**BÁO CÁO THỰC HÀNH – LAB 1**

**Tổng quan các lỗ hổng bảo mật web thường gặp**

|  |  |
| --- | --- |
| **Họ và tên** | **MSSV** |
| Phan Nguyễn Nhật Trâm | 22521501 |

**GVHD:** Ngô Đức Hoàng Sơn

**Lớp:** NT213.P12.ANTT

# Bài tập 1: Sử dụng repeater để thực hành Broken Access Control Lab 1

## Minh chứng:

A screenshot of a computer

Description automatically generated

Hình 1. Sử dụng repeat để sửa ‘admin=1’

## Giải thích:

* Chuyển request đến tab Repeat
* Tiến hành sửa request ở tab bên trái, chỉnh sửa admin = 1 và Send
* Kết quả request được hiển thị ở tab response: 0NLY\_F0R\_4DMIN5

# Bài tập 2: Báo cáo lỗ hổng đang được thực hành.

## Tiêu đề: Lỗ hổng Broken Access Control trong trang đăng nhập, cho phép tấn công giành quyền admin.

## Mô tả:

Lỗ hổng Broken Access Control là lỗ hổng trong kiểm soát truy cập, cho phép kẻ tấn công khai thác quyền không chính xác để truy cập các tài nguyên không được phép.

## Kịch bản:

Trong trường hợp này, lỗ hổng xảy ra khi cơ chế kiểm tra quyền hạn của người dùng được xác định khi đăng nhập, sau đó lưu trữ thông tin này ở vị trí do người dùng kiểm soát (admin=0 hoặc admin=1), dẫn đến việc kẻ tấn công có thể có thể sửa đổi giá trị và truy cập tài nguyên của admin.

## Các bước để thực hiện:

1. **Đăng nhập bằng tài khoản người dùng Jack:**

Tên đăng nhập: jack

Mật khẩu: jacktheripper

1. **Phân tích cookie:**

Sau khi đăng nhập, quan sát giá trị của cookie ở tab Request tại tab httpHistory của BurpSuite, có một trường cho quyền hạn của người dùng admin=0.

1. **Chỉnh sửa cookie:**

Thay đổi giá trị từ admin=0 thành admin=1 bằng cách chặn gói tin để chỉnh sửa trực tiếp tại tab Intercept hoặc bằng Repeat của BurpSuite.

1. **Kiểm tra kết quả:**

Gửi yêu cầu với quyền đã chỉnh sửa. Trình duyệt sẽ trả về khóa bí mật hoặc thông báo truy cập với quyền quản trị viên.

## Minh chứng:

A screenshot of a computer

Description automatically generated

*Hình 2. Kết quả tấn công lỗ hổng Broken Access Control*

## Tài liệu hỗ trợ và tham khảo:

<https://portswigger.net/web-security/access-control>

<https://owasp.org/Top10/A01_2021-Broken_Access_Control/>

## Mức độ ảnh hưởng của lỗ hổng:

Năm 2021, 94% ứng dụng đã được kiểm tra lỗ hổng Broken Access Control với tỷ lệ mắc lỗi trung bình là 3,81

Lỗ hổng Broken Access Control có tác động nghiêm trọng đến bảo mật. Kẻ tấn công có thể giành quyền quản trị, truy cập vào các tài nguyên quan trọng, thay đổi dữ liệu, hoặc thực hiện các hành động quản trị khác mà bình thường họ không được phép. Điều này có thể dẫn đến mất quyền kiểm soát hệ thống và rò rỉ thông tin nhạy cảm, mất mát dữ liệu và tài chính

## Khuyến cáo khắc phục:

* Ngoại trừ các tài nguyên công cộng, hãy từ chối theo mặc định.
* Sử dụng cùng một cơ chế kiểm soát truy cập cho toàn bộ ứng dụng để tránh những sai sót và lỗ hổng bảo mật tiềm ẩn. Giảm thiểu việc sử dụng Cross-Origin Resource Sharing (CORS) để hạn chế rủi ro về truy cập trái phép từ các nguồn khác nhau.
* Kiểm soát truy cập phải đảm bảo rằng chỉ người sở hữu bản ghi mới có quyền tạo, đọc, cập nhật hoặc xóa bản ghi đó. Không được giả định rằng tất cả người dùng đều có quyền truy cập vào tất cả dữ liệu…
* Đảm bảo rằng web server không liệt kê thư mục và không để lộ các siêu dữ liệu như .git hay các tệp sao lưu. Điều này sẽ ngăn chặn kẻ tấn công tìm hiểu thông tin nội bộ hoặc truy cập vào dữ liệu không mong muốn.
* Các lỗi truy cập phải được ghi lại và thông báo cho quản trị viên khi cần, đặc biệt là trong trường hợp lỗi lặp lại hoặc nghi ngờ tấn công.
* Giới hạn tốc độ truy cập API
* Các định danh phiên phải được vô hiệu hóa trên server ngay khi người dùng đăng xuất để ngăn chặn việc tái sử dụng. Đối với các mã thông báo JWT không có trạng thái, chúng phải có thời gian tồn tại ngắn để hạn chế cơ hội tấn công. Các mã thông báo tồn tại lâu hơn nên tuân thủ các tiêu chuẩn OAuth để có thể thu hồi quyền truy cập khi cần thiết

# Bài tập 3: Báo cáo lỗ hổng đang được thực hành.

## **Tiêu đề:** Lỗ hổng Cryptographic Failures trong trang đăng nhập, mật mã admin bị tiết lộ.

## Mô tả:

Cryptographic failures xảy ra khi các cơ chế mã hóa không được áp dụng đúng cách hoặc sử dụng các thuật toán, giao thức mã hóa lỗi thời, yếu kém, gây ra rủi ro bảo mật cho dữ liệu.

**Một số lỗ hổng thường gặp trong mã hóa**

* Sử dụng các giao thức không an toàn như HTTP, SMTP, FTP mà không áp dụng biện pháp bảo mật như TLS hoặc STARTTLS, đặc biệt nguy hiểm với dữ liệu trao đổi qua internet.
* Sử dụng thuật toán mã hóa cũ hoặc yếu
* Khóa mã hóa không an toàn
* Thiếu các tiêu đề HTTP hoặc chỉ thị bảo mật từ trình
* Xác thực chứng chỉ không chính xác
* Sử dụng các phương thức mã hóa không an toàn: Các yếu tố như vector khởi tạo (IV) không được xử lý đúng cách, hoặc việc sử dụng chế độ mã hóa như ECB (Electronic Codebook) thay vì các chế độ mã hóa an toàn hơn.
* Không sử dụng hàm dẫn xuất khóa dựa trên mật khẩu như PBKDF2, scrypt, hoặc Argon2
* Các số ngẫu nhiên được sử dụng trong mã hóa không đáp ứng các yêu cầu về tính ngẫu nhiên và khó dự đoán.
* Hàm băm lỗi thời hoặc không phù hợp: Các hàm băm như MD5 hay SHA1 đã lỗi thời, hoặc việc sử dụng hàm băm không mang tính mã hóa trong trường hợp cần bảo mật.
* Phương thức đệm mã hóa lỗi thời như PKCS #1 v1.5
* Các thông báo lỗi mã hóa hoặc thông tin rò rỉ

## Kịch bản:

Trong trường hợp này, lỗ hổng xảy ra khi mật khẩu của admin được mã hoá bằng hàm băm lỗi thời MD5, kẻ tấn công đã lấy được cơ sở dữ liệu tài khoản của người dùng và giải mã mật khẩu thông qua các trang web online trên mạng

## Các bước để thực hiện:

1. **Bản sao cơ sở dũ liệu của người dùng:**

|  |  |
| --- | --- |
| **username** | **password** |
| alex | 9d6edee6ce9312981084bd98eb3751ee |
| admin | c93ccd78b2076528346216b3b2f701e6 |
| rupak | 5ee3547adb4481902349bdd0f2ffba93 |

1. **Dự đoán hàm băm mã hoá**

Dựa vào độ dài 16 bytes và các ký tự bắt đầu từ 0, tối đa là f

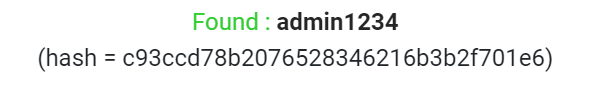
🡪 dự đoán password sử dụng hàm băm MD5

1. **Giải mã mật khẩu bằng trang web online:**

Sử dụng web: <https://www.md5online.org/md5-decrypt.html#google_vignette>

Lần lượt thay các mật khẩu và giải mã, kết quả brute-force là **admin1234**

## Minh chứng:

*Hình 3. Kết quả tấn công brute-force mật khẩu mã hoá bằng hàm băm MD5*

## Tài liệu hỗ trợ và tham khảo:

<https://owasp.org/Top10/A02_2021-Cryptographic_Failures/>

## Mức độ ảnh hưởng của lỗ hổng:

Lỗ hổng Cryptographic Failures có thể gây ra hậu quả nghiêm trọng cho cả cá nhân và tổ chức. Khi các cơ chế mã hóa không được áp dụng đúng cách, dữ liệu nhạy cảm như thông tin cá nhân, mật khẩu, thẻ tín dụng hoặc bí mật kinh doanh có thể bị xâm phạm. Việc truyền hoặc lưu trữ dữ liệu mà không có các biện pháp mã hóa thích hợp có thể dẫn đến:

* Rò rỉ dữ liệu nhạy cảm: Dữ liệu như thông tin thẻ tín dụng hoặc mật khẩu bị lộ ra bên ngoài có thể dẫn đến tổn thất tài chính nghiêm trọng và làm giảm niềm tin của khách hàng vào tổ chức.
* Vi phạm quy định pháp lý: Các quy định như GDPR hay PCI DSS yêu cầu bảo vệ dữ liệu cá nhân và tài chính. Vi phạm các quy định này có thể dẫn đến các khoản phạt tài chính đáng kể, cũng như mất uy tín và danh tiếng của tổ chức.
* Tấn công và xâm nhập hệ thống: Sử dụng các thuật toán hoặc khóa mã hóa yếu có thể giúp kẻ tấn công dễ dàng giải mã dữ liệu, cho phép chúng xâm nhập hệ thống và thực hiện các hành vi phá hoại.
* Tăng nguy cơ tấn công, chẳng hạn như tấn công padding oracle, có thể bị khai thác để làm lộ khóa mã hóa hoặc dữ liệu được bảo vệ

## Khuyến cáo khắc phục:

* Phân loại dữ liệu: Xác định dữ liệu nhạy cảm cần bảo vệ theo các yêu cầu pháp lý và kinh doanh. Chỉ lưu trữ những dữ liệu thực sự cần thiết và loại bỏ dữ liệu không cần thiết.
* Áp dụng mã hóa mạnh: Đảm bảo mã hóa tất cả dữ liệu nhạy cảm ở trạng thái lưu trữ và trong quá trình truyền tải bằng cách sử dụng các thuật toán mã hóa mạnh như AES và các giao thức bảo mật như TLS với các cipher có forward secrecy (FS).
* Quản lý khóa an toàn: Sử dụng các khóa mã hóa mạnh, được tạo ngẫu nhiên từ các CSPRNG (bộ sinh số ngẫu nhiên an toàn về mặt mật mã). Thiết lập quy trình quản lý khóa hiệu quả, bao gồm xoay vòng và lưu trữ khóa một cách an toàn.
* Bảo mật mật khẩu: Lưu trữ mật khẩu bằng các hàm băm thích ứng mạnh và có thêm muối (salt) như Argon2, scrypt, bcrypt hoặc PBKDF2 để tăng tính bảo mật và chống lại các cuộc tấn công brute-force.
* Sử dụng mã hóa có xác thực: Luôn áp dụng các thuật toán mã hóa có xác thực để đảm bảo tính toàn vẹn và bảo mật của dữ liệu, tránh các lỗ hổng trong quá trình mã hóa.
* Không sử dụng các hàm và giao thức mã hóa lỗi thời: Tránh sử dụng các thuật toán mã hóa lỗi thời như MD5, SHA1, hoặc các phương thức đệm như PKCS #1 v1.5. Luôn cập nhật hệ thống với các tiêu chuẩn và phương thức mã hóa mới nhất.
* Kiểm tra cấu hình bảo mật: Thực hiện kiểm tra độc lập và định kỳ để xác minh tính hiệu quả của các thiết lập và cấu hình bảo mật mã hóa, nhằm đảm bảo không có sai sót trong quá trình triển khai.

# Bài tập 4: Báo cáo lỗ hổng đang được thực hành.

## **Tiêu đề:** Lỗ hổng Injection trong trang đăng nhập, kẻ tấn công có thể đăng nhập với quyền admin.

## Mô tả:

94% các ứng dụng đã được thử nghiệm với tỷ lệ xảy ra tối đa là 19%, tỷ lệ trung bình là 3% và 274 nghìn trường hợp xảy ra.

Ứng dụng có thể bị tấn công khi dữ liệu do người dùng cung cấp không được xác thực, lọc, hoặc refresh bởi hệ thống.

Ngoài ra, việc sử dụng dữ liệu nguy hiểm trong các tham số tìm kiếm của hệ thống Object-Relational Mapping (ORM) có thể dẫn đến việc truy xuất các bản ghi nhạy cảm. Khi dữ liệu độc hại được sử dụng trực tiếp hoặc kết hợp vào các truy vấn SQL hoặc lệnh hệ điều hành, các dữ liệu và cấu trúc độc hại có thể xâm nhập vào các truy vấn động, lệnh, hoặc thủ tục lưu trữ.

Một số dạng injection phổ biến bao gồm SQL, NoSQL, OS Command, ORM, LDAP, và Expression Language (EL) hoặc Object Graph Navigation Library (OGNL).

## Kịch bản:

Trong trường hợp này, lỗ hổng xảy ra khi nhập mật khẩu của admin là **anything' OR '1' ='1** kẻ tấn công đã đăng nhập vào hệ thống với vai trò admin

## Các bước để thực hiện:

1. **Nhập thông tin đăng nhập:**

* Tại trường "Username" nhập admin
* Trong trường "Password", thay vì nhập mật khẩu hợp lệ, nhập chuỗi mã tấn công **anything' OR '1'='1**

1. **Kết quả:**

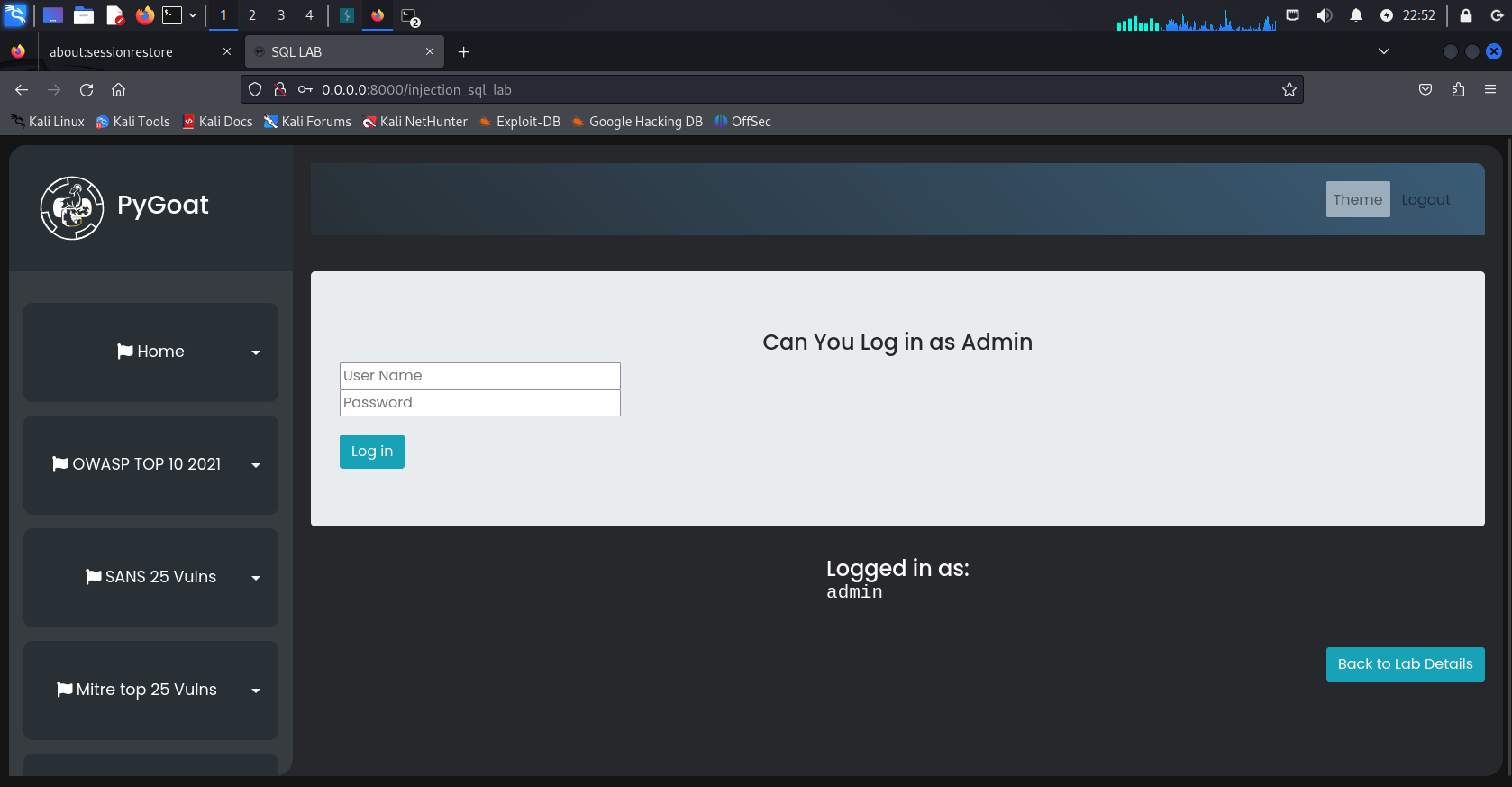
* Yêu cầu đăng nhập sẽ được xử lý với câu lệnh SQL bị injection, dẫn đến việc đăng nhập thành công với quyền admin, mặc dù không cung cấp mật khẩu hợp lệ.

## Minh chứng:

A screenshot of a computer

Description automatically generated

*Hình 4. Nhập mật khẩu là anything' OR '1'='1*



*Hình 5. Kết quả đăng nhập*

## Tài liệu hỗ trợ và tham khảo:

<https://owasp.org/Top10/A03_2021-Injection/>

## Mức độ ảnh hưởng của lỗ hổng:

Lỗ hổng SQL Injection là một trong những lỗ hổng bảo mật nghiêm trọng nhất, đặc biệt khi nó xuất hiện trên trang đăng nhập của hệ thống. Khi bị khai thác, kẻ tấn công có thể thực hiện các hành vi sau:

* Đăng nhập với quyền admin: Cho phép kẻ tấn công truy cập vào các tài nguyên và chức năng nhạy cảm trong hệ thống, dẫn đến việc đánh cắp thông tin cá nhân, dữ liệu quan trọng, hoặc thậm chí chiếm quyền kiểm soát toàn bộ hệ thống.
* Tiết lộ dữ liệu nhạy cảm: Kẻ tấn công có thể truy vấn cơ sở dữ liệu để xem, sửa đổi, hoặc xóa thông tin mà chỉ người quản trị mới có quyền truy cập.
* Thay đổi cấu trúc cơ sở dữ liệu: Việc thực hiện các lệnh SQL không được kiểm soát có thể dẫn đến việc sửa đổi, xóa bảng dữ liệu hoặc các thao tác nguy hiểm khác làm hỏng cơ sở dữ liệu, gây mất mát dữ liệu và thất thoát tài sản
* Phát tán cuộc tấn công quy mô lớn: Kẻ tấn công có thể sử dụng lỗ hổng này để tấn công các phần khác của hệ thống hoặc sử dụng nó như một bước đệm để thực hiện các cuộc tấn công lớn hơn.

## Khuyến cáo khắc phục:

1. Sử dụng API an toàn và chuẩn hóa**:** Ưu tiên sử dụng API an toàn, tránh việc sử dụng trực tiếp các lệnh SQL hoặc kết hợp dữ liệu với các truy vấn động. Các API có tham số hóa giúp giữ dữ liệu và lệnh riêng biệt, ngăn ngừa việc dữ liệu người dùng bị chèn trực tiếp vào các câu lệnh SQL.
2. Sử dụng các công cụ ORM (Object Relational Mapping):Di chuyển sang sử dụng các công cụ ORM giúp quản lý các truy vấn cơ sở dữ liệu một cách an toàn mà không cần phải viết SQL trực tiếp. Tuy nhiên, cần lưu ý các trường hợp ORM vẫn có thể bị injection nếu không được cấu hình cẩn thận.
3. Xác thực đầu vào phía máy chủ:Thực hiện xác thực đầu vào trên máy chủ bằng cách áp dụng các quy tắc xác thực chặt chẽ, chẳng hạn như chỉ cho phép những ký tự hợp lệ. Tuy nhiên, lưu ý rằng cách này không thể phòng ngừa hoàn toàn injection, nhất là khi ứng dụng yêu cầu nhập các ký tự đặc biệt (ví dụ: văn bản trong các API hoặc ứng dụng di động).
4. Thoát ký tự cho các truy vấn động:Nếu bắt buộc phải sử dụng truy vấn động, hãy đảm bảo thoát các ký tự đặc biệt bằng cú pháp thoát đặc thù cho từng loại hệ thống cơ sở dữ liệu, để ngăn việc dữ liệu độc hại xâm nhập vào truy vấn.
5. Giới hạn kết quả truy vấn: Sử dụng các biện pháp kiểm soát như LIMIT, OFFSET trong câu truy vấn SQL để giảm thiểu việc tiết lộ hàng loạt dữ liệu trong trường hợp lỗ hổng bị khai thác.
6. Thực hiện kiểm thử bảo mật ứng dụng: Sử dụng các công cụ kiểm thử bảo mật tự động như SAST, DAST và IAST trong quy trình CI/CD để phát hiện và vá lỗi injection trước khi triển khai sản phẩm ra môi trường sản xuất.

# Bài tập 5: Báo cáo lỗ hổng đang được thực hành.

## **Tiêu đề:** Lỗ hổng Insecure Design trong việc nhận vé xem phim miễn phí, kẻ tấn công có thể lấy toàn bộ vé.

## Mô tả:

Lỗ hổng Insecure Design đại diện cho một danh mục rộng gồm các điểm yếu, được mô tả là "thiếu hoặc thiết kế kiểm soát không hiệu quả." Điều này có nghĩa rằng các biện pháp bảo mật cần thiết để bảo vệ hệ thống khỏi các cuộc tấn công cụ thể không được thiết kế từ đầu dẫn đến những sai sót trong giai đoạn thiết kế và khiến lỗi phát sinh trong quá trình thực thi.

## Kịch bản:

Trong trường hợp này, lỗ hổng nằm ở việc hệ thống không có các biện pháp kiểm tra bảo mật chặt chẽ về số lượng vé và cách chúng được phân phối trong quá trình giao dịch.

## Các bước để thực hiện:

1. **Tạo tài khoản:**

* Tại trường "Username" nhập username bất kỳ
* Trong trường "Password" nhập password bất kỳ

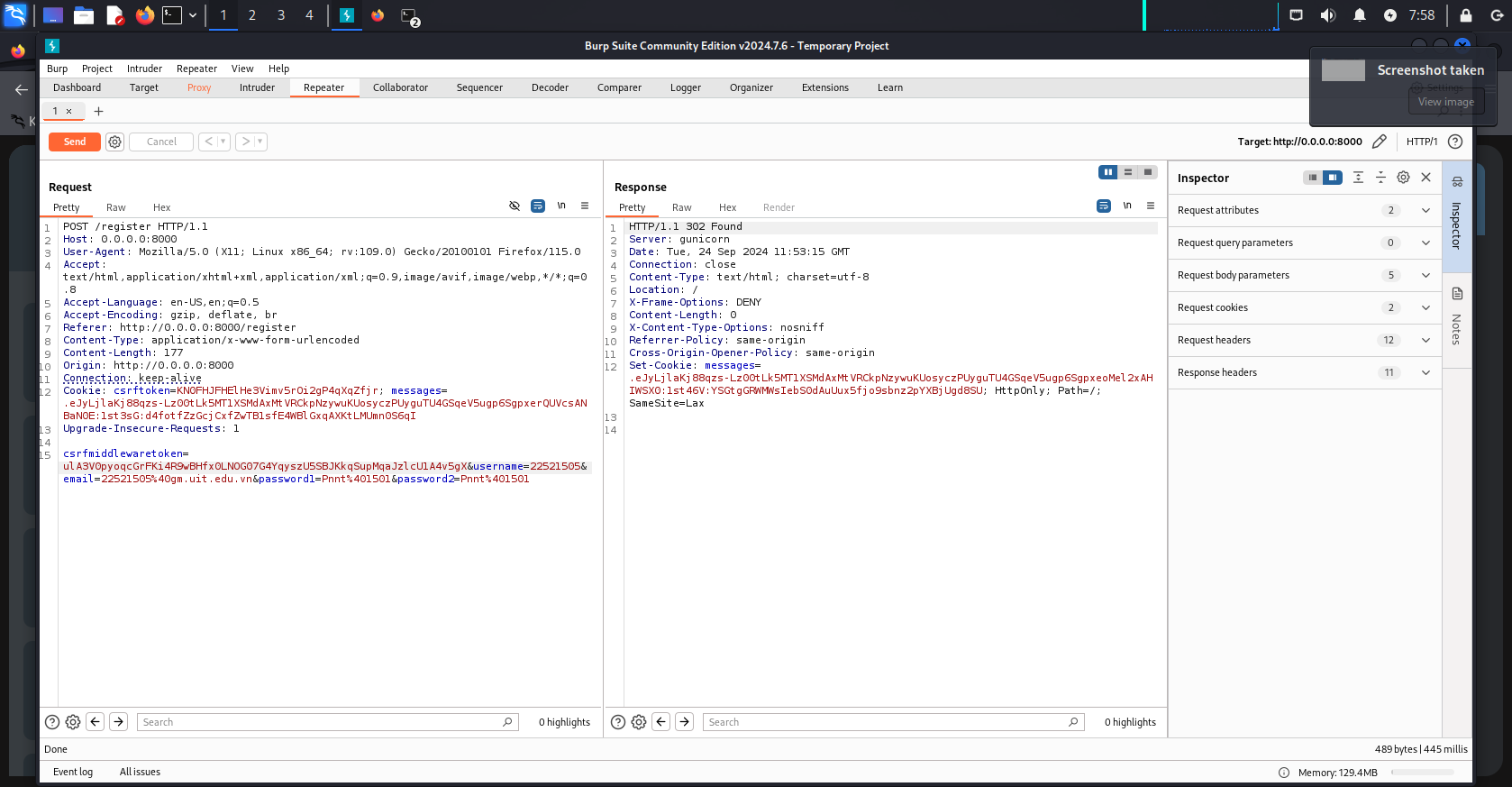
1. **Tạo tài khoản ảo để lấy vé:**

* Với những tài khoản còn lại ta dùng repeat để gửi các yêu cầu tạo tài khoản ảo với những tên khác nhau

1. **Lấy vé bằng các tài khoản ảo**

* Tương tự, ta dùng repeat để gửi yêu cầu đăng nhập bằng các tài khoản ảo. Với từng tài khoản ảo ta click lấy vé

## Minh chứng:





*Hình 6. Tạo tài khoản ảo bằng repeat của BurpSuite*

A screenshot of a computer

Description automatically generated

*Hình 7. Đăng nhập lần lượt các tài khoản ảo*

A screenshot of a computer

Description automatically generated

*Hình 8. Dùng tài khoản ảo để lấy vé*

## Tài liệu hỗ trợ và tham khảo:

<https://owasp.org/Top10/A04_2021-Insecure_Design/>

## Mức độ ảnh hưởng của lỗ hổng:

Lỗ hổng này có thể bị khai thác hàng loạt, dẫn đến việc mất doanh thu lớn cho rạp chiếu phim, cũng như làm giảm trải nghiệm của người dùng chân chính khi họ không thể mua được vé thực tế do hệ thống đã bị thao túng..

## Khuyến cáo khắc phục:

1. Tích hợp ngôn ngữ và kiểm soát bảo mật vào các user story: Tăng cường bảo mật bằng cách tích hợp ngôn ngữ và các biện pháp kiểm soát bảo mật vào quá trình xây dựng yêu cầu phần mềm từ giai đoạn đầu.
2. Thực hiện kiểm tra tính hợp lý ở mỗi tầng của ứng dụng: Kiểm tra tính hợp lý của các hoạt động ở mọi tầng (từ frontend đến backend) nhằm đảm bảo rằng hệ thống phản ứng chính xác với các luồng dữ liệu và không để lộ lỗ hổng.
3. Áp dụng cơ chế xác thực, đảm bảo rằng ưu đãi chỉ áp dụng sau khi toàn bộ vé đã được mua và giao dịch được xác thực.
4. Giới hạn hoặc sử dụng CAPTCHA để ngăn chặn bot và các yêu cầu giả mạo

# Bài tập 6: Báo cáo lỗ hổng đang được thực hành.

## **Tiêu đề:** Lỗ hổng Security Misconfiguration trong việc nhận vé xem phim miễn phí, kẻ tấn công có thể lấy toàn bộ vé.

## Mô tả:

Lỗ hổng xảy ra khi ứng dụng thiếu các biện pháp cấu hình bảo mật cần thiết hoặc các cài đặt được cấu hình không chính xác trên bất kỳ thành phần nào trong hệ thống. Đây là một trong những rủi ro phổ biến nhất, với 90% các ứng dụng được kiểm tra phát hiện ra dạng lỗi này.

Các ứng dụng có thể dễ dàng bị tấn công nếu gặp phải một hoặc nhiều tình huống sau:

* Thiếu các biện pháp bảo mật trong bất kỳ phần nào của hệ thống, hoặc cài đặt quyền hạn không đúng trên các dịch vụ đám mây.
* Kích hoạt hoặc cài đặt các tính năng không cần thiết, chẳng hạn như mở các cổng không dùng, bật dịch vụ hoặc tài khoản không cần thiết, hoặc cấp quyền hạn không phù hợp.
* Tài khoản mặc định và mật khẩu chưa bị vô hiệu hóa hoặc thay đổi, tạo cơ hội cho kẻ tấn công sử dụng các thông tin này để truy cập vào hệ thống.
* Quản lý lỗi không đúng cách, khiến hệ thống hiển thị các thông tin nhạy cảm như stack trace hoặc thông báo lỗi chi tiết cho người dùng, làm lộ thông tin về kiến trúc hệ thống.
* Máy chủ không gửi các header bảo mật hoặc các chỉ thị bảo mật không được cấu hình chính xác.

## Kịch bản:

Trong trường hợp này, lỗ hổng nằm ở việc hệ thống quản lý lỗi không đúng cách, khiến hệ thống thông báo lỗi chi tiết cho người dùng, làm lộ thông tin về kiến trúc hệ thống.

## Các bước để thực hiện:

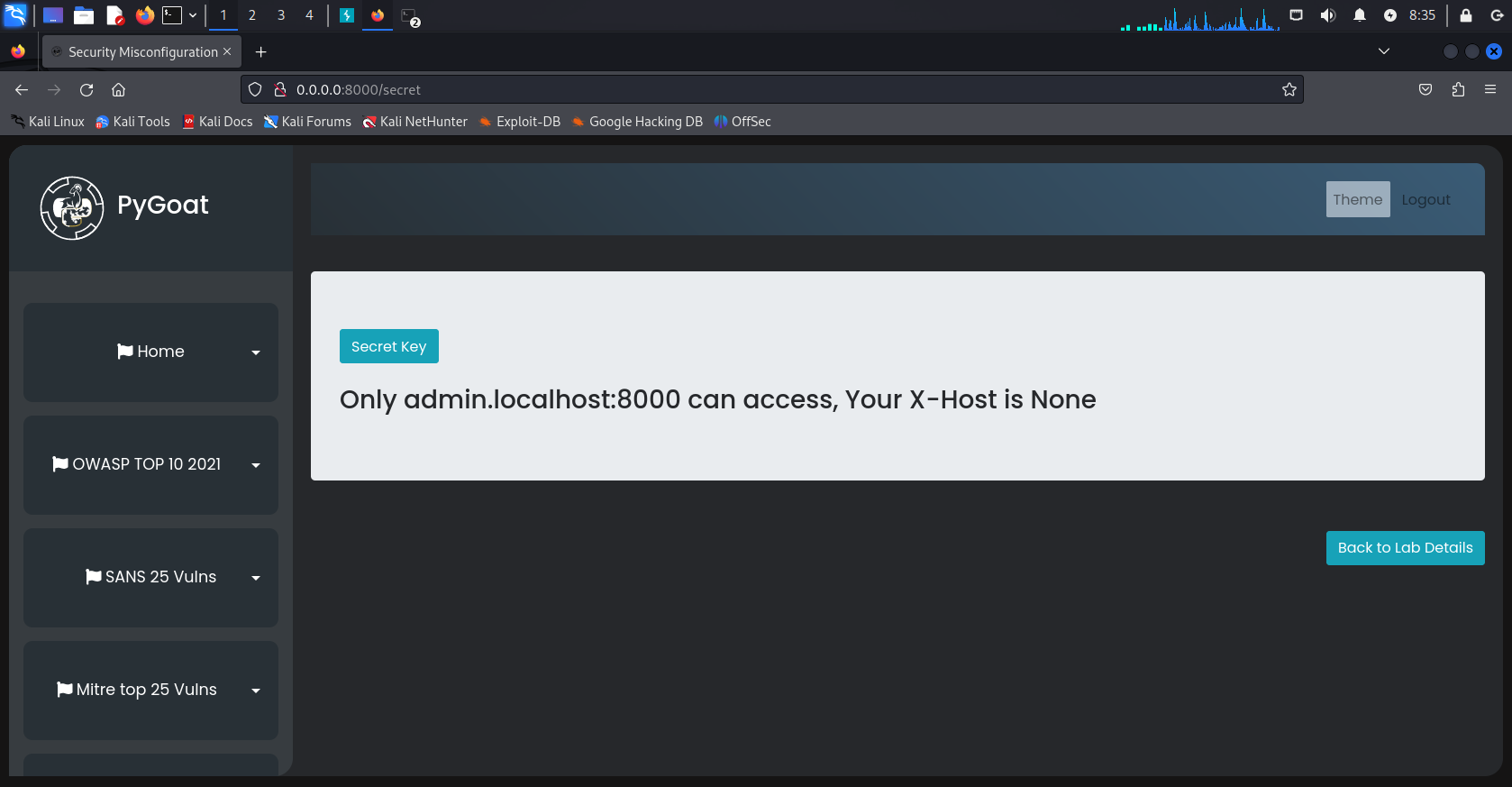
1. **Click nút ‘Secret Ket’ để lấy khoá:**

* Hệ thống thông báo lỗi ‘Only admin.localhost:8000 can access, Your X-Host is None’

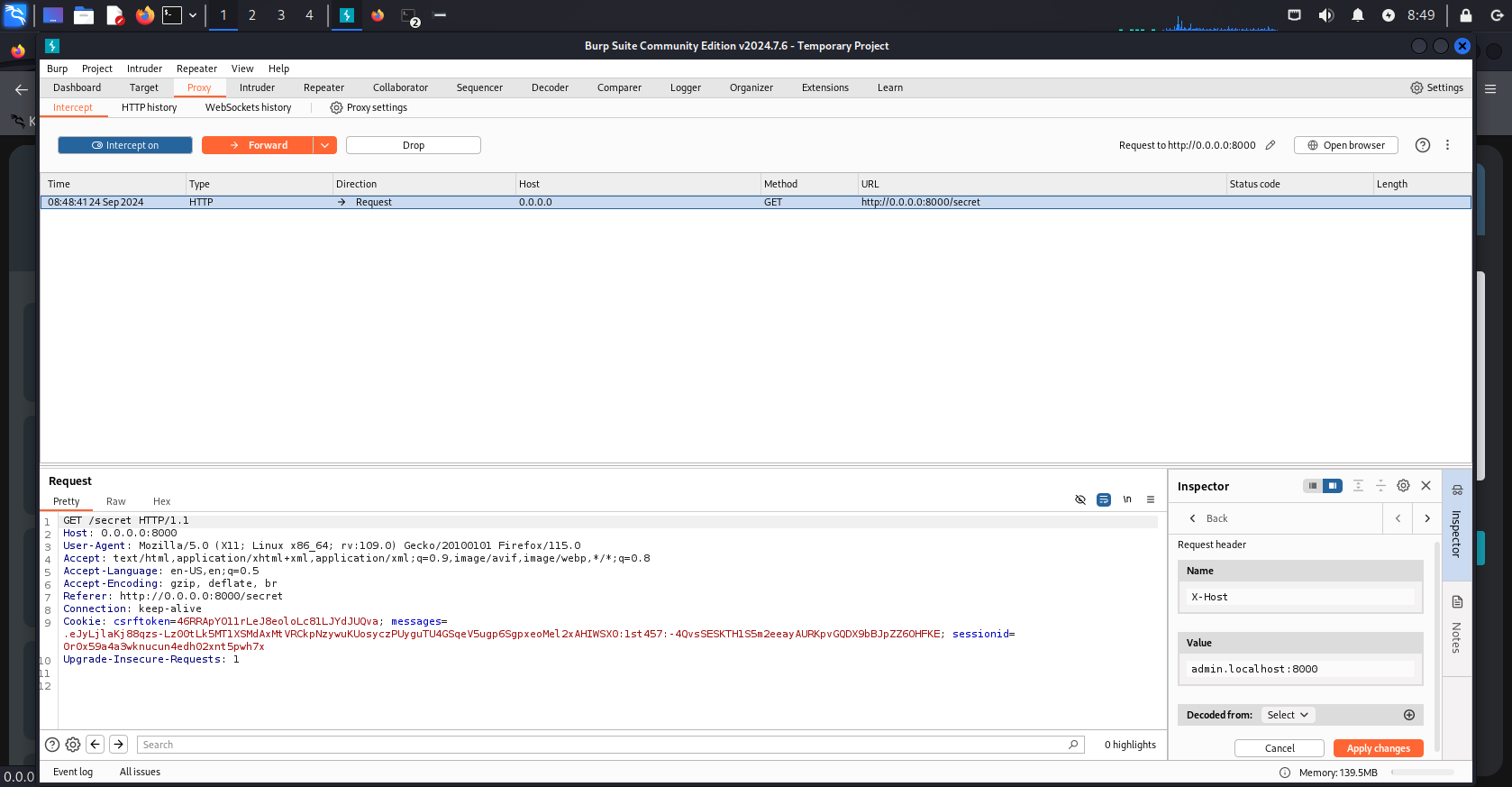
1. **Sử dụng BurpSuite phân tích lỗi**

* Ta bật Intercept và bắt đầu chặn các request
* Click ‘Secret Key’
* Do thông báo lỗi X-Host, nên tại Request Header trong tab Inspect, ta sửa thành X-Host và giá trị của X-Host là ‘admin.localhost:8000’ sau đó Apply Changes và Forward

## Minh chứng:

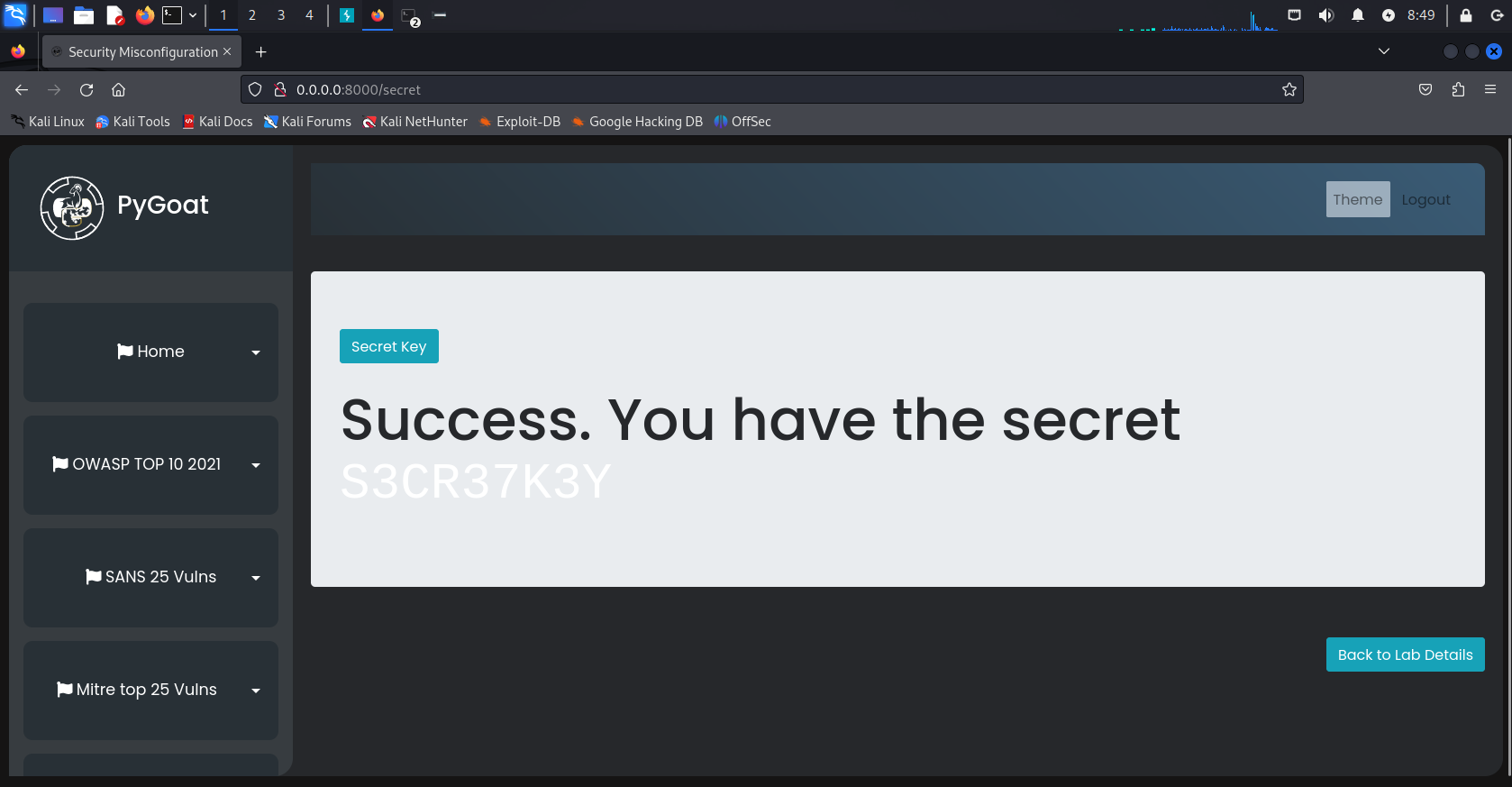


*Hình 9. Khi chưa tấn công*





*Hình 10. Thay đổi Header*



*Hình 11. Lấy secret key thành công*

## Tài liệu hỗ trợ và tham khảo:

<https://owasp.org/Top10/A05_2021-Security_Misconfiguration/>

## Mức độ ảnh hưởng của lỗ hổng:

Lỗ hổng misconfiguration có thể gây ra những hậu quả nghiêm trọng đối với hệ thống và tổ chức, bao gồm:

* Truy cập trái phép: kẻ tấn công truy cập vào các tài nguyên hệ thống, dữ liệu nhạy cảm hoặc thậm chí chiếm quyền điều khiển hệ thống.
* Các tính năng không cần thiết hoặc không bảo mật, nếu không được vô hiệu hóa, có thể trở thành điểm yếu dễ bị khai thác, dẫn đến các cuộc tấn công từ chối dịch vụ (DoS), leo thang đặc quyền, hoặc chiếm đoạt hệ thống.
* Rò rỉ thông tin: Lỗi trong quản lý lỗi hoặc cấu hình không bảo mật của các máy chủ và dịch vụ có thể làm lộ thông tin hệ thống, cho phép kẻ tấn công thu thập thông tin quan trọng để thực hiện các cuộc tấn công tiếp theo.
* Mất mát tài chính và uy tín: Việc hệ thống bị tấn công có thể gây tổn thất tài chính, làm gián đoạn dịch vụ, hoặc phá hủy uy tín của tổ chức.

## Khuyến cáo khắc phục:

1. Sử dụng nền tảng chỉ bao gồm các thành phần cần thiết, loại bỏ các tính năng, framework, tài liệu, và ví dụ không cần thiết hoặc không sử dụng đến.
2. Đảm bảo các cấu hình được xem xét và cập nhật theo các bản vá bảo mật, cập nhật phần mềm và các ghi chú bảo mật. Áp dụng quy trình quản lý bản vá nghiêm ngặt để giữ hệ thống luôn được bảo vệ trước các lỗ hổng mới nhất.
3. Sử dụng các security headers để cung cấp các chỉ thị bảo mật cho các ứng dụng web, giảm thiểu nguy cơ bị tấn công từ phía người dùng.\

**HẾT**