OWASP TOP 10

**Web Application Security Testing**

**I. Giới thiệu và mục tiêu của kiểm thử**

1. Khái niệm:

- Kiểm tra bảo mật là phương pháp để đánh giá mức độ an toàn của hệ thống máy tính hoặc mạng bằng phương pháp xác nhận và xác thực tính hiệu quả của biện pháp kiểm soát bảo mật ứng dụng

- Kiểm tra bảo mật ứng dụng web chỉ tập trung vào đánh giá tính bảo mật của ứng dụng web.

- Quá trình này bao gồm chủ động phân tích điểm yếu của ứng dụng, sai sót kỹ thuật (techniquals flaws) hoặc lỗ hổng bảo mật (vuln)

- Mọi vấn đề bảo mật sẽ được thông báo cho chủ sở hữu hệ thống, cùng đánh giá về tác động, đề xuất giảm thiểu hoặc giải pháp kỹ thuật

2. OWASP Testing methodology:

- Dựa trên phương pháp kiểm thử hộp đen (black-box) – người kiểm thử không biết gì hoặc có rất ít thông tin về ứng dụng cần kiểm thử

\* Mô hình kiểm thử:

+ Người kiểm thử (Tester): Người thực hiện việc kiểm thử

+ Tools and Methodology

+ Application: Black box được kiểm thử

\* 2 giai đoạn kiểm thử:

+ Phase 1: Passive mode:

* Mục tiêu: Cố gắng hiểu logic của ứng dụng và làm việc với ứng dụng. Cuối giai đoạn, người dùng phải hiểu tất cả các điểm truy cập (cổng) của ứng dụng (HTTP header, parameter, cookie). Mỗi cổng được tìm thấy sẽ đại diện cho một điêm thử nghiệm
* Tools có thể được sử dụng để thu thập thông tin

+ Phase 2: Active mode:

* Ở giai đoạn này, người kiểm thử bắt đầu sử dụng các phương pháp để kiểm tra như là:

+ Thu thập thông tin

+ Kiểm tra quản lý triển khai và cấu hình

+ Kiểm tra quản lý danh tính

+ Kiểm tra xác thực

+ Kiểm tra ủy quyền

+ Kiểm tra quản lý phiên

+ Kiểm tra xác thực đầu vào

+ Xử lý lỗi

+ Mật mã

+ Kiểm tra logic nghiệp vụ

+ Kiểm tra phía khách hàng

**II. Thu thập thông tin trong kiểm thử (Information Gathering):**

1. Phát hiện rỏ rỉ thông tin (information leakage) bằng công cụ tìm kiếm (cách tìm kiếm) hoặc trinh sát (reconnaissance):

\* Tổng quan:

- Có 2 phương pháp:

+ Trực tiếp (direct):

- Liên quan đến việc tìm kiếm các chỉ mục và nội dung liên quan từ bộ đệm

+ Gián tiếp (indirect):

- Liên quan đến thu thập thông tin cấu hình và thiết kế nhạy cảm bằng cách tìm kiếm trên các diễn đàn, nhóm tin và trang web đấu thầu

- Sau khi công cụ tìm kiếm robot hoàn thành thu thập thông tin => nó bắt đầu lập chỉ mục trang web dựa trên các thẻ và các thuộc tính liên quan (<TITLE>) để trả về kết quả tìm kiếm có liên quan.

- Nếu các file robot.txt không được cập trong thời gian trang web tồn tại và các thẻ nội tuyến meta HTML hướng dẫn robot không được chỉ mục nội dung chưa được sử dụng => các chỉ mục có thể chứa nội dung web không được chủ sở hữu đưa vào.

- Chủ sở hữu trang web có thể sử dụng robots.txt, thẻ meta HTML, xác thực và các tools được cung cấp bởi công cụ tìm kiếm trước đó để xóa nội dung

\* Mục đích của kiểm thử:

- Để hiểu những thông tin thiết kế và cấu hình nhạy cảm nào của ứng dụng/hệ thống/tổ chức được hiển thị trực tiếp (trên trang web của tổ chức) hoặc gián tiếp (trên trang web của bên thứ 3)

\* Kiểm thử ntn ?  
 - Sử dụng công cụ tìm kiếm tìm các thông tin:

+ Sơ đồ và cấu hình mạng

+ Các bài đăng và email được lưu trữ bởi quản trị viên và nhân viên chủ chốt khác

+ Thủ tục đăng nhập và định dạng tên người dùng

+ Tên người dùng và mật khẩu

+ Nội dung thông báo lỗi

+ Các phiên bản phát triển, thử nghiệm UAT và dàn dựng của trang web

\*\*UAT (User Acceptance Testing)?

+ Là kiểm thử chấp nhận người dùng

+ Là giai đoạn kiểm thử cuối cùng trước khi sản phẩm được đưa ra thị trường

+ Quá trình này chủ yếu dựa vào đánh giá của người dùng ⬄Người dùng sẽ trải nghiệm xem có sự xuất hiện của lỗi tiềm ẩn hoặc vấn đề liên quan đến trải nghiệm không. Sau đó họ sẽ cung cấp phản hồi cho bên phát triển để có điều chỉnh phù hợp

+ Các bước thực hiện UAT:

- Phân tích yêu cầu nghiệp vụ của phần mềm

- Tạo kế hoặc kiểm thử UAT

- Xác định các kịch bản kiểm thử

- Tạo các trường hợp kiểm thử UAT

- Chuẩn bị dữ liệu kiểm thử (phán ánh dữ liệu thực tế nhất có thể)

- Thực hiện kiểm thử

- Ghi nhận kết quả

- Xác nhận chức năng của sản phẩm

\* Thuật toàn tìm kiếm:

+ Toán tử sử dụng “**site:**” nâng cao giúp hạn chế kết quả tìm kiểm cho một tên miền cụ thể

A screenshot of a web page

Description automatically generated

*(Chrome)*

+ Không giới hạn công cụ tìm kiếm do thời điểm các nhà cung cấp thu thập nội dung khác nhau => Kết quả tìm kiếm được khác nhau:

A screenshot of a web page

Description automatically generated

*(Microsoft Edge)*

+ Google cung cấp toán tử tìm kiếm “**cache:**” nâng cao ⬄nhấp vào “Cached” bên cạnh mỗi kết quả Google search => Việc sử dụng toán tử “**site:**” nâng cao và nhấp “Cached” được ưu tiên hơn. Google SOAP Search API hỗ trợ doGetCachedPage và các tin nhắn doGetCachedPageRespond SOAP có liên quan để hỗ trợ truy xuất các điểm trang được lưu trong bộ nhớ đệm

\* Google Hacking Database:

+ Là một danh sách các truy vấn tìm kiếm hữu dụng cho Google. Chúng được đặt trong một số loại:

- Foothodls

- Files containing usernames

- Sensitive Directories

- Web Server Detection

- Vulnerable Files

- Vulnerable Servers

- Error Messages

- Files containing juicy info

- Files containing passwords

- Sensitive Online Shopping Info

\* Cách khắc phục:

+ Xem xét tính nhạy cảm của thông tin thiết kế vào cấu hình trước khi đưa lên mạng

+ Định kỳ xem xét độ nhạy cảm của thông thiết kế và cấu hình hiện tại có được đăng trực tuyến không

2. Fingerprint Web Server:

\* Tổng quan:

- là nhiệm vụ quan trọng để biết phiên bản và loại máy chủ web đang chạy => cho phép người kiểm tra xác định những lỗ hổng đã biết + cách khai thác thích hợp để sử dụng trong quá trình thử nghiệm

- Thông tin có thể được lấy bằng cách gửi yêu cầu cho máy chủ web và phân tích kết quả nhận về. Do mỗi máy phiên bản máy chủ web phản hồi khác nhau => sau khi nhận phản hồi, so sánh kết quả với cơ sở dữ liệu về các loại đã biết.

- Lưu ý: thường phải sử dụng một số lệnh khác nhau để xác định chính xác máy chủ web => Tăng tính chính xác cho dự đoán của người kiểm thử

\* Mục tiêu kiểm thử:

- Tìm phiên bản + loại web server đang chạy để biết được các lỗ hổng và xác định cách khai thác thích hợp trong quá trình kiểm thử

\* Kiểm thử như thế nào?

+ Black Box testing:

- Hình thức đơn giản, cơ bản nhất để xác định máy chủ web là xem trường Máy chủ trong tiêu đề phản hồi HTTP

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Trường server: Apache/1.3.3 (Unix) (Red Hat/Linux)

Một số server khác:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

- Dựa vào Protocol Behavior

- Dựa vào thứ tự các trường HTTP header:

+ Qua các hình trên thấy được thứ các trường trong mỗi phản hồi ở mỗi server là khác nhau

- Yêu cầu không đúng định dạng:

+ Dựa vào kết quả trả về của yêu cầu không đúng định dạng để phân biệt

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Tools:

+ Automated testing: httpprint

+ Online testing: Netcraft

- Có thể truy xuất thông tin về hệ điều hành, máy chủ web đang sử dụng, thời gian hoạt động của máy chủ, chủ sở hữu netblock, lịch sử thay đổi liên quan đến máy chủ web và hệ điều hành

VD: https://digicom .vtc.vn

A group of text on a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

A group of text on a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A white background with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

A white background with a blue and yellow dot

Description automatically generated

3. Review Webserver Metafiles for Information Leakage

* Tổng quan**:**

**-** Liên quan đến việc kiểm tra file robots.txt xem có rò rỉ thông tin về (các đường dẫn) thư mục hoặc thư mục của ứng dụng web hay không.

**-** Hơn nữa, danh sách các thư mục cần bị tránh bởi Spiders, Robots, hoặc Crawlers có thể được tạo dưới dạng phần phụ thuộc cho các đường dẫn thực thi bản đồ thông qua ứng dụng

* Mục tiêu của kiểm tra:

1. Tìm kiếm rò rỉ thông tin về các đường dẫn thư mục hoặc thư mục cửa ứng dụng web
2. Tạo danh sách các thư mục cần tránh bởi Spiders, Robots, Crawlers

* Cách kiểm tra:

\* robots.txt:

\* robots.txt in webroot - with “wget” or “curl”

\* robots.txt in webroot - with rockspider

\* Analyze robots.txt using Google Webmaster Tools

- Chủ sở hữu trang web có thể sử dụng chức năng “Analyze robots.txt” của Google để phân tích trang web như 1 phần của “Google Webmaster Tools”. Công cụ này hỗ trợ kiểm tra như sau:

+ Đăng nhập vào Google Webmaster Tools bằng tài khoản Google

+ Trên bảng điều khiển, viết URL của trang web cần phân tích

+ Chọn giữa các phương pháp có sẵn và làm theo hướng dẫn trên màn hình

* META Tag

- Được đặt trong phần HEAD của mỗi tài liệu HTML và phải nhất quán trên toàn bộ trang web trong trường hợp điểm bắt đầu điểm bắt đầu của /robot/spider/crawler không bắt đầu với một liên kết khác ngoài webroot (liên kết sâu)

- Nếu không có thành phần “<META NAME = “ROBOTS”…>” thì giao thức loại trừ robot mặc định sẽ là INDEX, FOLLOW => Nếu không hợp lệ là NOINDEX, NOFOLLOW

4. Enumerate Applications on Webserver (Liêt kê các ứng dụng trên máy chủ web)

\* Tổng quan:

- Mục tiêu: Liệt kê tất cả các ứng dụng web trong phạm vi tồn tại trên máy chủ web

- Kiểm tra:

+ Black box testing:

\_ Khám phá ứng dụng web là một quá trình nhằm xác định các ứng dụng web trên cơ sở hạ tầng nhất định

\_ Cái sau thường được chỉ định dưới dạng một tập hợp các địa chỉ IP (có thể là một khối mạng), nhưng có thể bao gồm một tập hợp các tên tượng trưng DNS hoặc kết hợp cả hai

\_ Có ba yếu tố ảnh hưởng đến số lượng ứng dụng có liên quan đến một tên DNS (hoặc địa chỉ IP nhất định):

1. **URL cơ sở khác nhau:**

**-**

1. **Cổng không chuẩn:**

**- Cổng mặc định của các ứng dụng web: 80 (http) và 443 (https)**

**- Nhưng có thể đổi cổng (do các ứng dụng web có thể được liên kết với các cổng TCP tùy ý)**

1. **Máy chủ ảo:**

**- DNS cho phép 1 địa chỉ IP duy nhất liên kết với 1 hoặc nhiều tên tượng trưng, các tên không nhất thiết tất cả đều thuộc cùng một miền DNS**

**-** Mối quan hệ 1-n này có thể được phản ánh để phục vụ các nội dung khác nhau bằng cách sử dụng cái gọi là máy chủ ảo.

- Giải quyết:

1. URL không chuẩn:

- Nếu máy chủ web bị định cấu hình sai và cho phép duyệt thư mục, có thể phát hiện ra các ứng dụng này. Máy quét lỗ hổng có thể giúp ích về mặt này

- Sử dụng toán tử “site” để kiểm tra truy vấn cho url

- Một số tùy chọn khác: giao diện người dùng, giao diện quản trị,…

Chú ý: Không có cách nào để xác định đầy đủ sự tồn tại của các ứng dụng web có tên không chuẩn

2. Cổng không chuẩn:

- Sử dụng trình quét **nmap**

- Truy cập URL bằng trình duyệt web; Sử dụng các lệnh GET, HEAD,…

3. Máy chủ ảo:

- Chuyển vùng DNS:

- Truy vấn nghịch đảo DNS

- Dịch vụ IP ngược

- Tìm kiếm trên Google

# 5. Review Webpage Content for Information Leakage:

# \* Tổng quan:

# - Việc các lập trình viên đưa các nhận xét và siêu dữ liệu chi tiết vào mã nguồn của họ => có thể tiết lộ thông tin nội bộ mà những kẻ tấn công tiềm năng không được cung cấp

# 🡺 Cần thực hiện nhận xét và xem xét siêu dữ liệu để xác định xem có bất kỳ thông tin nào bị rò rỉ hay không

# - Đối với các ứng dụng web hiện đại, việc sử dụng JavaScript phía máy khách cho giao diện người dùng đang trở nên phổ biến hơn => nhiều lập trình viên cũng mã hóa cứng thông tin nhạy cảm trong các biến JavaScript ở giao diện người dùng (Khóa API riêng tư, IP nội bộ, tuyến đường nhạy cảm (đường đến các trang quản trị ẩn),…)

# - Từ tệp bản đồ nguồn (source map)

# \* Mục tiêu kiểm tra:

# - Xem lại nhận xét và siêu dữ liệu của trang web để tìm bất kỳ rò rỉ thông tin nào

# - Thu thập các tệp JavaScript và xem lại mã JS để hiểu rõ hơn về ứng dụng và tìm ra bất kỳ rò rỉ thông tin nào

# - Xác định xem có tồn tại tệp bản đồ nguồn hoặc tệp gỡ lỗi giao diện người dùng khác hay không

# \* Kiểm tra ntn?

# - Xem lại nhận xét và siêu dữ liệu của trang web:

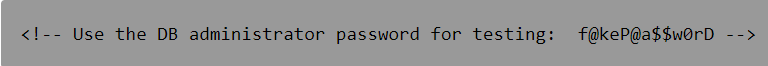
+ Tìm kiếm các nhận xét bắt đầu bằng <!--

+ Kiểm tra mã nguồn HTML để biết các nhận xét chứa thông tin nhạy cảm có thể giúp kẻ tấn công hiểu rõ hơn về ứng dụng. Đó có thể là mã SQL, tên người dùng và mật khẩu, địa chỉ IP nội bộ hoặc thông tin gỡ lỗi

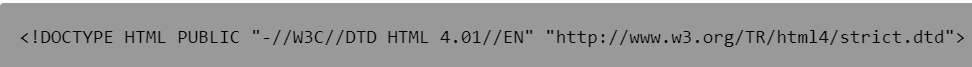
A screenshot of a computer code

Description automatically generated

Hoặc:



+ Kiểm tra thông tin phiên bản HTML để biết số phiên bản hợp lệ và URL Định nghĩa Loại Dữ liệu (DTD):



Strict.dtd – DTD nghiêm ngặt mặc định

Loose.dtd – DTD lỏng lẻo

Frameset.dtd – DTD cho tài liệu frameset

### - Xác định mã JavaScript và thu thập các tệp JavaScript:

+ kiểm tra mã nguồn HTML và tìm mã JavaScript giữa thẻ <script></script>

+ xác định các tệp JavaScript bên ngoài để xem lại mã (tệp có phần mở rộng .js và tên của tệp Javascript thường được đặt trong <src của thẻ <script>)

+ Các giá trị tìm kiếm: khóa API, địa chỉ IP nội bộ, tuyến đường nhạy cảm hoặc thông tin xác thực

A screenshot of a computer code

Description automatically generated

A grey background with black text

Description automatically generated

+ Khi tìm thấy khóa API => kiểm tra các hạn chế của Khóa API được đặt cho mỗi dịch vụ hay theo IP, liên kết giới thiệu HTTP, ứng dụng, SDK,…

VD:

A computer screen shot of a computer code

Description automatically generated

A screen shot of a computer

Description automatically generated

- Xác định các tệp bản đồ nguồn:

+ Thường được tải khi DevTools mở. Người kiểm tra cũng có thể tìm thấy các tệp bản đồ nguồn bằng cách thêm phần mở rộng “.map” sau phần mở rộng của mỗi tệp JavaScript bên ngoài.

- Kiểm tra hộp đen:

+ Kiểm tra các tệp bản đồ nguồn để tìm bất kỳ thông tin nhạy cảm nào có thể giúp kẻ tấn công hiểu rõ hơn về ứng dụng

VD:

A screen shot of a computer code

Description automatically generated

Khi các trang web tải tệp bản đồ nguồn, mã nguồn giao diện người dùng sẽ trở nên dễ đọc và dễ gỡ lỗi hơn

# 6. Identify Application Entry Points:

# \* Tổng quan:

# + Việc liệt kê ứng dụng và bề mặt tấn công là bước quan trọng trước khi thực hiện bất kỳ thử nghiệm, nó cho phép ngưởi thử nghiệm xác định các khu vực có khả năng bị yếu.

# \* Mục tiêu:

# + Xác định các điểm đầu vào và điểm tiêm có thể thông qua phân tích yêu cầu và phản hồi

# \* Kiểm tra như thế nào?

# + Hiểu rõ về ứng dụng, cách thức trình duyệt giao tiếp với nó

# + Chú ý tất cả các yêu cầu HTTP cũng như mọi trường tham số và biểu mẫu được chuyển đến ứng dụng

# + Chú ý đến thời điểm sử dụng GET/POST để truyền tham số cho ứng dụng và khi sử dụng các phương thức khai thác cho dịch vụ RESTful

# Lưu ý: Xem tham số được gửi trong nội dung yêu cầu, cần sử dụng 1 công cụ như proxy chặn. Proxy sẽ theo dõi mọi yêu cầu và phản hồi giữa người thử nghiệm và ứng dụng. Thời điểm này, người kiểm tra thường bẫy mọi yêu cầu và phản hồi để họ có thể thấy chính xác mọi tiêu đề, tham số,… đang được chuyển đến ứng dụng và trả về.

# + Một số chú ý khi kiểm thử Request và Respond (Đối với phương thức GET và POST):

# \* Request:

# - Xác định nơi sử dụng GET và POST

# - Xác định tất cả các tham số được sử dụng trong yêu cầu POST (nằm trong phần nội dung của Request)

# - Trong POST Request, chú ý đến mọi thông số ẩn. Khi một POST được gửi, tất cả các trường biểu mẫu (hồm cả các tham số ẩn) sẽ được gửi trong phần nội dung của thông báo HTTP tới ứng dụng. Chúng thường không được tìm thấy trừ khi sử dụng proxy hoặc xem mã nguồn HTML

# - Xác định các tham số được sử dụng trong yêu cầu GET (⬄ URL), đặc biệt là chuỗi truy vấn (thường sau dấu ?)

# - Xác định tất cả các tham số của chuỗi truy vấn. Chúng thường ở định dạng cặp. Nhiều tham số có thể nằm trong một chuỗi truy vấn, được phân tách nhau bằng dấu &,\~,: hoặc bất kỳ ký tự hoặc mã hóa đặc biệt nào

# - Trong POST Request, một số hoặc tất cả các tham số sẽ cần thiết để thuộc hiện các cuộc tấn công. Người kiểm tra cần xác định tất cả các tham số (ngay cả khi được mã hóa hoặc mã hóa) và xác định những tham số nào được ứng dụng xử lý

# - Chú ý đến bất kỳ tiêu đề loại bổ sung hoặc tùy chỉnh nào thường không được nhìn thấy (VD: debug:false)

# \* Respond:

# - Xác định nơi đặt cookie mới (Set-Cookie header), sửa đổi hoặc thêm vào

# - Xác định nơi có bất kỳ chuyển hướng nào (Mã trạng thái HTTP 3xx), mã trang thái 400 đặc biệt là 403 Forbidden và 500 lỗi máy chủ nội bộ trong các phản hồi thông thường (tức là yêu cầu chưa được sửa đổi)

# - Lưu ý nơi sử dụng bất kỳ tiêu đề nào

# 7. Map Execution Paths Through Application (Ánh xạ đường dẫn thực thi thông qua ứng dụng)

# \* Tổng quan:

# - Trước khi kiểm tra bảo mật, việc hiểu cấu trúc của ứng dụng là tối quan trọng. Nếu không sẽ rất khó để kiểm tra kỹ lưỡng ứng dụng

# \* Mục tiêu:

# - Lập bản đồ ứng dụng mục tiêu và hiểu quy trình công việc chính

# \* Kiểm tra như thế nào?

# - Trong thử nghiệm hộp đen (black box tesing), việc kiểm tra toàn bộ cơ sở mã là cực kỳ khó khăn do người kiểm tra không có quyền xem các đường dẫn mã thông qua ứng dụng và nếu xem được cũng rất tốn thời gian

# => Giải quyết: Ghi lại những đường dẫn mã nào đã được phát hiện và thử nghiệm

# - Một số cách tiếp cận việc kiểm tra và đo lương mức độ bao phủ của mã:

# + Đường dẫn:

# - Là kiểm tra từng đường dẫn thông qua một ứng dụng bao gồm thử nghiệm phân tích giá trị biên và tổ hợp cho từng đường dẫn quyết định. Mặc dù mang lại sự kỹ lưỡng nhưng số lượng đường dẫn phải kiểm tăng lên rất nhanh với mỗi nhánh quyết đinh (theo cấp số nhân)

# + Luồng dữ liệu (hoặc phân tích vết):

# - Là kiểm tra việc gán các biến thông qua tương tác bên ngoài (thường là người dùng). Tập trung vào việc ánh xạ luồng, chuyển đổi và sử dụng dữ liệu trong toàn bộ ứng dụng

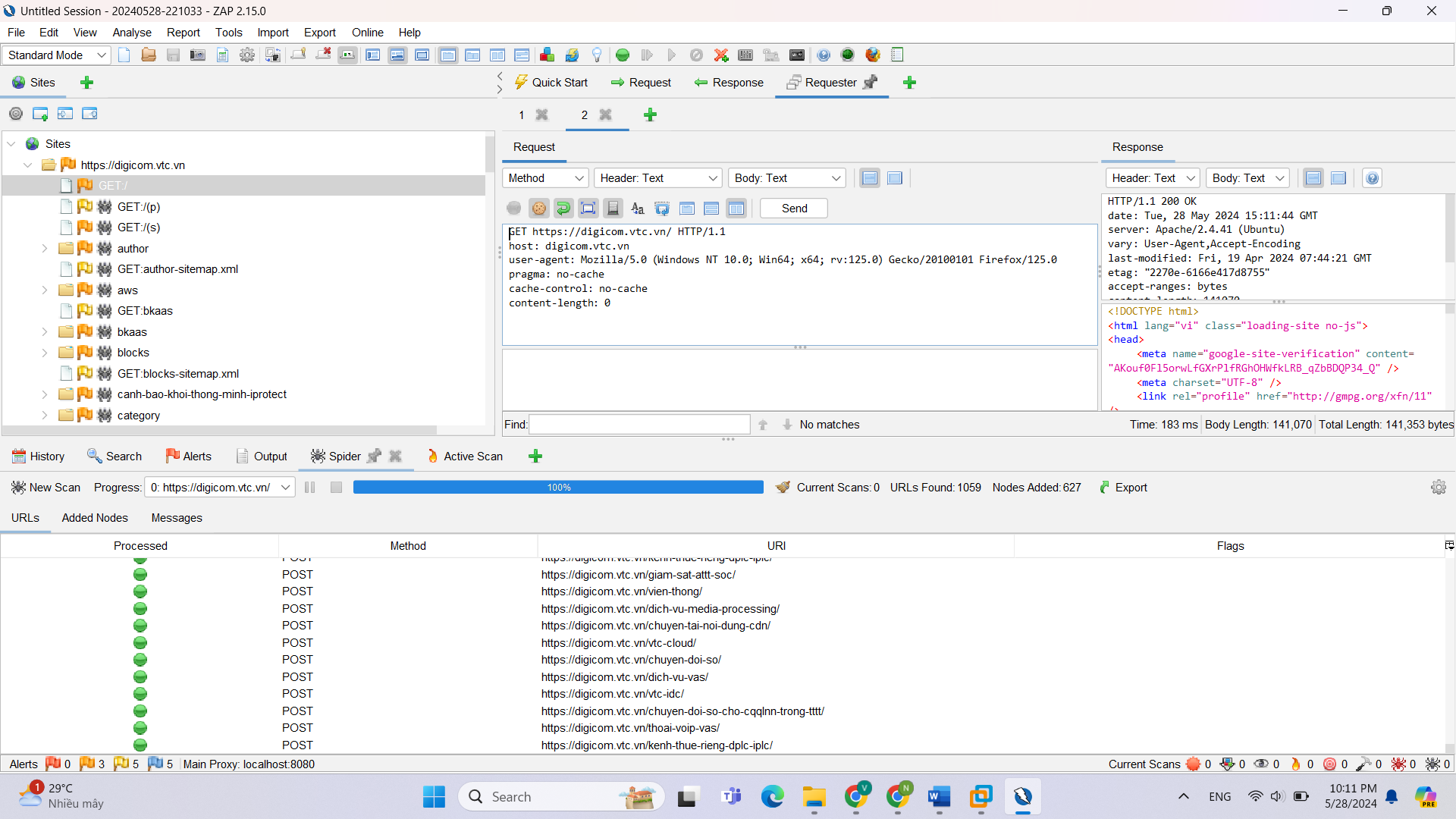
# + Race:

# - Là kiểm tra nhiều phiên bản đồng thời của ứng dụng thao tác cùng một dữ liệu

# - Automatic Spidering:

# + Là công cụ được sử dụng để tự động khám phá các tài nguyên (URL) mới trên một trang web cụ thể.

# + Nó bắt đầu bằng một danh sách các URL cần truy cập được gọi là hạt giống, tùy thuộc vào cách Spider bắt đầu



8. Khung ứng dụng web vân tay:

\* Tổng quan

\* Mục tiêu kiểm thử:

- Fingerprint của các thành phần đang được sử dụng bởi các ứng dụng web

\* Kiểm thử như thế nào?

+ Kiểm thử hộp đen:

* Có một số vị trí phổ biến cần xem xét để xác định các khung thành phần:

+ Tiêu đề HTTP

+ Bánh quy

+ Mã nguồn HTML

+ Các tập tin và thư mục cụ thể

+ Phần mở rộng tập tin

+ Thông báo lỗi

+ Tiêu đề HTTP:

* Xác định khung trang web bằng trường X-Powered-By trong tiêu đề phản hồi HTTP.
* Công cụ: Sử dụng netcat

A grey rectangular object with a white border

Description automatically generated

* Từ trường X-Powered-By, ta biết được khung ứng dụng web là Mono, nhưng phương pháp này không hiệu quả 100% mặc dù nhanh. Có thể vô hiệu hóa X-Powered bằng cấu hình phù hợp (xáo trộn tiêu đề PHP)

VD:

A screenshot of a computer

Description automatically generated

Có thể có nhiều tiêu đề trỏ đến một khung nhất định:

A screenshot of a computer

Description automatically generated

Trường X-Powerwed-By chứa phiên bản PHP, nhưng thực ra X-Generator mới chỉ ra khung thực sự là Swiftlet

=> Giúp người tấn công mở rộng vecto tấn công

+ Cookie:

* Là xác định cookie dành riêng cho khung:

A computer code on a white background

Description automatically generated

* Yêu cầu về cookie CAKEPHP đã được đặt tự động cung cấp về thông tin khung đang được sử dụng.
* Vẫn tồn tại những hạn chế khi dựa vào nhận dạng này => thay đổi được tên của cookie

+ Mã nguồn HTML

* Dựa trên việc tìm kiếm các mẫu nhất định trong mã nguồn HTML.

+ Các tập tin và thư mục cụ thể

* Mỗi thành phần web đều có cấu trúc tệp và thư mục cụ thể riêng trên máy chủ. người ta có thể thấy đường dẫn cụ thể từ nguồn trang HTML nhưng đôi khi chúng không được trình bày rõ ràng ở đó và vẫn nằm trên máy chủ
* Để phát hiện ra chúng, một kỹ thuật được gọi là duyệt bắt buộc hoặc "làm hỏng" (disburting) được sử dụng. Dirbusting là hành vi ép buộc mục tiêu có tên thư mục và tệp đã biết, đồng thời theo dõi các phản hồi HTTP để liệt kê nội dung máy chủ. Thông tin này có thể được sử dụng để tìm các tệp mặc định và tấn công chúng cũng như để lấy dấu vân tay của ứng dụng web.

+ Phần mở rộng tệp:

* URL có thể bao gồm phần mở rộng tệp, điều này cũng có thể giúp xác định nền tảng hoặc công nghệ web
* VD:

A grey rectangular sign with black text

Description automatically generated

Một số phần mở rộng tệp web phổ biến và các công nghệ liên quan:

+ .php -PHP

+ .aspx – Microsoft ASP.NET

+ .jsp – Trang máy chủ Java

\* Mã định danh chung

+ Cookie:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A close up of a sign

Description automatically generated

+ Mã nguồn HTML:

A screenshot of a computer program

Description automatically generated

+ Điểm đánh dấu chung:

* %framework\_name%
* Powered by
* Built upon
* Running

+ Điểm đánh dấu cụ thể

A screenshot of a computer program

Description automatically generated

\* Công cụ

+ WhatWeb

+ Wappalyzer

# 9. Map Application Architecture

# \* Tổng quan:

# - cơ sở hạ tầng web: phức tạp, không đồng nhất => quản lý, xem xét cấu hình thành bước cơ bản trong kiểm thử và triển khai ứng dụng

# - Chỉ 1 lỗ hổng nhỏ => suy yếu bảo mật của cơ sở

# => Giải quyết: thực hiện đánh giá chuyên sâu về cấu hình và các vấn đề bảo mật đã biết. Trước đó cần phải lập bản đồ kiến ​​trúc mạng và ứng dụng để xác định các yếu tố tạo nên hạ tầng, cách tương tác giữa chúng, cách chúng ảnh hưởng đến bảo mật

# \* Mục tiêu:

# - Tạo bản đồ dựa trên nghiên cứu đã tiến hành

# \* Kiểm thử như thế nào?

# - Lập bản đồ kiến trúc ứng dụng:

# + Kiến trúc đơn giản: 1 máy chủ duy nhất phục vụ các chức năng

# + Kiến trúc phức tập: Nhiều máy chủ tham gia gồm proxy, máy chủ web ngoại vi, … Mỗi máy chủ sẽ được sử dụng cho các mục đích khác nhau và có thể được tách biệt trong các mạng khác nhau có tường lửa giữa chúng.

# Tạo ra các mạng khác nhau để truy cập vào máy chủ web => tách biệt giữa các thành phần => không tổn hại đến kiến trúc khi có sự cố

# + Trước tiên người kiểm thử sẽ giả định rằng có một thiết lập đơn giản (1 máy chủ duy nhất) => lấy thông tin từ các thí nghiệm khác và rút ra các yếu tố khác nhau, đặt câu hỏi giả định này và mở rộng bản đồ kiến trúc.

# VD:

# + Có tường lửa bảo vệ máy chủ web không? => Dựa trên kết quả quét mạng với mục tiêu là máy chủ web, phân tích xem liệu các cổng mạng của máy chủ web có đang được lọc ở biên mang hay không (không nhận được trả lời hoặc ICMP không truy cập được)

# + Máy chủ có được kết nối trực tiếp trực tiếp với Internet (tức là trả về các gói RST cho tất cả các cổng không nghe)

# 

# => Từ đó nâng cao hơn: loại tường lửa đang sử dụng (dựa trên kiểm tra gói mạng) || đây có phải tường lửa trạng thái hay bộ lọc danh sách truy cập trên bộ định tuyến || Được cấu hình như thế nào ? || Có phải tường lửa ứng dụng web chính hay không ?

# 

# + Phát hiện proxy ngược:

# - Phân tích biểu ngữ máy chủ web => tiết lộ tồn tại proxy ngược

# - Thu thập respond của máy chủ web, so sánh với cơ sở dữ liệu

# VD: proxy ngược hoạt động như IPS (Hệ thống ngăn xâm nhập) bằng cách chặn các cuộc tấn công đã biết nhắm vào máy chủ web. Nếu máy chủ web được biết là sẽ trả lời bằng thông báo 404 cho một yêu cầu nhắm đến 1 trang không khả dụng và trả về một thông báo một thông báo lỗi khác đối với một số cuộc tấn công web phổ biến như các cuộc tấn công do máy quét lỗ hổng thực hiện

# => dấu hiệu của proxy ngược (hoặc tường lửa cấp ứng dụng) đang lọc yêu cầu và trả về một trang lỗi khác với trang dự kiến.

# + Bộ cân bằng tải:

# - thường các hệ thống này sẽ cân bằng một số cổng TCP/IP nhất định cho nhiều máy chủ dựa trên các thuật toán khác nhau (quay vòng,…) => phát hiện bằng cách kiểm tra nhiều request và so sánh kết quả để xác định xem các request đó có đến cùng một máy chủ web không

# III. Configuration and Deployment Management Testing

# 3.1 Test Network Infrastructure Configuration

# \* Tổng quan:

# + Hạ tầng => phức tạp => kiểm tra => tránh lỗ hổng

# + Các bước thực hiện:

# - các yếu tố tạo nên, cách chúng tương tác với ứng dụng web, cách chúng ảnh hưởng đến bảo mật của ứng dụng đó

# - xem xét các yếu tố của hạ tầng, đảm bảo không chứa bất kỳ lỗ hổng nào đã biết

# - xem xét công cụ quản trị được sử dụng để duy trì

# - xem xét hệ thống xác thực, đảm bảo chúng phục vụ cho ứng dụng và không bị người dùng bên ngoài thao túng, tận dụng quyền truy cập

# - Danh sách các cổng cho ứng dụng phải được duy trì, kiểm soát, thay đổi

# \* Mục tiêu:

# + Xem lại cấu hình của các ứng dụng được đặt trên mạng và xác thực rằng chúng không dễ bị tấn công

# + Xác thực rằng các khung và hệ thống đã sử dụng là an toàn và không dễ gặp phải các lỗ hổng đã biết do phần mềm không được bảo trì hoặc cài đặt và thông tin xác thực mặc định

# \* Kiểm thử ntn ?

# + Các lỗ hổng máy chủ đã biết:

# - Các lỗ hổng được tìm thấy trong các khu vực khác nhau của kiến ​​trúc ứng dụng, có thể là trong máy chủ web hoặc trong cơ sở dữ liệu phía sau, có thể gây tổn hại nghiêm trọng đến chính ứng dụng đó

# - Nếu thông qua thâm nhập mù => xem xét lỗ hổng khó thực hiện => sử dụng công cụ tự động >< mang lại kết quả không đoán trước trên máy chủ web và việc kiểm tra những lỗ hổng khác (không thưc hiện được do thời gian ngừng hoạt động của dịch vụ liên quan nếu kiểm thử thành công)

# - Một số công cụ tự động gán cờ các lỗ hổng dựa trên phiên bản máy chủ web truy xuất => kết quả dương tính giả và âm tính giả:

# + TH1: Nếu phiên bản máy chủ web đã bị quản trị viên trang web cục bộ xóa hoặc che khuất => công cụ quét không gán cờ cho máy chủ là dễ tổn tuonh

# + TH2: nếu nhà cung cấp phần mềm không cập nhật phiên bản máy chủ web khi các lỗ hổng được khắc phục =>công cụ quét sẽ gắn cờ các lỗ hổng không tồn tại

# => TH2 phổ biến hơn vì một số nhà cung cấp hệ điều hành hỗ trợ chuyển các bản vá lỗ hổng bảo mật cho phần mềm mà họ cung cấp trong hệ điều hành nhưng không tải toàn bộ lên phiên bản phần mềm mới nhất (phổ biến ở các bản phân phối GNU/Linux như Debian, Red Hat hoặc SuSE)

# => Trong hầu hết các trường hợp: việc quét lỗ hổng của kiến ​​trúc ứng dụng sẽ chỉ tìm thấy các lỗ hổng liên quan đến các thành phần “bị lộ” của kiến ​​trúc (như máy chủ web) và thường sẽ không thể tìm thấy các lỗ hổng liên quan đến các thành phần không bị lộ trực tiếp (phần phụ trợ xác thực, cơ sở dữ liệu phụ trợ hoặc proxy ngược đang sử dụng)

# 

# - không phải tất cả các nhà cung cấp phần mềm đều tiết lộ các lỗ hổng một cách công khai => những điểm yếu này không được đăng ký trong cơ sở dữ liệu về các lỗ hổng được công khai

# - Thông tin này chỉ được tiết lộ cho khách hàng hoặc được công bố thông qua các bản sửa lỗi không có tư vấn đi kèm => giảm tính hữu ích của các công cụ quét lỗ hổng

# 🡺 việc xem xét các lỗ hổng được thực hiện tốt nhất khi người kiểm tra được cung cấp thông tin nội bộ của phần mềm được sử dụng gồm: các phiên bản và bản phát hành được sử dụng cũng như các bản vá được áp dụng cho phần mềm

# + Công cụ quản trị:

# - Bất kỳ cơ sở hạ tầng máy chủ web nào cũng yêu cầu sự tồn tại của các công cụ quản trị và cập nhật thông tin được ứng dụng sử dụng.

# - Thông tin được ứng dụng sử dụng: nội dung tĩnh (trang web, tệp đồ họa), mã nguồn ứng dụng, cơ sở dữ liệu xác thực người dùng,…

# - Các công cụ quản trị sẽ khác nhau tùy thuộc vào trang web, công nghệ hoặc phần mềm được sử dụng

# VD: quản lý = giao diện quản trị: iPlanet || quản lý = các tệp cấu hình văn bản thuần túy: Apache || sử dụng GUI: web server IIS của Microsoft hoặc ASP.Net

# - Sau khi ánh xạ các giao diện quản trị được sử dụng để quản lý các phần khác nhau của kiến trúc, phải xem lại. Cần:

**+** Xác định các cơ chế kiểm soát quyền truy cập vào các giao diện này và tính nhạy cảm liên quan của chúng. Thông tin này có thể có sẵn trực tuyến.

**+** Thay đổi tên người dùng và mật khẩu mặc định

# - Một số công ty chọn nhờ các bên khác quản lý nội dung do ứng dụng web cung cấp. Công ty bên ngoài có thể chỉ cung cấp 1 phần nội dung/ quản trị hoàn toàn máy chủ web. Người ta thường tìm thấy các giao diện quản trị có sẵn trên Internet trong những trường hợp này

# 3.2 Test Application Platform Configuration

# \* Tổng quan:

# - Việc cấu hình đúng các thành phần đơn lẻ trong kiến trúc => ngăn sai sót ảnh hưởng đến bảo mật

# - Xem xét kiểm tra cấu hình do nhiều hệ thống khác nhau thường được cung cấp các cấu hình chung => không phù hợp với nhiệm vụ mà chúng sẽ thực hiện trên trang web cụ thể mà chúng được cài đặt

# \* Mục tiêu:

# - Đảm bảo rằng các tập tin mặc định và đã biết đã bị xóa

# - Xác thực rằng không còn mã gỡ lỗi hoặc tiện ích mở rộng nào trong môi trường sản xuất

# - Xem lại cơ chế ghi nhật ký được thiết lập cho ứng dụng

# \* Kiểm thử ntn?

# - Kiểm thử hộp đen

# + Các tệp và thư mục mẫu đã biết:

# - Trong cài đặt mặc đinh, nhiều máy chủ web và máy chủ ứng dụng cung cấp các ứng dụng và tệp mẫu vì lợi ích của nhà phát triển + kiểm tra máy chủ có hoạt động bình thường sau khi cài đặt không

# - Máy quét CGI bao gồm danh sách chi tiết các tệp đã biết và mẫu thư mục được cung cấp bởi các máy chủ ứng dụng hoặc web khác nhau và có thể là cách nhanh chóng để xác định xem các tệp này có hiện diện hay không

# + Đánh giá comment:

# -  thực hiện để xác định xem có bất kỳ thông tin nào bị rò rỉ qua comment hay không

# - Việc xem xét này chỉ có thể được thực hiện kỹ lưỡng thông qua phân tích nội dung tĩnh và động của máy chủ web cũng như thông qua tìm kiếm tệp

# - Nội dung được truy xuất này sau đó có thể được tìm kiếm để phân tích bất kỳ comment HTML nào có sẵn trong code

# - Kiểm thử hộp xám (Gray box testing):

# + Đánh giá cấu hình:

# - Cấu hình máy chủ web hoặc máy chủ ứng dụng đóng vai trò quan trọng trong việc bảo vệ nội dung của trang web

# - Cấu hình được đề xuất sẽ khác nhau tùy thuộc vào chính sách của trang web và chức năng mà phần mềm máy chủ sẽ cung cấp

# - Trong hầu hết các trường hợp, phải tuân theo các nguyên tắc cấu hình (do nhà cung cấp phần mềm hoặc các bên bên ngoài cung cấp) để xác định xem máy chủ có được bảo mật đúng cách hay không

# - Một số nguyên tắc chung:

# + Chỉ kích hoạt các mô-đun máy chủ (phần mở rộng ISAPI trong trường hợp IIS) cần thiết cho ứng dụng => giảm bề mặt tấn công vì máy chủ bị giảm kích thước và độ phức tạp khi các mô-đun phần mềm bị vô hiệu hóa || ngăn chặn các lỗ hổng có thể xuất hiện trong phần mềm của nhà cung cấp ảnh hưởng đến trang web nếu chúng chỉ xuất hiện trong các mô-đun đã bị vô hiệu hóa

# + Xử lý lỗi máy chủ (40x hoặc 50x) bằng các trang tùy chỉnh thay vì bằng các trang máy chủ web mặc định ⬄đảm bảo rằng bất kỳ lỗi ứng dụng nào sẽ không được trả lại cho người dùng cuối và không có mã nào bị rò rỉ qua các lỗi này

# + Đảm bảo rằng phần mềm máy chủ chạy với các đặc quyền được giảm thiểu trong hệ điều hành => ngăn lỗi trong phần mềm máy chủ ảnh hưởng trực tiếp đến toàn bộ hệ thống (mặc dù kẻ tấn công có thể nâng cao đặc quyền khi chạy code với tư cách là máy chủ web)

# + Đảm bảo phần mềm máy chủ ghi nhật ký chính xác cả truy cập hợp pháp và lỗi

# + Đảm bảo rằng máy chủ được cấu hình để xử lý đúng tình trạng quá tải và ngăn chặn các cuộc tấn công Từ chối dịch vụ. Đảm bảo rằng máy chủ đã được điều chỉnh hiệu suất đúng cách

# + Không bao giờ cấp quyền truy cập phi quản trị (ngoại trừ NT SERVICE\VMSvc) vào applicationHost.config, redirection.config và management.config (quyền truy cập Đọc hoặc Ghi). Điều này bao gồm Network Service, IIS\_IUSRS, IUSR hoặc bất kỳ danh tính tùy chỉnh nào được nhóm ứng dụng IIS sử dụng.

# + Không bao giờ chia sẻ applicationHost.config, redirection.config và management.config trên mạng. Khi sử dụng Cấu hình được chia sẻ, hãy ưu tiên xuất applicationHost.config sang một vị trí khác

# + Tất cả người dùng đều có thể đọc các tệp .NET Framework machine.config và root web.config theo mặc định. Không lưu trữ thông tin nhạy cảm trong các tệp này nếu nó chỉ dành cho quản trị viên

# + Mã hóa thông tin nhạy cảm mà chỉ tiến trình của nhân viên IIS mới được đọc chứ không phải bởi những người dùng khác trên máy

# + Không cấp quyền truy cập Ghi vào danh tính mà máy chủ Web sử dụng để truy cập vào tệp applicationHost.config. Danh tính này chỉ có quyền truy cập Đọc

# + Sử dụng danh tính riêng để xuất bản applicationhost.config lên phần chia sẻ. Không sử dụng danh tính này để định cấu hình quyền truy cập vào cấu hình được chia sẻ trên máy chủ Web

# + Sử dụng mật khẩu mạnh khi xuất khóa mã hóa để sử dụng với cấu hình chia sẻ

# + Duy trì quyền truy cập hạn chế vào phần chia sẻ chứa cấu hình và khóa mã hóa được chia sẻ. Nếu chia sẻ này bị xâm phạm, kẻ tấn công sẽ có thể đọc và ghi bất kỳ cấu hình IIS nào cho máy chủ Web của bạn, chuyển hướng lưu lượng truy cập từ trang Web của bạn đến các nguồn độc hại và trong một số trường hợp giành quyền kiểm soát tất cả các máy chủ web bằng cách tải mã tùy ý vào nhân viên IIS quá trình

# + cân nhắc việc bảo vệ phần chia sẻ này bằng các quy tắc tường lửa và chính sách IPsec để chỉ cho phép các máy chủ web thành viên kết nối

# - Logging:

# + là một tài sản quan trọng về tính bảo mật của kiến ​​trúc ứng dụng, vì nó có thể được sử dụng để phát hiện các sai sót trong ứng dụng cũng như các cuộc tấn công liên tục từ những người dùng lừa đảo

# + Thường được tạo chính xác bởi web và phần mềm máy chủ khác

# + Mục đích chính của nhật ký ứng dụng là tạo ra đầu ra gỡ lỗi mà lập trình viên có thể sử dụng để phân tích một lỗi cụ thể

# + Trong cả hai trường hợp (nhật ký máy chủ và ứng dụng), một số vấn đề cần được kiểm tra và phân tích dựa trên nội dung nhật ký:

# Nhật ký có chứa thông tin nhạy cảm không?

# Nhật ký có được lưu trữ trong một máy chủ chuyên dụng không?

# Việc sử dụng nhật ký có thể tạo ra điều kiện Từ chối dịch vụ không?

# Chúng được xoay như thế nào? Nhật ký có được lưu giữ đủ thời gian không?

# Nhật ký được xem xét như thế nào? Quản trị viên có thể sử dụng những đánh giá này để phát hiện các cuộc tấn công có chủ đích không?

# Sao lưu nhật ký được bảo tồn như thế nào?

# Dữ liệu đang được ghi lại có được xác thực dữ liệu (độ dài tối thiểu/tối đa, ký tự, v.v.) trước khi được ghi không?

# \*\* Thông tin nhạy cảm trong nhật ký:

# + Nhật ký sự kiện (Events log) thường sẽ chứa dữ liệu hữu ích cho kẻ tấn công (rò rỉ thông tin) hoặc có thể được sử dụng trực tiếp để khai thác:

# - Thông tin gỡ lỗi

# - Dấu vết ngăn xếp

# - Tên người dùng

# - Tên thành phần hệ thống

# - Địa chi IP nội bộ

# - Dữ liệu cá nhân ít nhạy cảm hơn (địa chỉ email, địa chỉ gửi thư và sđt liên quan đến cá nhân)

# - Dữ liệu kinh doanh

# + Danh sách rộng hơn các thông tin nhạy cảm là:

# - Mã nguồn ứng dụng

# - Giá trị nhận dạng phiên

# - Mã thông báo truy cập

# - Dữ liệu cá nhân nhạy cảm và một số dạng thông tin nhận dạng cá nhân (PII)

# - Mật khẩu xác thực

# - Chuỗi kết nối cơ sở dữ liệu

# - Khóa mã hóa

# - Chuỗi kết nối cơ sở dữ liệu

# - Khóa mã hóa

# - Dữ liệu tài khoản ngân hàng hoặc chủ thẻ thanh toán

# - Dữ liệu có mức độ bảo mật cao hơn hệ thống ghi nhật ký được phép lưu trữ

# - Thông tin nhạy cảm về mặt thương mại

# - Thông tin được thu thập bất hợp pháp trong phạm vi quyền hạn liên quan

# - Thông tin mà người dùng đã từ chối thu thập hoặc không đồng ý

# 

# - Vị trí nhật ký:

# + các máy chủ sẽ tạo nhật ký cục bộ về các hành động và lỗi của chúng, tiêu tốn đĩa của hệ thống mà máy chủ đang chạy

# + Nếu máy chủ bị xâm phạm => nhật ký bị xóa => quản trị viên không biết cuộc tấn công diễn ra ntn / nguồn tấn công ở đâu (log zapper)

# => Lưu giữ nhật ký ở vị trí khác máy chủ web => giúp dễ tổng hợp (từ các nguồn khác nhau), phân tích dễ dàng hơn, không ảnh hưởng đến máy chủ

# 

# - Lưu trữ nhật ký:

# + Nhật ký có thể đưa ra tình trạng Từ chối Dịch vụ nếu chúng không được lưu trữ đúng cách. Bất kỳ kẻ tấn công nào có đủ tài nguyên đều có thể tạo ra đủ số lượng yêu cầu để lấp đầy không gian được phân bổ cho các tệp nhật ký, nếu chúng không bị ngăn chặn cụ thể làm như vậy >< nếu máy chủ không được cấu hình đúng cách, các tệp nhật ký sẽ được lưu trữ trong cùng phân vùng đĩa với phân vùng được sử dụng cho phần mềm hệ điều hành hoặc chính ứng dụng ⬄nếu đĩa bị đầy thì hệ điều hành hoặc ứng dụng có thể bị lỗi do không thể ghi vào đĩa

# + Thông thường, nhật ký của hệ thống UNIX sẽ được đặt trong /var. Trong một số trường hợp và để ngăn nhật ký hệ thống bị ảnh hưởng, thư mục nhật ký của chính phần mềm máy chủ phải được lưu trữ trong một phân vùng chuyên dụng

# + Điều này không có nghĩa là nhật ký nên được phép phát triển để lấp đầy hệ thống tệp mà chúng cư trú. Cần theo dõi sự phát triển của nhật ký máy chủ để phát hiện tình trạng này vì nó có thể là dấu hiệu của một cuộc tấn công

# 

# - Xoay vòng nhật ký:

# + Hầu hết các máy chủ (nhưng một số ứng dụng tùy chỉnh) sẽ xoay vòng nhật ký để ngăn chúng lấp đầy hệ thống tệp mà chúng cư trú

# + Tính năng này cần được kiểm tra để đảm bảo rằng:

# - Nhật ký được lưu giữ trong thời gian được xác định trong chính sách bảo mật, không nhiều hơn và không ít hơn

# - Nhật ký được nén sau khi xoay (điều này rất tiện lợi vì điều đó có nghĩa là sẽ có nhiều nhật ký hơn được lưu trữ trên cùng một dung lượng đĩa trống)

# - Quyền hệ thống tệp của tệp nhật ký được xoay giống (hoặc chặt chẽ hơn) với quyền của chính tệp nhật ký

# - Một số máy chủ có thể xoay vòng nhật ký khi chúng đạt đến kích thước nhất định. Nếu điều này xảy ra, phải đảm bảo rằng kẻ tấn công không thể buộc nhật ký xoay để che giấu dấu vết của mình

# - Kiểm soát truy cập nhật ký:

# + Thông tin nhật ký sự kiện sẽ không bao giờ được hiển thị cho người dùng cuối.

# + Ngay cả quản trị viên web cũng không thể xem các nhật ký như vậy vì nó phá vỡ sự phân tách các biện pháp kiểm soát nhiệm vụ

# + Đảm bảo rằng mọi lược đồ kiểm soát truy cập được sử dụng để bảo vệ quyền truy cập vào nhật ký thô và mọi ứng dụng cung cấp khả năng xem hoặc tìm kiếm nhật ký đều không được liên kết với các lược đồ kiểm soát truy cập cho các vai trò người dùng ứng dụng khác.

# + Người dùng chưa được xác thực cũng không thể xem được bất kỳ dữ liệu nhật ký nào

# - Xem lại nhật ký:

# + Để phân tích các cuộc tấn công của máy chủ web, các tệp nhật ký lỗi của máy chủ cần được phân tích. Việc rà soát cần tập trung vào:

# - Thông báo lỗi 40x (không tìm thấy). Một lượng lớn trong số này từ cùng một nguồn có thể là dấu hiệu cho thấy công cụ quét CGI đang được sử dụng để chống lại máy chủ web

# - Thông báo 50x (lỗi máy chủ). Đây có thể là dấu hiệu cho thấy kẻ tấn công đang lạm dụng các phần của ứng dụng khiến ứng dụng bị lỗi ngoài dự kiến

# 3.3 Test File Extensions Handling for Sensitive Information

# \* Tổng quan:

# + Tác dụng của phần mở rộng trong máy chủ web: dễ dàng xác định công nghệ, ngôn ngữ và plugin nào phải được sử dụng để đáp ứng yêu cầu web

# + Việc kiểm tra tiện ích mở rộng thường được sử dụng để xác thực các tệp sẽ được tải lên => kết quả không mong muốn do nội dung không như mong đợi hoặc do việc xử lý tên tệp hệ điều hành không mong muốn

# + Việc xác định cách máy chủ web xử lý các yêu cầu tương ứng với các tệp có phần mở rộng khác nhau có thể giúp hiểu được hành vi của máy chủ web tùy thuộc vào loại tệp được truy cập

# \* Mục tiêu:

# + Xóa các tiện ích mở rộng tệp nhạy cảm hoặc tiện ích mở rộng có thể chứa dữ liệu thô

# + Xác thực rằng không có đường vòng hệ thống nào tồn tại trên bộ quy tắc

# \* Kiểm tra ntn?

# - Duyệt web bắt buộc:

# VD: Người kiểm tra đã xác định được sự tồn tại của tệp có tên connection.inc. Cố gắng truy cập trực tiếp vào nó sẽ trả lại nội dung của nó, đó là:

# A screenshot of a computer code Description automatically generated

# Người kiểm tra xác định sự tồn tại của phần phụ trợ MySQL DBMS và thông tin xác thực (yếu) được ứng dụng web sử dụng để truy cập nó

# - Máy chủ web không bao giờ được trả lại các phần mở rộng tệp sau đây vì chúng có liên quan đến các tệp có thể chứa thông tin nhạy cảm hoặc các tệp không có lý do gì để được cung cấp:

# + .asa

# + .inc

# + .config

# - Các phần mở rộng tệp sau đây có liên quan đến các tệp mà khi truy cập sẽ được trình duyệt hiển thị hoặc tải xuống và chúng không chưa thông tin nhạy cảm:

# + .zip, .tar, .gz, .tgz, .rar

# + .java

# + .txt

# + .pdf

# + .doc, .rtf, .xlsx, .pptx

# + .bak, .old

# - Tải lên tệp

# - Kiểm thử hộp xám (Gray box)

# + Giống như kiểm tra cấu hình của máy chủ web hoặc máy chủ ứng dụng tham gia vào kiến ​​trúc ứng dụng web và xác minh cách chúng được hướng dẫn phân phát các phần mở rộng tệp khác nhau

# 3.4 Review Old Backup and Unreferenced Files for Sensitive Information:

# 3.5 [Enumerate Infrastructure and Application Admin Interfaces](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/05-Enumerate_Infrastructure_and_Application_Admin_Interfaces)

# 3.6 [Test HTTP Methods](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/06-Test_HTTP_Methods)

# \* Tổng quan:

# \* Mục tiêu:

# + Liệt kê các phương thức HTTP được hỗ trợ

# + Kiểm tra bỏ qua kiểm soát truy cập

# + Kiểm tra lỗ hổng XST

# + Kiểm tra kỹ thuật ghi đè phương thức HTTP

# \* Kiểm tra ntn ?

# - Khám phá các phương pháp được hỗ trợ

# + sử dụng nmap với tập lệnh http-methods

# A screen shot of a computer Description automatically generated

# => Các phương pháp được hỗ trợ là POST, OPTIONS, HEAD, GET

# - Kiểm tra phương pháp PUT

# 1. Nắm bắt yêu cầu cơ bản của mục tiêu bằng proxy web

# 2. Thay đổi phương thức yêu cầu thành PUT và thêm tệp test.html rồi gửi yêu cầu đến máy chủ ứng dụng

# A grey rectangular object Description automatically generated

# 3. Nếu máy chủ phản hồi với mã thành công 2XX hoặc chuyển hướng 3XX rồi xác nhận bằng yêu cầu GET cho tệp test.html => Ứng dụng này có lỗ hổng

# + Nếu phương thức PUT không được phép trên URL, thử đường dẫn khác trong hệ thống

# Chú ý: Nếu tải lên web shell thành công, nên ghi đè lên nó hoặc đảm bảo rằng nhóm bảo mật của mục tiêu biết và xóa thành phần đó ngay sau khi chứng minh khái niệm của bạn

# - Kiểm tra bỏ qua kiểm soát truy cập:

# + Tìm một trang để truy cập có hạn chế về bảo mật sao cho yêu cầu GET thường buộc chuyển hướng 302 đến trang đăng nhập hoặc buộc đăng nhập trực tiếp

# + Đưa ra yêu cầu bằng nhiều phương pháp khác nhau như HEAD, POST, PUT, v.v. cũng như các phương thức được tạo tùy ý như BILBAO, FOOBAR, CATS, v.v.

# + Nếu ứng dụng web phản hồi bằng 1 HTTP/1.1 200 OK thì trang không phải là trang đăng nhập

# VD:

# A screenshot of a computer Description automatically generated

# + Nếu hệ thống có vẻ dễ tấn công => thực hiện tấn công giống CSRF để khai thác đây đủ hơn:

# - HEAD /admin/createUser.php?member=myAdmin

# - PUT /admin/changPw.php?member=myAdmin&passwd=foo123&confirm=foo123

# - CATS /admin/groupEdit.php?group=Admins&member=myAdmin&action=add

# => Sử dụng 3 lệnh trên, sửa đổi phù hợp với ứng dụng theo yêu cầu => một người dùng mới được tạo, mật khẩu được chỉ định và người dùng tạo quản trị viên.

# 

# - Kiểm tra Cross-Site Tracing Potential:

# + Phương pháp TRACE nhằm mục đích kiểm tra và gỡ lỗi, hướng dẫn máy chủ web phản ánh tin nhắn đã nhận trở lại máy khách. Phương pháp này có thể được sử dụng để bỏ qua biện pháp bảo vệ và truy cập cookie ngay cả khi thuộc tính HttpOnly được cài đặt

# - Kiểm tra ghi đè phương thức HTTP

# \* Cách khắc phục

# \* Công cụ

# 3.7 [Test HTTP Strict Transport Security](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/07-Test_HTTP_Strict_Transport_Security):

# \* Tổng quan:

# - Tính năng Bảo mật truyền tải nghiêm ngặt HTTP cho phép ứng dụng web thông báo cho trình duyệt thông qua việc sử dụng tiêu đề phản hồi đặc biệt rằng nó sẽ không bao giờ thiết lập kết nối đến các máy chủ miền được chỉ định bằng HTTP không được mã hóa.

# - Tiêu đề bảo mật truyền tải nghiêm ngặt HTTP sử dụng 2 lệnh:

# + max-age: cho biết số giây mà trình duyệt sẽ tự động chuyển đổi tất cả các yêu cầu HTTP sang HTTPS

# + includeSubDomains: chỉ ra rằng tất cả các tên miền phụ có liên quan phải sử dụng HTTPS.

# + preload Unofficial: cho biết rằng (các) tên miền nằm trong (các) danh sách tải trước và các trình duyệt sẽ không bao giờ kết nối nếu không có HTTPS

# => Được hỗ trợ bởi tất cả các trình duyệt chính nhưng không phải là phần chính thức của thông số kỹ thuật

# 

# - VD triển khai tiêu đề HSTS:

# Strict-Transport-Security: max-age=3156000; includeSubDomains

# => Sử dụng tiêu đề này của các ứng dụng web phải được kiểm tra để tìm xem liệu các vấn đề bảo mật sau có thể xảy ra hay không:

# + lưu lượng mạng và truy cập thông tin được truyền qua kênh không được mã hóa

# + kẻ tấn công khai thác kẻ thao túng ở giữa cuộc tấn công vì vấn đề chấp nhận các chứng chỉ không đáng tin cậy

# + Người dùng đã nhập nhầm địa chỉ vào trình duyệt bằng HTTP thay vì HTTPS hoặc người dùng nhấp vào liên kết trong ứng dụng web chỉ ra nhầm việc sử dụng giao thức HTTP

# \* Mục tiêu:

# + Xem lại tiêu đề HSTS và tính hợp lệ của nó

# \* Kiểm tra ntn?

# + Kiểm tra phản hồi của máy chủ thông qua proxy chặn hoặc sử dụng curl:

# A close-up of a grey background Description automatically generated

# + Lệnh curl: Công cụ dòng lệnh để truyền dữ liệu bằng nhiều giao thức khác nhau, phổ biến nhất là HTTP/HTTPS

# + -s : chế độ im lặng (silent mode), tắt các thanh tiến độ và thông báo lỗi

# + -D-: Ghi tiêu đề phản hồi HTTP vào đâu ra tiêu chuẩn (stdout)

# + max-age=31536000: thời gian (tính bằng giây) mà trình duyệt sẽ nhớ rằng trang web này chỉ được truy cập qua HTTPS 31536000 giây ⬄ 1 năm

# 3.8 [Test RIA Cross Domain Policy](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/08-Test_RIA_Cross_Domain_Policy)

# \* Tổng quan:

# - RIA (Rich Internet Applications) đã áp dụng các tệp chính sách crossdomain.xml của Adobe => cho phép truy cập tên miền chéo có kiểm soát vào dữ liệu và mức tiêu thụ dịch vụ bằng cách sử dụng các công nghệ như Oracle Java, Silverlight và Adobe Flash

# => một miền có thể cấp quyền truy cập từ xa vào các dịch vụ của nó từ một miền khác

# - Thường thì các tệp chính sách mô tả các hạn chế truy cập được cấu hình kém

# + Cross-domain policy file là gì ?

# Là tệp chỉ định các quyền mà ứng dụng khách web như Java, Adobe Flash, Adobe Reader, v.v. sử dụng để truy cập dữ liệu trên các tên miền khác nhau

# với Silverlight, Microsoft đã sử dụng một tập hợp con crossdomain.xml của Adobe và còn tạo thêm tệp chính sách tên miền chéo của riêng mình: clientaccesspolicy.xml

# Bất cứ khi nào máy khách web phát hiện thấy tài nguyên phải được yêu cầu từ miền khác, trước tiên nó sẽ tìm tệp chính sách trong miền đích để xác định xem có cho phép thực hiện các yêu cầu tên miền chéo, bao gồm tiêu đề và kết nối dựa trên ổ cắm hay không

# Các tệp chính sách chính được đặt ở thư mục gốc của tên miền. Máy khách có thể được hướng dẫn tải một tệp chính sách khác nhưng nó sẽ luôn kiểm tra tệp chính sách chính trước để đảm bảo rằng tệp chính sách chính cho phép tệp chính sách được yêu cầu

# + Crossdomain.xml so với Clientaccesspolicy.xml:

# Với RIA: hỗ trợ crossdomain.xml

# Với Silverlight: hỗ trợ clientaccesspolicy / crossdomain.xml chỉ định rằng quyền truy cập được cho phép từ bất kỳ miền nào thì Silverlight hoạt động

# Quyền của tệp chính sách:

# + Các tệp chính sách được chấp nhận (Các tệp chính sách chính có thể vô hiệu hóa hoặc hạn chế các tệp chính sách cụ thể)

# + Quyền Sockets

# + Quyền Header

# + Quyền kết nối HTTP/HTTPS

# + Quyền truy cập dựa trên thông tin xác thực mật mã

# VD:

# A screenshot of a computer program Description automatically generated

# Tệp cấu hình trên được coi là dễ dãi vì:

# + dòng <site-control permitted-cross-domain-policies="all"/>: Cho phép tất cả các chính sách cross-domain

# + dòng <allow-access-from domain="\*" secure="false"/>: cho bất kỳ tên miền nào, không bắt buộc secure

# + dòng <allow-http-request-headers-from domain="\*" headers="\*" secure="false"/>: Cho phép mọi tên miền gửi yêu cầu HTTP, bất kỳ tiêu đề HTTP, không yêu cầu kết nối bảo mật

# + Bị lạm dụng ntn?

# Chính sách tên miền chéo quá dễ dãi

# Tạo phản hồi của máy chủ có thể được coi là tệp chính sách tên miền chéo

# Sử dụng chức năng tải tệp lên để tải lên các tệp có thể được coi là tệp chính sách tên miền chéo

# + Tác động:

# - Đánh bại các biện pháp bảo vệ CSRF

# - Đọc dữ liệu bị hạn chế hoặc được bảo vệ bởi các chính sách có nguồn gốc chéo

# \* Mục tiêu:

# - Xem xét và xác nhận các tập tin chính sách

# 

# \* Kiểm tra ntn?

# + Kiểm tra điểm yếu của tệp chính sách RIA:

# - Thử truy xuất các tệp chính sách crossdomain.xml và clientaccesspolicy.xml từ thư mục gốc của ứng dụng và từ mọi thư mục được tìm thấy (VD: https://digicom.vtc.vn/clientaccesspolicy.xml)

# - Sau khi truy xuất tất cả các tệp chính sách, các quyền được phép phải được kiểm tra theo nguyên tắc đặc quyền tối thiểu. Yêu cầu chỉ nên đến từ các miền, cổng hoặc giao thức cần thiết. Nên tránh những chính sách quá dễ dãi. Các chính sách \*trong đó cần được kiểm tra chặt chẽ

# => Kết quả mong đợi:

# + Một danh sách các tập tin chính sách được tìm thấy

# + Danh sách các cài đặt yếu trong chính sách

# \* Công cụ:

# + Nikto

# + OWASP Zed Attack Proxy Project

# + W3af

# 3.9[Test File Permission](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/09-Test_File_Permission):

# \* Tổng quan:

# -

# \* Mục tiêu:

# \* Kiểm tra ntn?

# \* Cách khắc phục:

# \* Công cụ:

# 3.10 [Test for Subdomain Takeover](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/10-Test_for_Subdomain_Takeover)

# 3.11 [Test Cloud Storage](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/11-Test_Cloud_Storage)

# IV. Identity Management Testing

# 4.1 Test Role Definitions

# \* Tổng quan:

# +

# \* Mục tiêu:

# + Xác định và ghi lại các vai trò được ứng dụng sử dụng

# + Cố gắng chuyển đổi, thay đổi hoặc truy cập vai trò khác

# + Xem lại mức độ chi tiết của các vai trò và nhu cầu đằng sau các quyền được cấp

# \* Kiểm tra ntn?

# + Role Identification:

# Người kiểm tra nên bắt đầu bằng cách xác định các vai trò ứng dụng đang được kiểm tra thông qua bất kỳ phương pháp nào sau đây:

# + Tài liệu ứng dụng

# + Hướng dẫn của nhà phát triển hoặc quản trị viên của ứng dụng

# + Ý kiến ​​ứng dụng (Application Comment)

# + Fuzz có thể có các vai trò:

# \_ biến cookie (role = admin, isAdmin=True)

# \_ biến tài khoản (Role: manager)

# \_ thư mục hoặc tập tin ẩn (/admin,/mod,/backups)

# \_ chuyển sang người dùng (admin, bakups)

# + Switching to Available Roles:

# Sau khi xác định các vectơ tấn công có thể xảy ra, người kiểm tra cần kiểm tra và xác thực rằng chúng có thể truy cập các vai trò có sẵn.

# + Review Roles Permissions:

# Sau khi có quyền truy cập vào các vai trò trên hệ thống, người kiểm tra phải hiểu các quyền được cung cấp cho từng vai trò

# Kỹ sư hỗ trợ không được phép thực hiện các chức năng quản trị, quản lý bản sao lưu hoặc thực hiện bất kỳ giao dịch nào thay cho người dùng

# Quản trị viên không nên có toàn quyền trên hệ thống. Chức năng quản trị nhạy cảm nên tận dụng nguyên tắc người kiểm tra hoặc sử dụng MFA để đảm bảo rằng quản trị viên đang tiến hành giao dịch

# + Tools:

# - Burp’s Autorize extension

# - ZAP’s Access Control Testing add-on

# 

# 4.2 Test User Registration Process

# \* Tổng quan:

# \* Mục tiêu:

# + Xác minh rằng các yêu cầu về danh tính để đăng ký người dùng có phù hợp với các yêu cầu về kinh doanh và bảo mật hay không

# + Xác thực quá trình đăng ký

# \* Kiểm tra ntn?

# + Xác minh rằng các yêu cầu về danh tính để đăng ký người dùng có phù hợp với các yêu cầu về kinh doanh và bảo mật hay không

# Có ai đăng ký được quyền truy cập không ?

# Các đăng ký có được con người xem xét kỹ lưỡng trước khi cung cấp hay chúng được tự động cấp nếu đáp ứng các tiêu chí?

# Cùng một người hoặc danh tính có thể đăng ký nhiều lần không?

# Người dùng có thể đăng ký các vai trò hoặc quyền khác nhau không?

# Cần có bằng chứng nhận dạng nào để đăng ký thành công

# Danh tính đã đăng ký có được xác minh không

# + Xác thực quá trình đăng ký:

# Thông tin nhận dạng có thể dễ dàng bị giả mạo hoặc làm giả không?

# Việc trao đổi thông tin nhận dạng có thể bị thao túng trong quá trình đăng ký không

# \* Khắc phục:

# + Triển khai các yêu cầu nhận dạng và xác minh tương ứng với yêu cầu bảo mật của thông tin mà thông tin xác thực bảo vệ

# \* Công cụ:

# + Proxy HTTP

# 4.3 Testing Account Provisioning Process:

# \* Tổng quan:

# \* Mục tiêu:

# + Xác minh tài khoản nào có thể cung cấp tài khoản khác và thuộc loại nào

# \* Kiểm tra ntn?

# Xác định vai trò nào có thể cấp phép cho người dùng và loại tài khoản nào họ có thể cấp phép

# + Có bất kỳ xác minh, kiểm tra và ủy quyền nào đối với các yêu cầu cung cấp không?

# + Có bất kỳ sự xác minh, kiểm tra và ủy quyền nào đối với các yêu cầu hủy cung cấp không?

# + Quản trị viên có thể cấp quyền cho quản trị viên khác hay chỉ người dùng không?

# + Quản trị viên hoặc người dùng khác có thể cấp các tài khoản có đặc quyền lớn hơn đặc quyền của họ không?

# + Quản trị viên hoặc người dùng có thể tự hủy cung cấp không?

# + Các tệp hoặc tài nguyên thuộc quyền sở hữu của người dùng đã hủy cấp phép được quản lý như thế nào? Chúng có bị xóa không? Quyền truy cập có được chuyển giao không?

# \* Công cụ:

# - Proxy HTTP

# 4.4 Testing for Account Enumeration and Guessable User Account

# \* Tổng quan:

# \* Mục tiêu:

# + Xem xét các quy trình liên quan đến nhận dạng người dùng ( ví dụ: đăng ký, đăng nhập, v.v.)

# + Liệt kê người dùng nếu có thể thông qua phân tích phản hồi

# \* Kiểm tra ntn?

# - Thông báo phản hồi HTTP

# + Kiểm tra thông tin xác thực hợp lệ:

# Ghi lại câu trả lời của máy chủ khi bạn gửi ID người dùng hợp lệ và mật khẩu hợp lệ

# + Kiểm tra người dùng hợp lệ với mật khẩu sai

# Bây giờ, người kiểm tra nên thử chèn ID người dùng hợp lệ và mật khẩu sai rồi ghi lại thông báo lỗi do ứng dụng tạo ra

# + Kiểm tra tên người dùng không tồn tại

# Bây giờ, người kiểm tra nên cố gắng chèn ID người dùng không hợp lệ và mật khẩu sai và ghi lại câu trả lời của máy chủ (người kiểm tra phải chắc chắn rằng tên người dùng không hợp lệ trong ứng dụng). Ghi lại thông báo lỗi và câu trả lời của máy chủ

# - Những cách khác để liệt kê người dùng

# + Phân tích mã lỗi nhận được trên mạng đăng nhập:

# Một số ứng dụng web đưa ra mã lỗi hoặc thông báo cụ thể mà chúng tôi có thể phân tích

# + Phân tích URL và chuyển hướng URL:

# Khi người kiểm tra cung cấp ID người dùng và mật khẩu cho ứng dụng web, họ sẽ thấy thông báo cho biết đã xảy ra lỗi trong URL

# + Thăm dò URI:

# Đôi khi máy chủ web phản hồi khác nhau nếu nó nhận được yêu cầu về một thư mục hiện có hay không. Ví dụ: trong một số cổng, mỗi người dùng được liên kết với một thư mục. Nếu người thử nghiệm cố gắng truy cập vào một thư mục hiện có, họ có thể gặp lỗi máy chủ web

# Một số lỗi phổ biến nhận được từ máy chủ web là:

# + 403 Forbidden

# + 404 Not Found

# + Phân tích tiêu đề trang web:

# Người kiểm tra có thể nhận được thông tin hữu ích về Tiêu đề của trang web, nơi họ có thể nhận được mã lỗi cụ thể hoặc thông báo cho biết liệu có sự cố xảy ra với tên người dùng hoặc mật khẩu hay không

# + Phân tích tin nhắn nhận được từ cơ sở khôi phục:

# Khi chúng tôi sử dụng phương tiện khôi phục (tức là chức năng quên mật khẩu), một ứng dụng dễ bị tấn công có thể trả về thông báo cho biết tên người dùng có tồn tại hay không

# + Thông báo lỗi 404:

# Khi chúng tôi yêu cầu một người dùng trong thư mục không tồn tại, không phải lúc nào chúng tôi cũng nhận được mã lỗi 404. Thay vào đó, chúng tôi có thể nhận được “200 ok” kèm theo một hình ảnh, trong trường hợp này, chúng tôi có thể giả định rằng khi chúng tôi nhận được hình ảnh cụ thể thì người dùng không tồn tại. Logic này có thể được áp dụng cho phản hồi của máy chủ web khác; mẹo nhỏ là phân tích tốt các thông báo của máy chủ web và ứng dụng web

# + Phân tích thời gian phản hồi:

# - Ngoài việc xem xét nội dung của các phản hồi, thời gian thực hiện phản hồi cũng cần được xem xét. Đặc biệt khi yêu cầu gây ra sự tương tác với dịch vụ bên ngoài (chẳng hạn như gửi email quên mật khẩu), điều này có thể thêm vài trăm mili giây vào phản hồi. Điều này có thể được sử dụng để xác định xem người dùng được yêu cầu có hợp lệ hay không

# - Đoán người dùng:

# + Trong một số trường hợp, ID người dùng được tạo theo chính sách cụ thể của quản trị viên hoặc công ty. Ví dụ: chúng ta có thể xem người dùng có ID người dùng được tạo theo thứ tự tuần tự

# + Đôi khi tên người dùng được tạo bằng bí danh REALM và sau đó là các số thứ tự (R1001)

# + Các khả năng khác là: - ID người dùng được liên kết với số thẻ tín dụng hoặc ở dạng số chung có mẫu. - ID người dùng được liên kết với tên thật

# - Kiểm tra hộp xám

# + Kiểm tra thông báo lỗi xác thực:

# \_ Xác minh rằng ứng dụng trả lời theo cách tương tự cho mọi yêu cầu của khách hàng tạo ra xác thực không thành công. Đối với vấn đề này, kiểm tra hộp đen và kiểm tra hộp xám có cùng khái niệm dựa trên việc phân tích các thông báo hoặc mã lỗi nhận được từ ứng dụng web

# \* Khắc phục:

# - Đảm bảo ứng dụng trả về các thông báo lỗi chung nhất quán do tên tài khoản, mật khẩu hoặc thông tin xác thực người dùng khác được nhập trong quá trình đăng nhập không hợp lệ

# - Đảm bảo các tài khoản hệ thống mặc định và tài khoản thử nghiệm sẽ bị xóa trước khi đưa hệ thống vào sản xuất (hoặc đưa hệ thống vào mạng không đáng tin cậy)

# \* Công cụ:

# - ZAP

# - curl

# - PERL

# 

# 4.5 Testing for Weak or Unenforced Username Policy:

# \* Tổng quan:

# -

# \* Mục tiêu:

# - Xác định xem cấu trúc tên tài khoản nhất quán có làm cho ứng dụng dễ bị liệt kê tài khoản hay không

# - Xác định xem thông báo lỗi của ứng dụng có cho phép liệt kê tài khoản hay không

# \* Test ntn?

# - Xác định cấu trúc tên tài khoản

# - Đánh giá phản hồi của ứng dụng đối với tên tài khoản hợp lệ và không hợp lệ

# - Sử dụng các phản hồi khác nhau cho tên tài khoản hợp lệ và không hợp lệ để liệt kê tên tài khoản hợp lệ

# - Sử dụng từ điển tên tài khoản để liệt kê tên tài khoản hợp lệ

# \* Cách khắc phục:

# - Đảm bảo ứng dụng trả về các thông báo lỗi chung nhất quán do tên tài khoản, mật khẩu hoặc thông tin xác thực người dùng khác được nhập trong quá trình đăng nhập không hợp lệ

# V. Authentication Testing

# 5.1 Testing for Credentials Transported over an Encrypted Channel (Thông tin xác thực được truyền qua kênh mã hóa)

# \* Tổng quan:

# - Việc kiểm tra việc truyền tải thông tin xác thực sẽ xác minh rằng các ứng dụng web mã hóa dữ liệu xác thực trong quá trình truyền tải

# - Khách hàng có thể gửi hoặc nhận dữ liệu xác thực trong các tương tác sau:

# + Một khách hàng gửi thông tin xác thực để yêu cầu đăng nhập

# + Máy chủ phản hồi đăng nhập thành công bằng mã thông báo phiên

# + Một khách hàng được xác thực sẽ gửi mã thông báo phiên để yêu cầu thông tin nhạy cảm từ trang web

# + Khách hàng gửi mã thông báo đến trang web nếu họ quên mật khẩu

# \* Mục tiêu:

# - Đánh giá xem bất kỳ trường hợp sử dụng nào của trang web hoặc ứng dụng có khiến máy chủ hoặc máy khách trao đổi thông tin đăng nhập mà không mã hóa hay không

# \* Kiểm tra ntn?

# - Nắm bắt lưu lượng truy cập giữa máy khách và máy chủ ứng dụng web cần thông tin xác thực. Kiểm tra thông tin đăng nhập được chuyển trong quá trình đăng nhập và trong khi sử dụng ứng dụng với phiên hợp lệ

# - Trong lưu lượng truy cập đã ghi lại, hãy tìm dữ liệu nhạy cảm bao gồm:

# + Cụm mật khẩu hoặc mật khẩu, thường nằm trong message body

# + Mã thông báo, thường nằm trong cookie

# + Mã đặt lại tài khoản hoặc mật khẩu

# VD: website: <http://testphp.vulnweb.com/login.php>

# A screenshot of a computer Description automatically generated

# - Đăng nhập dưới username test và password test:

# A screenshot of a computer Description automatically generated

# - Sửa đổi Name thàng vuong1 và update

# Kiểm tra trong wireshark:

# A screenshot of a computer Description automatically generated

# A screenshot of a computer Description automatically generated

# => Các thông tin giao tiếp giữa user và website đều không được mã hóa

#### 5.2 [Testing for Default Credentials](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/02-Testing_for_Default_Credentials)

\* Tổng quan:

\* Mục tiêu:

+ Liệt kê các ứng dụng để có thông tin xác thực mặc định và xác thực nếu chúng vẫn tồn tại

+ Xem xét và đánh giá các tài khoản người dùng mới và xem liệu chúng có được tạo bằng bất kỳ giá trị mặc định hoặc mẫu có thể nhận dạng nào hay không

\* Kiểm tra ntn?

+ Kiểm tra thông tin xác thực mặc định của các ứng dụng phổ biến

#### 5.3  [Testing for Weak Lock Out Mechanism](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/03-Testing_for_Weak_Lock_Out_Mechanism)

\* Mục tiêu:

- Đánh giá khả năng của cơ chế khóa tài khoản trong việc giảm thiểu việc đoán mật khẩu bằng brute force

- Đánh giá khả năng chống lại việc mở khóa tài khoản trái phép của cơ chế mở khóa

\* Kiểm tra ntn?

- Cơ chế khóa:

+ Thử đăng nhập không hợp lệ bằng mật khẩu không chính xác nhiều lần:

1. Cố gắng đăng nhập sai mật khẩu 3 lần
2. Đăng nhập thành công với đúng mật khẩu, qua đó thể hiện cơ chế khóa không kích hoạt sau 3 lần xác thực sai
3. Cố gắng đăng nhập sai mật khẩu 4 lần
4. Đăng nhập thành công với đúng mật khẩu, qua đó thể hiện cơ chế khóa không kích hoạt sau 4 lần xác thực sai
5. Cố gắng đăng nhập sai mật khẩu 5 lần
6. Cố gắng đăng nhập bằng mật khẩu chính xác. Ứng dụng trả về “Tài khoản của bạn đã bị khóa.”, qua đó xác nhận rằng tài khoản đã bị khóa sau 5 lần xác thực sai
7. Cố gắng đăng nhập bằng mật khẩu chính xác sau 5 phút. Ứng dụng trả về “Tài khoản của bạn đã bị khóa.”, qua đó thể hiện cơ chế khóa không tự động mở khóa sau 5 phút
8. Cố gắng đăng nhập bằng mật khẩu chính xác 10 phút sau. Ứng dụng trả về “Tài khoản của bạn đã bị khóa.”, qua đó thể hiện cơ chế khóa không tự động mở khóa sau 10 phút
9. Đăng nhập thành công với đúng mật khẩu 15 phút sau, qua đó cho thấy cơ chế khóa tự động mở khóa sau khoảng thời gian từ 10 đến 15 phút

- Mã CAPTCHA:

+ Lỗi của CAPTCHA:

1. Dễ dàng vượt qua thử thách, chẳng hạn như bộ câu hỏi số học hoặc giới hạn
2. CAPTCHA kiểm tra mã phản hồi HTTP thay vì phản hồi thành công
3. Logic phía máy chủ CAPTCHA được mặc định là giải quyết thành công
4. Kết quả thử thách CAPTCHA không bao giờ được xác thực phía máy chủ
5. Trường hoặc thông số nhập CAPTCHA được xử lý thủ công và được xác thực hoặc thoát không đúng cách

#### 5.4 [Testing for Bypassing Authentication Schema](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/04-Testing_for_Bypassing_Authentication_Schema)

\* Kiểm tra buộc duyệt trực tiếp vào bảng điều khiển nội bộ mà không cần đăng nhập

\* Kiểm tra dự đoán ID phiên

\* Kiểm tra giả mạo thông số xác thực

\* Kiểm tra việc chèn SQL trên trang đăng nhập

\* Kiểm tra để có quyền truy cập với sự trợ giúp của ID phiên

\* Kiểm tra có cho phép đăng nhập nhiều lần hay không?

#### 5.5  [Testing for Vulnerable Remember Password](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/05-Testing_for_Vulnerable_Remember_Password)

\* Đảm bảo rằng mật khẩu được lưu trữ được mã hóa

\* Đảm bảo rằng mật khẩu được lưu trữ ở phía máy chủ

#### 5.6 [Testing for Browser Cache Weaknesses](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/06-Testing_for_Browser_Cache_Weaknesses)

\* Đảm bảo cài đặt kiểm soát bộ đệm thích hợp trên các trang nhạy cảm

\* Đảm bảo không có dữ liệu nhạy cảm nào được lưu trữ trong bộ nhớ đệm của trình duyệt

#### 5.7 [Testing for Weak Password Policy](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/07-Testing_for_Weak_Password_Policy)

\* Đảm bảo chính sách mật khẩu được đặt ở mức mạnh

\* Kiểm tra khả năng sử dụng lại mật khẩu

\* Kiểm tra xem người dùng có bị ngăn sử dụng tên người dùng của mình làm mật khẩu không

\* Kiểm tra việc sử dụng các mật khẩu yếu phổ biến

\* Kiểm tra độ dài mật khẩu tối thiểu sẽ được đặt

\* Kiểm tra độ dài mật khẩu tối đa sẽ được đặt

#### 5.8[Testing for Weak Security Question Answer](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/08-Testing_for_Weak_Security_Question_Answer)

\* Kiểm tra độ phức tạp của câu hỏi

\* Kiểm tra brute-force

#### 5.9[Testing for Weak Password Change or Reset Functionalities](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/09-Testing_for_Weak_Password_Change_or_Reset_Functionalities)

\* Kiểm tra thông tin cần thiết để đặt lại mật khẩu

\* Kiểm tra chức năng đặt lại mật khẩu bằng HTTP

\* Kiểm tra tính ngẫu nhiên của mã thông báo đặt lại mật khẩu

\* Kiểm tra tính duy nhất của mã thông báo đặt lại mật khẩu

\* Kiểm tra giới hạn tốc độ trên mã thông báo đặt lại mật khẩu

\* Đảm bảo token phải hết hạn sau khi sử dụng

\* Đảm bảo token phải hết hạn sau một thời gian dài không được sử dụng

#### 5.10[Testing for Weaker Authentication in Alternative Channel](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/10-Testing_for_Weaker_Authentication_in_Alternative_Channel)

\* Kiểm tra xem mật khẩu cũ có được yêu cầu thay đổi không

\* Kiểm tra tính duy nhất của mật khẩu bị quên

\* Kiểm tra thay đổi mật khẩu trống

\* Kiểm tra chức năng đổi mật khẩu bằng HTTP

\* Đảm bảo mật khẩu cũ không hiển thị sau khi thay đổi

\* Đảm bảo các phiên khác đã bị hủy sau khi thay đổi mật khẩu

5.11 Test For Weak Authentication In Alternative Channel

\* Kiểm tra xác thực trên trình duyệt máy tính để bàn

\* Kiểm tra xác thực trên trình duyệt di động

\* Kiểm tra xác thực ở một quốc gia khác

\* Kiểm tra xác thực bằng ngôn ngữ khác

\* Kiểm tra xác thực trên ứng dụng máy tính để bàn

\* Kiểm tra xác thực trên ứng dụng di động

**VI. Authorization Testing**

#### 6.1 [Testing Directory Traversal File Include](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/05-Authorization_Testing/01-Testing_Directory_Traversal_File_Include)

\* Xác định điểm tiêm trên URL

\* Kiểm tra việc đưa tệp cục bộ vào

\* Kiểm tra việc bao gồm tệp từ xa

\* Kiểm tra truyền tải trên tham số URL

\* Kiểm tra truyền tải trên tham số cookie

#### 6.2 [Testing for Bypassing Authorization Schema](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/05-Authorization_Testing/02-Testing_for_Bypassing_Authorization_Schema)

\* Kiểm tra truyền tải bằng mã hóa Base64

\* Kiểm tra truyền tải bằng mã hóa URL

\* Kiểm tra truyền tải bằng bảng mã ASCII

\* Kiểm tra truyền tải bằng mã hóa HTML

\* Kiểm tra truyền tải bằng mã hóa Hex

\* Kiểm tra truyền tải bằng mã hóa nhị phân

\* Kiểm tra truyền tải bằng mã hóa bát phân

\* Kiểm tra truyền tải bằng mã hóa Gzip

#### 6.3 [Testing for Privilege Escalation](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/05-Authorization_Testing/03-Testing_for_Privilege_Escalation)

#### \* Xác định điểm tiêm

#### \* Kiểm tra khả năng vượt qua các biện pháp bảo mật

#### \* Kiểm tra khả năng duyệt bắt buộc

#### \* Kiểm tra IDOR

#### \* Kiểm tra việc giả mạo tham số đối với người dùng có đặc quyền

#### 6.4 [Testing for Insecure Direct Object References](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/05-Authorization_Testing/04-Testing_for_Insecure_Direct_Object_References)

\* Kiểm tra thay đổi thông số ID

\* Test thêm tham số tại điểm cuối

\* Kiểm tra ô nhiễm tham số HTTP

\* Kiểm tra bằng cách thêm phần mở rộng vào cuối

\* Thử nghiệm với các phiên bản API lỗi thời

\* Kiểm tra bằng cách gói ID bằng một mảng

\* Kiểm tra bằng cách gói ID bằng một đối tượng JSON

\* Kiểm tra ô nhiễm tham số JSON

\* Kiểm tra bằng cách thay đổi trường hợp

\* Kiểm tra khả năng truyền tải đường đi

\* Kiểm tra bằng cách thay đổi từ

\* Kiểm tra bằng cách thay đổi phương pháp

**VII. Session Management Testing**

7.1 [Testing for Session Management Schema](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/01-Testing_for_Session_Management_Schema)

#### 7.2 [Testing for Cookies Attributes](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes)

#### 7.3 [Testing for Session Fixation](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/03-Testing_for_Session_Fixation)

#### 7.4 [Testing for Exposed Session Variables](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/04-Testing_for_Exposed_Session_Variables)

#### 7.5 [Testing for Cross Site Request Forgery](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/05-Testing_for_Cross_Site_Request_Forgery)

#### 7.6 [Testing for Logout Functionality](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/06-Testing_for_Logout_Functionality)

#### 7.7 [Testing Session Timeout](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/07-Testing_Session_Timeout)

#### 7.8 [Testing for Session Puzzling](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/08-Testing_for_Session_Puzzling)

**VIII. Input Validation Testing**

#### 8.1 [Testing for Reflected Cross Site Scripting](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/01-Testing_for_Reflected_Cross_Site_Scripting)

\* Black box- 3 giai đoạn:

1. Phát hiện vecto đầu vào ⬄ xác định tất cả các biến do người dùng xác định của ứng dụng web và cách nhập chúng (gồm các đầu vào ẩn: tham số HTTP, dữ liệu POST,…)

2. Phân tích vecto đầu vào: Sử dụng dữ liệu đầu vào => kích hoạt phản hồi từ trình duyệt web => lỗ hổng

3. Kiểm tra tác động

\* Đảm bảo các ký tự này được lọc <>''&””

\* Kiểm tra với chuỗi thoát ký tự

\* Kiểm tra bằng cách thay thế < và > bằng các thực thể HTML < và >

\* Kiểm tra tải trọng bằng cả chữ thường và chữ hoa

\* Kiểm tra để phá vỡ biểu thức tường lửa bằng dòng mới /r/n

\* Kiểm tra bằng mã hóa kép

\* Thử nghiệm với bộ lọc đệ quy

\* Kiểm tra việc chèn thẻ neo không có khoảng trắng

\* Kiểm tra bằng cách thay thế khoảng trắng bằng dấu đầu dòng

\* Kiểm tra bằng cách thay đổi phương thức HTTP

#### 8.2 [Testing for Stored Cross Site Scripting](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/02-Testing_for_Stored_Cross_Site_Scripting)

\* Xác định các tham số đầu vào được lưu trữ sẽ phản ánh ở phía máy khách

\* Tìm kiếm thông số đầu vào trên trang hồ sơ

\* Tìm kiếm thông số đầu vào trên trang giỏ hàng

\* Tìm kiếm thông số đầu vào trên trang upload file

\* Tìm kiếm thông số đầu vào trên trang cài đặt

\* Tìm kiếm thông số đầu vào trên diễn đàn, trang bình luận

\* Kiểm tra tải lên một tệp có tải trọng XSS làm tên tệp

\* Kiểm tra bằng thẻ HTML

#### 8.3 [Testing for HTTP Verb Tampering](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/03-Testing_for_HTTP_Verb_Tampering)

#### 8.4 [Testing for HTTP Parameter Pollution](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/04-Testing_for_HTTP_Parameter_Pollution)

#### 8.5 [Testing for SQL Injection](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection)

\* Test for Oracle

+ Cổng PL/SQL:

- Hoạt động: Giống như proxy (nhận yêu cầu web từ người dùng, chuyển nó đến máy chủ cơ sở dữ liệu được thực thi)

+ Máy chủ web chấp nhận yêu cầu từ máy khách web và xác định xem yêu cầu đó có được PL/SQL Gateway xử lý hay không

+ Cổng PL/SQL xử lý yêu cầu bằng cách trích xuất tên gói, quy trình và các biến được yêu cầu

+ Gói và quy trình được yêu cầu được gói trong một khối PL/SQL ẩn danh và được gửi đến máy chủ cơ sở dữ liệu

+ Máy chủ cơ sở dữ liệu thực hiện quy trình và gửi kết quả trở lại Cổng dưới dạng HTML

+ Cổng gửi phản hồi, thông qua máy chủ web, quay lại máy khách

- URL cho các ứng dụng web PL/SQL: dễ nhận biết (thường bắt đầu bằng chuỗi sau)

+ <http://www.example.com/pls/xyz>

+ <http://www.example.com/xyz/owa>

+ <http://www.example.com/xyz/plsql>

+ Cổng PL/SQL có đang chạy không ?

* Tiêu đề phản hồi phổ biến:

A screenshot of a computer program

Description automatically generated

* Kiểm tra null:

A grey background with red blue and white text

Description automatically generated

A screenshot of a computer

Description automatically generated

* Quyền truy cập gói đã biết:

+ Trên các phiên bản cũ hơn của PL/SQL Gateway, một số gói truy cập được: OWA, HTTP,… (VD OWA\_UTIL chứa SIGNATURE => xuất ra chữ ký PL/SQL trong HTML)

VD:

A close-up of a computer screen

Description automatically generated

Nếu phản hồi là 403 Forbidden => PL/SQL gateway đang chạy

* Kiểm tra cổng PL/SQL để tìm lỗi
* Bypassing the PL/SQL Exclusion List (6 Methods)

\* Test for MySQL

+ Phiên bản và cách thức có thể bị tấn công trên MySQL:

- Từ phiên bản 4.0: UNION

- Từ phiên bản 4.1: Truy vấn con

- Từ Phiên bản 5.0: Thủ tục được lưu trữ, Hàm được lưu trữ và chế độ xem được đặt tên INFORMATION\_SCHEMA

- Từ phiên bản 5.0.2: Trình kích hoạt

**+ Truy vấn đơn**

**+ Truy vấn hỗn hợp**

**+ Thu thập thông tin**

- Lấy dấu vân tay MySQL ⬄ MySQL DBMS có làm csdl không?

Sử dụng ‘/\*! Sql here\*/’

- Phiên bản: @@version / hàm version() / ( /\*!40110 and 1=0\*/ = if(version >= 4.1.10) add 'and 1=0' to the query.)

- Người dùng: hàm USER() / CURRENT\_USER()

- Tên csdl đang được sử dụng database()

+ INFORMATION\_SCHEMA (Từ bản 5.0)

A blue and white information table

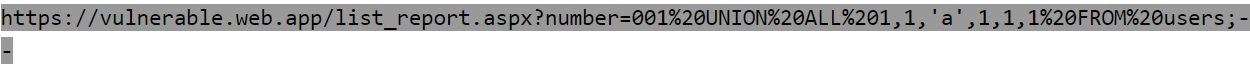
Description automatically generated with medium confidence

\* Test for SQL Server

+ Test trong GET Request:



* Nếu ứng dụng đang sử dụng truy vấn SQL động và chuỗi được thêm vào truy vấn xác thực thông tin đăng nhập của người dùng thì điều này có thể dẫn đến đăng nhập thành công vào ứng dụng



* Kiểm tra số cột

+ Test trong POST Request:

A screenshot of a computer

Description automatically generated

+ Tùy chỉnh xp\_cmdshell:

\*\* xp\_cmdshell thường bị tắt

+ SQL Server làm máy quét cổng: Sử dụng OPENBROWSET

A screenshot of a computer

Description automatically generated

\*\* OPENROWSET được bật theo mặc định trong SQL Server 2000 nhưng bị tắt trong SQL Server 2005

- Tấn công SQL Injection Blind

+ Phép thử và lỗi sai

+ Tấn công thời gian

+ Kiểm tra phiên bản và lỗ hổng

\* Test PostgreSQL

- Xác định công cụ phụ trợ có phải PostgreSQL không? => Sử dụng toán tử ép kiểu ::



- hàm version() => Trả về phiên bản:





- Blind Injection:

Một số hàm lưu ý:

+ Chiều dài chuỗi LENGTH(str)

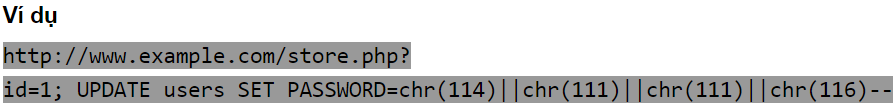
+ Trích xuất chuỗi con từ chuỗi đã cho SUBSTR(str, index, offset)

+ Biểu diễn chuỗi không có dấu ngoặc đơn CHR(104)||CHR(101)||

+ Hàm pg\_sleep(n) cho Time delay()

- Trích dẫn đơn:

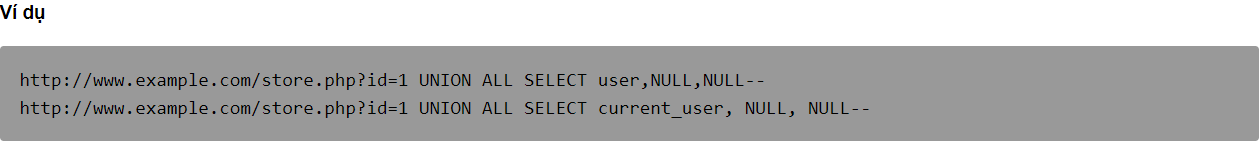
Chuỗi được mã hóa => ngăn dấu nháy đơn thoát ra => sử dụng hàm chr(n): Trả về ký tự ascii(n) tương ứng với n / ascii(n)



- Vector tấn công:

A grey rectangular object with black text

Description automatically generated



A close up of a text

Description automatically generated

A close up of words

Description automatically generated

A screenshot of a computer code

Description automatically generated

\* Test for MS Access

\* Test for NoSQL Injection

\* Test for ORM Injection

- Xác định lớp ORM (Information Gathering)

- Lạm dụng lớp ORM:

+ xác định các CVE liên quan đến thư viện đang được sử dụng

+ Đôi khi, các lớp ORM không được triển khai đúng cách và do đó cho phép người kiểm tra tiến hành [Tiêm SQL](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection) bình thường mà không cần lo lắng về lớp ORM

- Triển khai ORM yếu:

- Lớp ORM dễ bị tổn thương:

A screenshot of a computer program

Description automatically generated

\* Test for Client-Side

### - Xác định cách sử dụng Web SQL DB:

Sử dụng 3 lệnh:

+ openDatabase()

+ transaction()

+ executeSQL()

A close-up of a computer screen

Description automatically generated

### - Tiêm cơ sở dữ liệu SQL vào web

#### 8.6 [Testing for LDAP Injection](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/06-Testing_for_LDAP_Injection)

\* Sử dụng bộ lọc tìm kiếm LDAP

\* Thử LDAP Tiêm để bỏ qua kiểm soát truy cập

#### 8.7 [Testing for XML Injection](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/07-Testing_for_XML_Injection)

\* Kiểm tra xem ứng dụng có đang sử dụng XML để xử lý không

\* Xác định điểm chèn XML bằng siêu ký tự XML

\* Xây dựng payloads XSS trên XML

Các siêu ký tự XML là:

+ ‘ : khi không được làm sạch, ký tự này có thể đưa ra một ngoại lệ trong quá trình phân tích XML, nếu giá trị được chèn sẽ là 1 phần của giá trị thuộc tính trong thẻ

VD:

Giả sử có thuộc tính:



Nếu:



Được khởi tạo sau đó chèn dưới dạng giá trị attrib:



=> Tài liệu XML thu được không được định dạng đúng

+ “ : tương tự như dấu ‘ có thể được sử dụng nếu giá trị thuộc tính được đặt trong dấu ngoặc kép

VD:

A screenshot of a computer code

Description automatically generated

+ Dấu ngoặc đơn góc: > và < : thêm dấu ngoặc đơn góc mở hoặc đóng trong đầu vào của người dùng như:



=> ứng dụng sẽ xây dựng một nút mới

A computer screen shot of a computer code

Description automatically generated

Do sự hiện diện của ‘<’ mở, tài liệu XML kết quả không hợp lệ

+ Thẻ chú thích: <!-- / -->: phần đầu/cuối của chú thích

Khi tạo:



Nút được tạo:

A computer screen shot of a computer code

Description automatically generated

=> XML không hợp lệ

+ Dấu &: Biểu diễn thực thể trong xml. 1 thực thể được ánh xạ tới 1 ký tự trong bộ ký tự Unicode

VD:



+ Dấu phân cách phần CDATA: <!\[CDATA\[ / ]]> : Các ký tự nằm trong phần CDATA không được phân tích cú pháp bằng trình phân tích XML

#### 8.8 [Testing for SSI Injection](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/08-Testing_for_SSI_Injection)

\* Sử dụng Google dorks để tìm SSI

\* Xây dựng RCE trên SSI

\* Xây dựng các mũi tiêm khác trên SSI

\* Kiểm tra việc tiêm SSI vào các trang đăng nhập, trường tiêu đề, liên kết giới thiệu, v.v.

#### 8.9 [Testing for XPath Injection](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/09-Testing_for_XPath_Injection)

Xác định file xml lưu cơ sở dữ liệu, mô phỏng tấn công như SQL Injection

(<https://www.youtube.com/watch?v=6tV8EuaHI9M>)

\* Xác định điểm tiêm XPATH

\* Kiểm tra việc tiêm XPATH

#### 8.10 [Testing for IMAP SMTP Injection](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/10-Testing_for_IMAP_SMTP_Injection)

\* Xác định điểm tiêm IMAP SMTP

\* Hiểu luồng dữ liệu

\* Hiểu cấu trúc triển khai của hệ thống

\* Đánh giá tác động tiêm

#### 8.11 [Testing for Code Injection](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11-Testing_for_Code_Injection)

#### 8.12 [Testing for Command Injection](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/12-Testing_for_Command_Injection)

\* Xác định các điểm phun

\* Tìm từ khóa Command Insert

\* Kiểm tra việc chèn lệnh bằng các dấu phân cách khác nhau

\* Kiểm tra lệnh tiêm với danh sách tải trọng

\* Kiểm tra việc chèn lệnh với các lệnh hệ điều hành khác nhau

#### 8.13 [Testing for Format String Injection](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/13-Testing_for_Format_String_Injection)

\* Xác định các điểm phun

\* Sử dụng các tham số định dạng khác nhau làm payloads

\* Đánh giá tác động tiêm

#### 8.14 [Testing for Incubated Vulnerability](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/14-Testing_for_Incubated_Vulnerability)

#### 8.15 [Testing for HTTP Splitting Smuggling](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/15-Testing_for_HTTP_Splitting_Smuggling)

# IX. Testing for Error Handling

# 1. Testing for Improper Error Handling

# \* Xác định đầu ra lỗi

# \* Phân tích các kết quả đầu ra khác nhau được trả về

# \* Tìm kiếm các lỗi xử lý lỗi phổ biến

# \* Kiểm tra việc xử lý lỗi bằng cách sửa đổi tham số URL

# \* Kiểm tra xử lý lỗi bằng cách tải lên các định dạng tệp không được nhận dạng

# \* Kiểm tra xử lý lỗi bằng cách nhập dữ liệu đầu vào không được nhận dạng

# \* Kiểm tra việc xử lý lỗi bằng cách tạo ra tất cả các lỗi có thể xảy ra

# 2. Testing for Stack Traces

# X. Weak Cryptography Testing

# 1. Testing for Weak Transport Layer Security

# - Kiểm tra các phiên bản:

# Danh sách một số phiên bản giao thức, mật mã và phần mở rộng được TLS hỗ trợ đã cũ, có điểm yếu về mật mã:

# + SSLv2 (DROWN)

# + SSLv3 (POODLE)

# + TLSv1.0 (BEAST)

# + EXPORT ciphers suites (FREAK)

# + NULL ciphers (they only provide authentication).

# + Anonymous ciphers (these may be supported on SMTP servers, as discussed in RFC 7672)

# + RC4 ciphers (NOMORE)

# + CBC mode ciphers (BEAST, Lucky 13)

# + TLS compression (CRIME)

# + Weak DHE keys (LOGJAM)

# - Kiểm tra trên chứng chỉ số:

# + Độ mạnh của khóa: tối thiểu 2048 bit

# + Thuật toán chữ ký ít nhất phải là SHA-256. Không nên sử dụng các thuật toán cũ như MD5 và SHA-1

# \* Đảm bảo chứng thư số không sử dụng MDF và SHA-1

# \* Đảm bảo tính hợp lệ của chứng thư số

# \* Đảm bảo yêu cầu về độ dài khóa tối thiểu

# \* Tìm bộ mật mã yếu

# 2. Testing Padding Oracle

# 3. Testing for Sensitive Information Sent via Unencrypted Channels

# 4. Testing for Weak Encryption

# XI. BUSINESS LOGIC TESTING

# 1. Test for Business Logic

# \* Xác định logic về cách thức hoạt động của ứng dụng

# \* Xác định chức năng của tất cả các nút

# \* Kiểm tra bằng cách thay đổi các giá trị số thành giá trị cao hoặc âm

# \* Kiểm tra bằng cách thay đổi số lượng

# \* Kiểm tra bằng cách sửa đổi các khoản thanh toán

# \* Kiểm tra việc giả mạo thông số

# 2. Test for Malicious File Upload

# \* Kiểm tra việc tải lên tệp độc hại bằng cách tải lên tệp độc hại

# \* Kiểm tra việc tải lên tệp độc hại bằng cách đặt địa chỉ IP của bạn vào tên tệp

# \* Kiểm tra upload file độc ​​hại bằng cách ghi đè từ phải sang trái

# \* Kiểm tra tải lên tệp độc hại bằng tên tệp được mã hóa

# \* Kiểm tra upload file độc ​​hại bằng payload XSS trên tên file

# \* Kiểm tra tải lên tệp độc hại bằng tải trọng RCE trên tên tệp

# \* Kiểm tra upload file độc ​​hại bằng payload LFI trên tên file

# \* Kiểm tra tải lên tệp độc hại bằng tải trọng RFI trên tên tệp

# \* Kiểm tra tải lên tệp độc hại bằng tải trọng SQL trên tên tệp

# \* Kiểm tra việc tải lên tệp độc hại bằng cách tiêm khác vào tên tệp

# \* Kiểm tra việc upload file độc ​​hại bằng cách chèn payload vào bên trong ảnh bằng công cụ bmp.pl

# \* Kiểm tra upload file độc ​​hại bằng cách upload file dung lượng lớn (dẫn tới DOS)

# XII. Client-side Testing:

# 1. Test for DOM-Based Cross Site Scripting

# \* Cố gắng xác định phần chìm của DOM

# \* Xây dựng payloads cho loại phần chìm DOM đó

# 2. Test for URL Redirect

# \* Tìm tham số chuyển hướng URL

# \* Kiểm tra chuyển hướng URL trên các tham số tên miền

# \* Kiểm tra chuyển hướng URL bằng cách sử dụng danh sách tải trọng

# \* Kiểm tra chuyển hướng URL bằng cách sử dụng một từ nằm trong danh sách trắng ở cuối

# \* Kiểm tra chuyển hướng URL bằng cách tạo một tên miền phụ mới giống với tên miền mục tiêu

# \* Kiểm tra chuyển hướng URL bằng XSS

# \* Kiểm tra chuyển hướng URL bằng lỗi URL hồ sơ

# 3. Test for Cross Origin Resource Sharing

# \* Tìm “Access-Control-Allow-Origin” trên phản hồi

# \* Sử dụng mã khai thác HTML CORS để khai thác thêm

# 4. Test for Clickjacking

# \* Đảm bảo tiêu đề “X-Frame-Options” được bật

# \* Khai thác bằng mã HTML iframe cho POC