strong>Lỗi</strong>: Thư điện tử này đã được sử dụng, hãy chọn địa chỉ khác.
    <br />
34. </div>
35.         <form name="registerform" id="registerform" action="https://duticrm.info/
    wp-login.php?action=register" method="post" novalidate="novalidate">
36.             <p>
37.                 <label for="user_login">Tên người dùng</label>
38.                 <input type="text" name="user_login" id="user_login" class="input
    " value="ZAP" size="20" autocapitalize="off" />
39.             </p>
40.             <p>
41.                 <label for="user_email">Email</label>
42.                 <input type="email" name="user_email" id="user_email" class="inpu
    t" value="foo-bar@example.com" size="25" />
43.             </p>
44.                         <p id="reg_passmail">
45.                 Email xác nhận sẽ được gửi tới hộp thư của bạn.         </p>
46.             <br class="clear" />
47.             <input type="hidden" name="redirect_to" value="" />
48.             <p class="submit">
49.                 <input type="submit" name="wp-submit" id="wp-
    submit" class="button button-primary button-large" value="Đăng ký" />
50.             </p>
51.         </form>
52.
53.         <p id="nav">
54.             <a href="https://duticrm.info/wp-login.php">Đăng nhập</a>
55.                 |          <a href="https://duticrm.info/wp-
    login.php?action=lostpassword">Bạn quên mật khẩu?</a>
56.         </p>
57.                 <p id="backtoblog"><a href="https://duticrm.info/">
58.         ← Quay lại DutiCRM      </a></p>
59.             </div>
60.             <script type="text/javascript">
61.         try{document.getElementById('user_login').focus();}catch(e){}
62.         if(typeof wpOnload=='function')wpOnload();
63.         </script>
64.             <div class="clear"></div>
65.     </body>
66.     </html>
```

◉ Recommended:

- Phase: Architecture and Design
- Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
- For example, use anti-CSRF packages such as the OWASP CSRFGuard.

Tương tự, thực hành trên trang: https://hack-yourself-first.com/. Thực hiện các bước như Ví dụ 1, ta có kết quả như sau:

Bảng 3.2. Tổng hợp lỗ hổng bảo mật web: https://hack-yourself-first.com/

No	Type	Number issues	Severity
1	Cross Site Scripting (Reflected)	1	High
2	Path Traversal	3	High
3	SQL Injection - Microsoft SQL Server	1	High
4	X-Frame-Options Header Not Set	49	Medium
5	Absence of Anti-CSRF Tokens	75	Low
6	Cookie No HttpOnly Flag	1	Low
7	Cookie Without SameSite Attribute	1	Low
8	Cookie Without Secure Flag	2	Low
9	Incomplete or No Cache-control and Pragma HTTP Header Set	52	Low
10	Secure Pages Include Mixed Content	2	Low
11	Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	105	Low
12	Web Browser XSS Protection Not Enabled	52	Low
13	X-AspNet-Version Response Header Scanner	105	Low
14	X-Content-Type-Options Header Missing	102	Low
15	Information Disclosure - Suspicious Comments	3	Informational
16	Timestamp Disclosure - Unix	52	Informational

3.3 Đánh giá

- ZAP cung cấp cho các chuyên gia an ninh và kỹ sư kiểm thử phần mềm một loạt các tính năng tuyệt vời trong một gói đơn giản, liền mạch và rất mạnh mẽ:
  - Giao diện thân thiện dễ sử dụng. Phần mềm mã nguồn mở nên cập nhật thường xuyên bởi đội ngũ cộng tác viên trên thế giới.
  - Phù hợp với những người kiểm thử nghiệp dư đến chuyên nghiệp
  - Không cần có kiến thức sâu mà lập trình, hiểu biết chi tiết về hệ thống
  - Không yêu cầu truy cập vào mã nguồn nên có thể thực hiện nhanh chóng, thường xuyên
  - Có thể thực hiện quét tự động hoặc thủ công, tùy theo nhu cầu
  - Trong báo cáo có phân loại các lỗ hổng dựa trên các mức độ nghiêm trọng
  - Sử dụng được với bất kỳ ngôn ngữ lập trình nào và với cả các framework sẵn có hay đã tùy chỉnh.
- Ngoài một số điểm mạnh nêu trên, ZAP cũng có một số hạn chế như:
  - Phần mềm mã nguồn mở, chưa phải bản thương mại nên chưa thực sự hiệu quả. Nhiều loại lỗ hổng bảo mật rất khó tìm thấy như các vấn đề xác thực, các vấn đề kiểm soát truy cập, sử dụng mật mã không an toàn,...
  - Dữ liệu có thể bị ghi đè hoặc tải trọng độc hại được đưa vào trang web thực hiện kiểm thử.
  - Chỉ phù hợp với doanh nghiệp quy mô vừa và nhỏ

KẾT LUẬN

1. Những đóng góp của luận văn

Tấn công mạng là khá thường xuyên trong những thời điểm hiện tại. Việc tiến hành kiểm thử bảo mật web trên cơ sở quét lỗ hổng thường xuyên và kiểm tra thâm nhập để phát hiện các lỗ hổng là điều quan trọng và cấp thiết của mỗi cơ quan, doanh nghiệp và cá nhân đối với website của mình. Tần suất tiến hành kiểm tra xâm nhập phải phụ thuộc vào chính sách bảo mật của tổ chức. Tuy nhiên, tiến hành kiểm tra xâm nhập thường xuyên có thể xác định điểm yếu của hệ thống và giúp nó tránh các vi phạm an ninh. Một trong các cách là sử dụng các công cụ kiểm thử. Các công cụ kiểm thử cung cấp giao diện trực quan và đơn giản để xác định các lỗ hổng bảo mật cụ thể.

Luận văn đã thực hiện nghiên cứu về bảo mật website, kiểm thử bảo mật website đặc biệt đi sâu nghiên cứu các phương pháp kiểm thử cho website và sử dụng công cụ kiểm thử bảo mật Zed Attack Proxy (ZAP) để kiểm thử các trang web: https://duticrm.info và https://hack-yourself-first.com/ .

Cụ thể kết quả đạt được như sau:

- Luận văn đã trình bày về các vấn đề bảo mật website, kiểm thử bảo mật website. Đồng thời cũng trình bày các phương pháp kiểm thử website và các công cụ hiện có tương ứng. Thực hiện so sánh phân tích các ưu nhược điểm và khả năng áp dụng thực tế của của phương pháp và công cụ để lựa chọn công cụ áp dụng vào công việc kiểm thử bảo mật website thực tế.
- Tiếp theo luận văn trình bày việc sử dụng công cụ Zed Attack Proxy (ZAP) cho việc kiểm thử bảo mật website https://duticrm.info và https://hack-yourself-first.com/. Toàn bộ kết quả đạt được trong phần áp dụng thực nghiệm này là quá trình cố gắng của bản thân trong việc vận dụng lý thuyết, tìm tòi, nghiên cứu, học hỏi từ đồng nghiệp, bạn bè và một vài dự án trong thực tế.

- Kết quả nghiên cứu rất có ích cho việc kiểm thử bảo mật của website https://duticrm.info và https://hack-yourself-first.com/. Từ đó giúp tìm ra các lỗi, lỗ hổng bảo mật của website để kịp thời đưa ra các phương án khắc phục sớm đảm bảo website an toàn và không bị tấn công bởi tin tặc.
- Do thời gian và kiến thức còn hạn chế nên chưa thực hiện được kết hợp nhiều công cụ với nhau với nhiều phương pháp khác nhau trên nhiều website để đưa ra kết quả chính xác hơn.

2. Hướng phát triển

Luận văn sử dụng nghiên cứu các phương pháp, công cụ kiểm thử ứng dụng cho kiểm thử bảo mật website.

Vì vậy hướng nghiên cứu tiếp theo là có thể xây dựng được một hệ thống/công cụ áp dụng các phương pháp kiểm thử khác nhau để đưa vào triển khai ở đơn vị mà không phải tốn thêm chi phí mua bản quyền của công cụ.

TÀI LIỆU THAM KHẢO

[1] PGS.TSKH Hoàng Đăng Hải, *Quản lý an toàn thông tin*, 2018.

[2] Karin Klooster (2016), University of tartu, "*Applying a Security Testing Methodology: a Case Study*"

[3] Can Zhu (2017), Aalto University School of Science, "*Experimental study of vulnerabilities in a web application*"

[4] https://securitybox.vn/4509/10-lo-hong-trong-ung-dung-web/ (truy cập ngày 01/11/2019)

[5] http://softwaretestingfundamentals.com/white-box-testing/ (truy cập ngày 08/11/2019)

[6] http://softwaretestingfundamentals.com/black-box-testing/ (truy cập ngày 08/11/2019)

[7] https://owasp.org/www-project-zap/ (truy cập ngày 15/11/2019)

[8] https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_Top_10.html (truy cập ngày 15/10/2019)

[10] https://whitehat.vn/threads/pentest-chuan-pentest-website-theo-owasp.6989/ (truy cập ngày 10/11/2019)

[11] https://insights.sei.cmu.edu/sei_blog/2018/07/10-types-of-application-security-testing-tools-when-and-how-to-use-them.html (truy cập ngày 08/11/2019)

[12] https://www.breachlock.com/3-opensource-tools-for-dast/ (truy cập ngày 15/11/2019)

[13] https://www.acunetix.com/vulnerability-scanner/ (truy cập ngày 15/10/2019)

[14] http://rips-scanner.sourceforge.net/ (truy cập ngày 15/10/2019)

[15]https://www.ibm.com/support/knowledgecenter/en/SSS9LM_9.0.3/com.ibm.rational.appscansrc.common.doc/topics/intro_products.html (truy cập ngày 15/10/2019)

[16] https://tools.kali.org/information-gathering/nikto (truy cập ngày 25/10/2019)

[17] https://cystack.net/vi/resource/tinh-hinh-an-ninh-mang-2019/ (truy cập ngày 25/11/2019)

PHỤ LỤC 1: DANH SÁCH URL QUÉT ĐƯỢC

1. https://duticrm.info/
2. https://duticrm.info/'+t+'
3. https://duticrm.info/+t+
4. https://duticrm.info/author
5. https://duticrm.info/author/duticrm/
6. https://duticrm.info/comments
7. https://duticrm.info/comments/feed/
8. https://duticrm.info/embed/
9. https://duticrm.info/feed/
10. https://duticrm.info/page-sitemap.xml
11. https://duticrm.info/post-sitemap.xml
12. https://duticrm.info/robots.txt
13. https://duticrm.info/s
14. https://duticrm.info/s/521dd3.js
15. https://duticrm.info/s/7e4e02.js
16. https://duticrm.info/s/85b35f.js
17. https://duticrm.info/s/8fb1ff.css
18. https://duticrm.info/s/904306.js
19. https://duticrm.info/s/fd37bd.js
20. https://duticrm.info/search
21. https://duticrm.info/search/ZAP
22. https://duticrm.info/search/ZAP/feed
23. https://duticrm.info/search/ZAP/feed/rss2/
24. https://duticrm.info/search/feed
25. https://duticrm.info/search/feed/rss2/
26. https://duticrm.info/sitemap.xml
27. https://duticrm.info/sitemap_index.xml
28. https://duticrm.info/wp-admin
29. https://duticrm.info/wp-admin/
30. https://duticrm.info/wp-admin/admin-ajax.php
31. https://duticrm.info/wp-admin/css
32. https://duticrm.info/wp-admin/css/forms.min.css?ver=5.3
33. https://duticrm.info/wp-admin/css/l10n.min.css?ver=5.3
34. https://duticrm.info/wp-admin/css/login.min.css?ver=5.3
35. https://duticrm.info/wp-admin/images
36. https://duticrm.info/wp-admin/images/wordpress-logo.svg?ver=20131107
37. https://duticrm.info/wp-admin/js
38. https://duticrm.info/wp-admin/js/password-strength-meter.min.js?ver=5.3
39. https://duticrm.info/wp-admin/js/user-profile.min.js?ver=5.3
40. https://duticrm.info/wp-content
41. https://duticrm.info/wp-content/et-cache
42. https://duticrm.info/wp-content/et-cache/5
43. https://duticrm.info/wp-content/et-cache/5/et-core-unified-5-1574696153603.min.css
44. https://duticrm.info/wp-content/et-cache/global
45. https://duticrm.info/wp-content/et-cache/global/et-divi-customizer-global-15746962159519.min.css
46. https://duticrm.info/wp-content/themes
47. https://duticrm.info/wp-content/themes/Divi
48. https://duticrm.info/wp-content/themes/Divi/core
49. https://duticrm.info/wp-content/themes/Divi/core/admin
50. https://duticrm.info/wp-content/themes/Divi/core/admin/fonts
51. https://duticrm.info/wp-content/themes/Divi/core/admin/fonts/modules.ttf
52. https://duticrm.info/wp-content/themes/Divi/core/admin/js
53. https://duticrm.info/wp-content/themes/Divi/core/admin/js/common.js?ver=4.0.6
54. https://duticrm.info/wp-content/themes/Divi/js
55. https://duticrm.info/wp-content/themes/Divi/js/custom.min.js?ver=4.0.6
56. https://duticrm.info/wp-content/themes/Divi/style.css?ver=4.0.6

57. https://duticrm.info/wp-content/uploads
58. https://duticrm.info/wp-content/uploads/2019
59. https://duticrm.info/wp-content/uploads/2019/10
60. https://duticrm.info/wp-content/uploads/2019/10/Afbeelding-stand_klein-1.png
61. https://duticrm.info/wp-content/uploads/2019/10/business-coach-0034.jpg
62. https://duticrm.info/wp-content/uploads/2019/10/favicon-150x150.png
63. https://duticrm.info/wp-content/uploads/2019/10/favicon.png
64. https://duticrm.info/wp-content/uploads/2019/10/logo.png
65. https://duticrm.info/wp-includes
66. https://duticrm.info/wp-includes/css
67. https://duticrm.info/wp-includes/css/buttons.min.css?ver=5.3
68. https://duticrm.info/wp-includes/css/dashicons.min.css?ver=5.3
69. https://duticrm.info/wp-includes/css/dist
70. https://duticrm.info/wp-includes/css/dist/block-library
71. https://duticrm.info/wp-includes/css/dist/block-library/style.min.css?ver=5.3
72. https://duticrm.info/wp-includes/js
73. https://duticrm.info/wp-includes/js/jquery
74. https://duticrm.info/wp-includes/js/jquery/jquery-migrate.min.js?ver=1.4.1
75. https://duticrm.info/wp-includes/js/jquery/jquery.js?ver=1.12.4-wp
76. https://duticrm.info/wp-includes/js/underscore.min.js?ver=1.8.3
77. https://duticrm.info/wp-includes/js/wp-embed.min.js?ver=5.3
78. https://duticrm.info/wp-includes/js/wp-util.min.js?ver=5.3
79. https://duticrm.info/wp-includes/js/zxcvbn-async.min.js?ver=1.0
80. https://duticrm.info/wp-includes/js/zxcvbn.min.js
81. https://duticrm.info/wp-includes/wlwmanifest.xml
82. https://duticrm.info/wp-json
83. https://duticrm.info/wp-json/
84. https://duticrm.info/wp-json/oembed
85. https://duticrm.info/wp-json/oembed/1.0
86. https://duticrm.info/wp-json/oembed/1.0/embed?format=xml&url=https%3A%2F%2Fduticrm.info%2F
87. https://duticrm.info/wp-json/oembed/1.0/embed?url=https%3A%2F%2Fduticrm.info%2F
88. https://duticrm.info/wp-login.php
89. https://duticrm.info/wp-login.php?action=lostpassword
90. https://duticrm.info/wp-login.php?action=register
91. https://duticrm.info/wp-login.php?checkemail=confirm
92. https://duticrm.info/wp-login.php?redirect_to=https%3A%2F%2Fduticrm.info%2Fwp-admin%2F&reauth=1
93. https://duticrm.info/xmlrpc.php
94. https://duticrm.info/xmlrpc.php?rsd