BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN

PENETRATION TESTING

Nguyễn Công Khang – 21DH110770

Phạm Đức Thiên Phúc – 21DH112813

LỜI CẢM ƠN

Cách riêng em xin gửi lời cảm ơn chân thành đến thầy ThS. Phạm Đình Thắng – giảng viên chuyên ngành An ninh mạng đã hướng dẫn em tận tình trong quá trình em làm báo cáo môn học này. Em xin chúc thầy và gia đình luôn mạnh khỏe và bình an.

Cuối cùng em xin gửi lời cảm ơn đến anh chị em, bạn bè cùng những người thân thương đã luôn ủng hộ cách này hay cách khác, đóng góp ý kiến và giúp đỡ trong quá trình học tập, nghiên cứu và hoàn thành báo cáo này.

Em xin hứa luôn sống tốt và cống hiến cho xã hội bằng khả năng của mình khi có thể để đáp đền tình thương của mọi người dành cho em và những gì em đã được lãnh nhận.

Trong học tập tại trường HUFLIT, chắc chắn sẽ không tránh khỏi những thiếu xót. Chính vì vậy, em mong được thầy/ cô nhận xét góp ý để mình ngày càng hoàn thiện hơn.

Em xin chân thành cảm ơn!

LỜI MỞ ĐẦU

Trong thời đại số hóa ngày nay, với sự phát triển không ngừng của công nghệ kỹ thuật, việc bảo vệ thông tin và dữ liệu trở nên càng trở nên quan trọng hơn bao giờ hết. Trong bối cảnh đó, môn học Penetration Testing (Kiểm thử xâm nhập) đóng một vai trò quan trọng trong việc phát hiện và khắc phục các lỗ hổng bảo mật trong hệ thống thông tin.

Báo cáo này được thực hiện nhằm mục đích tổng kết và phản ánh lại quá trình học tập và nghiên cứu trong khóa học Penetration Testing. Trong quá trình này, chúng em đã tiến hành nghiên cứu sâu rộng về các kỹ thuật và phương pháp thực hiện kiểm thử xâm nhập, từ việc phân tích lỗ hổng đến việc triển khai các biện pháp bảo mật phù hợp.

Báo cáo này không chỉ là kết quả của sự nỗ lực của chúng em mà còn là sự kết hợp giữa kiến thức lý thuyết và kinh nghiệm thực tế từ các bài tập và dự án thực hiện trong khóa học. Chúng em đã áp dụng các công cụ và phương pháp thực hành để thực hiện kiểm thử xâm nhập trên các môi trường thực tế, từ đó nhận được cái nhìn sâu sắc hơn về tình hình bảo mật thông tin trong các hệ thống.

Báo cáo này sẽ trình bày chi tiết về quá trình thực hiện kiểm thử xâm nhập, từ việc phân tích môi trường đến việc đánh giá và báo cáo các lỗ hổng được phát hiện. Chúng em hy vọng rằng thông tin trong báo cáo sẽ cung cấp cái nhìn toàn diện về môn học Penetration Testing và đóng góp vào việc nâng cao nhận thức về vấn đề bảo mật thông tin.

Chúng em xin chân thành cảm ơn sự hướng dẫn và hỗ trợ từ giảng viên cũng như sự đồng lòng và cống hiến của tất cả các thành viên trong nhóm trong quá trình thực hiện báo cáo này.

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

TPHCM, Ngày …… tháng …… năm 20……

Giáo viên hướng dẫn

(Ký tên, đóng dấu)

MỤC LỤC

Chương 1: Tổng quan về đề tài

I. Penetration testing

1. Khái niệm

Penetration Testing, còn được gọi tắt là pen test, pentest hay ethical hacking, là cuộc tấn công mô phỏng nhắm vào hệ thống máy tính để kiểm tra những lỗ hổng có thể được khai thác. Trong bảo mật ứng dụng web, Penetration Testing (kiểm thử thâm nhập) thường được sử dụng để tăng cường tường lửa ứng dụng web (Web Application Firewall - WAF).

2. Mục tiêu

- Đánh giá mức độ bảo mật của hệ thống, mạng hoặc ứng dụng.
- Xác định các lỗ hổng bảo mật và rủi ro tiềm ẩn.
- Kiểm tra khả năng phòng thủ và phản ứng của tổ chức trước các cuộc tấn công.

3. Quá trình

Quá trình pen test có thể được chia thành 8 giai đoạn.

PENETRATION TEST

1

DEFINE SCOPE AND OBJECTIVES

- Xác định phạm vi
- Các URLs,điểm cuối,chức năng ứng dụng cụ thể.

2

INFORMATION GATHERING

- Tên miền, IP, tên miền phụ
- Liệt kê DNS (Whois)
- Xác định framework, server (whatweb)

3

MAPPING THE APPLICATION

- Cấu trúc ứng dụng(Burpsuite,ZAP,...).
- Tìm hiểu bằng cách thủ công.

4

IDENTIFY AND TEST FOR COMMON VULNERABILITIES

- SQLInjection
- XSS
- LFI&RFI
- IDOR
- Command Injection

5

AUTHENTICATION & AUTHORIZATIONTESTING

- Bruteforce
- Chính sách khóa tài khoản
- Lỗ hổng đặt lại mật khẩu
- Kiểm tra đặc quyền theo chiều ngang và dọc
- Đảm bảo user không truy cập chức năng/dữ liệu trái phép

6

INPUT VALIDATION TESTING

- Kiểm tra đầu vào người dùng (SQLi, XSS, OS command)
- Xác thực xử lý đầu vào phía client và server

7

BUSINESS LOGIC TESTING

- Xác định trường hợp và tình huống lạm dụng tiềm ẩn
- Kiểm tra lỗ hổng logic không rõ ràng qua quét tự động và thủ công

8

REPORTING

- Mô tả chi tiết về các lỗ hổng.
- Đánh giá rủi ro và tác động tiềm ẩn.
- Khuyến nghị khắc phục.

4. Phương pháp

Phương pháp Penetration Testing (pen test) là một quá trình kiểm tra và đánh giá tính bảo mật của một hệ thống máy tính hoặc mạng bằng cách mô phỏng các kỹ thuật tấn công mà một hacker có thể sử dụng. Mục tiêu của pen test là phát hiện các lỗ hổng bảo mật, thiết lập cơ sở để cải thiện bảo mật và đảm bảo rằng hệ thống hoặc mạng đủ mạnh mẽ để chống lại các cuộc tấn công.

Dưới đây là một số phương pháp thông thường được sử dụng trong pen test:

- **Black Box Testing:** Trong phương pháp này, nhà kiểm thử không có thông tin trước về hệ thống được kiểm tra. Họ phải tiến hành tìm kiếm lỗ hổng bằng cách quét mạng, phân tích ứng dụng và thực hiện các cuộc tấn công giả mạo từ bên ngoài.
- **White Box Testing:** Trái ngược với Black Box Testing, trong White Box Testing, nhà kiểm thử có toàn bộ thông tin về hệ thống, bao gồm mã nguồn và cấu trúc hệ thống. Điều này cho phép họ thực hiện một kiểm tra chi tiết và toàn diện hơn.
- **Gray Box Testing:** Gray Box Testing là sự kết hợp của Black Box và White Box Testing. Trong phương pháp này, nhà kiểm thử có một số thông tin về hệ thống, nhưng không đầy đủ như White Box Testing.
- **Automated Testing:** Sử dụng các công cụ tự động để thực hiện các phân tích và cuộc tấn công, giúp tăng hiệu suất và giảm thời gian kiểm tra. Các công cụ như Nessus, Nmap, Metasploit là những ví dụ phổ biến.
- **Manual Testing:** Kiểm tra thủ công được thực hiện bởi các chuyên gia bảo mật để phát hiện các lỗ hổng mà các công cụ tự động không thể phát hiện được. Điều này đòi hỏi kiến thức sâu rộng về bảo mật và kỹ năng tay nghề.
- **Social Engineering Testing:** Kiểm tra sự tồn tại của lỗ hổng trong các phương tiện giao tiếp xã hội và các chiêu trò lừa đảo như gửi email lừa đảo, cuộc gọi điện thoại giả mạo.
- **Wireless Testing:** Kiểm tra tính bảo mật của mạng không dây bằng cách sử dụng các kỹ thuật như kiểm tra các điểm truy cập không dây, mã hóa mạng và xâm nhập vào hệ thống qua các thiết bị không dây.

5. OWASP TOP 10

- **Injection**

Injection attack xảy ra khi dữ liệu không đáng tin cậy được gửi đến trình thông dịch mã (code interpreter) thông qua việc điền các form (biểu mẫu) hoặc một số dữ liệu khác gửi đến ứng dụng web. Ví dụ, kẻ tấn công có thể nhập SQL database

code vào một biểu mẫu yêu cầu username ở dạng plaintext. Nếu việc điền các biểu mẫu đó không được bảo mật đúng cách, điều này sẽ dẫn đến việc SQL code đó được thực thi. Đây được gọi là một cuộc tấn công SQL injection.

Các cuộc tấn công injection có thể được ngăn chặn bằng cách xác thực và / hoặc "khử trùng" dữ liệu do người dùng gửi. (Xác thực nghĩa là từ chối các dữ liệu đáng ngờ, trong khi "khử trùng" nghĩa là làm sạch các phần dữ liệu có vẻ đáng ngờ.) Ngoài ra, quản trị viên cơ sở dữ liệu có thể thiết lập các biện pháp kiểm soát để giảm thiểu lượng thông tin bị lộ ra qua một cuộc tấn công injection.

- **Broken Authentication**

Các lỗ hổng trong hệ thống xác thực (login) có thể cho phép kẻ tấn công truy cập vào tài khoản người dùng và thậm chí có khả năng xâm nhập toàn bộ hệ thống bằng tài khoản quản trị viên. Ví dụ: kẻ tấn công có thể lấy một danh sách chứa hàng nghìn tổ hợp tên người dùng / mật khẩu đã biết có được trong một lần vi phạm dữ liệu và sử dụng tập lệnh để thử tất cả các tổ hợp đó trên hệ thống đăng nhập để xem có tổ hợp nào hoạt động không.

Một số chiến lược để giảm thiểu lỗ hổng xác thực là sử dụng xác thực 2 yếu tố two-factor authentication (2FA) cũng như hạn chế hoặc trì hoãn các nỗ lực đăng nhập lặp lại bằng cách sử dụng giới hạn về số lần đăng nhập & thời gian giãn cách giữa các lần đăng nhập sai.

- **Sensitive Data Exposure**

Nếu các ứng dụng web không bảo vệ dữ liệu nhạy cảm như thông tin tài chính và mật khẩu, hacker có thể giành quyền truy cập vào dữ liệu đó và sử dụng cho các mục đích bất chính. Một phương pháp phổ biến để lấy cắp thông tin nhạy cảm là sử dụng một cuộc tấn công on-path attack.

Nguy cơ lộ dữ liệu có thể được giảm thiểu bằng cách mã hóa (encypt) tất cả dữ liệu nhạy cảm cũng như vô hiệu cache của bất kỳ thông tin nhạy cảm nào. Ngoài ra, các nhà phát triển ứng dụng web nên cẩn thận để đảm bảo rằng họ không lưu trữ bất kỳ dữ liệu nhạy cảm nào một cách không cần thiết.

Cache lưu trữ tạm thời dữ liệu để sử dụng lại. Ví dụ: trình duyệt web thường sẽ lưu vào bộ nhớ cache các trang web để nếu người dùng truy cập lại các trang đó trong một khoảng thời gian cố định, trình duyệt không phải tìm nạp các trang web đó từ đầu.

- **XML External Entities (XEE)**

Đây là một cuộc tấn công ứng dụng web bằng phân tích cú pháp đầu vào XML (parses XML input). Input này có thể tham chiếu đến một thực thể bên ngoài (external entity), đang cố gắng khai thác lỗ hổng trong trình phân tích cú pháp (parser). External entity có thể là một đơn vị lưu trữ, chẳng hạn như ổ cứng. XML parser có thể bị lừa để gửi dữ liệu đến một thực thể bên ngoài trái phép và chuyển trực tiếp dữ liệu nhạy cảm cho kẻ tấn công.

Các cách tốt nhất để ngăn chặn các cuộc tấn công XEE là để các ứng dụng web chấp nhận một loại dữ liệu ít phức tạp hơn, chẳng hạn như JSON, hoặc vô hiệu hóa việc sử dụng các thực thể bên ngoài trong một ứng dụng XML.

XML hay Extensible Markup Language là một markup language nhằm mục đích cho phép cả người & máy đều có thể đọc hiểu được. Do tính phức tạp và lỗ hổng bảo mật, XML hiện đang bị loại bỏ dần trong nhiều ứng dụng web.

JavaScript Object Notation (JSON) là một loại ký hiệu đơn giản được sử dụng để truyền dữ liệu qua internet. Mặc dù ban đầu được tạo cho JavaScript, JSON là ngôn ngữ có thể được thông dịch bởi nhiều ngôn ngữ lập trình khác nhau.

- **Broken Access Control**

Access Control hay kiểm soát truy cập đề cập đến một hệ thống kiểm soát quyền truy cập vào thông tin hoặc chức năng. Access Control chứa lỗ hổng cho phép kẻ tấn công bỏ qua ủy quyền (authorization) và thực hiện các tác vụ như thể là người dùng có đặc quyền, chẳng hạn như quản trị viên (admin). Ví dụ: một ứng dụng web có thể cho phép người dùng thay đổi tài khoản mà họ đã đăng nhập chỉ bằng cách thay đổi một phần của url mà không cần bất kỳ xác minh nào khác.

Kiểm soát truy cập có thể được bảo mật bằng cách đảm bảo rằng ứng dụng web sử dụng authorization tokens và đặt các kiểm soát chặt chẽ đối với các token này.

Nhiều dịch vụ cho phép sử dụng authorization tokens khi người dùng đăng nhập. Mọi yêu cầu đặc quyền mà người dùng đưa ra sẽ yêu cầu phải có authorization tokens. Đây là một cách an toàn để đảm bảo rằng đúng người dùng với đúng đặc quyền.

- **Security Misconfiguration**

Security misconfiguration hay lỗi cấu hình sai bảo mật là lỗ hổng phổ biến nhất trong danh sách và thường là kết quả của việc sử dụng cấu hình mặc định hoặc thông báo hiển thị lỗi quá nhiều thông tin. Ví dụ: một ứng dụng có thể hiển thị lỗi mô tả quá nhiều thông tin có thể tiết lộ các lỗ hổng trong ứng dụng. Điều này có

thể được giảm thiểu bằng cách loại bỏ bất kỳ tính năng không sử dụng nào trong code và đảm bảo rằng các thông báo lỗi sẽ mang tính tổng quát chung chung hơn.

- **Cross-Site Scripting**

Cross-Site Scripting xảy ra khi các ứng dụng web cho phép người dùng thêm code tùy chỉnh vào đường dẫn url hoặc vào một trang web mà những người dùng khác sẽ nhìn thấy. Lỗ hổng này có thể bị khai thác để chạy mã JavaScript độc hại (malicious JavaScript code) trên trình duyệt của nạn nhân. Ví dụ: kẻ tấn công có thể gửi email cho nạn nhân có vẻ là từ một ngân hàng đáng tin cậy, với một liên kết đến trang web của ngân hàng đó. Tuy nhiên, liên kết này có thể có một số mã JavaScript độc hại được gắn thẻ vào cuối url. Nếu trang web của ngân hàng không được bảo vệ thích hợp chống lại Cross-Site Scripting, thì mã độc hại đó sẽ được chạy trong trình duyệt web của nạn nhân khi họ nhấp vào liên kết.

Các chiến lược giảm thiểu tấn công Cross-Site Scripting bao gồm thoát các yêu cầu HTTP không đáng tin cậy cũng như xác thực và / hoặc loại bỏ các nội dung do người dùng thêm vào. Sử dụng các web development frameworks hiện đại như ReactJS và Ruby on Rails cũng cung cấp một số tính năng bảo vệ khỏi các cuộc tấn công Cross-Site Scripting.

- **Insecure Deserialization**

Tấn công này bao gồm Serialization & Deserialization.

- Serialization có nghĩa là lấy các đối tượng (object) từ mã ứng dụng (application code) và chuyển đổi chúng thành một định dạng có thể được sử dụng cho mục đích khác, chẳng hạn như lưu trữ dữ liệu vào đĩa hoặc phát trực tuyến dữ liệu đó.
- Deserialization thì ngược lại với Serialization.

Serialization giống như đóng gói đồ đạc vào các hộp trước khi chuyển đi, và deserialization giống như mở hộp và lắp ráp đồ đạc sau khi chuyển đi. Một cuộc tấn công deserialization giống như việc xáo trộn nội dung của các hộp trước khi chúng được giải nén trong quá trình di chuyển.

- **Sử dụng các thành phần có lỗ hổng đã biết**

Nhiều nhà phát triển (developer) web hiện nay sử dụng các thành phần như thư viện (libraries) và framework trong các ứng dụng web của họ. Những thành phần này là những phần mềm giúp các nhà phát triển tránh công việc thừa và cung cấp chức năng cần thiết; ví dụ phổ biến bao gồm các framework front-end như React

và các thư viện nhỏ hơn được sử dụng để thêm các biểu tượng chia sẻ hoặc a/b testing. Một số kẻ tấn công tìm kiếm các lỗ hổng trong các thành phần này mà sau đó chúng có thể sử dụng để điều phối các cuộc tấn công. Một số thành phần phổ biến hơn được sử dụng trên hàng trăm nghìn trang web; kẻ tấn công tìm thấy lỗ hổng bảo mật trong những thành phần này có thể khiến hàng trăm nghìn trang web bị khai thác.

Các nhà phát triển các thành phần này thường cung cấp các bản vá bảo mật và cập nhật để bổ sung các lỗ hổng đã biết, nhưng các nhà phát triển ứng dụng web không phải lúc nào cũng có các phiên bản được vá hoặc cập nhật mới nhất. Để giảm thiểu rủi ro khi chạy các thành phần có lỗ hổng đã biết, các nhà phát triển nên xóa các thành phần không sử dụng khỏi dự án, cũng như đảm bảo rằng đang nhận các thành phần từ một nguồn đáng tin cậy và đảm bảo chúng được cập nhật.

- **Kiểm tra log & giám sát không hiệu quả**

Nhiều ứng dụng web không thực hiện đủ các bước để phát hiện vi phạm dữ liệu. Thời gian phát hiện trung bình cho một vi phạm là khoảng 200 ngày sau khi đã xảy ra. Điều này cho phép những kẻ tấn công có nhiều thời gian để gây ra thiệt hại trước khi có bất kỳ phản ứng nào. OWASP khuyến nghị rằng các nhà phát triển web nên thực hiện ghi log và giám sát (monitor) cũng như lên các kế hoạch ứng phó sự cố để đảm bảo rằng họ nhận thức được các cuộc tấn công vào các ứng dụng.

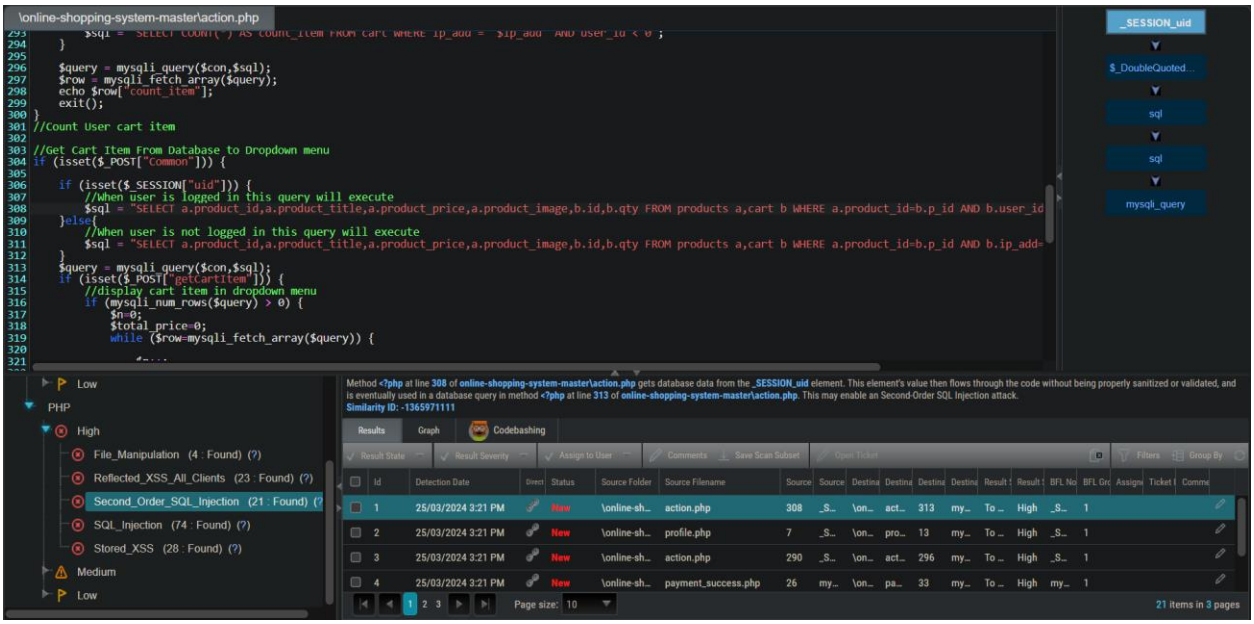
Chương 2: Công cụ học tập

I. Checkmarx

1. Khái niệm

Checkmarx là một công ty bảo mật có trụ sở tại Mỹ, chuyên trong lĩnh vực kiểm thử bảo mật phần mềm và phát triển các công cụ để phát hiện lỗ hổng bảo mật trong mã nguồn của ứng dụng. Cụ thể, Checkmarx tập trung vào việc phát triển các giải pháp tự động hóa kiểm thử bảo mật phần mềm, giúp các tổ chức phát hiện và khắc phục lỗ hổng bảo mật từ giai đoạn phát triển phần mềm đến khi ứng dụng được triển khai và hoạt động.

Công cụ kiểm thử bảo mật của Checkmarx có thể phát hiện ra các lỗ hổng bảo mật phổ biến như lỗ hổng SQL injection, Cross-site Scripting (XSS), lỗ hổng phân quyền, và nhiều loại lỗ hổng khác. Điều này giúp cho các nhà phát triển và nhà kiểm thử có thể cải thiện chất lượng mã nguồn và giảm thiểu rủi ro bảo mật trước khi phần mềm được triển khai.



2. Lợi ích

- **Công cụ đánh giá an ninh mã nguồn mạnh nhất:** Được đánh giá là công cụ có công nghệ phân tích mã nguồn tĩnh (**Static Code Analysis**) mạnh nhất hiện nay trên thị trường (Theo đánh giá của **Gartner**).
- **Tích hợp chặt với SDLC:** Tích hợp chặt chẽ với quy trình phát triển ứng dụng SDLC (**Software Development Life Cycle**), đảm bảo ứng dụng được phát triển xong sẽ an toàn trước khi đưa vào môi trường production.

- **Rà quét nhanh, mức độ chính xác cao:** Hiệu năng rà quét lớn, dò quét và phân tích bảo mật cho mã nguồn nhanh với kết quả False Positive là thấp nhất trong các vendor.
- **Cung cấp mô hình dò quét và phân tích bảo mật ứng dụng tổng thể:** Hỗ trợ dò quét và phân tích theo chế độ **Incremental Scan**, quét các ứng dụng lai/phức tạp (được viết và ghép bởi nhiều ngôn ngữ lập trình, nhiều framework khác nhau).
- **Đánh giá bảo mật theo chuẩn và tùy biến security script:** Hỗ trợ đánh giá bảo mật cho mã nguồn (source code) theo các chuẩn nổi tiếng bao gồm: **OWASP TOP 10 & Mobile**, **SANS TOP 25**, **CWE**, **PCI DSS**, **HIPAA**, **BSIMM**, … và cho phép tổ chức tự tùy biến/viết bổ sung các Security Script để kiểm tra các vấn đề về bảo mật.
- **Hỗ trợ nhiều nhất các ngôn ngữ lập trình bao gồm cả Mobile:** Hỗ trợ dò quét, phân tích và đánh giá bảo mật cho nhiều loại source code khác nhau (**hơn 15 ngôn ngữ lập trình khác nhau**) bao gồm: C++, JAVA, PHP, dot.Net, Ruby, Python, Perl, Android, Apple, …
- **Dễ dàng triển khai, sử dụng:** Kiến trúc đơn giản với giao diện trực quan, rõ ràng giúp dễ dàng trong tương tác phân tích mã, nhanh chóng định danh lỗ hổng bảo mật, điều tra cũng như thực thi sửa chữa.

3. Tính năng

- **Hỗ trợ tất cả các ngôn ngữ lập trình phổ biến: Checkmarx** hỗ trợ dò quét và đánh giá bảo mật cho nhiều nền tảng (**platform**) và ngôn ngữ lập trình ứng dụng khác nhau (hơn 15 ngôn ngữ lập trình phổ biến) bao gồm: Lập trình cho Desktop (**Web/Win Form**), Lập trình cho mobile, đảm bảo đáp ứng như cầu đánh giá bảo mật cho hệ thống ứng dụng của tổ chức. **Checkmarx** cũng hỗ trợ sẵn tích hợp với các công cụ/môi trường phát triển ứng dụng IDE.
- **Phân tích, đánh giá an ninh bảo mật mã nguồn theo các chuẩn bảo mật nổi tiếng và cho phép chỉnh sửa tùy biến: Checkmarx** cung cấp khả năng rà quét, đánh giá bảo mật cho hệ thống ứng dụng theo nhiều chuẩn tuân thủ nổi tiếng thế giới bao gồm: OWASP TOP 10 & Mobile, SANS TOP 25, CWE, PCI DSS, HIPAA, BSIMM, MISRA,… giúp tổ chức dễ dàng đánh giá bảo mật của ứng dụng theo các chuẩn tuân thủ kể từ giai đoạn phát triển ứng dụng.
- **Tích hợp toàn diện với vòng đời phát triển ứng dụng Software Develop Life Cycle (SDLC): Checkmarx** tích hợp chặt chẽ với quy trình/vòng đời phát triển của ứng dụng, đảm bảo khả năng rà quét bảo mật, khuyến cáo sửa chữa cho từng giai đoạn trong chu trình phát triển phần mềm: Phát triển, lưu

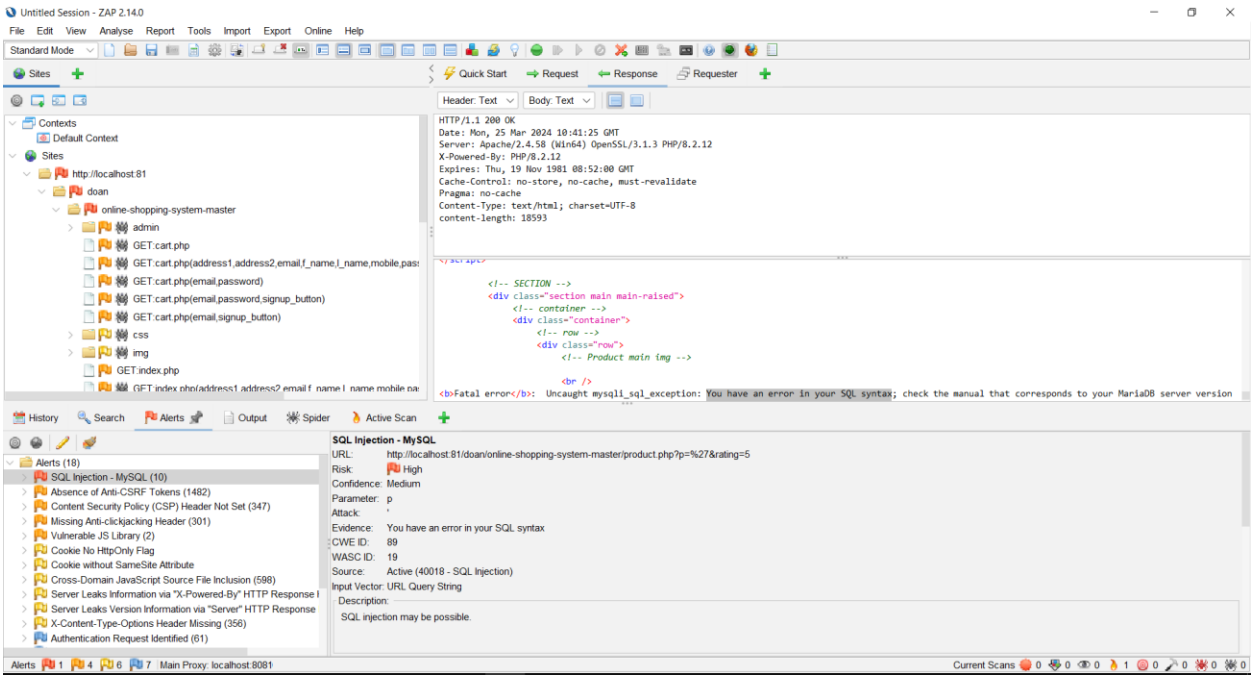
trữ, quản lý, kiểm thử, bug tracking, sửa chữa. Đảm bảo ứng dụng an toàn ngay sau khi được phát triển xong.

II. OWASP ZAP

1. Khái niệm

OWASP là viết tắt của Open Web Application Security Project là một tổ chức phi lợi nhuận quốc tế chuyên về bảo mật ứng dụng web. Một trong những nguyên tắc cốt lõi của OWASP là tất cả các tài liệu của tổ chức đều miễn phí và dễ dàng truy cập trên trang web chính thức http://owasp.org, giúp mọi người đặc biệt là ngành an ninh mạng có thể cải thiện tính bảo mật của ứng dụng web. Các tài liệu OWSAP cung cấp bao gồm tài liệu, công cụ, video và diễn đàn. OWASP được biết đến nhiều nhất qua OWASP Top 10.

OWASP Top 10 là một báo cáo được cập nhật thường xuyên về các nguy cơ bảo mật đối với bảo mật ứng dụng web, tập trung vào 10 rủi ro/lỗ hổng quan trọng nhất. Báo cáo được tổng hợp bởi một nhóm các chuyên gia bảo mật từ khắp nơi trên thế giới. OWASP đề cập đến Top 10 như một "tài liệu nâng cao nhận thức" và họ khuyến nghị tất cả các công ty nên kết hợp báo cáo này vào các quy trình của họ để giảm thiểu rủi ro bảo mật.



2. Kiểm tra được gì?

OWASP ZAP kiểm tra lỗ hổng của ứng dụng web bằng ba phương pháp kiểm tra sau.

- **Scan đơn giản**

"Scan" nghĩa là tấn công ứng dụng web mục tiêu. Khi nhập URL của ứng dụng Web cần kiểm tra trong OWASP ZAP, quá trình scan đơn giản sẽ bắt đầu. Tùy trường hợp mà việc scan có thể được thực hiện trong vài phút, nhưng hoặc có thể mất vài giờ.

OWASP ZAP sẽ gửi một số lượng lớn request đến ứng dụng Web mục tiêu. ("Request" theo thuật ngữ máy tính là "yêu cầu" hoặc "tin nhắn" được trao đổi trên hệ thống máy tính.) Khi quá trình scan đơn giản hoàn tất, các lỗ hổng được tìm thấy sẽ được hiển thị ở mục "Alert (warning)" trên màn hình.

- **Scan tĩnh**

Trong quá trình scan tĩnh, người thao tác OWASP ZAP (user) sẽ thực sự sử dụng thử ứng dụng Web mục tiêu. Ví dụ: nếu ứng dụng Web mục tiêu là trang EC, thì user sẽ mở trang sản phẩm, cho sản phẩm vào "Giỏ hàng" rồi thực hiện thanh toán. Bằng cách thực hiện các thao tác mà user luôn thực hiện ở ứng dụng Web, OWASP ZAP sẽ thực hiện kiểm tra là ứng dụng Web đã hoạt động như thế nào ứng với thao tác của user. Khi scan tĩnh hoàn thành, các lỗ hổng được tìm thấy cũng sẽ được hiển thị dưới dạng Warning.

- **Scan động**

Scan động được thực hiện bằng cách gửi một số lượng lớn request (giống như quá trình scan đơn giản) đối với các vị trí thực hiện scan tĩnh. Mục đích là "Thử cố gắng tấn công một lần nữa" nhằm tìm ra các điểm yếu đã bị sót ở scan đơn giản và scan tĩnh.

III. BURP SUITE

1. Khái niệm

Burp Suite là một bộ công cụ thử nghiệm bảo mật ứng dụng web được phát triển bởi PortSwigger Web Security. Nó được sử dụng rộng rãi trong lĩnh vực kiểm thử bảo mật để phát hiện và khai thác các lỗ hổng bảo mật trong ứng dụng web. Burp Suite bao gồm nhiều công cụ như proxy, crawler, scanner, repeater, intruder, và comparer, giúp người dùng thực hiện các hoạt động kiểm thử bảo mật như kiểm tra các lỗ hổng XSS (Cross-Site Scripting), SQL injection, hoặc sử dụng các kỹ thuật khai thác khác nhau.

2. Tính năng nổi bật

- **Interception Proxy:** được thiết kế để bắt các request từ đó có thể tùy ý sửa đổi trước khi các request này được gửi lên server.
- **Repeater:** cho phép sử dụng một request trước đó và tùy sửa đổi nội dung request một cách nhanh chóng nhiều lần khác nhau.
- **Intruder:** tự động hóa việc gửi hàng loạt các request có chứa các payload tương tự nhau lên server.
- **Decoder:** decode và encode string theo các format khác nhau (URL, Base64, HTML,…).
- **Comparer:** chỉ ra sự khác nhau giữa các requests/responses
- **Extender:** API để mở rộng chức năng của Burp Suite. Bạn có thể download các extensions thông qua Bapp Store.
- **Spider & Discover Content:** crawl link có trong ứng dụng web.
- **Scanner (chỉ có trong bản Pro):** đây là một mô đun khác mạnh mẽ, nó tự động quét các lỗ hổng trong ứng dụng web (XSS, SQLi, Command Injection, File Inclusion,…).

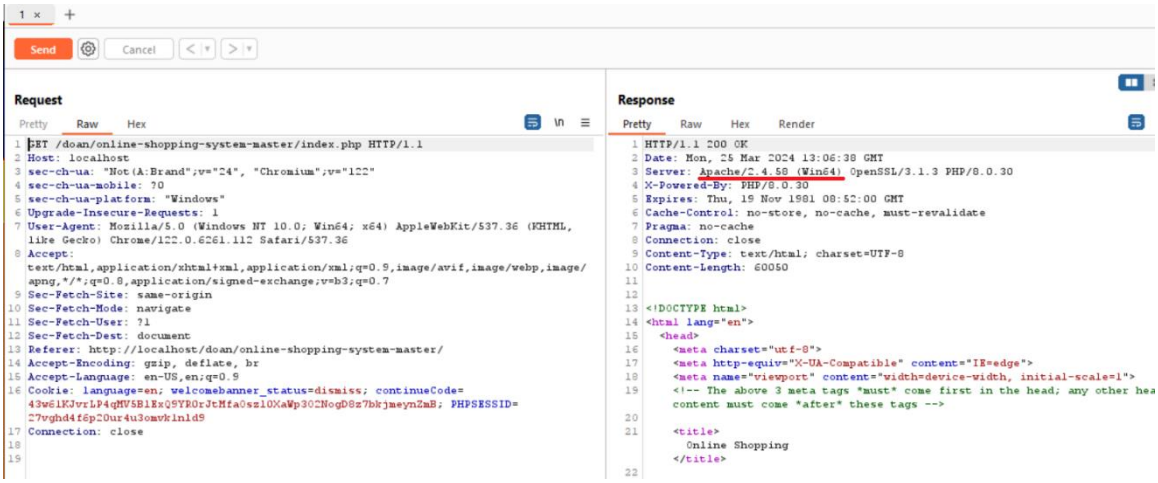
Chương 3: Triển khai đề tài

I. Information Gathering

Fingerprint Web Server

Các kiểu lấy biểu ngữ:

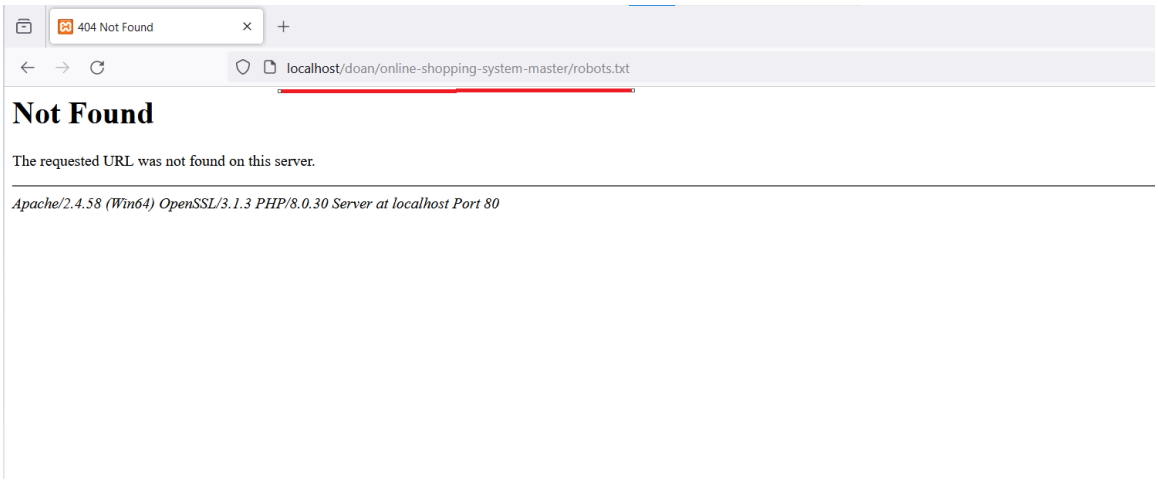
- **Active banner grabbing:** Trong phương pháp này, tin tặc gửi các gói đến máy chủ từ xa và phân tích dữ liệu phản hồi. Cuộc tấn công liên quan đến việc mở TCP hoặc kết nối tương tự giữa máy chủ gốc và máy chủ từ xa. Hệ thống phát hiện xâm nhập (IDS) có thể dễ dàng phát hiện biểu ngữ đang hoạt động.
- **Passive banner grabbing:** Phương pháp này cho phép tin tặc và nhà phân tích bảo mật có được thông tin giống nhau trong khi tránh tiết lộ kết nối ban đầu. Trong việc lấy biểu ngữ thụ động, kẻ tấn công triển khai phần mềm và phần mềm độc hại làm cổng ngăn chặn kết nối trực tiếp khi thu thập dữ liệu từ mục tiêu. Kỹ thuật này sử dụng các công cụ và dịch vụ mạng của bên thứ ba để thu thập và phân tích các gói nhằm xác định phần mềm và phiên bản đang được sử dụng chạy trên máy chủ.



dùng burpsuite để lấy biểu ngữ

Review Webserver Metafiles for Information Leakage

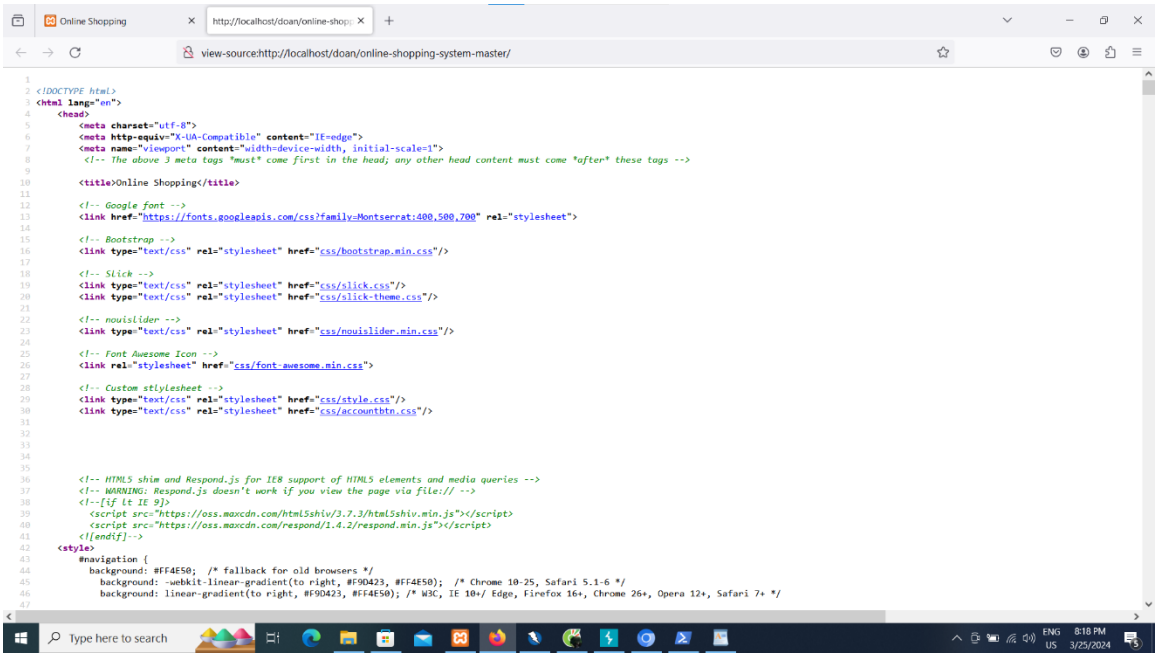
Tệp robots.txt là một tập tin văn bản đơn giản được đặt trên máy chủ web để chỉ định cho các robot của công cụ tìm kiếm (còn được gọi là web crawlers hoặc web spiders) cách tiếp cận và quét trang web của bạn. Đôi khi do sơ xuất lập trình viên cũng có thể để lộ thông tin riêng tư và là lỗ hổng cho các hacker khai thác

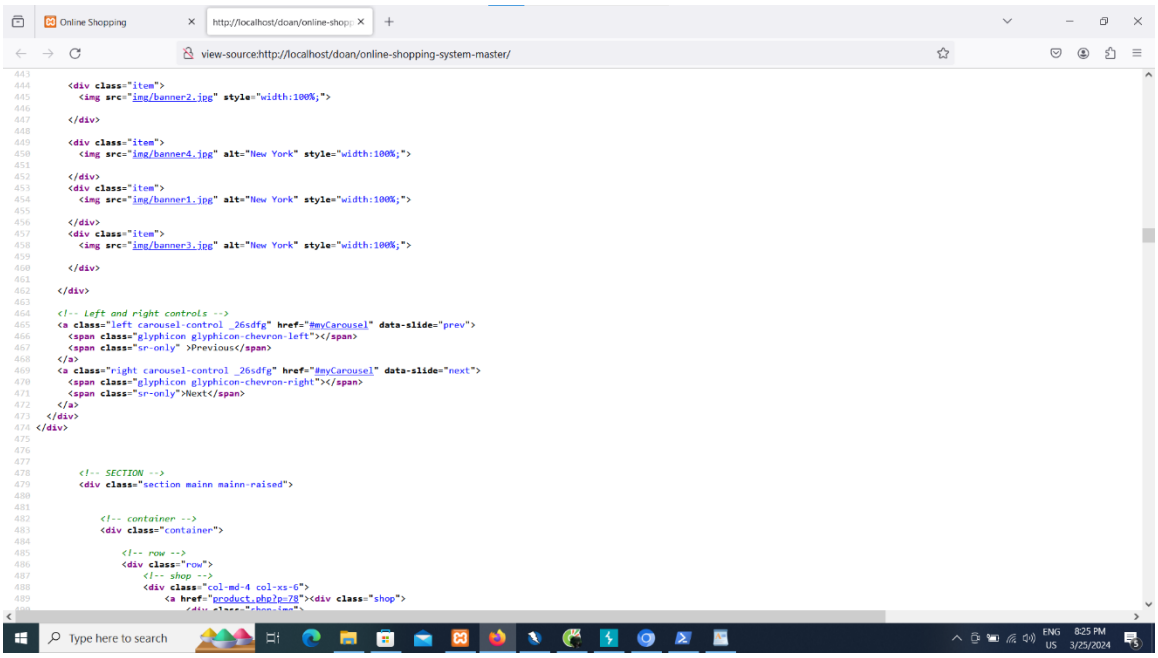


Tìm thông tin trên tệp robots.txt

Review Webpage Content for Information Leakage

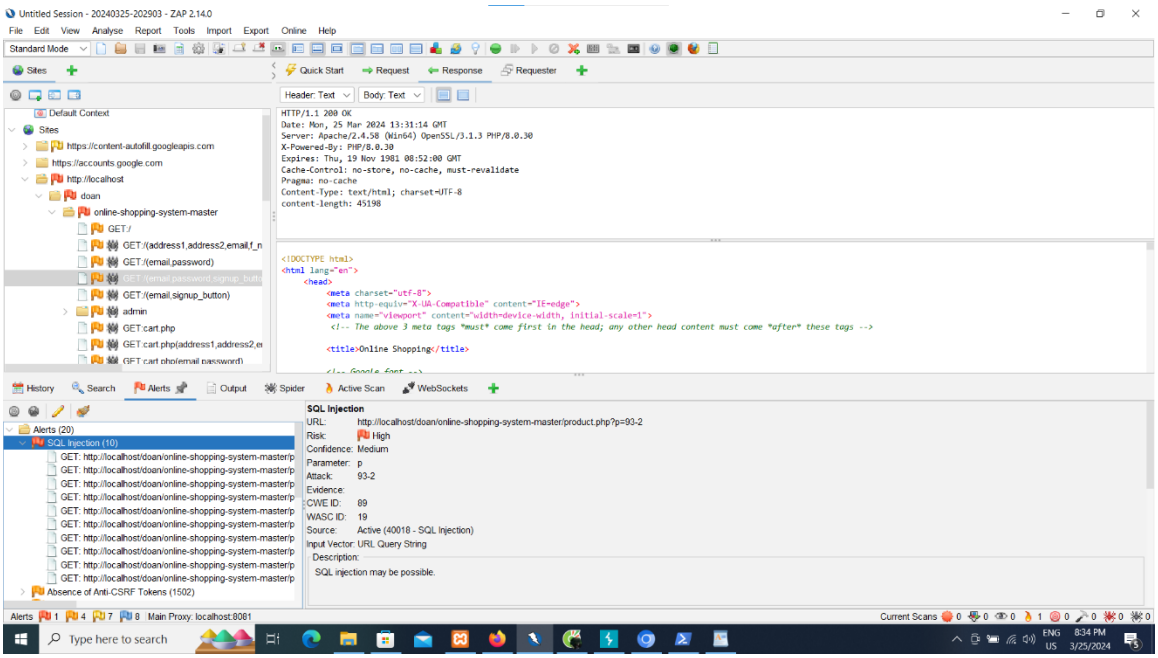
Trong View Page Source tìm kiếm những thông tin nhạy cảm từ các comment và metadata





Map Execution Paths Through Application

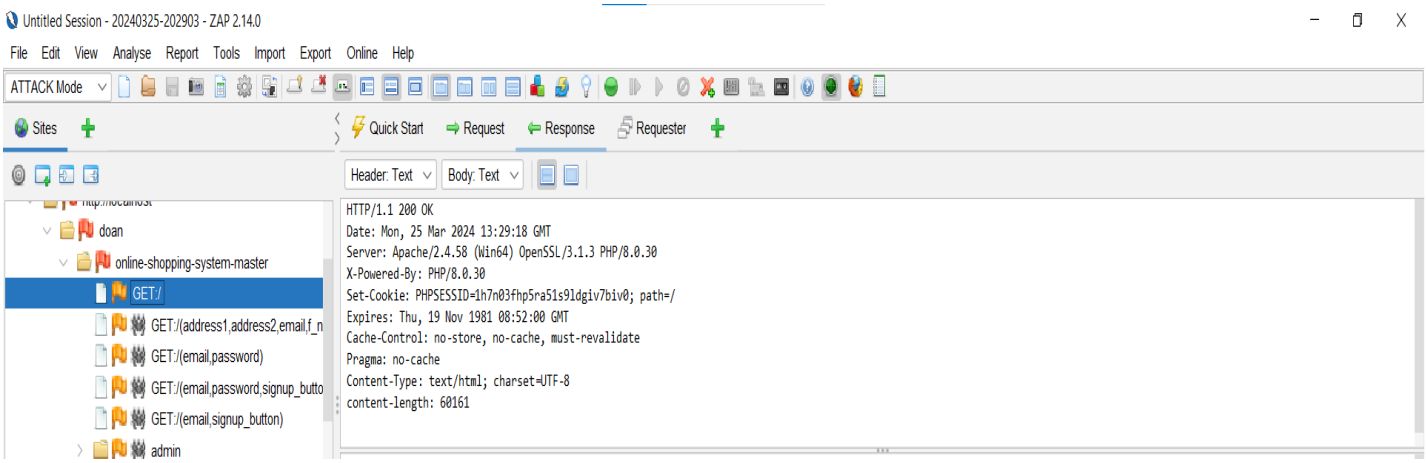
Dùng công cụ OWASP ZAP để ánh xạ các chức năng của ứng dụng và phát hiện các lỗ hổng trên website.



Dùng công cụ OWASP ZAP để ánh xạ chức năng

Fingerprint Web Application

Xác định platform và phiên bản của ứng dụng để có phương án khai thác thích hợp



Xác định platform và phiên bản của ứng dụng

Trang web sử dụng công nghệ Apache và PHP để cung cấp nội dung cho người dùng.Việc sử dụng phiên bản cụ thể của Apache và PHP có thể cung cấp thông tin về tính bảo mật và hiệu suất của trang web.

II. Configuration and Deployment Management Testing

Test HTTP Methods

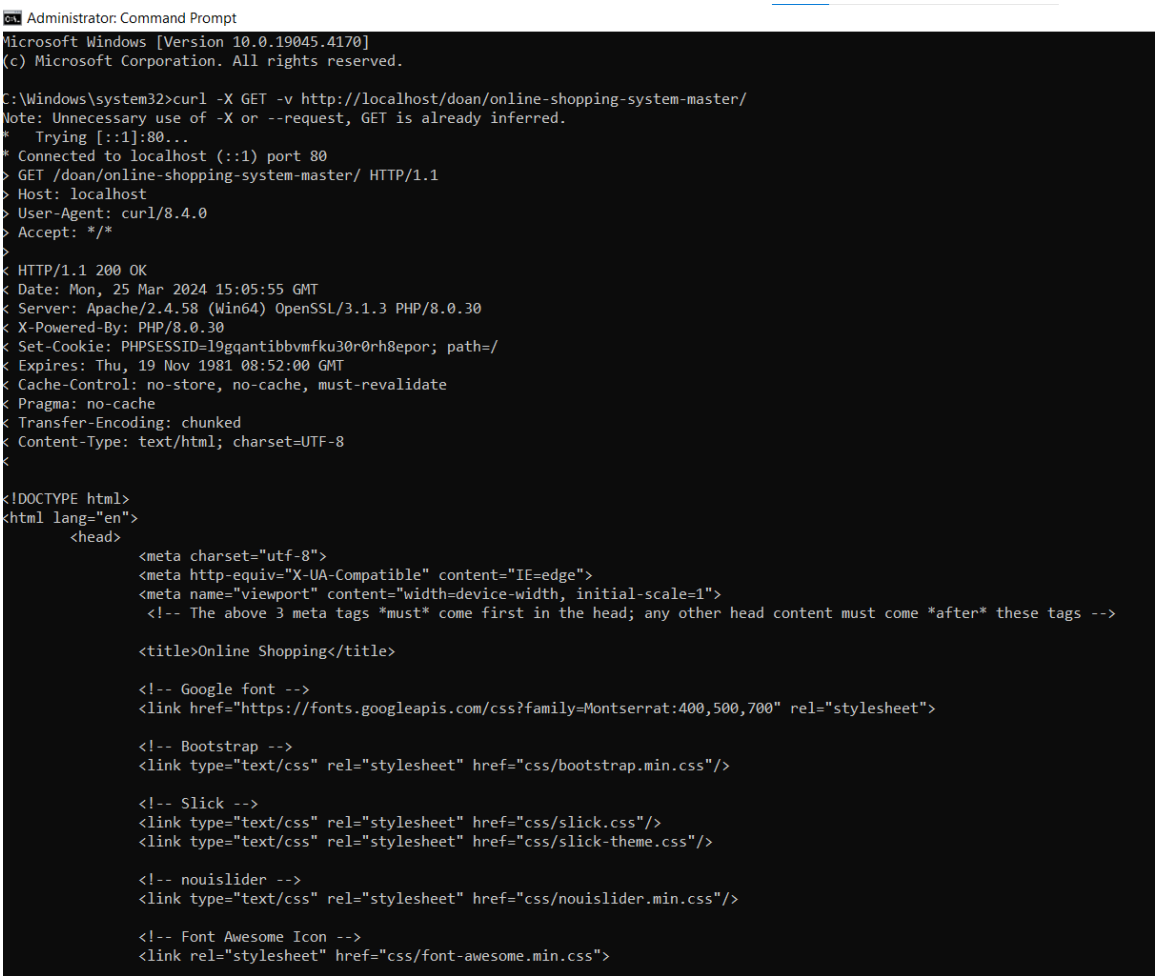
Mục tiêu thử nghiệm:

• Liệt kê các phương thức HTTP được hỗ trợ.

• Kiểm tra bỏ qua kiểm soát truy cập.

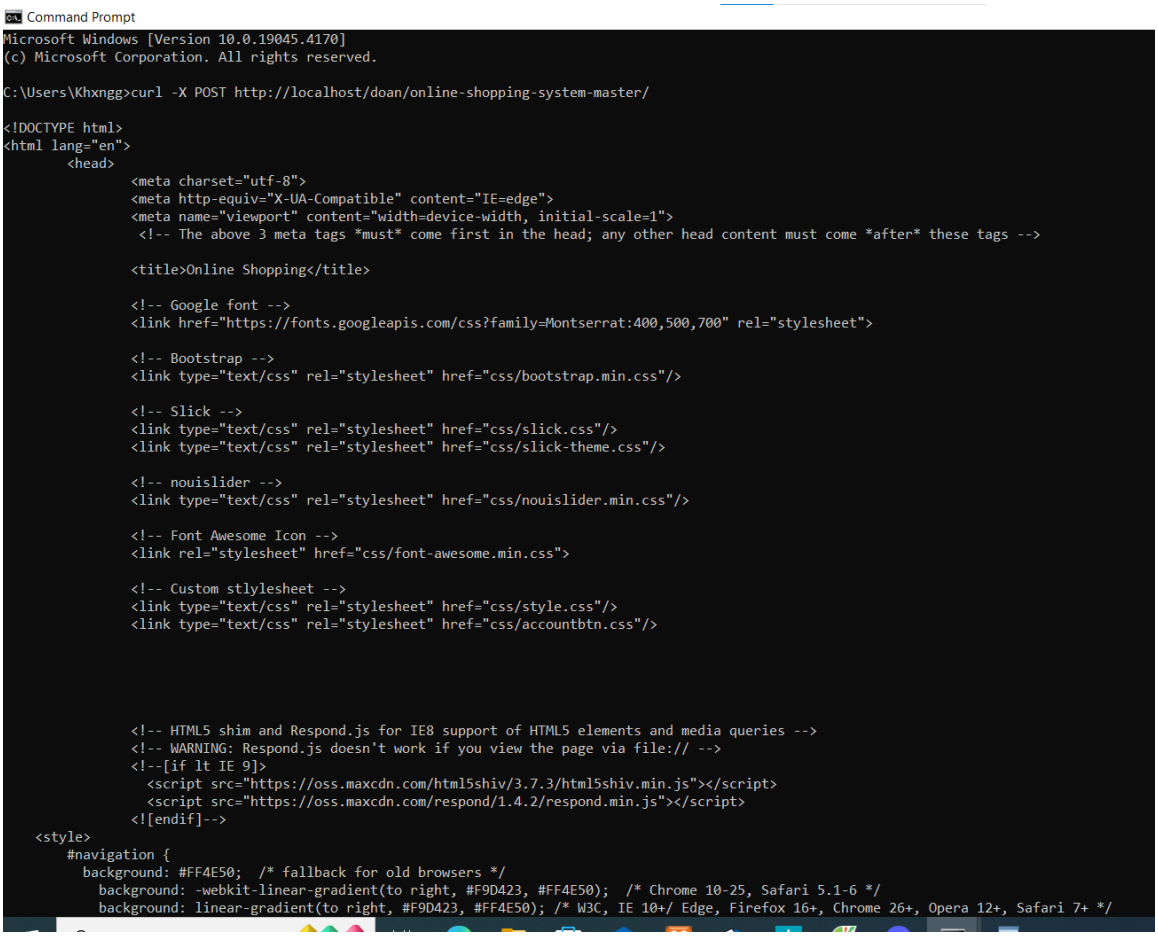
• Kiểm tra kỹ thuật ghi đè phương thức HTTP.

Kiểm tra GET



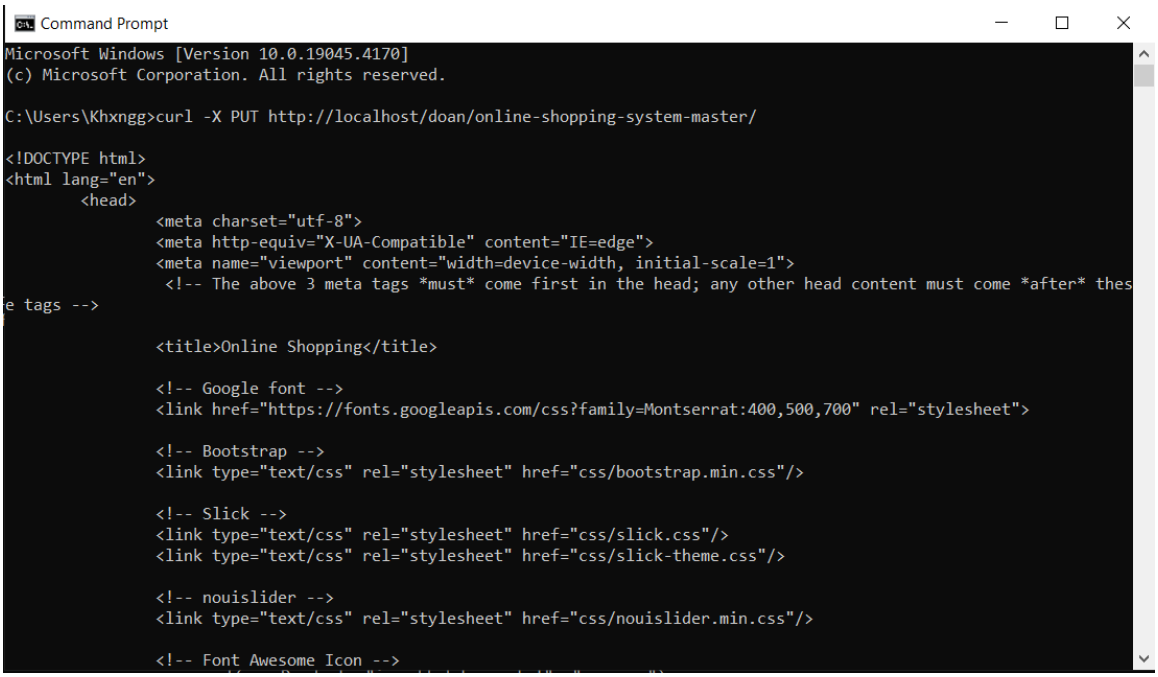
Testing với phương thức GET

Kiểm tra POST



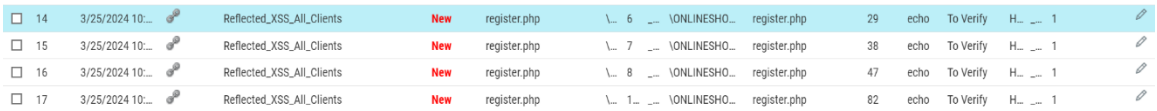
Testing với phương thức POST

Kiểm tra PUT



Testing phương thức PUT

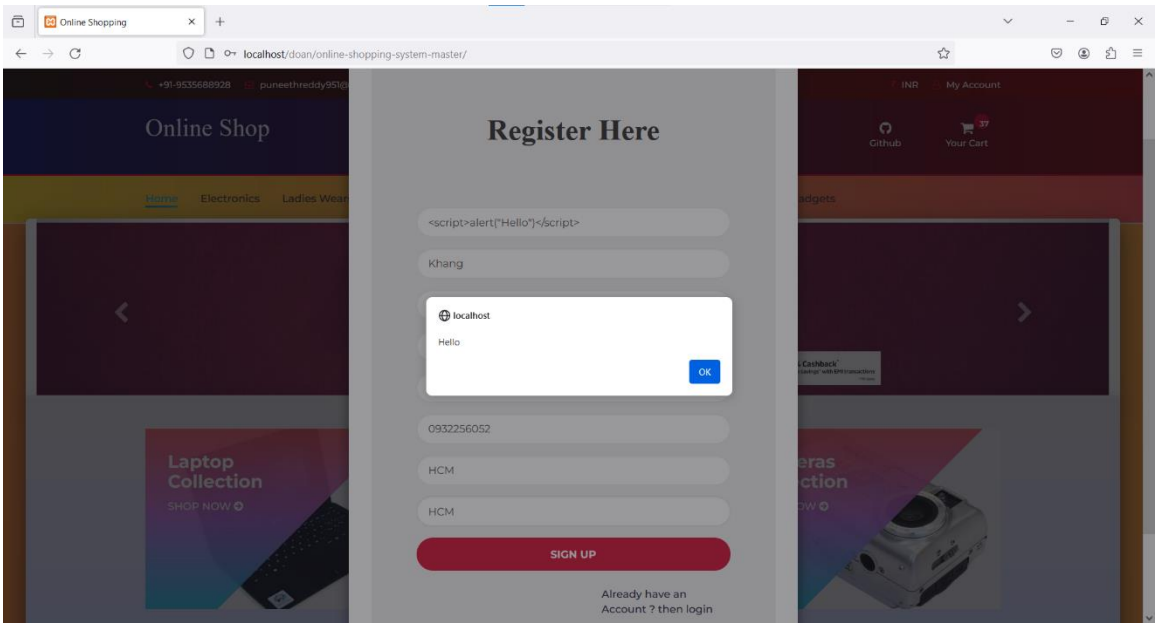
Lỗ hổng XSS



Lỗ hổng XSS

XSS là một loại lỗ hổng bảo mật thường gặp trên các ứng dụng web. Khi một ứng dụng web không kiểm tra và xử lý đầu vào từ người dùng một cách an toàn, kẻ tấn công có thể chèn các mã JavaScript hoặc HTML vào trang web và thực thi nó trên trình duyệt của người dùng.

Mức độ nguy hiểm của lỗ hổng XSS có thể rất cao, tùy thuộc vào cách mà kẻ tấn công sử dụng nó ví dụ như: Đánh Cắp Cookie,Đánh Cắp Dữ Liệu,Thực Thi Hành Động Bất Hợp Pháp,Phishing.....

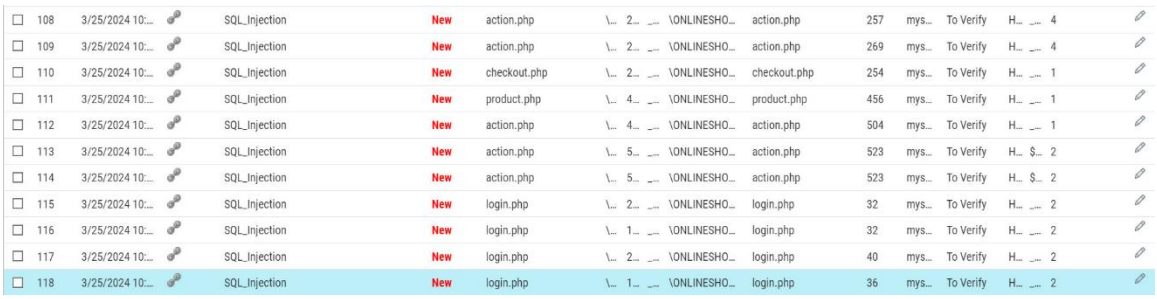


Script:<script>alert("Hello")</script>

Testing for SQL Injection

Từ kết quả scanning có thể thấy ứng dụng dễ bị tấn công bởi SQL Injection

45	3/25/2024 10:...	SQL_Injection	New	edituser.php	\...	5	...	\ONLINESHO...	edituser.php	16	mys...	To Verify	H...	$...	5
46	3/25/2024 10:...	SQL_Injection	New	edituser.php	\...	1...	...	\ONLINESHO...	edituser.php	16	mys...	To Verify	H...	$...	5
47	3/25/2024 10:...	SQL_Injection	New	edituser.php	\...	1...	...	\ONLINESHO...	edituser.php	16	mys...	To Verify	H...	$...	5
48	3/25/2024 10:...	SQL_Injection	New	edituser.php	\...	1...	...	\ONLINESHO...	edituser.php	16	mys...	To Verify	H...	$...	5
49	3/25/2024 10:...	SQL_Injection	New	edituser.php	\...	1...	...	\ONLINESHO...	edituser.php	16	mys...	To Verify	H...	$...	5
50	3/25/2024 10:...	SQL_Injection	New	products_list.php	\...	7	...	\ONLINESHO...	products_list.php	9	mys...	To Verify	H...	...	2
51	3/25/2024 10:...	SQL_Injection	New	electroniclist.php	\...	7	...	\ONLINESHO...	electroniclist.php	9	mys...	To Verify	H...	...	2
52	3/25/2024 10:...	SQL_Injection	New	manageuser.php	\...	7	...	\ONLINESHO...	manageuser.php	10	mys...	To Verify	H...	...	1
53	3/25/2024 10:...	SQL_Injection	New	products_list.php	\...	7	...	\ONLINESHO...	products_list.php	22	mys...	To Verify	H...	...	2
54	3/25/2024 10:...	SQL_Injection	New	electroniclist.php	\...	7	...	\ONLINESHO...	electroniclist.php	22	mys...	To Verify	H...	...	2
55	3/25/2024 10:...	SQL_Injection	New	addsuppliers.php	\...	8	...	\ONLINESHO...	addsuppliers.php	16	mys...	To Verify	H...	$...	7
56	3/25/2024 10:...	SQL_Injection	New	addsuppliers.php	\...	9	...	\ONLINESHO...	addsuppliers.php	16	mys...	To Verify	H...	$...	7
57	3/25/2024 10:...	SQL_Injection	New	addsuppliers.php	\...	1...	...	\ONLINESHO...	addsuppliers.php	16	mys...	To Verify	H...	$...	7
58	3/25/2024 10:...	SQL_Injection	New	addsuppliers.php	\...	1...	...	\ONLINESHO...	addsuppliers.php	16	mys...	To Verify	H...	$...	7
59	3/25/2024 10:...	SQL_Injection	New	addsuppliers.php	\...	1...	...	\ONLINESHO...	addsuppliers.php	16	mys...	To Verify	H...	$...	7
60	3/25/2024 10:...	SQL_Injection	New	addsuppliers.php	\...	1...	...	\ONLINESHO...	addsuppliers.php	16	mys...	To Verify	H...	$...	7
61	3/25/2024 10:...	SQL_Injection	New	addsuppliers.php	\...	1...	...	\ONLINESHO...	addsuppliers.php	16	mys...	To Verify	H...	$...	7
62	3/25/2024 10:...	SQL_Injection	New	add_products.php	\...	8	...	\ONLINESHO...	add_products.php	29	mys...	To Verify	H...	$...	7
63	3/25/2024 10:...	SQL_Injection	New	add_products.php	\...	9	...	\ONLINESHO...	add_products.php	29	mys...	To Verify	H...	$...	7
64	3/25/2024 10:...	SQL_Injection	New	add_products.php	\...	1...	...	\ONLINESHO...	add_products.php	29	mys...	To Verify	H...	$...	7
65	3/25/2024 10:...	SQL_Injection	New	add_products.php	\...	1...	...	\ONLINESHO...	add_products.php	29	mys...	To Verify	H...	$...	7
66	3/25/2024 10:...	SQL_Injection	New	add_products.php	\...	1...	...	\ONLINESHO...	add_products.php	29	mys...	To Verify	H...	$...	7



108	3/25/2024 10:...	SQL_Injection	New	action.php	\...	2...	.....	\ONLINESHO...	action.php	257	mys...	To Verify	H...	.....	4
109	3/25/2024 10:...	SQL_Injection	New	action.php	\...	2...	.....	\ONLINESHO...	action.php	269	mys...	To Verify	H...	.....	4
110	3/25/2024 10:...	SQL_Injection	New	checkout.php	\...	2...	.....	\ONLINESHO...	checkout.php	254	mys...	To Verify	H...	.....	1
111	3/25/2024 10:...	SQL_Injection	New	product.php	\...	4...	.....	\ONLINESHO...	product.php	456	mys...	To Verify	H...	.....	1
112	3/25/2024 10:...	SQL_Injection	New	action.php	\...	4...	.....	\ONLINESHO...	action.php	504	mys...	To Verify	H...	.....	1
113	3/25/2024 10:...	SQL_Injection	New	action.php	\...	5...	.....	\ONLINESHO...	action.php	523	mys...	To Verify	H...	$...	2
114	3/25/2024 10:...	SQL_Injection	New	action.php	\...	5...	.....	\ONLINESHO...	action.php	523	mys...	To Verify	H...	$...	2
115	3/25/2024 10:...	SQL_Injection	New	login.php	\...	2...	.....	\ONLINESHO...	login.php	32	mys...	To Verify	H...	.....	2
116	3/25/2024 10:...	SQL_Injection	New	login.php	\...	1...	.....	\ONLINESHO...	login.php	32	mys...	To Verify	H...	.....	2
117	3/25/2024 10:...	SQL_Injection	New	login.php	\...	2...	.....	\ONLINESHO...	login.php	40	mys...	To Verify	H...	.....	2
118	3/25/2024 10:...	SQL_Injection	New	login.php	\...	1...	.....	\ONLINESHO...	login.php	36	mys...	To Verify	H...	.....	2

Các chức năng bị lỗi:

- edituser.php
- action.php
- product.php
- checkout.php
- login.php
- manageuser.php
- electroniclist.php
- productlist.php

III. Báo cáo kết quả kiểm thử

Tóm lại, qua kết quả kiểm thử thâm nhập trên, chúng ta có phát hiện có 1 số lỗ hổng nghiêm trọng như:

- File Manipulation
- Reflected XSS All Clients
- Second Order SQL Injection
- SQL Injection
- Stored XSS

Tất cả các lỗ hổng trên đều mang lại rủi ro rất cao về an toàn thông tin cho người dùng cũng như về máy chủ. Hacker có thể đánh cấp thông tin từ lỗ hổng SQL Injection hay nghiệm trọng là XSS có thể thực hiện việc đánh cắp dữ liệu người dùng.

Đề xuất cách khắc phục:

- Prepared Statements: giúp việc truy vấn dữ liệu an toàn hơn
- Object-Relational Mapping: giúp tránh thao tác trực tiếp với câu lệnh SQL
- Content Security Policy: để giới hạn hoặc ngăn chặn việc thực thi mã JavaScript từ các nguồn không an toàn

TÀI LIỆU THAM KHẢO

https://quantrimang.com/cong-nghe/tim-hieu-ve-penetration-testing-162644

https://toancaujsc.com.vn/giai-phap-phan-tich-lo-hong-bao-mat-ma-nguon-checkmarx.html

https://vinsep.com/kien-thuc/security/owasp-la-gi-top-10-owasp-la-gi/

https://blog.neoscorp.vn/gioi-thieu-ve-cong-cu-chan-doan-bao-mat-owasp-zap/

https://viblo.asia/p/burp-suite-tro-thu-dac-luc-cho-tester-va-pentester-trong-kiem-tra-ung-dung-web-E375z4GWZGW