|  |
| --- |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| ĐỒ ÁN TỐT NGHIỆP  **NGHIÊN CỨU XÂY DỰNG CÔNG CỤ KIỂM THỬ AN TOÀN THÔNG MINH** |
| Ngành: An toàn thông tin  Mã số: 7.48.02.02  *Sinh viên thực hiện*:  **Cao Quang Huy – MSV: AT180123**  *Người hướng dẫn*:  **TS. Phạm Duy Trung**  Khoa An toàn thông tin – Học viện Kỹ thuật mật mã |
| Hà Nội, 2025 |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| ĐỒ ÁN TỐT NGHIỆP  **NGHIÊN CỨU XÂY DỰNG CÔNG CỤ KIỂM THỬ AN TOÀN THÔNG MINH** |
| Ngành: An toàn thông tin  Mã số: 7.48.02.02  *Sinh viên thực hiện*:  **Cao Quang Huy – MSV: AT180123**  *Người hướng dẫn*:  **TS. Phạm Duy Trung**  Khoa An toàn thông tin – Học viện Kỹ thuật mật mã |
| Hà Nội, 2025 |

Mục lục

[DANH MỤC TỪ VIẾT TẮT iii](#_Toc214386315)

[DANH MỤC HÌNH VẼ vi](#_Toc214386316)

[LỜI NÓI ĐẦU vii](#_Toc214386317)

[LỜI CẢM ƠN ix](#_Toc214386318)

[CHƯƠNG 1. TỔNG QUAN VỀ KIỂM THỬ AN TOÀN VÀ XU HƯỚNG AI TRONG BẢO MẬT 1](#_Toc214386319)

[1.1. Bối cảnh an toàn thông tin hiện nay 1](#_Toc214386320)

[1.2. Khái niệm và quy trình kiểm thử an toàn 4](#_Toc214386321)

[1.2.1. Khái niệm kiểm thử an toàn 4](#_Toc214386322)

[1.2.2. Quy trình kiểm thử an toàn theo Cyber Kill Chain 5](#_Toc214386323)

[1.2.3. Vai trò của kiểm thử an toàn trong thời đại AI 6](#_Toc214386324)

[1.3. Các công cụ kiểm thử truyền thống 6](#_Toc214386325)

[1.3.1. Nmap 6](#_Toc214386326)

[1.3.2. Nikto 7](#_Toc214386327)

[1.3.3. Nuclei 7](#_Toc214386328)

[1.3.4. crt.sh 8](#_Toc214386329)

[1.3.5. Metasploit 8](#_Toc214386330)

[1.3.6. Burp Suite 9](#_Toc214386331)

[1.4. Xu hướng tích hợp AI vào kiểm thử an toàn 9](#_Toc214386332)

[1.4.1. Vai trò cụ thể của AI trong chuỗi kiểm thử 10](#_Toc214386333)

[1.4.2. Các Kỹ thuật đang được áp dụng 10](#_Toc214386334)

[1.4.3. Lợi ích, thách thức và rủi ro khi tích hợp AI 11](#_Toc214386335)

[1.4.4. Triển vọng tương lai của AI trong kiểm thử an toàn 13](#_Toc214386336)

[1.5. Kết luận chương 1 13](#_Toc214386337)

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 14](#_Toc214386338)

[2.1. Kiến trúc tổng thể hệ thống 14](#_Toc214386339)

[2.1.1. Mục tiêu thiết kế 14](#_Toc214386340)

[2.1.2. Các thành phần chính của hệ thống 14](#_Toc214386341)

[2.1.3. Luồng hoạt động tổng thể 16](#_Toc214386342)

[2.2. Phân tích chi tiết từng mô hình chức năng 17](#_Toc214386343)

[2.2.1. Mô hình Thu Thập Subdomain crt.sh 17](#_Toc214386344)

[2.2.2. Mô hình Quét Mạng Nmap 20](#_Toc214386345)

[2.2.3. Mô hình Quét Web Nuclei + Nikto 23](#_Toc214386346)

[2.2.4. Mô hình AI Core 28](#_Toc214386347)

[2.2.5. Mô hình Enrich Bằng AI 32](#_Toc214386348)

[2.2.6. Mô hình Phân Tích Endpoint Và Lỗ Hổng Tiềm Năng 40](#_Toc214386349)

[2.2.7. Mô hình điều phối 46](#_Toc214386350)

[2.2.8. Mô hình API Backend 48](#_Toc214386351)

[2.2.9. Mô hình Frontend 50](#_Toc214386352)

[2.3. Kết luận chương 2 57](#_Toc214386353)

[CHƯƠNG 3. THỰC NGHIỆM 58](#_Toc214386354)

[3.1. Môi trường triển khai 58](#_Toc214386355)

[3.1.1. Hạ tầng phần cứng và hệ điều hành 58](#_Toc214386356)

[3.1.2. Phiên bản phần mềm chính và phụ thuộc 58](#_Toc214386357)

[3.1.3. Chuẩn bị môi trường 59](#_Toc214386358)

[3.2. Kịch bản thực nghiệm kiểm thử 60](#_Toc214386359)

[3.2.1. Kịch bản 1 60](#_Toc214386360)

[3.2.2. Kịch bản 2 68](#_Toc214386361)

[3.2.3. Kịch bản 3 72](#_Toc214386362)

[3.3. Đánh giá hiệu quả, hạn chế và hướng phát triển 83](#_Toc214386363)

[3.3.1. Đánh giá hiệu quả 83](#_Toc214386364)

[3.3.2. Hạn chế 83](#_Toc214386365)

[3.3.3. Hướng phát triển 84](#_Toc214386366)

[3.3.4. Kết luận chương 85](#_Toc214386367)

[KẾT LUẬN 87](#_Toc214386368)

[TÀI LIỆU THAM KHẢO 89](#_Toc214386369)

[PHỤ LỤC 91](#_Toc214386370)

Danh mục Từ viết tắt

|  |  |  |
| --- | --- | --- |
| **Từ viết tắt** | **Tiếng Anh** | **Tiếng Việt** |
| API | Application Programming Interface | Giao diện lập trình ứng dụng |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| CSS | Cascading Style Sheets | Ngôn ngữ tạo kiểu trang web |
| CVE | Common Vulnerabilities and Exposures | Lỗ hổng và điểm yếu phổ biến |
| DBIR | Data Breach Investigations Report | Báo cáo điều tra vi phạm dữ liệu |
| DL | Deep Learning | Học sâu |
| DNS | Domain Name System | Hệ thống tên miền |
| EDR | Endpoint Detection and Response | Giải pháp phát hiện và phản ứng tại điểm cuối |
| ENISA | European Union Agency for Cybersecurity | Cơ quan An ninh mạng Liên minh Châu Âu |
| GPT | Generative Pre-trained Transformer | Mô hình ngôn ngữ tạo sinh |
| HTML | HyperText Markup Language | Ngôn ngữ đánh dấu siêu văn bản |
| HTTP | HyperText Transfer Protocol | Giao thức truyền siêu văn bản |
| HTTPS | HyperText Transfer Protocol Secure | Giao thức truyền siêu văn bản bảo mật |
| IBM | International Business Machines | Tập đoàn Máy tính Quốc tế |
| IDS | Intrusion Detection System | Hệ thống phát hiện xâm nhập |
| IP | Internet Protocol | Giao thức Internet |
| IPID | IP Identification | Trường định danh IP |
| ISO | International Organization for Standardization | Tổ chức Tiêu chuẩn hóa Quốc tế |
| JSON | JavaScript Object Notation | Định dạng dữ liệu JSON |
| LFI | Local File Inclusion | Lỗ hổng sử dụng file hệ thống trên web |
| LLM | Large Language Model | Mô hình ngôn ngữ lớn |
| MFA | Multi-Factor Authentication | Xác thực đa yếu tố |
| ML | Machine Learning | Học máy |
| NIST | National Institute of Standards and Technology | Viện tiêu chuẩn và công nghệ quốc gia (Mỹ) |
| NSE | Nmap Scripting Engine | Công cụ kịch bản Nmap |
| OOB | Out-Of-Band | Ngoại băng tần |
| OS | Operating System | Hệ điều hành |
| OSINT | Open Source Intelligence | Tình báo nguồn mở |
| OWASP | Open Web Application Security Project | Dự án Bảo mật Ứng dụng |
| PCI DSS | Payment Card Industry Data Security Standard | Tiêu chuẩn bảo mật dữ liệu thẻ thanh toán |
| PDF | Portable Document Format | Định dạng tài liệu di động |
| PoC | Proof of Concept | Bằng chứng khái niệm |
| RaaS | Ransomware-as-a-Service | Ransomware dạng dịch vụ |
| RFI | Remote File Inclusion | Lỗ hổng sử dụng file bên ngoài trên web |
| RL | Reinforcement Learning | Học tăng cường |
| SDK | Software Development Kit | Bộ công cụ phát triển |
| SQLi | SQL Injection | Tấn công chèn câu lệnh SQL |
| SSL/TLS | Secure Sockets Layer / Transport Layer Security | Giao thức bảo mật SSL/TLS |
| SYN | Synchronize | Cờ SYN trong TCP |
| TCP | Transmission Control Protocol | Giao thức điều khiển truyền |
| UDP | User Datagram Protocol | Giao thức gói dữ liệu người dùng |
| URL | Uniform Resource Locator | Địa chỉ tài nguyên thống nhất |
| WEF | World Economic Forum | Diễn đàn Kinh tế Thế giới |
| XDR | Extended Detection and Response | Phát hiện và phản ứng mở rộng |
| XSS | Cross-Site Scripting | Tấn công chèn script chéo |
| XML | eXtensible Markup Language | eXtensible Markup Language |
| YAML | YAML Ain’t Markup Language | Ngôn ngữ đánh dấu YAML |

Danh mục hình vẽ

[Hình 1.1. Minh hoạ các nhóm tác nhân đe doạ, vectơ tấn công, công cụ tấn công và hệ thống phòng thủ tương ứng. 3](#_Toc452402311)

[Hình 1.2. Mô hình cyber kill train 5](#_Toc452402312)

[Hình 1.3. Vai trò của AI trong kiểm thử an toàn 10](#_Toc452402313)

[Hình 1.4. Rủi ro và biện pháp giảm thiểu khi dùng AI trong kiểm thử 12](#_Toc452402314)

[Hình 2.1. Sơ đồ luồng hoạt động tổng thể 17](#_Toc452402315)

[Hình 2.2. Giao diện quét 55](#_Toc452402316)

[Hình 2.3. Giao diện quản lý API key 57](#_Toc452402317)

Lời nói đầu

Trong bối cảnh chuyển đổi số diễn ra mạnh mẽ trên toàn cầu, các hệ thống thông tin, ứng dụng web và dịch vụ trực tuyến ngày càng đóng vai trò quan trọng trong hoạt động của doanh nghiệp, tổ chức và cơ quan nhà nước. Tuy nhiên, đi kèm với sự phát triển đó là nguy cơ ngày càng gia tăng về các cuộc tấn công mạng, xâm nhập trái phép và khai thác lỗ hổng bảo mật. Theo báo cáo của IBM Security năm 2024, chi phí trung bình cho một sự cố rò rỉ dữ liệu đã vượt ngưỡng 4.5 triệu USD, và hơn 80% các vụ tấn công bắt nguồn từ các lỗ hổng chưa được phát hiện trong ứng dụng web.

Trước thực trạng đó, việc kiểm thử bảo mật định kỳ và toàn diện trở thành yêu cầu bắt buộc đối với mọi hệ thống công nghệ thông tin. Tuy nhiên, quy trình kiểm thử truyền thống thường đòi hỏi chuyên môn cao, thời gian thực hiện dài và khó mở rộng, các công cụ kiểm thử truyền thống như Nmap, Nikto, Nuclei hay crt.sh thường hoạt động rời rạc, yêu cầu nhiều thao tác thủ công và thiếu khả năng phân tích ngữ cảnh sâu. Trong khi đó, các công trình nghiên cứu gần đây đã chỉ ra tiềm năng ứng dụng trí tuệ nhân tạo, đặc biệt là các mô hình ngôn ngữ lớn (LLM), để hỗ trợ phân tích kết quả quét, đánh giá lỗ hổng và đề xuất phương án khai thác một cách tự động và hiệu quả hơn.

Xuất phát từ nhu cầu thực tiễn và định hướng nghiên cứu hiện đại, đồ án này lựa chọn đề tài: “Nghiên cứu xây dựng công cụ kiểm thử an toàn thông minh” nhằm tích hợp các công cụ truyền thống này cùng công nghệ trí tuệ nhân tạo (AI) giúp hình thành một hệ thống kiểm thử tự động, thông minh và toàn diện.

Mục tiêu của đồ án là thiết kế và triển khai một hệ thống có khả năng tự động hóa toàn bộ quy trình kiểm thử bảo mật từ khâu trinh sát, quét mạng, quét web đến phân tích kết quả và sinh báo cáo. Hệ thống tích hợp các công cụ kiểm thử truyền thống như Nmap, Nikto, Nuclei cùng với lớp phân tích bằng AI để phân tích kết quả, phát hiện các điểm cuối (endpoint) tiềm ẩn và đánh giá khả năng khai thác lỗ hổng.

Các vấn đề chính cần giải quyết trong đồ án bao gồm:

* Tự động hóa quy trình kiểm thử bảo mật đầu cuối.
* Tích hợp hiệu quả các công cụ quét truyền thống với mô hình AI.
* Phân tích kết quả quét và enrich thông tin lỗ hổng bằng AI.
* Đánh giá khả năng khai thác lỗ hổng dựa trên phản hồi thực tế.
* Cung cấp giao diện đơn giản, dễ sử dụng cho người dùng không chuyên.

Cấu trúc đồ án gồm ba chương chính:

* **Chương 1**: Tổng quan về kiểm thử an toàn và xu hướng AI trong bảo mật.
* **Chương 2**: Phân tích thiết kế hệ thống.
* **Chương 3**: Thực nghiệm.

Lời cảm ơn

Trong quá trình thực hiện đồ án tốt nghiệp này, em đã nhận được sự giúp đỡ tận tình của cán bộ hướng dẫn là TS. Phạm Duy Trung – Giảng viên Khoa An toàn thông tin Học viện Kỹ thuật Mật mã, sự quan tâm sâu sát của cán bộ Hệ quản lý học viên quốc tế, sự động viên của người thân và bạn bè.

Xin cảm ơn tất cả mọi người đã tạo những điều kiện tốt nhất để tôi hoàn thành đồ án tốt nghiệp này!

|  |  |
| --- | --- |
|  | **SINH VIÊN THỰC HIỆN ĐỒ ÁN**  Cao Quang Huy |

|  |
| --- |
|  |

# Tổng Quan Về Kiểm Thử An Toàn Và Xu Hướng AI Trong Bảo Mật

Trong bối cảnh chuyển đổi số diễn ra mạnh mẽ, an toàn thông tin đã trở thành yếu tố cốt lõi quyết định sự ổn định và phát triển bền vững của các tổ chức, doanh nghiệp cũng như cơ quan nhà nước. Sự gia tăng nhanh chóng của các mối đe dọa mạng cùng với sự phức tạp ngày càng cao của hạ tầng công nghệ thông tin khiến cho các phương pháp bảo mật truyền thống không còn đủ khả năng ứng phó hiệu quả. Các cuộc tấn công mạng ngày nay không chỉ xuất phát từ mục tiêu tài chính mà còn mang tính chính trị, phá hoại và gián điệp công nghiệp, đòi hỏi cách tiếp cận chủ động, thông minh và liên tục hơn trong việc phát hiện, phòng ngừa, khắc phục.

Đồng thời, sự phát triển vượt bậc của trí tuệ nhân tạo (AI) và tự động hóa đã mở ra kỷ nguyên mới cho lĩnh vực kiểm thử an toàn thông tin (penetration testing). Việc tích hợp AI vào quy trình kiểm thử cho phép hệ thống không chỉ tự động phát hiện lỗ hổng mà còn có thể học hỏi, phân tích ngữ cảnh và đề xuất phương án khắc phục với tốc độ và độ chính xác cao hơn con người. Điều này hình thành nên khái niệm “kiểm thử an toàn thông minh” (intelligent penetration testing), đây là hướng đi tất yếu giúp nâng cao năng lực phòng thủ, tối ưu hóa nguồn lực và rút ngắn thời gian phản ứng trong chiến lược bảo mật hiện đại.

Chương này sẽ trình bày tổng quan về bối cảnh an toàn thông tin hiện nay, khái niệm và quy trình kiểm thử an toàn, các công cụ kiểm thử phổ biến, cùng với phân tích xu hướng ứng dụng trí tuệ nhân tạo trong bảo mật, qua đó hình thành cơ sở lý thuyết cho việc nghiên cứu và xây dựng công cụ kiểm thử an toàn thông minh trong các chương tiếp theo.

## Bối Cảnh An Toàn Thông Tin Hiện Nay

Trong vài năm gần đây, thế giới chứng kiến một sự thay đổi căn bản về mức độ, tính đa dạng và mức độ tự động hóa của các mối đe dọa mạng. Các tổ chức hiện nay không chỉ phải đối mặt với các cuộc tấn công đơn lẻ mà còn với chuỗi tấn công phối hợp, tận dụng đồng thời lỗ hổng công nghệ, yếu tố con người và chuỗi cung ứng phần mềm.

Theo Báo cáo Điều tra Vi phạm Dữ liệu Toàn cầu (Verizon DBIR 2024), ransomware vẫn là mối đe dọa hàng đầu, và yếu tố con người tiếp tục đóng vai trò then chốt trong phần lớn các sự cố bảo mật; trong nhiều trường hợp, con người chính là điểm yếu làm lộ thông tin hoặc mở đường cho tấn công tiếp theo [1].

Gần đây, tác nhân đe dọa gồm nhiều thành phần: tội phạm mạng vì lợi nhuận (mô hình ransomware-as-a-service, mã độc đánh cắp thông tin), nhóm do nhà nước hậu thuẫn (state-sponsored) thực hiện gián điệp kinh tế/chính trị, hacktivist (các nhóm, cá nhân tấn công mạng chính trị) thực hiện tấn công vì mục đích chính trị/ý thức hệ, và những kẻ lợi dụng sai sót con người (phishing, credential stuffing). Cơ quan An ninh mạng của Liên minh châu Âu ENISA (European Union Agency for Cybersecurity) ghi nhận xu hướng gia tăng các cuộc tấn công nhắm vào nhằm làm gián đoạn hoạt động hoặc khiến hệ thống mất khả năng truy cập (ransomware/denial-of-service) cùng với mối nguy về dữ liệu và chuỗi cung ứng phần mềm điều này cho thấy mục tiêu tấn công ngày càng đa dạng và có tác động lan rộng [2].

Một thay đổi quan trọng trong bối cảnh an ninh mạng hiện nay là sự xuất hiện và lan rộng của tự động hóa cùng trí tuệ nhân tạo (AI) trong cả hai phía: tấn công và phòng thủ. Ở phía tấn công, các dịch vụ tấn công theo mô hình thuê (tấn công như một dịch vụ), công cụ quét tự động quy mô lớn và phần mềm độc hại được cung cấp như dịch vụ đã làm giảm đáng kể rào cản kỹ thuật, giúp tội phạm mạng có thể triển khai các cuộc tấn công nhanh hơn, tinh vi hơn và được tổ chức chặt chẽ hơn. Trong khi đó, ở phía phòng thủ, các tổ chức và chuyên gia an ninh mạng đang tận dụng AI và học máy để phát hiện hành vi bất thường, dự đoán sớm mối đe dọa, cũng như tự động hóa quy trình phản ứng và xử lý sự cố, nhằm rút ngắn thời gian phát hiện, khắc phục, giảm thiểu thiệt hại và nâng cao khả năng phòng vệ chủ động.

Theo TechRadar, tốc độ và tần suất các cuộc tấn công tự động đã tăng mạnh, buộc các tổ chức phải triển khai kiến trúc phòng thủ hiện đại như Zero Trust, EDR, XDR nhằm duy trì khả năng giám sát thời gian thực [4].

Hệ quả: tốc độ phát hiện, phản ứng phải nhanh hơn. Khoảng cách giữa phát hiện và khắc phục (time-to-remediate) vẫn là một vấn đề lớn: chậm trễ trong patching và xử lý lỗ hổng tạo cửa sổ tấn công rộng cho kẻ xấu. DBIR 2024 nhấn mạnh rằng nhiều tổ chức vẫn mất hàng chục ngày để vá hoặc giảm thiểu rủi ro đối với lỗ hổng nghiêm trọng [1].

Một số kỹ thuật và hình thức tấn công phổ biến được ghi nhận trong giai đoạn 2023–2025 bao gồm:

* Ransomware và tống tiền kỹ thuật số: vẫn là hình thức tấn công phổ biến nhất, gây thiệt hại tài chính trực tiếp; mô hình RaaS khiến nhóm tấn công vận hành như doanh nghiệp [1].
* Đánh cắp thông tin định danh (Credential Theft): rò rỉ tài khoản, vượt qua xác thực đa lớp (MFA bypass) và đánh cắp thông tin quan trọng đang gia tăng mạnh. Báo cáo Check Point 2025 ghi nhận số vụ đánh cắp thông tin đăng nhập tăng hơn 30% so với năm 2023 [3].
* Tấn công tự động và khai thác lỗ hổng (Automated Exploitation): các mạng botnet và scan farm (cụm hệ thống quét lớn) được sử dụng để quét hàng loạt lỗ hổng, lên tới hàng chục nghìn yêu cầu mỗi giây, điều này khiến việc phát hiện và lọc nhiễu trở thành thách thức lớn [4].

Lỗ hổng ứng dụng web: theo OWASP, lỗi cấu hình sai, xác thực yếu và injection vẫn là nguồn gốc chủ yếu dẫn đến tấn công web [5].

A diagram of information on a white background

AI-generated content may be incorrect.

Hình 1.1 Minh hoạ các nhóm tác nhân đe doạ, vectơ tấn công, công cụ tấn công và hệ thống phòng thủ tương ứng.

Các cuộc tấn công không chỉ gây mất mát tài chính trực tiếp mà chúng còn làm gián đoạn hoạt động, tổn hại uy tín, rò rỉ dữ liệu khách hàng và đôi khi gây hậu quả cho an ninh quốc gia (khi nhắm vào hạ tầng quan trọng). Báo cáo toàn cầu của WEF và các phân tích của các hãng an ninh lớn chỉ ra rằng sự bất bình đẳng về năng lực phòng thủ giữa tổ chức lớn và SME (small-medium enterprises) đang gia tăng: những tổ chức nhỏ thường dễ bị tổn thương hơn do thiếu nguồn lực và cơ chế phản ứng [7].

Trong bối cảnh đó, phương pháp kiểm thử an toàn truyền thống (thủ công, thực hiện theo chu kỳ) không còn đáp ứng đủ yêu cầu hiện đại:

* Cần tự động hóa để bao quát bề mặt tấn công ngày càng mở rộng.
* Cần phân loại thông minh (triage) để giảm cảnh báo sai (false positives) và tập trung vào lỗ hổng nghiêm trọng.
* Cần tích hợp tri thức thời gian thực (nguồn dữ liệu CVE, chứng chỉ số, threat intelligence).

Do đó, xu hướng tất yếu là kết hợp công cụ quét truyền thống (Nmap, Nikto, Nuclei, crt.sh…) với phân tích dựa trên trí tuệ nhân tạo (AI/LLM) nhằm tạo báo cáo có giá trị cao, giúp đội ngũ bảo mật ưu tiên xử lý và ra quyết định nhanh hơn. Mục tiêu của đề tài này nằm trong xu hướng đó: xây dựng hệ thống kiểm thử an toàn thông minh, có khả năng không chỉ quét mà còn hiểu, phân tích và tự động hóa phần lớn quy trình trinh sát và đánh giá ban đầu giúp giảm tải đáng kể cho chuyên gia bảo mật con người.

## Khái Niệm Và Quy Trình Kiểm Thử An Toàn

### Khái niệm kiểm thử an toàn

Kiểm thử xâm nhập (Pentest) là quá trình đánh giá an toàn của hệ thống, ứng dụng hoặc hạ tầng mạng bằng cách mô phỏng các cuộc tấn công thực tế. Mục tiêu là phát hiện các lỗ hổng bảo mật, cấu hình sai, hoặc điểm yếu trong quy trình vận hành, từ đó đưa ra biện pháp khắc phục trước khi các kẻ tấn công khai thác [6].

Kiểm thử an toàn đóng vai trò quan trọng trong việc:

* Bảo vệ dữ liệu và tài sản quan trọng của tổ chức [8].
* Đảm bảo tuân thủ các tiêu chuẩn bảo mật như ISO 27001, PCI DSS, hoặc NIST [10] [9].
* Tăng khả năng phát hiện sớm các mối đe dọa mới, giảm nguy cơ bị khai thác.

Theo báo cáo của ENISA, việc thực hiện kiểm thử định kỳ giúp giảm 60–70% nguy cơ bị khai thác các lỗ hổng phổ biến, đồng thời cải thiện khả năng phòng thủ trước các tấn công tinh vi [11].

### Quy trình kiểm thử an toàn theo Cyber Kill Chain

Mô hình Cyber Kill Chain do Lockheed Martin đề xuất giúp mô tả các giai đoạn trong một cuộc tấn công mạng[12]. Áp dụng vào kiểm thử an toàn, các bước cơ bản gồm:

1. Thu thập thông tin (Reconnaissance): Xác định mục tiêu và thu thập dữ liệu từ các nguồn công khai (OSINT) [13]. Ví dụ: quét miền phụ, tìm địa chỉ IP, xác định các dịch vụ đang chạy.
2. Công cụ hóa (Weaponization): Chuẩn bị các công cụ, payload hoặc kịch bản kiểm thử để khai thác lỗ hổng [12].
3. Vận chuyển (Delivery): Thử nghiệm đưa payload đến mục tiêu theo cách an toàn, ví dụ thông qua môi trường thử nghiệm [12].
4. Khai thác (Exploitation): Khai thác lỗ hổng đã xác định để đánh giá mức độ nguy hiểm thực tế [12].
5. Cài đặt kiểm soát (Installation): Thực hiện các hành động tạm thời nhằm giữ quyền truy cập trong môi trường kiểm thử, chỉ để đánh giá khả năng duy trì truy cập [12].
6. Kiểm soát từ xa (Command & Control): Mô phỏng việc kiểm soát hệ thống sau khi xâm nhập để đánh giá nguy cơ từ xa [12].
7. Hành động (Actions on Objectives): Thử nghiệm các tác động có thể xảy ra nếu lỗ hổng bị khai thác, ví dụ truy xuất dữ liệu nhạy cảm, leo thang đặc quyền, hoặc phá hủy dữ liệu (chỉ trong môi trường thử nghiệm) [12].

A diagram of a computer virus

AI-generated content may be incorrect.

Hình 1.2 Mô hình cyber kill chain

### Vai trò của kiểm thử an toàn trong thời đại AI

Việc tích hợp trí tuệ nhân tạo (AI) trong kiểm thử giúp tự động hóa và nâng cao hiệu quả [14]:

* Thu thập và phân tích thông tin: AI tổng hợp dữ liệu từ nhiều nguồn, phân loại thông tin quan trọng, giảm sai sót.
* Đánh giá lỗ hổng: AI nhận diện các mẫu (pattern) bất thường và dự đoán lỗ hổng tiềm năng, giảm đáng kể tỷ lệ cảnh báo sai (false positives) và bỏ sót (false negatives).
* Báo cáo và khắc phục: AI sinh báo cáo chi tiết bằng ngôn ngữ tự nhiên, mô tả mức độ nghiêm trọng, gợi ý biện pháp vá lỗ hổng.

Các nghiên cứu gần đây cho thấy AI có thể phân tích kết quả quét và đề xuất phương án khắc phục với độ chính xác 85–92%, giúp kỹ sư bảo mật ra quyết định nhanh chóng và chính xác hơn [14].

## Các công cụ kiểm thử truyền thống

Mục này giới thiệu các công cụ phổ biến trong kiểm thử xâm nhập, phần lớn là công cụ mã nguồn mở hoặc công cụ nền tảng mà bất kỳ đồ án, bài lab hay hệ thống kiểm thử tự động nào cũng nên tích hợp hoặc tham khảo.

### Nmap

Mục đích chính: phát hiện host, cổng mở, dịch vụ chạy trên cổng, phiên bản dịch vụ, và thu thập thông tin hệ điều hành cơ bản; là bước nền tảng để xác định bề mặt tấn công.

Nguyên lý & tính năng nổi bật:

* Nmap dùng kỹ thuật gửi/gói tin tùy biến để kiểm tra trạng thái cổng (open/closed/filtered). Hỗ trợ nhiều kiểu quét: SYN scan (fast, stealth), TCP connect, UDP scan, Idle scan (IPID side-channel), …; đồng thời có khả năng dò tìm phiên bản dịch vụ (version detection) và nhận diện hệ điều hành (OS detection) [15].
* Nmap Scripting Engine (NSE): cho phép viết script bằng Lua để tự động hoá kiểm tra chuyên sâu (ví dụ kiểm tra lỗ hổng cụ thể, tấn công vét cạn, thu thập thông tin). NSE là lý do Nmap không chỉ là công cụ “quét port” mà còn dùng để tìm các lỗ hổng đơn giản [15].

Các tùy chọn và chiến lược thực tế hay dùng:

* -sS (SYN scan) + -Pn (không ping) cho quét nhanh stealth.
* -sV để dò version, kết hợp --script để chạy các NSE script phù hợp (ví dụ vuln, auth, discovery).
* Quét theo dải IP lớn: phân đoạn, điều phối công việc (parallelization) để tránh quá tải mạng/IDS.

Tích hợp tự động: Nmap có đầu ra XML/grepable/JSON, dễ chuẩn hóa bằng Python (thư viện python-nmap hoặc trực tiếp chuẩn hóa XML) để lưu kết quả vào cơ sở dữ liệu, kết nối với mô hình phân tích sau đó [15].

Lưu ý khi sử dụng: cần xin phép/ghi rõ phạm vi; tốc độ quét cao có thể gây gián đoạn dịch vụ hoặc bị hệ thống phát hiện; các kết quả OS/version có thể sai nếu hệ thống được cấu hình tối ưu chống nhận dạng đặc trưng (fingerprinting).

### Nikto

Mục đích chính: kiểm tra cấu hình và lỗ hổng tại lớp web server, file nhạy cảm, CGI/backup files, cấu hình SSL/TLS yếu, header nguy hiểm, phiên bản server lỗi thời.

Nguyên lý & tính năng nổi bật:

* Nikto thực hiện hàng nghìn kiểm tra (test signatures) chống lại một máy chủ web và đưa ra danh sách các điểm nguy cơ (thông tin có thể cần xác minh thủ công). Nikto kiểm tra hàng nghìn mục (thông tin nhà phát triển cho biết con số lớn ~7.000+ tests) [16].
* Hỗ trợ nhiều định dạng báo cáo (txt, html, csv, xml) thuận tiện cho việc tích hợp vào chuỗi kiểm thử tự động [16].

Cách dùng thực tế:

* Chạy Nikto cho từng host/virtual-host: nikto -h http://target hoặc từ container. Sử dụng tùy chọn output để lưu kết quả cho mô hình phân tích.
* Kết hợp Nikto với Burp Suite (proxy) để thực hiện kiểm tra sâu hơn trên endpoint mà Nikto phát hiện.

Lưu ý: Nikto thường phát hiện nhiều “vấn đề cấu hình” nhưng có thể có false positives, cần bước xác minh thủ công (hoặc dùng AI để enrich và loại bỏ false positives) [15].

### Nuclei

Mục đích chính: quét nhanh, quy mô lớn dựa trên template mô tả các mẫu đặc trưng của lỗ hổng (CVE, misconfiguration, endpoint nguy hiểm).

Nguyên lý & tính năng nổi bật:

* Nuclei sử dụng các template YAML (cộng đồng đóng góp) để mô tả cách phát hiện từng lỗ hổng cụ thể, điều này cho phép cập nhật nhanh khi có CVE mới và tạo template tuỳ chỉnh cho môi trường đặc thù [17].
* Thiết kế cho tốc độ với khả năng quét song song và mở rộng tốt. Nuclei thường dùng cho quét hàng loạt hosts/subdomains rất hiệu quả [17].

Chiến lược sử dụng:

* Dùng Nuclei cho giai đoạn quét mở rộng (sau khi thu thập subdomain) để lọc nhanh các endpoint nghi vấn; kết quả quét vào mô hình enrich (AI) để phân loại mức độ nghiêm trọng.
* Duy trì kho template: tự động cập nhật template từ cộng đồng để đảm bảo phát hiện CVE mới.

Lưu ý: mặc dù nhanh, Nuclei phụ thuộc vào chất lượng template, template yếu hoặc lỗi có thể gây false negative/positive; cần chính sách quản lý template (cơ chế kiểm duyệt, cách ly và rà soát định kỳ) [17].

### crt.sh

Mục đích chính: tìm miền phụ bằng cách tra cứu các chứng chỉ SSL/TLS đã được phát hành cho domain (Certificate Transparency logs). Đây là một nguồn OSINT hữu hiệu để mở rộng phạm vi mục tiêu.

Nguyên lý & sử dụng:

* crt.sh là công cụ (giao diện web và truy vấn) để tra cứu các bản ghi trong Certificate Transparency và liệt kê các tên miền con đã xuất hiện trên chứng chỉ. Bằng cách phân tích tên trong các chứng chỉ, có thể thu thập miền phụ ít được chú ý [18].
* Kết hợp crt.sh với các nguồn OSINT khác (Passive DNS, sublist3r, Amass) để có danh sách miền phụ toàn diện.

Lưu ý: crt.sh trả về dữ liệu lịch sử, cần lọc kết quả cũ/không hợp lệ; một số miền phụ có thể là tạm thời hoặc do sinh ra tự động cho mục đích thử nghiệm. Việc xác thực tồn tại thực tế (HTTP/HTTPS) vẫn cần thiết để đảm bảo độ chính xác của kết quả.

### Metasploit

Mục đích chính: phát triển, kiểm thử và thực thi khai; xác nhận lỗ hổng bằng PoC/khai thác thực tế trong môi trường kiểm thử được phép.

Nguyên lý & tính năng: Metasploit là framework dạng mô‑đun (exploit, payload, auxiliary modules). Hỗ trợ từ thử nghiệm khai thác đến tạo payload (meterpreter), các kỹ thuật hậu khai thác (post‑exploitation) như leo thang đặc quyền và di chuyển giữa các máy trong nội bộ hệ thống [19].

Ứng dụng thực tế: sau khi phát hiện lỗ hổng từ Nmap/Nuclei, kiểm thử viên có thể dùng Metasploit để kiểm chứng mức độ khai thác (exploit validation), lưu ý phải giới hạn phạm vi và luôn có phép.

Lưu ý: Metasploit rất mạnh nhưng cũng dễ gây hậu quả (payload gây crash, backdoor) nên chỉ dùng trong phạm vi cho phép và nên chạy trên môi trường kiểm thử được phép.

### Burp Suite

Mục đích chính: hỗ trợ kiểm thử ứng dụng web từ mức tương tác thủ công (proxy, repeater, intruder) đến quét tự động (Burp Scanner trong bản Pro).

Tính năng chính: proxy HTTP(S) để can thiệp request/response, repeater để kiểm thử thủ công, intruder cho fuzzing, scanner cho quét tự động, collaborator để thu thập OOB interaction [20].

Tích hợp workflow: thường dùng sau bước quét tự động để phân tích sâu các endpoint nghi vấn, sinh PoC cho lỗ hổng logic, authentication bypass, XSS, SQLi…

Lưu ý: Burp Pro là công cụ trả phí; Burp Community có giới hạn. Tùy chọn tự động hoá qua Burp extensions (BApp store) hữu ích khi tích hợp vào chuỗi khai thác tự động.

## Xu hướng tích hợp AI vào kiểm thử an toàn

Tích hợp trí tuệ nhân tạo (AI) vào kiểm thử an toàn đang trở thành xu hướng tất yếu và có ảnh hưởng sâu rộng đến mọi khâu của quy trình kiểm thử, từ thu thập thông tin, phân tích lỗ hổng, ưu tiên xử lý đến soạn thảo báo cáo và hỗ trợ quyết định. AI không còn là khái niệm tương lai xa; các mô hình học máy (ML), học sâu (DL) và đặc biệt là các mô hình ngôn ngữ lớn (LLM) đã bắt đầu được áp dụng để tự động hóa những nhiệm vụ lặp đi lặp lại, rút ngắn thời gian phân tích và tăng tỷ lệ phát hiện những lỗ hổng mới hoặc phức tạp, hỗ trợ phân loại và ưu tiên lỗ hổng theo ngữ cảnh tổ chức, đồng thời sinh các nội dung báo cáo kỹ thuật và tóm tắt điều hành. Nhiều bài nghiên cứu và dự án prototype đã chứng minh rằng khi được thiết kế và kiểm soát đúng, AI có thể rút ngắn chu kỳ kiểm thử, giảm chi phí vận hành và tăng tỉ lệ phát hiện các vấn đề phức tạp mà phương pháp signature/rule-based truyền thống khó bắt được [21].

### Vai trò cụ thể của AI trong chuỗi kiểm thử

AI đóng vai trò ở nhiều tầng khác nhau của chuỗi kiểm thử. Ở tầng thu thập và tiền xử lý, AI hỗ trợ tối ưu hoá dữ liệu OSINT: lọc miền phụ, phân loại host theo mức độ xâm phạm, chuẩn hoá log và chuyển đổi đầu ra đa nguồn về một cấu trúc chung để thuận tiện cho phân tích tiếp theo. Ở tầng phát hiện, mô hình ML/DL có thể phát hiện bất thường dựa trên mẫu lưu lượng, chuỗi request hoặc kết quả quét mà các tập luật tĩnh có thể bỏ sót. Ở tầng phân tích, LLM có khả năng đọc hiểu ngữ nghĩa (code, HTTP response, log), từ đó sinh ra giải thích chi tiết, gợi ý priority và thậm chí đề xuất PoC hoặc bước khai thác tiếp theo, tuy nhiên mọi đề xuất này cần được xác thực trước khi thực hiện trên môi trường triển khai. Các công cụ dạng thử nghiệm như PentestGPT cho thấy LLM có thể tổng hợp nhiều nguồn thông tin để hướng dẫn luồng kiểm thử, nhưng vẫn cần con người trong vòng lặp kiểm soát cuối cùng. Những ứng dụng này đã được ghi nhận trong nhiều nghiên cứu gần đây, cho thấy AI có thể làm tăng tính hiệu quả của quy trình kiểm thử nhưng đồng thời đặt ra nhiều thách thức về độ tin cậy, an toàn dữ liệu và đạo đức [22].

A diagram of a diagram

AI-generated content may be incorrect.

Hình 1.3 Vai trò của AI trong kiểm thử an toàn

### Các Kỹ thuật đang được áp dụng

Đi sâu vào khả năng thực tế, LLM (ví dụ các dòng GPT) đã chứng minh năng lực nổi bật trong việc phân tích ngôn ngữ tự nhiên và mã nguồn, vì vậy chúng phù hợp để hỗ trợ phân tích code bảo mật, mô tả lỗ hổng bằng ngôn ngữ dễ hiểu, sinh bản vá gợi ý và tạo phần tóm tắt điều hành (executive summary) cho báo cáo. Nhiều hướng kỹ thuật đã được thử nghiệm:

* LLM-empowered workflows: dùng LLM để phân tích đầu ra các scanner, viết reporter template, hướng dẫn bước kế tiếp.
* Reinforcement learning (RL) để học chuỗi hành động tối ưu trong môi trường mô phỏng, ví dụ như trong công cụ DeepExploit hoặc các nghiên cứu về RL cho hậu khai thác (post‑exploit).
* Hybrid ML + rules systems để kết hợp ưu điểm của cả hai (sử dụng rules để chặn false positive rõ ràng, dùng ML để phát hiện mẫu phức tạp).

Các dự án thực nghiệm như PentestGPT hoặc các dự án mã nguồn mở tương tự cho thấy LLM có thể hướng dẫn người trong việc chọn công cụ, thiết lập lệnh quét và diễn giải kết quả, nhưng nghiên cứu cũng ghi nhận LLM thường “bịa” các lệnh hoặc gợi ý sai tức là hiện tượng “hallucination” vẫn là vấn đề lớn cần quản lý.điều này gây giới hạn lớn khi chuyển sang môi trường thật do rủi ro gây hỏng hệ thống và đa dạng hoá môi trường mục tiêu.

Ngoài LLM, reinforcement learning (RL) và các phương pháp học tăng cường đã được áp dụng cho các nhiệm vụ tự động hóa tấn công, như chọn chuỗi hành động (scanning → exploit → post-exploit) để tối ưu hóa việc tìm kiếm leo thang đặc quyền hay tự động làm khai thác hậu xâm nhập. Những nghiên cứu trong vài năm gần đây đã trình diễn rằng RL có thể học chuỗi hành động hiệu quả trong môi trường mô phỏng, tuy vậy việc chuyển trực tiếp sang môi trường thực tế gặp khó khăn do tính phi tuyến, đa dạng hệ điều hành, và rủi ro gây hỏng hệ thống. Vì vậy, ứng dụng RL thực tế thường giới hạn trong môi trường lab/staging hoặc dùng để gợi ý cho chuyên gia chứ không tự động thực hiện trên hạ tầng sản xuất [24].

### Lợi ích, thách thức và rủi ro khi tích hợp AI

AI giúp tăng tốc quá trình phân tích bằng cách tự động tóm tắt và phân loại hàng nghìn kết quả phát hiện (findings), giảm khối lượng công việc thủ công trong các bước sàng lọc (triage) và báo cáo, đồng thời ưu tiên các lỗ hổng dựa trên bối cảnh doanh nghiệp (business impact) và mở rộng khả năng quét liên tục (continuous pentest). Khi được kết hợp chặt chẽ với các luồng chuẩn hóa dữ liệu, AI còn có thể nâng cao tỷ lệ phát hiện các lỗ hổng logic hoặc phức tạp mà các công cụ scanner truyền thống dễ bỏ sót. Những lợi ích này đã được chứng minh thông qua các thử nghiệm học thuật và các dự án mã nguồn mở [21].

Tuy AI đem lại nhiều lợi ích, tồn tại các rủi ro lớn cả về kỹ thuật lẫn pháp lý. Về mặt kỹ thuật, vấn đề “hallucination”(tạo thông tin sai) của LLM có thể khiến hệ thống sinh ra kết luận sai hoặc sinh lệnh nguy hiểm; bản thân các mô hình sinh mã (code generation) cũng thường tạo ra đoạn mã có lỗi bảo mật. Các báo cáo ngành gần đây thậm chí cho thấy tỉ lệ phần trăm không nhỏ mã do AI sinh ra chứa lỗ hổng cần khắc phục. Về mặt pháp lý và đạo đức, AI có thể xử lý dữ liệu nhạy cảm (ví dụ token, thông tin người dùng) khiến nguy cơ rò rỉ gia tăng khi không có cơ chế masking/mã hóa; đồng thời agent tự động nếu không được giám sát chặt chẽ có thể vượt quá phạm vi cho phép và gây thiệt hại thực tế. Các phân tích an toàn mô hình và các bản đánh giá rủi ro do nhà cung cấp công bố cũng nhấn mạnh rằng các mô hình hiện tại vẫn có “mức rủi ro” nhất định và cần biện pháp giảm thiểu khi tích hợp vào chu trình bảo mật. Do đó, trong mọi triển khai thực tế, AI phải được cấu trúc như một “trợ lý thông minh” cung cấp gợi ý, còn quyết định hành động cuối cùng phải do con người hoặc hệ thống kiểm soát có điều kiện thực hiện [24].

A black and white text on a white background

AI-generated content may be incorrect.

Hình 1.4 Rủi ro và biện pháp giảm thiểu khi dùng AI trong kiểm thử

Xét về mặt lợi ích chiến lược, tích hợp AI giúp nâng cao khả năng mở rộng của hoạt động kiểm thử: hệ thống có thể chạy quét định kỳ, ưu tiên tự động những phát hiện mang tính rủi ro cao, sinh báo cáo kỹ thuật và bản tóm tắt cho lãnh đạo, từ đó đẩy nhanh chu kỳ khắc phục lỗ hổng và giảm chi phí vận hành so với phương thức thủ công hoàn toàn. Tuy nhiên sự phụ thuộc vào AI cũng cần đi kèm với cơ chế đo lường: thiết lập các chỉ số hiệu suất rõ ràng như tỷ lệ dương tính giả (false positive), thời gian từ phát hiện đến phân loại xử lý, độ chính xác phân loại khi phân loại mức độ nghiêm trọng do AI gợi ý so với chuyên gia, để đảm bảo hệ thống thực sự cải thiện chất lượng kiểm thử. Nhiều nghiên cứu cho rằng AI nên được coi là một phần của đội kiểm thử xâm nhập (con người + AI), nơi AI hỗ trợ công việc lặp lại và cung cấp thông tin chi tiết, còn con người chịu trách nhiệm ra quyết định cuối cùng [25].

### Triển vọng tương lai của AI trong kiểm thử an toàn

Tương lai của kiểm thử sẽ là “con người + AI” làm việc phối hợp: AI chịu trách nhiệm tự động hoá, phân tích khối lượng lớn và cung cấp thông tin hữu ích , những hệ thống này sẽ kết hợp khả năng đọc mã nguồn, chạy test tự động trong môi trường sandbox, xác minh các cách khai thác trong môi trường mô phỏng và sinh các gợi ý còn con người giữ vai trò kiểm soát chiến lược, phê duyệt hành động nguy hiểm và xử lý những tình huống chưa có tiền lệ. Xu hướng phát triển agent tự chủ (như Aardvark) cho thấy khả năng tăng tốc mạnh mẽ nhưng đồng thời đòi hỏi khuôn khổ pháp lý, tiêu chuẩn an toàn và Cơ chế kiểm tra/ghi nhật ký tập trung trước khi được áp dụng rộng rãi trên môi trường triển khai. Tóm lại, AI là công cụ có tiềm năng thay đổi căn bản cách thức kiểm thử an toàn được thực hiện, nhưng để đạt được lợi ích bền vững cần tích hợp cẩn trọng, minh bạch và luôn giữ con người ở vai trò kiểm soát cuối cùng.

## Kết luận chương 1

Chương 1 đã trình bày tổng quan về bối cảnh an toàn thông tin hiện nay, các nguyên lý và công cụ kiểm thử an toàn truyền thống, đồng thời làm rõ xu hướng tất yếu của việc ứng dụng trí tuệ nhân tạo vào lĩnh vực này. Qua phân tích, có thể thấy rằng các công cụ hiện nay dù mạnh mẽ nhưng vẫn rời rạc và thiếu tính tự động, trong khi AI mang lại khả năng tổng hợp, phân tích và ra quyết định thông minh.

Sự kết hợp giữa AI và kiểm thử an toàn không chỉ là bước tiến kỹ thuật mà còn là hướng phát triển chiến lược, giúp hình thành hệ sinh thái kiểm thử chủ động, thông minh và liên tục. Từ những cơ sở đó, đề tài “Nghiên cứu và xây dựng công cụ kiểm thử an toàn thông minh” được lựa chọn nhằm mục tiêu tạo ra giải pháp mới đáp ứng yêu cầu tự động hóa, chính xác và phù hợp với thực tiễn ứng dụng trong bối cảnh an toàn thông tin hiện nay.

# PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## Kiến trúc tổng thể hệ thống

### Mục tiêu thiết kế

Hệ thống kiểm thử an toàn thông minh (Pentest-Auto) được thiết kế với mục tiêu tự động hóa toàn bộ quy trình kiểm thử bảo mật từ khâu trinh sát, quét lỗ hổng, phân tích kết quả đến sinh báo cáo. Để đạt được điều này, hệ thống cần đảm bảo các tiêu chí sau:

* Tự động hóa toàn bộ quy trình kiểm thử bảo mật: từ khi người dùng nhập URL đến khâu trinh sát, quét mạng, quét web đến phân tích lỗ hổng và sinh báo cáo kết quả đến người dùng.
* Tích hợp công cụ kiểm thử truyền thống: sử dụng các công cụ mạnh mẽ như Nmap, Nikto, Nuclei, crt.sh để đảm bảo độ tin cậy và khả năng phát hiện lỗ hổng thực tế.
* Tích hợp AI: sử dụng mô hình ngôn ngữ lớn (LLM) để phân tích kết quả, phân tích lỗ hổng và đề xuất khai thác, giúp hỗ trợ đánh giá mức độ nghiêm trọng của lỗ hổng.
* Cung cấp giao diện web đơn giản, dễ sử dụng: cung cấp giao diện sử dụng đơn giản để người dùng không chuyên cũng có thể sử dụng.
* Khả năng mở rộng: dễ dàng thêm công cụ mới, thay đổi mô hình AI, hoặc triển khai trên nhiều môi trường khác nhau.

### Các thành phần chính của hệ thống

#### Frontend (Giao diện người dùng)

Được xây dựng bằng HTML, CSS và JavaScript với các file pentest\_web/ frontend/index.html, pentest\_web/frontend/add\_api\_key.html , pentest\_web/ frontend /app.js, style.css. Cho phép người dùng nhập URL hoặc domain cần kiểm thử, chọn mô hình AI, thêm cookie nếu cần, và khởi chạy quá trình quét thông qua form nhập target + model + cookie, quản lý API keys, khởi chạy quét (POST /api/scan), kiểm tra trạng thái quét (GET /api/status/{job\_id}), gửi kết quả (GET /api/results/\*) và tạo bảng. Hiển thị kết quả kiểm thử dưới dạng bảng, JSON hoặc raw text và giao tiếp với backend thông qua các API RESTful.

#### Backend API (FastAPI)

Được xây dựng bằng Python sử dụng framework FastAPI với chức năng nhận yêu cầu scan, API quản lý API keys, endpoint kiểm tra trạng thái, endpoint trả kết quả. Backend sẽ bao gồm các endpoints chính:

* POST /api/scan: khởi chạy quét (main entry: main.py, orchestrator\_runner.py).
* GET /api/status/{job\_id}: kiểm tra trạng thái quét (jobs dict in-memory).
* GET /api/results/{tool} và GET /api/results/job/{job\_id}: lấy kết quả theo tool hoặc theo job (results.py).
* GET/POST: /api/api-keys. quản lý API key

Các file liên quan bao gồm Main entry: main.py và các file điều phối đến các mô hình kiểm thử và AI thông qua orchestrator: orchestrator\_runner.py, results.py, model.py. Main entry: main.py

#### Orchestrator (Điều phối kiểm thử)

Là trung tâm điều phối toàn bộ quy trình kiểm thử. Thực hiện điều phối luồng công việc theo kịch bản các bước tuần tự: thu thập subdomain → resolve IP → quét mạng → quét web → phân tích bằng AI → phân tích endpoint → lưu kết quả. Orchestrator sẽ được triển khai trong file orchestrator\_interactive.py. (hàm run\_scan thực thi luồng kiểm thử tự động), orchestrator\_runner.py nhận request và hàng đợi job.

#### Scanners (Công cụ quét)

Gồm ba mô hình chính:

* nmap\_runner.py: quét mạng, phát hiện dịch vụ và hệ điều hành. Bằng cách chạy Nmap (xml output), phân tích và đọc dữ liệu từ file/XML thành cấu trúc dữ liệu dạng từ điển sau khi đã giải dữ liệu (các dịch vụ), trả về dict gồm parsed (kết quả đã được phân tích), stdout (chuỗi string chứa đầu ra chuẩn), stderr (chuỗi chứa lỗi hoặc thống báo lỗi), ts (dấu thời gian). (Nên cài Nmap trên host; sử dụng cờ mặc định NMAP\_DEFAULT).
* nikto\_runner.py: quét lỗ hổng trên web server bằng Nikto, đọc dữ liệu đầu ra, chuẩn hoá thành các bản ghi (records).
* nuclei\_runner.py: quét lỗ hổng theo template CVE. Bằng cách tạo câu lệnh cmd cho Nuclei, chạy, chuẩn hóa JSONL hoặc đọc stdout, trả về danh sách normalized findings + raw path.

Các mô hình scanner được thiết kế làm wrapper gọi tiến trình ngoài (subprocess) và chuyển tiếp output vào outputs/<tool>\_results.jsonl

#### Collectors (Thu thập thông tin)

crtsh\_collector.py: truy vấn crt.sh để thu thập danh sách miền phụ của domain mục tiêu, chuẩn hoá danh sách miền phụ và lưu vào file ra outputs/subs\_results.jsonl và mở rộng phạm vi kiểm thử, giúp phát hiện các miền phụ tiềm ẩn rủi ro.

#### AI Layer (Phân tích bằng trí tuệ nhân tạo)

Thực hiện công việc quan trong nhất bao gồm phân tích, đánh giá thông tin thu thập được từ các bước trên. Bao gồm:

* ai\_common.py: quản lý API key (analyzers/data/api\_keys.txt), tạo session với retry/backoff, gọi OpenAI API thông qua hàm ask\_openai với API key được lưu trữ , xử lý lỗi và quay vòng key.
* ai\_enricher.py: chuẩn hóa thông tin dịch vụ bằng cách đọc các kết quả quét từ các công cụ scan bằng file JSONL (load\_nmap\_services, load\_nuclei\_findings, load\_nikto\_findings), tìm CVE liên quan thông qua các hàm ai\_find\_cves\_for\_service, enrich\_from\_parsed\_nmap để ghi outputs/enriched\_nmap\_results.jsonl., phân tích kết quả quét từ Nuclei và Nikto.
* ai\_recursive\_probe.py: phát hiện endpoint bằng cách dùng LLM ai\_detect\_endpoints\_only(html, base\_url) , phân tích tham số, thử payload từ vulns\_payloads.json và đánh giá khả năng khai thác.

#### Storage (Lưu trữ kết quả)

Sau cùng, kết quả được lưu dưới dạng file .jsonl trong thư mục outputs/, bao gồm:

* nmap\_results.jsonl, nikto\_results.jsonl, nuclei\_results.jsonl
* enriched\_nmap\_results.jsonl, enriched\_nikto\_results.jsonl, enriched\_ nuclei\_results.jsonl
* subs\_results.jsonl, endpoints\_results.jsonl, vulns\_results.jsonl

Một số thông tin tổng hợp được lưu trong cơ sở dữ liệu SQLite (assets.db), phục vụ cho việc truy vấn nhanh và báo cáo.

### Luồng hoạt động tổng thể

Dưới đây là mô tả luồng hoạt động của hệ thống từ khi người dùng khởi chạy kiểm thử đến khi nhận được kết quả:

1. Người dùng nhập URL hoặc domain trên giao diện web và nhấn nút "Quét".
2. Frontend gửi yêu cầu POST đến API /api/scan, kèm theo thông tin target, model AI, cookie và tùy chọn xóa dữ liệu cũ.
3. Backend tạo một job mới, lưu trạng thái là "queued", và khởi chạy background task run\_job.
4. Orchestrator thực hiện các bước kiểm thử:
   * Truy vấn crt.sh để thu thập subdomain.
   * Resolve IP của các subdomain.
   * Quét mạng bằng Nmap.
   * Quét web bằng Nikto và Nuclei.
   * Enrich kết quả bằng AI.
   * Phân tích endpoint và thử payload.
5. Kết quả được lưu vào thư mục outputs/ dưới dạng JSONL và SQLite.
6. Frontend liên tục kiểm tra trạng thái job qua API /api/status/{job\_id}.
7. Khi job hoàn tất, người dùng có thể xem kết quả qua các tab: Subdomains, Scanners, AI Enrich, AI Recursive.

A diagram of a computer hardware system

AI-generated content may be incorrect.

Hình 2.1 Sơ đồ luồng hoạt động tổng thể

## Phân tích chi tiết từng mô hình chức năng

### Mô hình Thu Thập Subdomain crt.sh

Trong kiểm thử bảo mật, việc mở rộng phạm vi kiểm thử là yếu tố then chốt để phát hiện các điểm yếu tiềm ẩn. Nhiều hệ thống không chỉ chạy trên domain chính mà còn phân tán trên các miền phụ như admin.example.com, api.example.com, dev.example.com, v.v. Những miền phụ này thường bị bỏ quên trong quá trình bảo trì, cập nhật, hoặc kiểm thử, dẫn đến nguy cơ tồn tại lỗ hổng nghiêm trọng.

Mô hình crtsh\_collector.py được xây dựng để tự động thu thập danh sách subdomain của một domain mục tiêu thông qua dịch vụ công khai [crt.sh](https://crt.sh/), vốn lưu trữ các chứng chỉ số SSL/TLS đã được cấp phát. Bằng cách phân tích các chứng chỉ này, ta có thể suy ra các subdomain đã từng được sử dụng hoặc đang hoạt động.

Luồng hoạt động tổng quát:

1. Nhận domain đầu vào từ orchestrator.
2. Gửi truy vấn đến https://crt.sh/?q=%.domain&output=json.
3. Nhận kết quả dưới dạng JSON hoặc text.
4. Parse và chuẩn hóa tên miền:

* Loại bỏ wildcard (\*.).
* Loại bỏ dấu chấm cuối (.).
* Lọc ký tự không hợp lệ.
* Chỉ giữ các tên kết thúc bằng domain chính.

1. Trả về danh sách subdomain duy nhất, đã chuẩn hóa, sắp xếp.

Trích mã nguồn và phân tích chức năng:

Đoạn mã chính nằm trong file collectors/crtsh\_collector.py:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | def fetch\_crtsh\_subdomains(domain: str, timeout: int = 30):  q = quote(f"%.{domain}")  url = f"{CRT\_BASE}?q={q}&output=json"  s = requests\_session()  try:  resp = s.get(url, timeout=timeout)  resp.raise\_for\_status()  except Exception as e:  raise RuntimeError(f"crt.sh request failed: {e}")  try:  data = resp.json()  except ValueError:  data = \_safe\_parse\_json\_lines(resp.text)  subs = set()  domain\_dot = "." + domain.lower()  for item in data:  name = item.get("name\_value") or item.get("common\_name") or ""  for line in str(name).splitlines():  line = line.strip().lower()  if not line:  continue  if line.startswith("\*."):  line = line[2:]  line = line.strip().rstrip(".")  if not re.match(r"^[a-z0-9\.-]+$", line):  continue  if line == domain.lower() or line.endswith(domain\_dot):  subs.add(line)  return sorted(subs) |

* **Truy vấn crt.sh**: sử dụng cú pháp “%.domain” (cú pháp SQL LIKE được crt.sh hỗ trợ. Dấu % đại diện cho bất kỳ chuỗi nào, còn “.” là dấu phân cách subdomain) để tìm mọi chứng chỉ có subdomain của domain. Với domain example.com, truy vấn %.example.com sẽ trả về mọi chứng chỉ có tên như www.example.com, api.example.com, v.v. Đây là cú pháp phổ biến trong OSINT để mở rộng phạm vi kiểm thử.
* Gửi yêu cầu HTTP và xử lý lỗi: sử dụng requests\_session() tạo một session có retry/backoff để tăng độ ổn định sau đó gửi yêu cầu HTTP GET đến crt.sh và kiểm tra mã trạng thái HTTP thông qua raise\_for\_status().
* **Xử lý kết quả**:
* Nếu kết quả là JSON, parse trực tiếp bằng resp.json() .
* Nếu không phải JSON (do lỗi hoặc định dạng khác), dùng \_safe\_parse\_json\_lines để xử lý từng dòng bằng cách cố gắng parse từng dòng thành JSON object.
* **Chuẩn hóa tên miền:**
* Dùng line.startswith("\*.") để xác định các chứng chỉ có tên như \*.example.com, \*.api.example.com nhằm loại bỏ wildcard \*. để tránh trùng lặp không cần thiết.
* Dùng line.strip().rstrip(".") Loại bỏ dấu chấm cuối để tránh sai lệch khi so sánh.
* Dùng regex ^[a-z0-9\.-]+$ để lọc tên miền có ký tự lạ (ví dụ: unicode, khoảng trắng).
* Dùng line == domain.lower() or line.endswith(domain\_dot) để giữ lại các tên miền có hậu tố chính xác (ví dụ: api.example.com hợp lệ, api.fakeexample.com bị loại).
* **Kết quả trả về**: danh sách subdomain duy nhất, đã chuẩn hóa, sắp xếp theo thứ tự chữ cái.

### Mô hình Quét Mạng Nmap

Quét mạng là bước quan trọng trong kiểm thử bảo mật nhằm xác định: Các địa chỉ IP đang hoạt động, các cổng (port) đang mở, các dịch vụ đang chạy trên từng cổng với các lỗ hổng tiềm năng thông qua script NSE và phiên bản phần mềm, hệ điều hành, thông tin bổ sung (banner, CPE).

Mô hình nmap\_runner.py được xây dựng để tự động hóa việc gọi công cụ Nmap, thu thập kết quả dưới dạng XML, chuẩn hóa dữ liệu và chuẩn hóa thành cấu trúc JSON dễ xử lý. Kết quả này sẽ được lưu vào file và cơ sở dữ liệu để phục vụ các bước enrich bằng AI và phân tích tiếp theo.

Luồng hoạt động tổng quát:

1. Nhận địa chỉ IP cần quét từ orchestrator.
2. Gọi Nmap với các tùy chọn mạnh mẽ (-sS, -sV, -A, v.v.).
3. Nhận kết quả dưới dạng XML qua stdout.
4. Parse XML để trích xuất thông tin dịch vụ, hệ điều hành, CPE.
5. Trả về kết quả dưới dạng dictionary gồm: ip, success, stdout, stderr, ts, parsed, cmd.
6. Lưu kết quả vào file JSONL và SQLite.

Trích mã nguồn và phân tích chức năng:

Đoạn mã chính nằm trong file collectors/crtsh\_collector.py:

**Cấu hình lệnh Nmap:**

|  |  |
| --- | --- |
| 1 | NMAP\_DEFAULT = ["nmap", "-sS", "-sV", "-Pn", "-A", "-T4", "--min-rate", "2000"] |

* Chức năng: xác định các tùy chọn mặc định khi gọi Nmap.
* Giải thích:
* -sS: quét SYN (stealth scan), phổ biến và nhanh.
* -sV: phát hiện phiên bản dịch vụ.
* -Pn: bỏ qua bước ping, quét luôn cả host không phản hồi ICMP.
* -A: bật OS detection, script scan, traceroute.
* -T4: tăng tốc độ quét.
* --min-rate 2000: yêu cầu tốc độ tối thiểu 2000 packet/giây.
* Ý nghĩa:
* Tập hợp các tùy chọn giúp thu được nhiều thông tin nhất trong thời gian ngắn.
* Phù hợp cho kiểm thử nội bộ hoặc môi trường cho phép quét mạnh.

**Chức năng gọi Nmap và xử lý kết quả:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | def run\_nmap(ip, timeout=900):  args = NMAP\_DEFAULT + [ip, "-oX", "-"]  p = subprocess.run(args, check=True, stdout=subprocess.PIPE,  stderr=subprocess.PIPE, text=True, timeout=timeout)  xml\_out = p.stdout or ""  parsed = parse\_nmap\_xml(xml\_out)  return {  "ip": ip,  "success": True,  "stdout": xml\_out,  "stderr": p.stderr,  "ts": ts(),  "parsed": parsed,  "cmd": args  } |

* Chức năng: thực thi lệnh Nmap và xử lý kết quả.
* Giải thích:
* subprocess.run(...): gọi Nmap và lấy kết quả qua stdout.
* -oX -: yêu cầu Nmap xuất kết quả dưới dạng XML ra stdout.
* parse\_nmap\_xml(xml\_out): gọi hàm phân tích XML để trích xuất thông tin.
* Trả về dictionary chứa toàn bộ thông tin quét.
* Xử lý lỗi:
* Nếu Nmap trả về lỗi (non-zero exit code), vẫn cố gắng parse stdout.
* Nếu bị timeout, trả về trạng thái thất bại.

**Chức năng phân tích kết quả XML:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | def parse\_nmap\_xml(xml\_text):  parsed = {"services": [], "hosts": []}  root = ET.fromstring(xml\_text)  for host in root.findall("host"):  host\_info = {"addresses": [], "hostnames": [], "services": []}  ...  for port in host.findall(".//port"):  portid = port.get("portid")  proto = port.get("protocol")  state\_el = port.find("state")  state = state\_el.get("state") if state\_el is not None else None  svc = port.find("service")  svc\_name = svc.get("name") if svc is not None else None  product = svc.get("product") if svc is not None else None  version = svc.get("version") if svc is not None else None  extrainfo = svc.get("extrainfo") if svc is not None else None  cpe\_list = [cpe.text for cpe in port.findall(".//cpe") if cpe.text]  svc\_obj = {  "port": portid,  "proto": proto,  "state": state,  "service": svc\_name,  "product": product,  "version": version,  "extrainfo": extrainfo,  "cpe": cpe\_list  }  host\_info["services"].append(svc\_obj)  parsed["services"].append(svc\_obj)  parsed["hosts"].append(host\_info)  return parsed |

* Chức năng: phân tích XML để trích xuất thông tin dịch vụ.
* Giải thích:
* Dùng root = ET.fromstring(xml\_text) để chuẩn hóa chuỗi XML thành cây DOM.
* Duyệt qua từng host trong XML vừa chuẩn hóa được, với mỗi host, khởi tạo dict host\_info để lưu thông tin riêng biệt.
* Chạy vòng for (for port in host.findall(".//port")) nhằm tìm toàn bộ port trong host
* Sau khi có được port, thực hiện trích xuất:
  + port, protocol, state: thông tin cơ bản.
  + service, product, version, extrainfo: banner dịch vụ.
  + cpe: Common Platform Enumeration — định danh phần mềm chuẩn hóa.
* Lưu vào parsed["services"] để phục vụ enrich bằng AI.
* Ý nghĩa:
* Đây là dữ liệu đầu vào quan trọng để tìm CVE, đánh giá mức độ nguy hiểm.
* CPE giúp liên kết với cơ sở dữ liệu lỗ hổng như NVD.

### Mô hình Quét Web Nuclei + Nikto

Trong kiểm thử bảo mật, việc quét các dịch vụ web là bước quan trọng nhằm phát hiện các lỗ hổng ứng dụng, endpoint nguy hiểm, cấu hình sai và thông tin nhạy cảm bị lộ. Nhiều hệ thống web tồn tại trên các subdomain hoặc cổng không phổ biến, dễ bị bỏ qua trong quá trình kiểm thử.

Mô hình quét web nhằm phát hiện lỗ hổng, misconfiguration và endpoint nguy hiểm ở tầng HTTP/HTTPS. Nó kết hợp hai công cụ chính:

* Nuclei: template-based, tốc độ cao, phù hợp phát hiện CVE và các pattern theo template.
* Nikto: heuristic checks trên web server, phát hiện file nhạy cảm, header không an toàn, cấu hình sai phổ biến.

Kết quả trả về gồm cả bản sao raw (file) và kết quả đã chuẩn hóa (normalized) để lưu vào cơ sở dữ liệu, hiển thị trong giao diện hoặc đưa cho lớp AI enricher.

Luồng hoạt động tổng quát:

1. Nhận danh sách target (URL hoặc host:port) từ orchestrator.
2. Mô hình kiểm tra tồn tại binary (NUCLEI\_BIN / NIKTO\_BIN); nếu không, trả lỗi rõ ràng.
3. Xây lệnh chạy, nếu targets là danh sách, Mô hình tạo file tạm chứa targets và dùng option tương ứng.
4. Thực thi scanner:

* Nuclei: chạy bằng lệnh chặn (subprocess.run), tức là chương trình đợi lệnh Nuclei chạy xong mới tiếp tục và thu toàn bộ stdout và stderr trả về để dùng cho phân tích sau.
* Nikto: khởi chạy tiến trình không chặn (subprocess.Popen) và tạo một luồng đọc (reader thread) riêng để stream đầu ra stdout vừa ghi vào file lưu trữ, vừa giữ một bản sao tạm trong bộ nhớ để xử lý hoặc xuất kết quả tức thì.

1. Lưu bản thô stdout và nếu có stderr thì lưu vào file timestamped trong thư mục scanner tương ứng.
2. Phân tích và chuẩn hóa output:

* Nuclei: ưu tiên đọc ở định dạng JSON lines (mỗi dòng là một JSON) vì dễ chuyển thành bản ghi chuẩn; nếu không có JSON, fallback sang phân tích dòng text dạng người đọc được bằng cách dùng biểu thức chính quy (regex) để trích thông tin.
* Nikto: phân tích đầu ra dạng văn bản bằng phương pháp heuristic. Ví dụ coi những dòng bắt đầu bằng ký tự + là một phát hiện (issue) và cố gắng trích địa chỉ mục tiêu, cổng (IP/Port) nếu có trong dòng đó.

1. Trả về cấu trúc chuẩn gồm: raw\_path, err\_path/stderr, normalized (list record), success (bool), cmd (string).

Trích mã nguồn và phân tích chức năng:

**Nuclei:** Đoạn mã chính nằm trong file scanners/nuclei\_runner.py

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | def run\_nuclei(targets=None, targets\_file=None, extra\_args=None,  timeout=DEFAULT\_TIMEOUT):  bin\_path = \_find\_binary()  if not bin\_path:  return {....,"success": False, "stderr": f"nuclei-not-found:  {NUCLEI\_BIN}", ...}  cmd = [bin\_path]  args = []  if extra\_args:  args += list(extra\_args)  # Xử lý targets  if targets\_file:  args += ["-l", targets\_file]  elif targets:  if isinstance(targets, list):  ...  else:  args += ["-target", str(targets)]  cmd += args  proc = subprocess.run(cmd, stdout=..., stderr=..., timeout=timeout)  # Xử lý kết quả  stdout = proc.stdout or ""  stderr = proc.stderr or ""  normalized = []  for line in stdout.splitlines():  ...  if not normalized and (stderr or stdout):  placeholder = {  "scanner": "nuclei",  "target": ...,  ...........................  "raw": stdout + ("\nERR:\n" + stderr if stderr else "")  }  normalized.append({...})  return {"raw\_path": ..., "err\_path": ..., "normalized": normalized,  "success": ...} |

* Chức năng: Thực thi lệnh gọi công cụ Nuclei rồi thu thập kết quả quét từ stdout/stderr và chuẩn hóa kết quả thành định dạng JSON để phục vụ lưu trữ và phân tích.
* Giải thích:
* Gọi \_find\_binary() để xác định đường dẫn thực thi của Nuclei, nếu không tìm thấy thì trả về lỗi và kết thúc.
* Nếu có targets\_file thì dùng trực tiếp, nếu có targets dạng list thì ghi ra file tạm, nếu là chuỗi thì truyền trực tiếp.
* Ghép đường dẫn thực thi và danh sách các tham số (option) thành lệnh đầy đủ rồi dùng subprocess.run để thực thi rồi lấy stdout, stderr từ việc thực thi.
* Duyệt qua từng dòng stdout, nếu là JSON thì thực hiện việc chuẩn hóa.
* Nếu không chuẩn hóa được dòng nào, tạo bản ghi giả để lưu lại dữ liệu thô.
* Trả về kết quả gồm đường dẫn file raw, file lỗi, danh sách kết quả chuẩn hóa và trạng thái thành công.
* Ý nghĩa:
* Đây là dữ liệu đầu vào quan trọng cho bước phân tích bằng AI (hàm enrich\_nuclei\_findings()).
* Các phát hiện từ Nuclei giúp AI phân tích mức độ nghiêm trọng, bằng chứng, và đề xuất khắc phục.

**Nikto:** Đoạn mã chính nằm trong file scanners/nikto\_runner.py

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | def run\_nikto(targets=None, targets\_file=None, extra\_args=None,  timeout=DEFAULT\_TIMEOUT):  cmd = [NIKTO\_BIN, "-ask", "no"]  if extra\_args:  cmd += list(extra\_args)  elif targets:  if isinstance(targets, list):  cmd += ["-h", targets[0]]  else:  cmd += ["-h", str(targets)]  try:  proc = subprocess.Popen(cmd, stdout=subprocess.PIPE,  stderr=subprocess.STDOUT, text=True, bufsize=1) # gọi tiến trình Nikto  except FileNotFoundError as e:  ... # xử lý lỗi không tìm thấy binary  def reader():  for line in proc.stdout:  combined\_lines.append(line)  # đọc từng dòng từ stdout và ghi vào bộ nhớ  t = threading.Thread(target=reader, daemon=True)  t.start()  try:  proc.wait(timeout=timeout)  except subprocess.TimeoutExpired:  ... # xử lý timeout  t.join(timeout=2)  raw\_fh.close()  combined = ... # ghép các dòng lại  normalized = \_normalize\_from\_text(combined, target\_hint=...)  success = (proc.returncode == 0) or bool(normalized)  return {"raw\_path": raw\_path, normalized": normalized, success":  success, "stderr": "", "cmd": cmd\_display, streamed": False} |

* Chức năng: Thực thi lệnh gọi công cụ Nikto rồi thu thập kết quả quét từ stdout/stderr và chuẩn hóa kết quả thành định dạng JSON để phục vụ lưu trữ và phân tích.
* Giải thích:
* cmd = [NIKTO\_BIN, "-ask", "no"] tạo lệnh gọi Nikto với các tham số cơ bản và bổ sung.
* Nếu có targets\_file thì dùng trực tiếp, nếu có targets dạng list thì lấy target đầu tiên, nếu là chuỗi thì truyền trực tiếp.
* Sử dụng shlex.quote() để đưa các đoạn có kí tự đặc biệt vào trong ' ', giúp tạo bản hiển thị lệnh để ghi log hoặc debug.
* Gọi Nikto bằng subprocess.Popen, nếu không tìm thấy thì ghi lỗi và trả kết quả thất bại.
* Tạo hàm reader() đọc từng dòng từ stdout của Nikto, ghi vào file và lưu vào bộ nhớ.
* Ghép các dòng lại, chuẩn hóa kết quả từ text sang JSON và trả về kết quả cuối cùng của quá trình quét
* Ý nghĩa:
* Đây là dữ liệu đầu vào quan trọng cho bước enrich bằng AI (hàm enrich\_nikto\_findings()).
* Các phát hiện từ Nuclei giúp AI phân tích mức độ nghiêm trọng, bằng chứng, và đề xuất khắc phục.

### Mô hình AI Core

Mô hình ai\_common.py là nền tảng cho toàn bộ hệ thống enrich bằng AI. Nó cung cấp các hàm dùng chung để quản lý API key (đọc, ghi, xóa an toàn); gửi yêu cầu tới OpenAI (qua ask\_openai) với khả năng xoay vòng key, xử lý lỗi mạng, quota, xác thực; tạo session HTTP có retry/backoff để tăng độ ổn định. Các mô hình khác như ai\_enricher.py, ai\_recursive\_probe.py đều phụ thuộc vào ai\_common.py để thực hiện các tác vụ AI.

Luồng hoạt động tổng quát:

1. Đọc danh sách API key từ file api\_keys.txt.
2. Tạo session HTTP có retry để gọi OpenAI.
3. Gửi prompt tới OpenAI qua ask\_openai, xử lý phản hồi JSON.
4. Nếu key bị lỗi xác thực hoặc quota, tự động xóa khỏi file.
5. Nếu bị rate-limit hoặc lỗi mạng, tự động xoay vòng sang key khác.
6. Trả về nội dung phản hồi hoặc None nếu thất bại.

Trích mã nguồn và phân tích chức năng:

Đoạn mã chính nằm trong file analyzers/ai\_common.py:

**Chức năng tạo session HTTP có retry:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | def create\_retry\_session(retries: int = 3, backoff\_factor: float = 0.5,  status\_forcelist: Tuple[int, ...] = (500, 502, 504)) -> requests.Session:  session = requests.Session()  retry = Retry(  total=retries,  read=retries,  connect=retries,  backoff\_factor=backoff\_factor,  status\_forcelist=status\_forcelist,  allowed\_methods=frozenset(['GET', 'POST'])  )  adapter = HTTPAdapter(max\_retries=retry)  session.mount("http://", adapter)  session.mount("https://", adapter)  return session |

* Chức năng: Tạo một phiên làm việc (requests.Session) có khả năng tự động retry khi gặp lỗi mạng hoặc lỗi máy chủ (HTTP 500, 502, 504).
* Giải thích:
* Tạo một phiên HTTP mới để sử dụng cho các request.
* Cấu hình retry cho số lần thử lại tối đa cho mỗi loại lỗi, thời gian chờ giữa các lần retry, danh sách mã lỗi HTTP sẽ kích hoạt retry và chỉ retry với POST, GET.
* Gắn cầu hình retry vào cả giao thức HTTP và HTTPS.
* Ý nghĩa: Đảm bảo tính ổn định khi gọi API bên ngoài (ví dụ: OpenAI), tránh bị gián đoạn do lỗi tạm thời hoặc timeout. Đây là một bước quan trọng trong việc tăng độ tin cậy cho các mô hình AI như ask\_openai().

**Chức năng quản lý API key từ file:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | def load\_api\_keys() -> List[str]:  if not os.path.exists(API\_KEYS\_FILE):  return []  try:  with open(API\_KEYS\_FILE, "r", encoding="utf-8") as f:  return [line.strip() for line in f if line.strip()]  except Exception as e:  LOG.error("Không đọc được %s: %s", API\_KEYS\_FILE, e)  return []  def save\_api\_keys\_atomic(keys: List[str]) -> None:  d = os.path.dirname(API\_KEYS\_FILE) or "."  fd, tmp = tempfile.mkstemp(dir=d, prefix=".\_tmp\_keys\_")  try:  with os.fdopen(fd, "w", encoding="utf-8") as tf:  for k in keys:  tf.write(k.strip() + "\n")  os.replace(tmp, API\_KEYS\_FILE)  except Exception as e:  LOG.error("Không lưu được API keys atomically: %s", e)  ...  def remove\_key\_at\_index\_safe(idx: int) -> bool:  with API\_KEYS\_LOCK:  keys = load\_api\_keys()  if 0 <= idx < len(keys):  removed = keys.pop(idx)  save\_api\_keys\_atomic(keys)  LOG.info("Removed API key at index %d (redacted)", idx)  return True  return False |

* Chức năng: Quản lý danh sách API key dùng để gọi OpenAI (hoặc các dịch vụ AI khác).
* Giải thích:
* Đọc file api\_keys.txt và trả về danh sách các dòng không rỗng.
* Ghi danh sách key vào file tạm, sau đó thay thế file gốc bằng file tạm.
* Gắn Xóa một key tại vị trí idx trong danh sách, ghi lại danh sách mới vào file.
* Ý nghĩa: Đảm bảo hệ thống có thể đọc, ghi, và cập nhật danh sách key một cách an toàn, tránh lỗi ghi đè hoặc mất dữ liệu khi có nhiều tiến trình truy cập đồng thời.

**Chức năng gửi yêu cầu tới OpenAI với xử lý xoay vòng key:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | def ask\_openai(prompt: str, model: str = "gpt-4o-mini", temperature: float = 0,  max\_tokens: int = 2000, per\_request\_timeout: int = 5000) -> Optional[str]:  keys = load\_api\_keys()  if not keys:  return None  session = requests.Session()  key\_index = 0  while keys:  api\_key = keys[key\_index]  headers = {"Authorization": f"Bearer {api\_key}", "Content-Type":  "application/json"}  payload = {"model": model, "messages": [{"role": "user", "content": prompt}],  “temperature": temperature, "max\_tokens": max\_tokens}  resp = session.post("https://api.openai.com/v1/chat/completions", headers=  headers, json=payload, timeout=per\_request\_timeout)  status = resp.status\_code  try:  body = resp.json()  except Exception:  body = resp.text  if status == 200:  return body["choices"][0]["message"]["content"].strip()  # xử lý lỗi xác thực  if status in (401, 403):  ...  remove\_key\_at\_index\_safe(pos)  keys = load\_api\_keys()  key\_index = 0  continue  # xử lý rate-limit  if status == 429 or is\_rate\_limit:  ...  key\_index = (key\_index + 1) % len(keys)  continue  # lỗi server  if 500 <= status < 600:  key\_index = (key\_index + 1) % len(keys)  continue  return None  LOG.error("Hết API key, không thành công.")  return None |

* Chức năng: Gửi yêu cầu tới API của OpenAI để nhận phản hồi từ mô hình ngôn ngữ, đồng thời xử lý các lỗi mạng, xác thực, giới hạn tốc độ và xoay vòng API key khi cần.
* Giải thích:
* Kiểm tra xem có API key nào khả dụng không. Nếu không có, ghi log và dừng hàm.
* Tạo phiên làm việc để gửi request, bắt đầu từ key đầu tiên bằng session = requests.Session().
* Tạo vòng lặp gửi yêu cầu và xử lý lỗi. Trong vòng lặp sẽ gửi các request với API và câu hỏi được gửi đến tới OpenAI.
* Trong mỗi lần gửi yêu cầu đến OpenAI, nếu phản hồi trả về bị lỗi hoặc key bị giới hạn thời gian thì sẽ tự động quay vòng sang key khác. Còn trường hợp giới hạn thời gian quá lâu (trên 1 ngày) hoặc key hết hạn thì sẽ thực hiện xóa key và dừng nếu không còn key.
* Ý nghĩa: Đây là hàm trung tâm trong hệ thống AI của dự án, đảm bảo việc gọi OpenAI diễn ra ổn định, liên tục và không bị gián đoạn khi một key bị lỗi hoặc hết hạn.

### Mô hình Enrich Bằng AI

Mô hình enrich AI có nhiệm vụ phân tích sâu các kết quả quét bảo mật (Nmap, Nikto, Nuclei) bằng cách sử dụng mô hình ngôn ngữ lớn (LLM) như GPT để chuẩn hóa thông tin dịch vụ (product, version, cpe), truy xuất CVE liên quan đến dịch vụ đang chạy, diễn giải các phát hiện từ công cụ quét web dưới dạng JSON có cấu trúc, phân tích phản hồi web để phát hiện endpoint, tham số và gợi ý lỗ hổng tiềm năng.

Việc enrich giúp tăng độ chính xác, độ sâu và khả năng tự động hóa trong phân tích bảo mật, đồng thời tạo đầu vào cho các bước đánh giá rủi ro, đề xuất khắc phục và báo cáo.

#### Enrich từ Nmap

Hàm enrich\_from\_parsed\_nmap có nhiệm vụ phân tích sâu kết quả quét mạng từ Nmap bằng cách chuẩn hóa thông tin dịch vụ (service, product, version, cpe) truy xuất các lỗ hổng bảo mật (CVE) liên quan đến từng dịch vụ trả về danh sách các bản ghi enrich có cấu trúc rõ ràng để phục vụ hiển thị, đánh giá rủi ro và báo cáo. Hàm này được gọi trực tiếp từ orchestrator sau khi Nmap quét xong từng IP, giúp enrich theo từng host một cách tuần tự và dễ kiểm soát.

Luồng hoạt động tổng quát:

1. Nhận đầu vào là dict parsed từ hàm run\_nmap\_and\_store, chứa danh sách dịch vụ.
2. Duyệt từng dịch vụ, chuẩn hóa thông tin bằng cách gọi AI (qua ai\_find\_cves\_for\_service).
3. Với mỗi CVE được AI trả về:

* Trích xuất các trường: id, cve\_url, exploit\_url, msf\_module, complexity, short\_description.
* Gắn thêm thông tin IP và sản phẩm.

1. Gom toàn bộ bản ghi enrich vào danh sách và trả về.

Trích mã nguồn và phân tích chức năng

Hàm chính: enrich\_from\_parsed\_nmap:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | def enrich\_from\_parsed\_nmap(parsed: Dict[str, Any], ip: str, model: str = "gpt-4o-mini",  max\_cves\_per\_service: int = 10) -> List[Dict[str, Any]]:  if not parsed:  return []  services\_raw = parsed.get("services", [])  if not services\_raw:  return []  svc\_list: List[Dict[str, Any]] = []  for svc in services\_raw:  s = {"ip": ip,  "port": svc.get("port"),  "service": svc.get("service") or "",  "product": svc.get("product") or "",  "version": svc.get("version") or "",  "cpe": svc.get("cpe") or [] }  svc\_list.append(s)  services = merge\_services(svc\_list)  enriched: List[Dict[str, Any]] = []  total = len(services)  for i, svc in enumerate(services, start=1):  print(f"[NMAP] Enriching {svc.get('service')}:{svc.get('port')}")  cves = ai\_find\_cves\_for\_service(svc, model=model, max\_cves=  max\_cves\_per\_service)  for c in cves:  enriched.append(c)  return enriched |

* Chức năng: Hàm này thực hiện phân tích sâu các dịch vụ đi cùng phiên bản của chúng đã được quét bởi Nmap, từ các kết quả quét của nmap công cụ sẽ sử dụng AI để tìm kiếm các lỗ hổng bảo mật (CVE), mã khai thác và msf module (nếu có) liên quan đến từng dịch vụ và phiên bản đi kèm.
* Giải thích:
* if not parsed..., services\_raw = parsed.get("services", []) kiểm tra dữ liệu đầu vào có hợp lệ không; nếu không có dịch vụ thì trả về rỗng.
* Vòng for svc in services\_raw, tạo svc\_list Tạo vòng lặp gửi yêu cầu và xử lý lỗi. Gom thông tin từng dịch vụ thành dict chuẩn gồm ip, port, service, product, version, cpe.
* Dùng merge\_services() để gộp các dịch vụ trùng IP + port + product để tránh enrich trùng lặp.
* Chạy vòng for với các services lấy được và sử dụng hàm ai\_find\_cves\_for\_service() để gọi AI để tìm các CVE liên quan đến từng dịch vụ. Nếu có kết quả, in ra để kiểm tra.
* Chuẩn hóa từng CVE thành một bản ghi có cấu trúc rõ ràng để lưu trữ hoặc hiển thị.
* Ý nghĩa: Đây là bước chuyển đổi từ dữ liệu kỹ thuật (port, service, version, CPE) sang thông tin bảo mật có giá trị, giúp xác định mức độ nguy hiểm, khả năng khai thác, và liên kết với các công cụ như Metasploit.

Hàm phụ: ai\_find\_cves\_for\_service:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | def ai\_find\_cves\_for\_service(service: Dict[str, Any], model: str = "gpt-4o-mini",  max\_cves: int = 100) -> List[Dict[str, Any]]:  ctx = {"ip": service.get("ip"),  "port": service.get("port"),  "product": service.get("product") or service.get("service"),  "version": service.get("version"),  "cpe": service.get("cpe", []) }  prompt = SERVICE\_CVE\_PROMPT\_VN.format(input=json.dumps(ctx,  ensure\_ascii=False))  ai\_res = \_call\_ai\_json(prompt, model=model, max\_tokens=1600)  out: List[Dict[str, Any]] = []  if isinstance(ai\_res, list):  for item in ai\_res[:max\_cves]:  if not isinstance(item, dict): continue  cid = item.get("id")  ip = item.get("ip")  product= item.get("product")  cve\_url = item.get("cve\_url")  exploit\_url = item.get("exploit\_url") or "Không có"  msf\_module= item.get("msf\_module") or "Không có"  complexity = item.get("complexity") or "Không rõ"  short\_desc = item.get("short\_description")  out.append({...  return out  return [] |

* Chức năng: Hàm này sử dụng AI để phân tích thông tin về một dịch vụ (service, product, version, CPE) và trả về danh sách các lỗ hổng bảo mật (CVE) liên quan.
* Giải thích:
* Tạo ngữ cảnh đầu vào cho AI dưới dạng JSON, bao gồm thông tin kỹ thuật của dịch vụ vào biến ctx.
* Dùng hàm call\_ai\_json() để gọi AI phân tích và trả về danh sách CVE dưới dạng JSON.
* Sau khi có kết quả là danh sách các CVE, chuẩn hóa từng bản ghi và trả về. Nếu không, trả về danh sách rỗng.
* Ý nghĩa: Đây là bước quan trọng trong việc enrich dữ liệu kỹ thuật thành thông tin bảo mật có giá trị, giúp xác định mức độ nguy hiểm, khả năng khai thác, và liên kết với các công cụ như Metasploit.

**PROMPT: SERVICE\_CVE\_PROMPT\_VN:**

|  |
| --- |
| Bạn là trợ lý phân tích bảo mật. Tôi cung cấp thông tin sản phẩm và phiên bản dưới dạng JSON: product, service, version, cpe. HÃY TRẢ VỀ MỘT đối tượng JSON (chỉ JSON) gồm các đối tượng CVE. Mỗi đối tượng CVE phải có các trường sau:  - ip:port: chua ip va port duoc dua vao(ví dụ:10.10.10.10:22), tất cả giá trị ở trường này đều như nhau trong mỗi đối tượng CVE\n"  - pro+ver: chứa cả product + version được đưa vào(ví dụ OpenSSH  8.2p1 Ubuntu 4ubuntu0.11), tất cả giá trị ở trường này đều như nhau trong  mỗi đối tượng CVE  - id: chuỗi CVE (ví dụ CVE-2021-12345)  - cve\_url: đường dẫn chi tiết CVE  - exploit\_url: link tới mã khai thác nếu có, nếu không có thì điền Không có  - msf\_module: link hoặc tên mô hình Metasploit nếu có, nếu không có thì điền Không có  - complexity: mức độ khó/độ phức tạp khai thác (thấp/trung bình/cao)  - short\_description: mô tả ngắn 1-2 câu  Nếu không tìm thấy CVE phù hợp thì trả về mảng rỗng [].  Dữ liệu đầu vào:\n{input}\n\nChỉ trả về JSON hợp lệ |

#### Enrich từ Nuclei

Hàm có nhiệm vụ phân tích nội dung raw từ công cụ quét web Nuclei bằng cách sử dụng AI để trích xuất các phát hiện bảo mật dưới dạng JSON có cấu trúc; chuẩn hóa thông tin như template, severity, confidence, evidence, description, remediation; tạo đầu vào rõ ràng cho các bước đánh giá rủi ro, hiển thị UI và báo cáo. Hàm này giúp chuyển đổi output thô từ Nuclei thành dữ liệu có thể xử lý tự động, dễ đọc và dễ phân tích.

Luồng hoạt động tổng quát:

* 1. Đọc file JSONL chứa các bản ghi raw từ việc quét trên công cụ Nuclei (nuclei\_results.jsonl).
  2. Với mỗi bản ghi:
* Trích targer và raw .
* Tạo prompt tiếng Việt yêu cầu AI phân tích và trích xuất findings.
* Gửi prompt tới OpenAI qua ask\_openai.
* Parse kết quả JSON trả về.
  1. Nếu kết quả hợp lệ và có kết quả enrich, thêm vào danh sách enrich.
  2. về danh sách các bản ghi enrich đã chuẩn hóa.

Trích mã nguồn và phân tích chức năng

Hàm chính: enrich\_nuclei\_findings

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | def enrich\_nuclei\_findings(nuclei\_jsonl: str, model: str = "gpt-4o-mini") ->  List[Dict[str, Any]]:  findings = load\_nuclei\_findings(nuclei\_jsonl)  summaries = []  for i, rec in enumerate(findings, start=1):  target = rec.get("target") or rec.get("primary\_target") or ""  raw = rec.get("raw") or ""  if not raw.strip():  continue  print(f"[NUCLEI] Phân tích raw {i}/{len(findings)} cho {target}")  prompt = (f"Bạn là trợ lý phân tích bảo mật. Tôi cung cấp nội dung raw từ công  cụ Nuclei quét web:\n\n"  f"Target: {target}\n\n"  f"Raw:\n{raw}\n\n"  f"Hãy trích xuất các phát hiện bảo mật dưới dạng JSON như sau:\n"  f"{{\n \"primary\_target\": \"{target}\",\n \"findings\": [\n{{\n \"template\":  \"...\",\n\"severity\": \"...\",\n\"confidence\": \"...\" ,\n \"evidence\": \"đưa bằng chứng là 1  lời giai thich \",\n \"description\": \"...\",\n \"remediation\": \"...\"\n}}\n ]\n}}\n"  f"Trả về JSON hợp lệ dung y nhu tren bao gom primary\_target va findings. Nếu  không có phát hiện thì trả về mảng findings rỗng.")  parsed = \_call\_ai\_json(prompt, model=model, max\_tokens=1500)  if isinstance(parsed, dict) and parsed.get("findings"):  summaries.append(parsed)  print(json.dumps(parsed, ensure\_ascii=False, indent=2))  return summaries |

* Chức năng: Hàm này sử dụng AI để phân tích kết quả quét thô từ công cụ Nuclei, trích xuất các phát hiện bảo mật và chuẩn hóa chúng thành định dạng JSON có cấu trúc.
* Giải thích:
* Đọc toàn bộ kết quả quét từ file nuclei\_results.jsonl và khởi tạo danh sách lưu kết quả enrich.
* Trích target và raw, tạo prompt tiếng Việt yêu cầu AI phân tích với mỗi bản ghi.
* Dùng hàm call\_ai\_json() để gọi AI phân tích và trả về danh sách dưới dạng JSON, sau đó thêm vào danh sách enrich.
* Ý nghĩa: Đây là bước enrich quan trọng giúp chuyển đổi dữ liệu quét web thô thành thông tin bảo mật có giá trị, dễ hiểu và có thể sử dụng trong báo cáo, cảnh báo hoặc phân tích sâu.

#### Enrich từ Nikto

Hàm enrich\_nikto\_findings có nhiệm vụ phân tích nội dung raw từ công cụ quét web Nikto bằng cách sử dụng AI để trich xuất các phát hiện bảo mật dưới dạng JSON có cấu trúc; chuẩn hóa thông tin như path, title, severity, confidence, evidence, impact, exploitability, remediation, references; tạo đầu vào rõ ràng cho các bước đánh giá rủi ro, hiển thị giao diện và báo cáo. Hàm này giúp chuyển đổi output thô từ Nikto thành dữ liệu có thể xử lý tự động, dễ đọc và dễ phân tích.

Luồng hoạt động tổng quát:

1. Đọc file JSONL chứa các bản ghi raw từ Nikto (nikto\_results.jsonl).
2. Với mỗi bản ghi:

* Trích target và raw.
* Tạo prompt tiếng Việt yêu cầu AI phân tích và trích xuất findings.
* Gửi prompt tới OpenAI qua ask\_openai.
* Parse kết quả JSON trả về.

1. Nếu kết quả hợp lệ và có findings, thêm vào danh sách enrich.
2. Trả về danh sách các bản ghi enrich đã chuẩn hóa.

Trích mã nguồn và phân tích chức năng

Hàm chính: enrich\_nikto\_findings

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | def enrich\_nikto\_findings(nikto\_jsonl: str, model: str = "gpt-4o-mini") ->  List[Dict[str, Any]]:  findings = load\_nikto\_findings(nikto\_jsonl)  summaries = []  for i, rec in enumerate(findings, start=1):  target = rec.get("target") or rec.get("primary\_target") or ""  raw = rec.get("raw") or ""  if not raw.strip():  continue  print(f"[NIKTO] Phân tích raw {i}/{len(findings)} cho {target}")  prompt = (f"Bạn là trợ lý phân tích bảo mật. Tôi cung cấp nội dung raw từ công cụ  Nikto quét web:\n\n"  f"Target: {target}\n\n Raw:\n{raw}\n\n "  f"Hãy trích xuất các phát hiện bảo mật dưới dạng JSON như sau:\n"  f"{{\n \"primary\_target\": \"{target}\",\n \"findings\": [\n {{\n \"path\": \"...\",\n  \"title\": \"...\",\n \"severity\": \"...\",\n \"confidence\": \"...\",\n \"description\": \"...\",  \n\"evidence\": \"chua bang chung\",\n \"impact\": \"...\",\n \"exploitability\": \"...\",\n  \"remediation\": \"...\",\n \"references\": \"...\"\n }}\n ]\n}}\n"  f"Trả về JSON hợp lệ. Nếu không có phát hiện thì trả về mảng findings rỗng” )  parsed = \_call\_ai\_json(prompt, model=model, max\_tokens=1500)  if isinstance(parsed, dict) and parsed.get("findings"):  summaries.append(parsed)  print(json.dumps(parsed, ensure\_ascii=False, indent=2))  return summaries |

* Chức năng: Hàm này sử dụng AI để phân tích kết quả quét thô từ công cụ Nikto, trích xuất các phát hiện bảo mật và chuẩn hóa chúng thành định dạng JSON có cấu trúc.
* Giải thích:
* Đọc toàn bộ kết quả quét từ file nuclei\_results.jsonl và khởi tạo danh sách lưu kết quả enrich.
* Trích target và raw, tạo prompt tiếng Việt yêu cầu AI phân tích với mỗi bản ghi.
* Dùng hàm call\_ai\_json() để gọi AI phân tích và trả về danh sách dưới dạng JSON, sau đó thêm vào danh sách enrich.
* Ý nghĩa: Đây là bước enrich quan trọng giúp chuyển đổi dữ liệu quét web thô thành thông tin bảo mật có giá trị, dễ hiểu và có thể sử dụng trong báo cáo, cảnh báo hoặc phân tích sâu.

### Mô hình Phân Tích Endpoint Và Lỗ Hổng Tiềm Năng

Mô hình ai\_recursive\_probe.py có nhiệm vụ phân tích động các ứng dụng web bằng cách:

* Phát hiện endpoint và tham số từ HTML hoặc URL đầu vào.
* Gợi ý các lỗ hổng tiềm năng theo từng tham số (XSS, SQLi, Path traversal, v.v.).
* Gửi payload thử nghiệm và đánh giá khả năng khai thác bằng AI.
* Trả về danh sách endpoint, tham số, payload và mức độ rủi ro.

Đây là bước kiểm thử nâng cao, giúp phát hiện các điểm yếu logic hoặc lỗ hổng chưa được công cụ quét truyền thống nhận diện.

Luồng hoạt động tổng quát:

1. Nhận URL đầu vào từ orchestrator hoặc CLI.
2. Tách endpoint và tham số từ URL nếu có.
3. Gửi request tới trang chủ để lấy HTML.
4. Gọi AI để phân tích HTML và tìm endpoint + tham số.
5. Gộp endpoint từ URL và từ AI.
6. Với mỗi endpoint:

* Gửi request thử nghiệm.
* Gọi AI để gợi ý lỗ hổng tiềm năng theo từng tham số.
* Với mỗi lỗ hổng tiềm năng sẽ gửi payload thử nghiệm và gọi AI để đánh giá khả năng khai thác.

1. Trả về danh sách endpoint + lỗ hổng + payload + mức độ rủi ro.

Trích mã nguồn và phân tích chức năng

Hàm chính: run\_security\_scan:

**Chức năng tiền xử lý URL và tách endpoint:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | parsed = urlparse(input\_url)  if not parsed.scheme:  logger.error("URL phải có scheme (http:// hoặc https://).")  return  base\_url = f"{parsed.scheme}://{parsed.netloc}"  initial\_endpoints = []  if parsed.path and parsed.path != "/":  ep\_path = parsed.path  params = {}  if parsed.query:  q = parse\_qs(parsed.query, keep\_blank\_values=True)  for k, vals in q.items():  params[k] = vals[0] if vals else ""  logger.info(f"Tách endpoint từ input: base\_url={base\_url},  endpoint={ep\_path}, params={params}")  initial\_endpoints.append({"endpoint": ep\_path,  "method": "GET",  "params": params }) |

* Chức năng: Trích và chuẩn hóa endpoint khởi đầu (path + query) từ URL đầu vào để dùng làm seed cho quy trình quét/thu thập endpoint tiếp theo.
* Giải thích:
* Gọi urlparse(input\_url), nếu thiếu scheme thì ghi log lỗi và trả None.
* Tạo base\_url dùng để gọi trang chủ hoặc làm gốc khi ghép các endpoint tương đối.
* Tách endpoint từ path và query nếu parsed.path tồn tại và khác "/", thì lấy ep\_path = parsed.path; phân tích parsed.query bằng parse\_qs(..., keep\_blank\_values=True) và chọn giá trị đầu tiên của mỗi key làm default
* Lưu thông tin của endpoint tách được.
* Ý nghĩa: Cung cấp một endpoint ban đầu rõ ràng (endpoint, method, params) và base\_url để tiếp tục các bước kiểm thử tự động.

**Chức năng tìm kiếm các endpoint từ URL ban đầu:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | headers = {}  if cookie\_header:  headers["Cookie"] = cookie\_header  r = session.get(base\_url, timeout=10, headers=headers)  if r.status\_code >= 400:  logger.warning(f"....")  html = r.text  detected\_from\_home = ai\_detect\_endpoints\_only(html, base\_url)  endpoints = []  seen\_signatures = set()  def add\_ep(ep):  sig = (ep.get("endpoint"), ep.get("method", "GET").upper(), tuple(sorted(ep.get  ("params", {}).keys())))  if sig not in seen\_signatures:  seen\_signatures.add(sig)  endpoints.append(ep)  for ep in initial\_endpoints:  add\_ep(ep)  for ep in detected\_from\_home:  add\_ep(ep) |

* Chức năng: Lấy HTML trang chủ (URL ban đầu), gọi AI để phát hiện các endpoint từ HTML đó, rồi gộp các endpoint khởi tạo (từ URL đầu vào) và endpoint do AI phát hiện thành danh sách endpoint duy nhất để tiếp tục bước quét mở rộng.
* Giải thích:
* Gửi GET tới base\_url; hỗ trợ gửi Cookie nếu người dùng cung cấp. Sau đó thu thập phản hồi từ yêu cầu gửi tới.
* Gọi hàm ai\_detect\_endpoints\_only để trích xuất endpoint.
* Với endpoint trích xuất được, thực hiện tạo chữ ký và kiểm tra trong bộ chữ ký. Nếu chưa tồn tại thì thêm endpoint và tập endpoints và chữ ký vào bộ chữ ký.
* Ý nghĩa: Cung cấp các endpoint để tiếp tục các bước kiểm thử tự động.

**Chức năng gửi request thử nghiệm tới từng endpoint và phân tích lỗ hổng web thông qua OpenAI:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | while i < len(endpoints):  if total\_requests >= max\_total\_requests: break  ep = endpoints[i]; i += 1  ensure\_default\_params(ep) # set missing params -> "test"  for k in list(ep.get("params", {}).keys()):  if ep["params"].get(k) in (None, ""):  ep["params"][k] = "test"  resp = send\_test\_request(session, base\_url, ep["method"], ep["endpoint"],  ep["params"], cookie\_header)  total\_requests += 1  # detect thêm endpoint từ response bằng AI  new\_eps = ai\_detect\_endpoints\_only(resp.text, base\_url)  for new\_ep in new\_eps:  add\_ep(ep)  ai\_vuln\_prompt= " (hỏi AI: với response này, param nào có khả năng dính lỗ hổng) "  ai\_vuln\_response = ask\_openai(ai\_vuln\_prompt, model=ai\_model)  parsed\_vuln = safe\_json\_load(ai\_vuln\_response)  per\_endpoint\_vuln\_suggestions[(ep["endpoint"], ep["method"])] =  parsed\_vuln |

* Chức năng: Tạo vòng lặp mở rộng endpoints, gửi request thử tới từng endpoint, thu nhận response, dùng AI để phát hiện endpoint mới từ response, và hỏi AI gợi ý tham số có khả năng dính lỗ hổng; lưu tóm tắt kết quả (scanned) và gợi ý lỗ hổng per\_endpoint\_vuln\_suggestions..
* Giải thích:
* Lấy endpoint hiện tại và chuẩn bị, gán param giá trị mặc định (dùng "test") cho các param chưa có giá trị trước khi gửi request.
* Tiếp tục gửi resquest đến endpoint hiện tại với các giá trị tương ứng để lấy phản hồi.
* Từ phản hồi được trả về, lưu nội dung và tiếp tục gọi sang AI để tìm kiếm thêm endpoint. Ngoài ra, gọi sang AI để thực hiện dò tìm các lỗ hổng có thể xảy ra.
* Ý nghĩa: Dò quét hết endpoint có thể cùng với các lỗ hổng có khả năng xảy ra trên endpoint đó để tạo input cho bước test payload thực tế.

**Chức năng gửi request thử nghiệm tới từng endpoint với thông tin lỗ hổng trước đó cùng payload tương ứng và phân tích kết quả thông qua OpenAI:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | vuln\_findings\_all = []  for (endpoint\_path, method), param\_map in per\_endpoint\_vuln\_suggestions  .items():  matching = None  for e in endpoints:  if e["endpoint"] == endpoint\_path and e["method"] == method:  matching = e  break  if not matching: continue  base\_params = matching.get("params", {}).copy()  for param\_name, vuln\_list in param\_map.items():  if not vuln\_list: continue  for vuln\_type in vuln\_list:  payloads = COMMON\_VULNS\_PAYLOADS.get(vuln\_type, [])  vuln\_note = COMMON\_VULNS\_NOTES.get(vuln\_type, "") or ""  for payload in payloads:  test\_params = base\_params.copy()  test\_params[param\_name] = payload  resp = send\_test\_request(session, base\_url, method, endpoint\_path, t  est\_params, cookie\_header=cookie\_header)  ai\_exploit\_prompt "... (yêu cầu AI trả 1 JSON of {result,reason} cho kết quả  phân tích lỗ hổng từ phản hồi được gửi cùng payload)"  ai\_exploit\_response = ask\_openai(ai\_exploit\_prompt, model=ai\_model)  exploit\_parsed = {}  if ai\_exploit\_response:  exploit\_parsed = json.loads(ai\_exploit\_response.strip())  result\_label = exploit\_parsed.get("result", "Không rõ") if i  sinstance(exploit\_parsed, dict) else "Không rõ"  reason = exploit\_parsed.get("reason", "") if  isinstance(exploit\_parsed, dict) else (ai\_exploit\_response or "")  vuln\_findings\_all.append({ "endpoint": endpoint\_path,  "method": method,  "param": param\_name,  "payload": payload.replace("\n", " "),  "likelihood": result\_label,  "reason": reason.replace("\n", " ")  }) |

* Chức năng: Từ danh sách endpoints và gợi ý lỗ hổng của AI, hàm này tạo bảng tóm tắt endpoints (rows), rồi thực thi các payload thử nghiệm cho các param có khả năng dính lỗ hổng, gọi AI để đánh giá mức khả năng khai thác và thu về danh sách kết quả khai thác (vuln\_findings\_all).
* Giải thích :
* Duyệt qua từng (endpoint\_path, method), param\_map trong per\_endpoint\_vuln\_suggestions.items().Với mỗi cặp, tìm endpoint tương ứng trong endpoints (matching), nếu không tìm thấy thì bỏ qua.
* Duyệt qua từng param\_name và vuln\_list trong param\_map. Sau đó tìm kiếm payload tương ứng với vulnlist trong tập payload có sẵn. Nếu có thì thay giá trị của param trong endpoint tương ứng với param\_name và gửi request đến endpoint đó với param chứa payload
* Từ phản hồi được trả về, lưu nội dung và tiếp tục gọi sang AI để phân tích khả năng khai thác của payload.
* Lưu kết quả phân tích
* Ý nghĩa: Chuyển gợi ý lý thuyết (param → loại lỗ hổng) thành bằng chứng tạm thời (response + đánh giá của AI) để ưu tiên xử lý.

Hàm phụ: ai\_detect\_endpoints\_only:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | def ai\_detect\_endpoints\_only(html, base\_url, max\_items=50):  prompt = "... (yêu cầu AI trả 1 JSON array of {endpoint, method, params})"  ai\_resp = ask\_openai(prompt)  if not ai\_resp: return []  # extract JSON substring and parse with fallback  parsed = try\_parse\_json\_from\_text(ai\_resp)  results = []  for item in parsed:  method = (item.get("method") or "GET").upper()  params = item.get("params", {}) if isinstance(item.get("params", {}), dict) else {}  params\_norm = {}  results.append({  "endpoint": ep\_path,  "method": method,  "params": params\_norm})  return results |

* Chức năng: Từ HTML trả về, sử dụng AI để chỉ liệt kê các endpoint tìm được (endpoint path hoặc URL), phương thức HTTP và tham số kèm giá trị mặc định; chuẩn hóa kết quả thành danh sách object JSON {"endpoint", "method", "params"}.
* Giải thích:
* Dựng prompt tiếng Việt yêu cầu "chỉ LIỆT KÊ" endpoint, method, params; kèm Base URL và phần HTML.
* Gọi AI và kiểm tra phản hồi thông qua ai\_resp = ask\_openai(prompt).
* Dùng try\_parse\_json\_from\_text trích JSON thô từ text trả về.
* Chuẩn hóa từng mục từ JSON thô tách được và lưu thông tin dưới dạng List[{"endpoint","method","params"}].
* Ý nghĩa: Tạo endpoint có chất lượng cao cho bước mở rộng và thử payload; giảm sai lệch bằng cách tận dụng khả năng hiểu ngữ cảnh của AI.

### Mô hình điều phối

Mô hình orchestrator\_interactive.py là trung tâm điều phối toàn bộ quy trình kiểm thử bảo mật. Nó thực hiện các nhiệm vụ nhận URL đầu vào và khởi tạo quá trình quétl; gọi các mô hình thu thập miền phụ, quét mạng, quét web, enrich AI và phân tích recursive; ghi kết quả ra file JSONL và cơ sở dữ liệu SQLite để phục vụ hiển thị, phân tích và điều tra.

Luồng hoạt động tổng quát:

1. Nhận URL đầu vào từ orchestrator hoặc CLI.
2. Gọi fetch\_crtsh\_subdomains để thu thập subdomain.
3. Gọi resolve\_host để phân giải IP từ subdomain.
4. Với mỗi IP:

* Gọi run\_nmap\_and\_store để quét mạng.
* Gọi enrich\_from\_parsed\_nmap để enrich kết quả.

1. Gọi run\_web\_scanners để quét web bằng Nuclei và Nikto.
2. Gọi run\_ai\_enrichment\_and\_recursive để enrich web findings và phân tích endpoint.
3. Ghi kết quả vào file JSONL và SQLite.

Trích mã nguồn và phân tích chức năng

Hàm chính: run\_scan:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | def run\_scan(target: str, model: str = "gpt-4o-mini", cookie: str | None = None,  clean: bool = True, input\_type: str = "url"):  clean\_outputs()  if input\_type == "ip":  parsed\_nmap = run\_nmap\_and\_store(target, "", "") #ip với url  enriched = []  if enrich\_from\_parsed\_nmap:  enriched = enrich\_from\_parsed\_nmap(parsed\_nmap, target, model=model)  dump\_enriched(enriched, prefix="enriched\_nmap")  return {"status": "ok", "summary": .....}  original\_url = target #Bắt đầu test với trường hợp đầu vào url  parsed = urlparse(original\_url)  domain = parsed.hostname  if not domain:  return {"status": "error", "message": "..."}  # 1) subdomain discovery  subs = fetch\_crtsh\_subdomains(domain) or []  if domain not in subs:  subs.insert(0, domain)  rec = {"domain": domain, "subdomains": subs}  dump\_enriched(rec, prefix="subs")  # 2) resolve -> collect IPs  all\_ips = set()  for sub in subs:  for ip in resolve\_host(sub):  all\_ips.add((ip, sub))  scanned\_ips = []  for ip, hostname in sorted(all\_ips):  parsed\_nmap = run\_nmap\_and\_store(ip, hostname, domain)  scanned\_ips.append(ip)  if enrich\_from\_parsed\_nmap:  enriched = enrich\_from\_parsed\_nmap(parsed\_nmap, ip, model=model)  if dump\_enriched:  dump\_enriched(enriched, prefix="enriched\_nmap")  web\_targets = run\_web\_scanners([original\_url])  run\_ai\_enrichment\_and\_recursive(web\_targets, original\_url, model=model,  cookie=cookie)  return {"status": "ok", "summary": ....} |

* Chức năng: Điểm vào chương trình để chạy toàn bộ quy trình quét an ninh cho một mục tiêu (URL hoặc IP) theo luồng đã thiết kế: dọn output, khởi tạo DB, khám phá miền phụ, resolve IP, quét Nmap, quét web, enrich bằng AI và probe đệ quy.
* Giải thích:
* gọi clean\_outputs() để xóa các file kết quả từ lần quét trước.
* Kiểm tra trước giá trị input\_type, nếu là IP thì chỉ thực hiện việc gọi Nmap và đưa kết quả sang AI để phân tích. Còn lại đều coi như là url và thực hiện toàn bộ chức năng.
* Với URL thực hiện hành động trích hostname làm domain, tìm kiếm các subdomain rồi chuyển chúng thành IP. Với mỗi IP lại thực hiện việc quét bằng Nmap và phân tích bằng AI. Sau đó thực hiện quét bằng Nikto và Nuclei. Cuối cùng là phân tích kết quả từ Nikto và Nuclei và quét tìm các endpoint, lỗ hổng trên trang web.
* Ý nghĩa: orchestrator cao cấp trong quy trình; chịu trách nhiệm tuần tự hoá các bước quét, lưu giá trị quét, phân tích kết quả bằng AI và ghi kết quả trung gian (dump\_enriched).

### Mô hình API Backend

Các Mô hình backend API này cung cấp giao diện RESTful để frontend và các hệ thống khác tương tác với hệ thống kiểm thử bảo mật. Chúng thực hiện các nhiệm vụ khởi tạo và theo dõi tiến trình quét (orchestrator\_runner.py); truy xuất kết quả quét từ file JSONL (results.py); quản lý API key phục vụ cho các mô hình AI (api\_keys.py).

Luồng hoạt động tổng quát:

1. Người dùng gửi yêu cầu quét qua /api/scan.
2. Hệ thống tạo job\_id, chạy nền run\_scan, lưu trạng thái vào jobs.
3. Người dùng truy vấn trạng thái qua /api/status/{job\_id}.
4. Khi quét xong, kết quả được truy xuất qua /api/results/job/{job\_id} hoặc /api/results/{tool}.
5. API key được thêm qua /api/api-keys và liệt kê qua /api/api-keys.

Trích mã nguồn và phân tích chức năng

File orchestrator\_runner.py:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | router = APIRouter()  jobs = {} # job\_id -> {status, result, error, started\_at, finished\_at}  def run\_job(job\_id, req: ScanRequest):  jobs[job\_id]["status"] = "running"  jobs[job\_id]["started\_at"] = time.time()  try:  result = run\_scan(req.target, req.model, req.cookie, req.clean, req.input\_type)  jobs[job\_id]["result"] = result  jobs[job\_id]["status"] = "success"  except Exception as e:  jobs[job\_id]["status"] = "failed"  jobs[job\_id]["error"] = str(e)  jobs[job\_id]["finished\_at"] = time.time()  @router.post("/scan")  def scan(req: ScanRequest, background\_tasks: BackgroundTasks):  job\_id = str(uuid.uuid4())  jobs[job\_id] = {"status": "queued", "created\_at": time.time()}  background\_tasks.add\_task(run\_job, job\_id, req)  return {"status": "started", "job\_id": job\_id} |

* Chức năng: Tạo API endpoint không đồng bộ cho phép khởi chạy tác vụ quét (scan) ở background, theo dõi trạng thái quét và truy xuất kết quả khi hoàn thành.
* Giải thích:
* Khởi tạo cấu trúc lưu quét, lưu trạng thái đơn giản, không phụ thuộc cơ sở dữ liệu; phù hợp demo/PoC.
* Hàm chạy trạng thái quét thực tế: run\_job(job\_id, req). Tạo trạng thái quét hiện tại rồi gọi đến hàm run\_scan() trong orchestrator\_interactive.py để thực hiện quét.
* Sau khi quét thành công trả về tạng thái quét là success.
* Ý nghĩa: Mô hình điều phối HTTP cho hệ thống quét, nhận yêu cầu, lưu tiến trình quét , chạy run\_scan trong nền, lưu trạng thái và trả kết quả khi có.

File result.py:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | @router.get("/results/{tool}")  def get\_results(tool: str):  path = os.path.join(OUTPUT\_DIR, f"{tool}\_results.jsonl")  try:  with open(path, encoding="utf-8") as f:  lines = [json.loads(line) for line in f if line.strip()]  return lines  except Exception:  return [] |

* Chức năng: Tạo API endpoint {tool} nhằm mở file kết quả được lưu từ các lần quét và trả về file đó.
* Giải thích:
* Lưu đường dẫn với giá trị tool được gửi tới thành {tool\_result.jsonl.
* Thực hiện mở file với đường dẫn đã được lưu, chuyển mỗi dòng thành một giá trị trong mảng.
* Ý nghĩa: Là nơi chuyển giao kết quả quét tới giao diện để hiện thị chi người dùng.

### Mô hình Frontend

Frontend của hệ thống Pentest-Auto cung cấp giao diện người dùng đơn giản, trực quan để nhập URL, chọn mô hình AI, và khởi động quá trình quét; theo dõi tiến trình quét và hiển thị kết quả theo từng công cụ; quản lý API key phục vụ cho các module AI; hiển thị dữ liệu dưới dạng bảng hoặc khối raw JSON, có hỗ trợ tooltip để xem chi tiết.

Luồng hoạt động tổng quát:

1. Người dùng truy cập index.html để bắt đầu quét.
2. Nhập URL, chọn mô hình AI, thêm cookie nếu cần → nhấn "Quét".
3. Giao diện gọi startScan() trong app.js, gửi yêu cầu tới /api/scan.
4. Giao diện gọi pollJob() để theo dõi tiến trình quét.
5. Khi quét xong, kết quả được hiển thị qua loadAndShow(tool) → gọi /api/results/{tool}.
6. Người dùng có thể chuyển sang add\_api\_key.html để thêm API key.
7. Giao diện gọi addApiKey() và loadApiKeys() để cập nhật danh sách key

Trích mã nguồn và phân tích chức năng

**Giao diện khởi động quét nằm ở file index.html:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <section id="scan-form">  <select id="input-type">.......</select>  <input type="text" id="target" placeholder="Nhập URL hoặc IP" />  <select id="model">....</select>  <button onclick="toggleCookie()">Cookies</button>  <div id="cookie-input" style="display:none;">  <input type="text" id="cookie" placeholder="Nhập Cookie" />  </div>  <button onclick="startScan()">Quét</button>  </section>  <section id="scan-summary"></section>  <section id="results-tabs">  <!-- menu groups: Subdomains, Scanners, AI Enrich, AI Recursive -->  <div id="result-display"></div>  </section> |

* Chức năng: Giao diện web (HTML fragment) để khởi tạo tác vụ quét cho phép người dùng nhập target (URL hoặc IP), chọn model AI, nhập cookie tùy chọn, rồi xem các loại kết quả (subdomains, kết quả scanner, AI enrich, AI recursive).
* Giải thích:
* input-type (select): "url" hoặc "ip".
* target (text): Chuỗi URL / domain / IP.
* model (select): Chọn model AI.
* cookie (text, ẩn/hiện): Cookie tuỳ chọn.
* Quét: Khởi chạy tác vụ scan.
* Result tab: Gồm nhiều nút gọi loadAndShow(key) để hiển thị các kết quả quét trong result-display (subs, Nmap, Nikto, Nuclei, enriched\_\*, endpoints, vulns)
* Ý nghĩa: Front-end quản lý đầu vào và điều hướng chúng tới backend để thực thi các tác vụ nền.

**Đoạn mã javascript ứng với giao diện quét:**

**Khởi động quét với hàm startScan():**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | async function startScan() {  const input\_type = document.getElementById("input-type").value;  const target = document.getElementById("target").value;  const model = document.getElementById("model").value;  const cookie = document.getElementById("cookie").value;  const clean = document.getElementById("clean")?.checked || false;  const summary = document.getElementById("scan-summary");  try {  const res = await fetch("/api/scan", {  method: "POST",  headers: { "Content-Type": "application/json" },  body: JSON.stringify({ target, model, cookie, clean, input\_type })  });  const data = await res.json();  clearInterval(loading);  if (data.job\_id) {  summary.innerText = "Scan started. Đang xử lý...";  pollJob(data.job\_id);  } else  summary.innerText = " Không nhận được job\_id.";  } catch (err) {  clearInterval(loading);  summary.innerText = " Lỗi khi gửi yêu cầu quét: " + err.message;  }  } |

* Chức năng: khởi tạo một scan bằng cách gửi POST tới endpoint /api/scan với payload gồm target, model, cookie, clean, input\_type; sau khi nhận job\_id, hiển thị thông báo rồi gọi hàm pollJob(job\_id) để theo dõi tiến trình.
* Giải thích:
* Lấy giá trị được nhập từ người dùng trên trang web bao gồm input\_type, target, model, cookie.
* Gọi fetch POST /api/scan, gói body JSON {target, model, cookie, clean, input\_type} và header Content-Type: application/json.
* Chờ response, parse JSON. Gọi clearInterval(loading). Nếu response chứa data.job\_id, hiển thị "Scan started. Đang xử lý..." rồi gửi data.job\_id tới hàm pollJob để cập nhật trạng thái quét.
* Ý nghĩa: Hàm này đóng vai trò điểm bắt đầu của quy trình quét phía client trong quy trình làm việc bất đồng bộ (asynchronous workflow). Nó đảm nhiệm việc bắt đầu tác vụ, giám sát trạng thái, và nhận kết quả cuối cùng từ máy chủ mà không làm gián đoạn giao diện người dùng.

**Hiển thị kết quả quét với hàm loadAndShow(file):**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | async function loadAndShow(file) {  const el = document.getElementById("result-display");  const res = await fetch(`/api/results/${file}`);  const data = await res.json();  if (file === "enriched\_nikto" || file === "enriched\_nuclei") {  el.innerHTML = renderTable(toRowsFromFindings(data));  return;  }  if (file.startsWith("enriched") || file === "endpoints" || file === "vulns") {  el.innerHTML = renderTable(data);  return;  }  // array of raw records  if (Array.isArray(data)) {  if (data.length === 0) { el.innerHTML = "<p>Không có dữ liệu.</p>"; return; }  const parts = data.map((rec, i) => {  const title = rec.scanner || rec.target || rec.template\_name || `Record ${i+1}`;  let raw = rec.raw ?? rec.output ?? rec.stdout ?? "";  if (raw && typeof raw !== "string") raw = JSON.stringify(raw, null, 2);  const meta = `<div class="raw-meta"><strong>${escapeHtml(title)}</strong> -  ${escapeHtml(rec.timestamp || "")}</div>`;  const rawBlock = `<pre class="raw-output">${escapeHtml(raw ||  JSON.stringify(rec, null, 2))}</pre>`;  return `<div class="raw-record">${meta}${rawBlock}</div>`;  }).join("<hr/>");  el.innerHTML = `<h3>${file.toUpperCase()} Raw</h3>${parts}`;  return;  }  el.innerHTML = `<h3>${file.toUpperCase()} Raw</h3><pre class="raw-  output">${escapeHtml(JSON.stringify(data, null, 2))}</pre>`; |

* Chức năng: lấy dữ liệu JSON từ endpoint /api/results/{file} và hiển thị vào vùng #result-display. Tuỳ theo tên file và cấu trúc dữ liệu, nó chọn 1 trong các kiểu hiển thị: bảng (renderTable), bảng chuyển đổi từ findings (toRowsFromFindings), danh sách bản raw (meta + cho mỗi record) hoặc JSON.
* Giải thích:
* Gọi fetch GET /api/results/{file} và lấy kết quả Json từ response.
* Với từng file kết quả sẽ hiển thị khác nhau.
  + Với file bắt đầu bằng "enriched" hoặc là "endpoints" / "vulns": gọi renderTable(data) trực tiếp để hiển thị bảng. Tuy nhiên với file "enriched\_nikto" hoặc "enriched\_nuclei" thì sẽ dùng hàm toRowsFromFindings(data) để lấy mỗi data trong giá trị findings của file rồi hiện thi chúng ra bảng.
  + Nếu data là mảng: với mỗi phần tử cố gắng chọn tiêu đề (scanner/target/template\_name/Record N), lấy trường raw/ output/stdout nếu có (stringify nếu là object), dựng khối HTML meta + pre và ghép các record bằng thẻ <hr/>.
  + Nếu data là object (không phải mảng): trả kết quả chuỗi json bình thường..
* Ý nghĩa: cầu nối giữa menu chọn kết quả hiển thị trên giao diện người dùng và vùng hiển thị chi tiết, hỗ trợ nhiều định dạng output (scanner findings, enriched results, endpoints, raw outputs).

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.2 Giao diện quét

**Giao diện quản lý API key nằm ở file add\_api\_key.html:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | <h1>Thêm API Key</h1>  <input type="password" id="api-key" placeholder="sk-..." />  <button onclick="addApiKey()">Thêm</button>  <h2>Danh sách API Key</h2>  <table id="key-table">  <thead>  <tr><th>API Key</th></tr>  </thead>  <tbody></tbody>  </table>  <script>  window.onload = loadApiKeys;  function addApiKey() {  const key = document.getElementById("api-key").value;  const status = document.getElementById("status");  if (!key.trim()) return;  fetch("/api/api-keys", {  method: "POST",  headers: { "Content-Type": "application/json" },  body: JSON.stringify({ key })  })  .then(res => res.json())  .then(data => {  clearInterval(loading); // Dừng hiệu ứng  if (data.status === "ok") {  status.innerText = "Đã thêm!";  document.getElementById("api-key").value = "";  loadApiKeys();  } else {  status.innerText = "Thêm thất bại.";  }  });  }  </script> |

* Chức năng: Giao diện quản lý API key, cho phép nhập một API key (dạng "sk-..."), lưu/đăng ký nó vào hệ thống, và hiển thị danh sách các API key hiện có trong bảng).
* Giải thích:
* Mỗi khi trang tải, window.onload gọi loadApiKeys để tải danh sách hiện có từ GET /api/api-keys.
* Hiển thị API key ở dạng bảng với mỗi dòng là một API key.
* Input thêm API key thông qua thẻ input có dạng password.
* Nút "Thêm" (onclick="addApiKey()"): gọi hàm JS xử lý thêm khoá. Hàm addApiKey được gọi, lấy giá trị input, kiểm tra rỗng rồi POST tới /api/api-keys với body JSON { key }.
* Khi backend trả về, hiển thị trạng thái thành công hoặc thất bại, Nếu thành công gọi loadApiKeys để làm mới bảng hiển thị.
* Ý nghĩa: Cung cấp cho người dùng cách thêm (và xem) khoá truy cập để backend có thể sử dụng khi gọi dịch vụ bên thứ ba (ví dụ OpenAI).

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.3 Giao diện quản lý API key

## Kết luận chương 2

Chương 2 đã trình bày đầy đủ kiến trúc tổng thể và phân tích chi tiết các thành phần trong hệ thống kiểm thử bảo mật thông minh Pentest-Auto. Hệ thống được thiết kế theo mô hình client-server, tích hợp các công cụ kiểm thử truyền thống như Nmap, Nikto, Nuclei cùng với lớp phân tích bằng trí tuệ nhân tạo (AI) để nâng cao độ sâu và độ chính xác trong việc phát hiện lỗ hổng.

Thông qua việc phân tích từng mô hình từ frontend, backend API, orchestrator, scanners, collectors, đến AI layer và lưu trữ, chương này đã làm rõ vai trò, luồng xử lý, và mối liên kết giữa các thành phần trong hệ thống. Việc sử dụng mô hình ngôn ngữ lớn (LLM) giúp phân tích kết quả quét, phân tích endpoint và đánh giá khả năng khai thác, từ đó hỗ trợ người dùng không chuyên tiếp cận quy trình kiểm thử một cách hiệu quả.

Kiến trúc hệ thống đảm bảo khả năng mở rộng, dễ bảo trì, và có thể triển khai trên nhiều môi trường khác nhau. Đây là nền tảng vững chắc để bước sang chương 3, nơi các kịch bản kiểm thử và đánh giá hiệu quả hệ thống sẽ được thực hiện nhằm kiểm chứng tính đúng đắn và khả năng ứng dụng thực tế của Pentest-Auto.

# Thực Nghiệm

## Môi trường triển khai

### Hạ tầng phần cứng và hệ điều hành

**Mục tiêu**: cung cấp môi trường đủ tài nguyên để chạy cùng lúc Nmap, Nuclei, Nikto và các cuộc gọi AI (OpenAI), đồng thời lưu trữ kết quả JSONL, raw logs.

**Khuyến nghị cấu hình (môi trường kiểm thử / demo):**

* CPU: 4 vCPU (Intel/AMD)
* RAM: 8–16 GB
* Ổ cứng: 50 GB (giao động tùy lượng raw output); SSD ưu tiên để I/O nhanh
* Mạng: kết nối Internet ổn định (ưu tiên < 100 ms latency tới dịch vụ OpenAI nếu dùng cloud)

**Hệ điều hành:**

* Ubuntu LTS (22.04/20.04) hoặc Debian tương đương.
* Hoạt động tốt trên Windows Subsystem for Linux (WSL2) nhưng khuyến nghị dùng Linux nguyên bản cho subprocess/binary.

### Phiên bản phần mềm chính và phụ thuộc

**Ngôn ngữ lập trình Python:**

* Khuyến nghị sử dụng: Python phiên bản 3.10 hoặc 3.11 để đảm bảo tương thích và hiệu năng tốt.
* Môi trường ảo (virtual environment): Nên sử dụng venv hoặc virtualenv để cô lập các gói phụ thuộc, tránh xung đột giữa các dự án.

**Các binary scanner:**

* Nmap: >= 7.80 (bất kỳ phiên bản ổn định hiện hành, có hỗ trợ -oX XML output).
* Nuclei: phiên bản 2.x hoặc mới hơn (nhị phân accessible bằng biến môi trường NUCLEI\_BIN hoặc PATH).
* Nikto: v2.x (được gọi bằng lệnh Nikto; script giả định -ask no có sẵn).

**Thư viện Python cần thiết:** requests, beautifulsoup4, fastapi, uvicorn, pydantic, python-multipart, urllib3, py\_markdown\_table.

**Mô hình AI/API:**

* Sử dụng API OpenAI (hoặc endpoint tương tự). File chứa API keys: analyzers/data/api\_keys.txt (mỗi dòng 1 key).
* Mặc định mã dùng model tên gpt-4o-mini nhưng có thể thay đổi sang gpt-4 / gpt-3.5-turbo trong UI hoặc cấu hình.

### Chuẩn bị môi trường

**Cài Python + virtualenv:**

* sudo apt update && sudo apt install -y python3 python3-venv python3-pip build-essential.
* python3 -m venv .venv && source .venv/bin/activate

**Cài thư viện Python**: pip install fastapi uvicorn requests beautifulsoup4 pydantic py\_markdown\_table.

**Cài đặt binary scanners:**

* Nmap: sudo apt install -y nmap
* Nikto: sudo apt install -y nikto
* C: sudo apt install -y nuclei

**Chuẩn bị API key:** Tạo thư mục analyzers/data nếu chưa có, tạo file api\_keys.txt, thêm API key hợp lệ (mỗi dòng 1 key).

**Khởi chạy FastAPI** (để dùng giao diện người dùng): uvicorn main:app --reload.

**Truy cập giao diện web thực hiện thêm API key:**

Bước 1: Truy cập và OpenAI để tạo API key:

A screenshot of a computer

AI-generated content may be incorrect.

Bước 2: Nhập API key đã nhận được:

A white background with a black border

AI-generated content may be incorrect.

Bước 3: Bấm nút thêm và thành công thêm API key vào danh sách:

A screenshot of a computer

AI-generated content may be incorrect.

## Kịch bản thực nghiệm kiểm thử

Mục tiêu phần này là mô tả chi tiết từng kịch bản kiểm thử thực tế thực hiện trên hệ thống, bao gồm: mục tiêu, môi trường lab chi tiết, dữ liệu đầu vào, tham số cấu hình, bước thực hiện từng bước (cả ở mức API và CLI), cách thu thập bằng chứng/đầu ra, tiêu chí đánh giá và các biện pháp an toàn. Phần này tập trung vào ba kịch bản: IP đơn có SMB (Windows 7 + CVE với Metasploit module đề xuất), Web app vuln (XSS/SQLi etc.).

### Kịch bản 1

#### Mục tiêu

Mục tiêu của kịch bản này là thực hiện quét trên một địa chỉ IP đơn lẻ để kiểm tra các cổng dịch vụ nhằm xác định chính xác cổng đang mở; thu thập thông tin product/version từ banner hoặc fingerprinting; chuẩn hoá (normalize) các trường product, version, cổng vào định dạng chuẩn của hệ thống; và chạy bước AI-enrichment để gợi ý các CVE liên quan phù hợp với phiên bản phát hiện được. Nếu có ánh xạ sang module Metasploit khả dụng, ghi nhận thêm trường msf module chứa tên module tương ứng để hỗ trợ bước khai thác/poC.

Kết quả mong đợi: một bản ghi chuẩn hoá cho host này gồm ip, port, service, product/version chuẩn hoá, danh sách CVE gợi ý kèm mức độ tin cậy, và nếu tìm thấytrường msf\_module chứa module Metasploit tương ứng cùng bằng chứng (raw output) để phục vụ phân tích và tái tạo.

#### Môi trường lab chuẩn bị

 Hệ điều hành: Windows 7 (thường là Windows 7 Professional/Ultimate)

* IP: 10.30.10.143

 Cổng mở: port 445/tcp phải mở và có dịch vụ SMB (microsoft-ds) đang nghe.

 SMB version: SMBv1 được bật (với một số lỗ hổng như MS17-010 / EternalBlue yêu cầu SMBv1). Một số lỗ hổng khác có thể nhắm SMBv2/3 nhưng cần kiểm tra từng CVE cụ thể.

 Máy chưa được vá: máy chưa cài bản vá liên quan (ví dụ chưa cài bản vá MS17-010 cho EternalBlue). Nói chung là patch level thấp đối với các CVE liên quan.

 Không có bảo vệ bổ sung: tính năng bảo vệ như SMB signing bắt buộc, IDS/IPS chặn exploit hoặc firewall chặn port 445 không được cấu hình để ngăn exploit.

 Mạng có thể truy cập: máy tấn công phải truy cập thành công tới port 445 (không bị NAT/ACL chặn giữa hai bên).

 Quyền truy cập dịch vụ: tùy exploit, không cần credentials (exploit RCE thường không cần login), nhưng một số khai thác hoặc kiểm chứng có thể yêu cầu credentials hợp lệ.

#### Thực nghiệm

**Bước 1**: **Truy cập vào giao diện web của công cụ**

A screenshot of a computer

AI-generated content may be incorrect.

Giao diện hiển thị là một trang web đơn giản và trực quankèm các nút chức năng như chọn URL hay IP của mục tiêu cần kiểm thử để quét; chọn model quét, Cookies, Quit cùng với nút Thêm API Key và Trang chủ Ngoài ra ở bên dưới là phần Kết quả có các module nhanh: Subdomains, Scanners, AI Enrich và AI Recursive để hiển thị các kết quả sau quá trình quét.

**Bước 2**: **Khởi tạo quá trình quét**

Bắt đầu quét bằng cách chọn chức năng IP rồi nhập IP mục tiêu là 10.30.10.143, chọn model AI gpt-4o-mini, Cuối cùng bấm nút Quét để bắt đầu thực hiện chức năng quét và phân tích bằng AI:

A screenshot of a computer

AI-generated content may be incorrect.

Sau khi bấm nút Quét trên giao diện đã hiển thị 1 thông báo trạng thái "Scan started. Đang xử lý...". Để rõ hơn các tiến trình, truy cập vào burpsuite để xem request được gửi đi:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả cho thấy web đã gửi 1 yêu cầu POST tới endpoint /api/scan với nôik dung JSON chứa target, model, cookie, clean và input\_type. Máy chủ trả về HTTP 200 với JSON xác nhận scan đã khởi chạy: status = "started" và một job\_id duy nhất (ví dụ "9761c65c-aa92-4064-88fc-4d1f3f46d066").

Theo như phân tích từ chương 2, mỗi khi quét thì hệ thống sẽ tạo một tiến trình quét và gọi đến nó để xem tiến độ. Mỗi khi hệ thống kiểm tra sẽ gửi một yêu cầu GET tới endpoint /api/status/<job\_id>. Vì vậy truy cập burp suite để xem kĩ chức năng này:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả là phản hồi JSON cho biết trạng thái công việc: status: running, error: null, result\_available: false. Điều này cho thấy tác vụ quét đã được khởi tạo và đang được xử lý trên server.

Sau một thời gian, quá trình quét đã được thực hiện xong, truy cập burp suite để theo dõi endpoint /api/status/<job\_id>:

A screenshot of a computer screen

AI-generated content may be incorrect.

Phản hồi đã trả về JSON với trạng thái ok cùng với 1 bản tóm tắt kết quả quét. Thử truy cập lại vào giao diện web thì cũng thấy đã có bản tóm tắt kết quả hiển thị. Điều này chứng minh việc quét trên IP 10.30.10.143 đã hoàn thành:

A white screen with a couple of text boxes

AI-generated content may be incorrect.

**Bước 3: Truy cập vào các mục kết quả để xem kết quả quét và phân tích**

Thực hiện bấm vào Scanner, chọn Nmap để xem kết quả quét:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả quét Nmap cho host 10.30.10.149 cho thấy có một vài dịch vụ đang được mở. Theo như nhưng gì đã được hiển thị trong ảnh thì có 2 dịch vụ đang được mở ở cổng 135 và 139. Ngoài ra còn cách dịch vụ khác nhưng không đủ chỗ hiển thị nên sẽ được ghi ở phần phụ lục.

Theo như phân tích chương 2, khi bấm vào mục kết quả tương ứng với công cụ thì hệ thống sẽ gọi tới api/results/{tool} để đọc file kết quả quét và hiển thị lên mà hình. Thực hiện truy cập vào burp suite để kiểm tra lại và đọc được mục tiêu của bài này ở port 445:

A screenshot of a computer screen

AI-generated content may be incorrect.

Kết quả quét đã hiển thị toàn bộ thông tin liên quan đến trạng thái cổng, dịch vụ và phiên bản của dịch vụ tương ứng.

Tiếp đó chuyển qua mục AI\_Enrich để xem phân tích của OpenAI từ kết quả của việc quét bằng Nmap:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả trên burpsuite:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả phân tích cho thấy hệ thống ghi nhận tổng cộng bốn lỗ hổng (enriched\_count = 4), tương ứng với bốn dòng dữ liệu trong bảng kết quả. Bốn mục này thực chất là hai mã CVE gồm CVE-2020-0601 và CVE-2019-0708, mỗi CVE xuất hiện trên hai cổng dịch vụ khác nhau là 139 và 445.

Đáng chú ý, ở cổng 445 (microsoft-ds), cả hai lỗ hổng đều được ánh xạ sang module khai thác Metasploit, giúp hỗ trợ kiểm thử khai thác tự động. Trong đó:

* CVE-2020-0601 có module exploit/windows/smb/ms17\_010\_eternalblue (đây là module mục tiêu của kịch bản này).
* CVE-2019-0708 ngoài module Metasploit exploit/windows/rdp/cve \_2019\_0708\_rdp còn có đường dẫn đến mã khai thác độc lập, cho phép khai thác mà không cần Metasploit.

**Bước 4: Thực hiện khai thác với module tìm được**

Thực hiện khai thác CVE-2020-0601 bằng cách truy cập vào metasploit, sử dụng module exploit/windows/smb/ms17\_010\_eternalblue. Đặt RHOST là IP 10.30.10.143 của mục tiêu và khởi chạy module để tiến hành khai thác:

A screenshot of a computer program

AI-generated content may be incorrect.

Thành công truy cập được vào máy tính mục tiêu sau khi khai thác bằng metasploit. Tiếp đó sử dụng lệnh shell để thực hiện truy cập vào shell của máy tính mục tiêu:

A screen shot of a computer

AI-generated content may be incorrect.

Kết quả được hiển thị cho thấy rằng module đã khai thác thành công vào CVE-2020-0601 thông qua module exploit/windows/smb/ms17\_010\_eternalblue. Module này đã thành công truy cập được vào shell của máy tính Windows 7 chạy SMB trên cổng 445 dưới quyền nt authority, đây là quyền cao nhất của window.

Tiếp theo, với mã khai thác của CVE-2019-0708 là 44256, sử dụng searchsploit để thực hiện lấy mã khai thác và chạy:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả cho thấy để sử dụng thành công file khai thác này thì máy bị tấn công phải sử dụng dịch vụ web. Vì vậy điều này là không khả thi.

Để có thêm thông tin về CVE-2019-0708, thực hiện tìm kiếm thêm trên mạng:

A screenshot of a computer program

AI-generated content may be incorrect.

Kết quả cho thấy rằng CVE-2019-0708 chỉ hoạt động trên RDP trong khi cổng 445 chạy SMB trực tiếp. Điều này chứng tỏ rằng không thể khai thác được với CVE-2019-0708,

Ngoài ra ở cổng 139 đã được cài đặt là NetBIOS Session Service là SMB chạy thông qua NetBIOS, không phải trực tiếp như cổng 445 nên CVE-2020-0601 không thể khai thác được, điều đó được chứng minh qua kết quả khai thác bằng metasploit dưới đây:

A screen shot of a computer

AI-generated content may be incorrect.

Lý do xảy ra điều này là cổng 139 yêu cầu thiết lập IPC$ session trước khi gửi gói SMB TRANS2. Nếu IPC$ không cho phép session (do policy), module check không thể tiếp tục.

#### Kết luận

Như vậy kịch bản đã cho thấy khả năng xác minh trạng thái các cổng dịch vụ, thu thập và chuẩn hoá thông tin dịch vụ, lưu trữ bằng chứng thô và sử dụng thành phần AI để gợi ý các CVE tiềm năng cùng khả năng ánh xạ sang module Metasploit của công cụ Pentest-Auto; kết quả trên host 10.30.10.143 cho thấy dấu hiệu của nhiều CVE có rủi ro thực thi mã từ xa trên nền Windows cũ, khẳng định tính hiệu quả của hướng tiếp cận tích hợp tự động hóa, thu thập dữ liệu chủ động kết hợp với phân tích ngữ nghĩa bằng AI, trong hỗ trợ phát hiện lỗ hổng ban đầu.

### Kịch bản 2

#### Mục tiêu

Mục tiêu của kịch bản này là đánh giá khả năng thu thập tên miền phụ (subdomain) và đánh giá chuỗi xử lý quét tự động của hệ thống Pentest-Auto trên một mục tiêu mẫu (example.com). Cụ thể:

* Xác minh khả năng thu thập subdomain từ nguồn chứng chỉ công khai (crt.sh) thông qua cơ chế truy vấn tự động của công cụ; so sánh và đánh giá độ phủ (coverage) và mức độ chính xác của kết quả thu được.
* Dùng Nmap để thực hiện quét các địa chỉ IP tương ứng với từng subdomain thu thập được, nhằm kiểm tra trạng thái cổng, dịch vụ và thông tin phiên bản (banner/version) của các dịch vụ chạy trên mỗi IP.
* Đánh giá khả năng phân tích kết quả quét bằng mô-đun AI của hệ thống: chuyển kết quả thô thành các bản ghi đã chuẩn hoá (normalized findings), xác định lỗ hổng tiềm ẩn và gợi ý hành động khuyến nghị.

Kết quả mong đợi: thu được bộ danh sách subdomain từ crt.sh, thực hiện quét Nmap trên các IP tương ứng, và nhận về báo cáo phân tích hợp lệ từ mô-đun AI, kèm bằng chứng (raw output) và các bản ghi chuẩn hoá phục vụ phân tích tiếp theo.

#### Thực nghiệm

**Bước 1**: **Khởi tạo quá trình quét**

Trên giao diện web, chọn chế độ nhập URL rồi nhập URL http://example.com vào rồi thực hiện quét:

A screenshot of a computer

AI-generated content may be incorrect.

Sau một thời gian, giao diện web hiển thị bản tóm tắt quá trình quét cho thấy rằng đã quét thành công và phát hiện chín địa chỉ IP liên quan tới tên miền chính:

A screenshot of a computer

AI-generated content may be incorrect.

**Bước 2: Kiểm tra kết quả**

Chọn mục Subdomains để xem thông tin quét các miền phụ:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả cho thấy hệ thống có nhiều tên miền phụ, mỗi tên miền được ánh xạ tới địa chỉ IP tương ứng; ví dụ dev.example.com, m.example.com, ... . Mỗi miền phụ đều có IP tương ứng của mình. Vì vậy, chuyển qua xem kết quả quét bằng Nmap:

A screenshot of a computer

AI-generated content may be incorrect.

Giao diện hiển thị cho thấy rằng kết quả quét Nmap trên từng ip, trong đó với bản ghi đầu tiên của ip 125.235.4.59 không phát hiện cổng, dịch vụ nào cả. Tương tự, các IP khác cũng không phát hiện dịch vụ nào. Do đó hệ thống sẽ không thể phân tích kết quả quét Nmap bằng AI được:

A screenshot of a computer

AI-generated content may be incorrect.

Chuyển qua xem kết quả quét bằng Nuclei:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả cho thấy rằng có bốn template được phát hiện. Tuy nhiên cả bốn template này đều chỉ ở mức thông tin (info) như sau:

* Template azure-domain-tenant phát hiện rằng example.com có thể trỏ hoặc liên quan đến tenant Azure AD có ID.
* Template external-service-interaction cho thấy rằng server có phản hồi khi gửi request HTTP từ Nuclei, nhưng không phát hiện bất kỳ tương tác outbound bất thường nào.
* Template waf-detect:akamai xác định website đang được bảo vệ bởi WAF Akamai.
* Template request-based-interaction:dns kiểm tra khả năng tạo tương tác DNS ra bên ngoài (out-of-band).

Tất cả template này chỉ cung cấp thêm thông tin về ứng dụng web chứ chưa có nguy hiểm gì xảy ra. Điều đó cũng được ghi lại ở dưới kết quả phân tích AI (Toàn bộ kết quả được lưu ở phần phụ lục):

A screenshot of a computer

AI-generated content may be incorrect.

Chuyển qua kết quả quét của Nikto:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả quét của Nikto cũng phát hiện ra một vài lỗ hổng. Tiếp đó chuyển qua xem phần phân tích của AI:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả phân tích cho thấy có hai mục được xếp ở mức medium là thiếu X-Frame-Options và X-Content-Type-Option. Điều này có thể dẫn đến tấn công clickjacking và tấn công MIME sniffing. Tuy nhiên trang web này chỉ thiết kế là một trang tĩnh để quét nên việc thiếu tiêu đề này cũng không có ảnh hướng gì cả. Tương tự hai mục còn lại nói về việc thay đổi server banner từ AkamaiNetStorage sang AkamaiGHost và đăng nhập thử với tài khoản mặc định của Akamai Storage cũng không có tác dụng, phần quét tìm endpoint và lỗ hổng trên trang web cũng không hoạt động được ở mục tiêu kịch bản này.

#### Kết luận

Kịch bản đã cho thấy hệ thống Pentest-Auto vận hành đúng quy trình: thu thập subdomain từ crt.sh, ánh xạ IP và thực hiện quét tự động bằng Nmap, Nuclei và Nikto. Kết quả cho thấy hầu hết IP của example.com không mở dịch vụ đáng chú ý, nên Nmap không tạo ra bản ghi phân tích AI, phù hợp với đặc tính của domain mẫu. Trong khi đó, Nuclei và Nikto ghi nhận một số thông tin ở mức informational và medium như WAF Akamai, thiếu một số header bảo mật hoặc thay đổi banner. Mô-đun AI phân loại chính xác mức rủi ro và đưa ra giải thích hợp lý. Tổng thể, kịch bản chứng minh pipeline của hệ thống hoạt động ổn định, tự động hóa tốt và phân tích AI chính xác, tạo nền tảng để triển khai trên các mục tiêu phức tạp hơn trong các kịch bản tiếp theo.

### Kịch bản 3

#### Mục tiêu

Mục tiêu của kịch bản này là đánh giá toàn diện khả năng của công cụ Pentest-Auto khi tiến hành kiểm thử trên một máy chủ web được xây dựng theo mô hình dễ khai thác. Thông qua kịch bản này, hệ thống sẽ được kiểm tra khả năng tích hợp và vận hành đồng thời nhiều công cụ kiểm thử bảo mật khác nhau, bao gồm Nmap, Nikto và Nuclei, nhằm thu thập thông tin dịch vụ, phát hiện cấu hình sai, và nhận diện các lỗ hổng tiềm ẩn. Kết quả từ các công cụ quét này sau đó sẽ được chuyển sang mô-đun AI để phân tích chuyên sâu, chuẩn hóa dữ liệu và đưa ra đánh giá rủi ro một cách hệ thống.

Bên cạnh khả năng quét truyền thống, kịch bản còn hướng đến việc kiểm tra tính thông minh của công cụ trong khâu phân tích mở rộng. Cụ thể, mô-đun AI sẽ được sử dụng để tự động phát hiện các endpoint trên ứng dụng web mục tiêu, bao gồm cả các đường dẫn và tham số có thể không được liệt kê rõ ràng trong giao diện. Trên từng endpoint thu được, AI sẽ tiến hành phân tích phương thức xử lý (method), tham số đầu vào, và mô hình phản hồi để dự đoán các lỗ hổng tiềm năng như SQL Injection, XSS, LFI/RFI, Command Injection, File Upload…, từ đó xây dựng danh sách rủi ro có khả năng xảy ra trên từng khu vực của hệ thống.

Ngoài ra, kịch bản cũng kiểm tra khả năng sinh payload của AI dựa trên từng loại lỗ hổng được dự đoán. Các payload được tạo ra phải bám sát hành vi từng endpoint, phù hợp với đặc điểm lỗ hổng, và có khả năng cao hỗ trợ kiểm thử thực tế. Việc phân tích này giúp đánh giá năng lực hỗ trợ khai thác bán tự động của công cụ, hướng tới xây dựng một pipeline kiểm thử đầy đủ từ quét – phân tích – phát hiện, gợi ý khai thác.

Kết quả mong đợi: bao gồm một bản ghi chi tiết về toàn bộ các endpoint được AI trích xuất, kèm theo phương thức truy cập, tham số liên quan và danh sách lỗ hổng tiềm năng tương ứng. Đồng thời, một bản ghi phân tích payload cũng được tạo ra, thể hiện nhóm payload phù hợp cho từng lỗ hổng và đánh giá khả năng áp dụng vào thực tế. Qua đó, kịch bản này giúp xác định rõ mức độ thông minh và độ chính xác của hệ thống trong việc tự động hóa kiểm thử an toàn ứng dụng web.

#### Môi trường lab chuẩn bị

* Hệ điều hành: Linux Debian.s
* IP: 10.30.10.130.
* Cổng mở: 22/tcp cho SSH, 80/tcp cho HTTP Apache, 3306/tcp cho MySQL.
* Ứng dụng web: Web vulnerable chứa các cơ chế upload file, xử lý session bị dính các lỗ hổng web File Upload, LFI.
* Không có cơ chế bảo vệ: IDS/IPS chặn exploit hoặc firewall không được cấu hình để chặn cổng web/SSH.
* Máy tấn công có thể kết nối đầy đủ đến các cổng 22/80/3306 mà không bị NAT, ACL hoặc filtering.

#### Thực nghiệm

**Bước 1**: **Khởi tạo quá trình quét**

Trên giao diện web, chọn chế độ nhập URL rồi nhập URL http://10.30.10.130 vào và thực hiện quét:

A white rectangular object with a blue stripe

AI-generated content may be incorrect.

Sau một thời gian, giao diện web hiển thị bản tóm tắt quá trình quét cho thấy đã quét thành công:

A screenshot of a computer

AI-generated content may be incorrect.

**Bước 2: Kiểm tra kết quả**

Kiểm tra kết quả quét Nmap:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả quét Nmap cho IP của ứng dụng web cho thấy có một vài dịch vụ đang được mở. Theo như nhưng gì đã được hiển thị trong ảnh thì có 3 dịch vụ đang được mở ở các cổng:

* 80 cho dịch vụ web server Apache.
* 111/ cho dịch vụ rcpbind.
* 3306 cho dịch vụ MySQL.

Với các thông tin trên, chuyển qua phần AI Enrich cho Nmap để xem phân tích của AI:

A screenshot of a computer

AI-generated content may be incorrect.

Từ kết quả phân tích, mỗi dịch vụ đều có hai CVE nhưng AI không tìm thấy mã khai thác và msf module tương ứng. Hai CVE ở dịch vụ web server Apache mô tả về lỗ hổng trong việc thực hiện tấn công từ chối dịch vụ; hai CVE ở dịch vụ rpc leo thang đặc quyền và CVE-2013-2018 được cho là AI tạo ra thông tin sai lệch vì CVE này chỉ liên quan đến nginx có phiên bản 1.3.9 tới 1.4.0; ở dịch vụ MySQL thì cả hai CVE đều liên quan đến leo thang đặc quyền. Và thực tế các CVE liên quan đều không thể thực thi được vì hệ thống thử nghiệm đều đã được cấu hình để tránh các cách khai thác CVE này dù cho phiên bản dịch vụ có phù hợp.

Xem kết quả quét Nuclei:

A screenshot of a computer screen

AI-generated content may be incorrect.

Xem kết quả đánh giá của AI:

A screenshot of a computer

AI-generated content may be incorrect.

Các kết quả đều cho thấy ứng dụng web thiếu các header bảo mật (CSP, Permissions‑Policy, Referrer‑Policy, X‑Frame‑Options, X‑Content‑Type‑Options, X‑XSS‑Protection, Clear‑Site‑Data, Cross‑Origin policies). Ngoài ra mức độ nghiêm trọng chỉ được báo là “info”, chỉ cung cấp thông tin và không ảnh hưởng trực tiếp đến an toàn hệ thống. Trong bối cảnh kịch bản 3 này, các thiếu sót trên không gây ra nguy hiểm.

Xem kết quả quét Nikto:

A screenshot of a computer

AI-generated content may be incorrect.

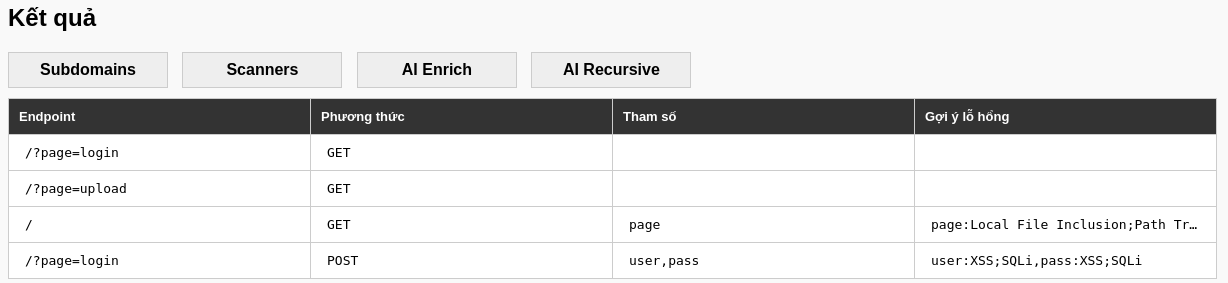
Xem kết quả phân tích của AI:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả quét và phân tích từ Nikto cho biết rằng ứng dụng web có một số endpoint đáng chú ý, bao gồm login.php, config.php, #wp-config.php, cùng một số thư mục và tệp tin khác (/images/, /icons/README). Ngoài ra kết quả cũng cho biết ứng dụng web thiếu các header bảo mật quan trọng như X-Frame-Options và X-Content-Type-Options như trong Nuclei trước đó.

Xem kết quả quét của module AI Recursive:



Kết quả quét từ hệ thống Pentest-Auto cho thấy ứng dụng web đang cung cấp endpoint gốc “/”, trong đó tham số page được dùng để điều hướng tới các tệp chức năng khác như login và upload, tạo thành các đường dẫn kiểu:

* /?page=login
* /?page=upload

Cơ chế xử lý này cho thấy ứng dụng có khả năng tải các tệp dựa trên giá trị do người dùng truyền vào. Theo phân tích tự động của mô-đun AI-enricher, tham số page tiềm ẩn nguy cơ bị khai thác dẫn đến các lỗ hổng:

* Path Traversal: kẻ tấn công có thể thao túng tham số để truy cập thư mục hoặc tệp ngoài phạm vi cho phép.
* Local File Inclusion (LFI): ứng dụng có thể vô tình tải và hiển thị nội dung từ các tệp nội bộ thông qua tham số page.
* Remote File Inclusion (RFI): ứng dụng có thể vô tình tải và hiển thị nội dung từ các tệp bên ngoài thông qua tham số page.

Ngoài ra, tại endpoint /?page=login, hệ thống phát hiện rằng các tham số user và pass có thể dễ bị khai thác bởi:

* Cross-Site Scripting (XSS): dữ liệu đầu vào có khả năng không được lọc hoặc mã hóa đúng cách trước khi phản hồi về trình duyệt.
* SQL Injection (SQLi): dữ liệu truyền vào có thể được đưa trực tiếp vào truy vấn cơ sở dữ liệu mà không qua cơ chế kiểm tra an toàn, cho phép kẻ tấn công can thiệp hoặc thực thi truy vấn độc hại.

Từ kết quả này, AI sẽ thực hiện gửi các payload tương ứng với lỗ hổng đã được gợi ý. Thực hiện xem kết quả phân tích của AI:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả phân tích chi tiết cho biết lỗ hổng LFI tại tham số page có mức độ khả năng xảy ra cao vì khi đưa vào payload php://filter/convert.base64-encode/resource=index trang web đã trả về chuỗi mã hóa base64 của index.php. Và như mong đợi, các lỗ hổng khác đều có khả năng khai thác thấp vì những lỗ hổng như XSS, SQLi đã được cấu hình hoặc hạn chế một phần, khiến việc khai thác các lỗ hổng này không thể xảy ra một cách trực tiếp hoặc dễ dàng.

**Bước 3: Thực hiện khai thác với kết quả quét được:**

Như ở trên kết quả phân tích bằng Nikto, công cụ Pentest-Auto đã phát hiện một số endpoint quan trọng trong đó có endpoint config.php, đây là một tệp cấu hình nhạy cảm chứa thông tin quan trọng như cấu hình cơ sở dữ liệu, khóa bí mật hoặc thông tin đăng nhập. Kết hợp với payload LFI vừa phân tích được với mức độ khai thác chi tiết cao, thực hiện tạo payload php://filter/convert.base64-encode/resource=config để thực hiện đọc mã nguồn của file config.php từ dạng base64:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả thu được một đoạn chuỗi base64 cho payload vừa được gửi. Thực hiện giải mã base64:

A screenshot of a computer

AI-generated content may be incorrect.

Sau khi giải mã, đã thu được tệp cấu hình chứa thông tin đăng nhập cơ sở dữ liệu ở bản rõ, bao gồm: Server: localhost, Username: root, Password: H4u%QJ\_H99, Database: Users.

Căn cứ vào kết quả quét trước đó của Nmap cho thấy dịch vụ MySQL đang mở công khai trên cổng 3306, kẻ tấn công có thể sử dụng trực tiếp các thông tin này để đăng nhập vào hệ quản trị cơ sở dữ liệu:

A screenshot of a computer screen

AI-generated content may be incorrect.

Sau khi đăng nhập thành công vào hệ quản trị MySQL với quyền root sử dụng thông tin thu được từ config.php, bước tiếp theo là thực hiện truy vấn dữ liệu từ cơ sở dữ liệu Users, cụ thể là bảng users bằng lệnh: select \* from users để đọc thông tin người dùng và lấy được user, pass của bảng users. Kết quả trả về cho thấy toàn bộ thông tin người dùng đã được truy xuất, bao gồm tên đăng nhập (user) và mật khẩu (pass) dưới dạng base64.

Thực hiện giải mã mật khẩu:

A white rectangular object with a black border

AI-generated content may be incorrect.

Tiếp tục đăng nhập vào người dùng kent với mật khẩu tương ứng đã được giải mã được:

A screenshot of a computer

AI-generated content may be incorrect.

Sau khi đăng nhập thành công, truy cập thành công vào endpoint /?page=upload:

A screenshot of a computer

AI-generated content may be incorrect.

**Bước 4:** **Khởi tạo lại quá trình quét**

Với việc đăng nhập thành công, hệ thống đã cấp cho trình duyệt một cookie phiên hợp lệ. Cookie này đóng vai trò định danh người dùng và là điều kiện bắt buộc để truy cập các chức năng hoặc endpoint yêu cầu quyền xác thực, cụ thể là endpoint /?page=upload.

Để đánh giá đầy đủ bề mặt tấn công của ứng dụng ở chế độ đã xác thực, tiến hành chạy lại công cụ Pentest Auto với việc bổ sung giá trị cookie thu được vào cấu hình quét:

A close-up of a computer screen

AI-generated content may be incorrect.

Kiểm tra kết quả quét endpoint của công cụ:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả lần này đã xuất hiện thêm phương thức POST trong endpoint /?page=upload cùng với lỗ hổng File upload. Từ đó chuyển qua xem kết quả phân tích payload với lỗ hổng tương ứng của công cụ:

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả cho biết khi upload file shell.gif và exploit.png chứa mã độc bên trong có khả năng cao khai thác thành công lỗ hổng File upload nhằm thực hiện RCE.

**Bước 5: Thực hiện khai thác với kết quả quét được**

Vì kết quả quét chỉ ở mức gợi ý, vẫn chưa thể khẳng định lỗ hổng xảy ra được. Do đó, bước tiếp theo trong quá trình kiểm thử là tiến hành khai thác thủ công (manual exploitation) dựa trên những gợi ý mà công cụ Pentest Auto đưa ra.

Sử dụng burp suite lấy request POST /?page=upload với mục tiêu là kiểm tra khả năng upload file reverse shell với:

* Tên tệp: reverseshell.gif.
* Content-Type: image/png (đặt giả để đánh lừa bộ lọc).
* Magic byte trong nội dung: GIF87a.

A screenshot of a computer

AI-generated content may be incorrect.

Sau khi gửi payload thành công, tiến hành truy cập vào endpoint chứa reverse shell. Tại thời điểm này, quay trở lại phiên Netcat đang lắng nghe để nhận kết nối reverse shell từ máy chủ:

A computer screen shot of white text

AI-generated content may be incorrect.

Kết quả cho thấy việc khai thác đã thành công, cho phép truy cập vào shell của máy chủ web.

#### Kết luận

Kịch bản đã chứng minh rằng công cụ Pentest-Auto có khả năng vận hành hiệu quả toàn bộ chuỗi kiểm thử an toàn ứng dụng web từ thu thập thông tin, phân tích lỗ hổng, gợi ý khai thác đến hỗ trợ tấn công bán tự động. Hệ thống đã phát hiện đúng các điểm yếu quan trọng như LFI và File Upload, đồng thời mô-đun AI xử lý tốt việc phân tích endpoint, dự đoán rủi ro và sinh payload phù hợp. Việc khai thác thủ công dựa trên gợi ý của công cụ dẫn đến truy cập được vào cơ sở dữ liệu và đạt được shell trên máy chủ web, cho thấy khả năng hỗ trợ kiểm thử thực tế của hệ thống. Qua thực nghiệm, công cụ đáp ứng tốt mục tiêu đánh giá, thể hiện mức độ chính xác và tính thông minh cần thiết cho một pipeline kiểm thử an toàn tự động hóa.

## Đánh giá hiệu quả, hạn chế và hướng phát triển

### Đánh giá hiệu quả

Công cụ hoạt động ổn định và tích hợp tốt các trình quét như Nmap, Nuclei và Nikto. Dữ liệu thô sau khi quét được chuyển đổi thành dạng chuẩn hóa, có cấu trúc rõ ràng, thống nhất và thuận tiện cho việc phân tích. Bên cạnh đó, chức năng tự động thu thập tên miền phụ từ nền tảng crt.sh hoạt động chính xác, trả về danh sách đầy đủ và không bị bỏ sót.

Module AI enrich có khả năng đọc hiểu dữ liệu đầu vào, nhận dạng dịch vụ, đối chiếu phiên bản và đưa ra gợi ý mã lỗi bảo mật (CVE) phù hợp. Điều này đặc biệt thể hiện rõ trong kịch bản 1 với độ chính xác cao. Ngoài ra, module AI recursive phân tích được endpoint động (dynamic endpoint discovery) hiệu quả, dự đoán đúng các nhóm lỗ hổng tiềm năng , dự đoán đúng các nhóm lỗ hổng tiềm ẩn, hỗ trợ quá trình kiểm thử nhanh và chính xác hơn.

Khả năng hỗ trợ khai thác của hệ thống thể hiện qua các điểm sau:

* Module Metasploit được gợi ý chính xác cho SMB.
* Payload khai thác LFI được AI đề xuất trực tiếp và thử nghiệm thành công.
* Kết hợp nhiều nguồn dữ liệu (AI + Nmap + cấu hình web) để đưa ra kết quả đầy đủ, có giá trị thực tiễn.

Hệ thống vận hành theo một chu trình hàon chỉnh Scanning → Normalization → AI Enrichment → Evidence Storage → Suggest Exploit → Verification, mang lại hiệu quả cao trong cả thử nghiệm hạ tầng và ứng dụng web.

### Hạn chế

Mặc dù hệ thống Pentest-Auto đạt được nhiều kết quả tích cực trong ba kịch bản thực nghiệm, quá trình đánh giá cũng chỉ ra một số hạn chế quan trọng cần xem xét. Thứ nhất, khả năng nhận diện các lỗ hổng phức tạp vẫn chưa thực sự toàn diện. Hệ thống mới dừng lại ở việc phát hiện các lỗ hổng mang tính mẫu hình (pattern-based) hoặc dựa trên hành vi đơn lẻ, chưa đủ năng lực phân tích mối liên hệ nhiều bước, chẳng hạn như chuỗi khai thác kết hợp (multi‑stage exploitation) hoặc các lỗ hổng đòi hỏi hiểu sâu logic ứng dụng. Điều này khiến hệ thống có thể bỏ sót các tình huống đặc thù trong ứng dụng web hoặc môi trường thực tế.

Thứ hai, độ chính xác của AI phụ thuộc lớn vào chất lượng dữ liệu đầu vào. Khi dữ liệu thu thập không đầy đủ hoặc thiếu nhất quán, kết quả phân tích có thể bị nhiễu và dẫn đến tình trạng AI suy diễn, tạo ra thông tin không chính xác. Điều này khiến một số gợi ý về mã lỗi bảo mật (CVE) không phù hợp với ngữ cảnh thực tế. Bên cạnh đó, AI chỉ dừng lại ở mức đề xuất loại lỗ hổng tiềm ẩn mà không thể cung cấp payload khai thác do ràng buộc về mặt pháp lý. Vì vậy, toàn bộ payload đều phải được xây dựng thủ công và cần được mô tả rõ ràng để tránh tình trạng cảnh báo sai hoặc phát sinh dữ liệu giả gây nhiễu quá trình phân tích.

Thứ ba, hiệu suất của hệ thống chưa tối ưu khi áp dụng cho môi trường quy mô lớn. Khối lượng dữ liệu sinh ra từ nhiều công cụ quét cùng lúc làm tăng thời gian xử lý, ảnh hưởng đến tốc độ phân tích và khả năng cung cấp kết quả theo thời gian thực. Ngoài ra, khả năng tổng hợp và xuất báo cáo vẫn còn hạn chế. Hệ thống chưa tạo được báo cáo hoàn chỉnh theo chuẩn công nghiệp (OWASP, MITRE, CVSS) và chưa hỗ trợ tích hợp dễ dàng với các nền tảng quản lý an ninh khác (SIEM/SOAR).

Cuối cùng, giao diện và trải nghiệm người dùng còn ở mức cơ bản, chưa hỗ trợ phân quyền, quản lý phiên làm việc, hay lưu trữ lâu dài các cấu hình quét. Điều này hạn chế khả năng sử dụng trong các tổ chức lớn hoặc các nhóm kiểm thử yêu cầu hệ thống quản trị chuyên nghiệp.

### Hướng phát triển

Để khắc phục các hạn chế đã được nêu ra và nâng cao năng lực của hệ thống trong tương lai, một số định hướng phát triển quan trọng được đề xuất như sau:

Trước hết, mô-đun AI cần được nâng cấp để hiểu ngữ cảnh sâu hơn và có khả năng phân tích logic ứng dụng ở mức cao hơn. Việc ứng dụng mô hình học sâu, phân tích đồ thị luồng xử lý (request–response graph) hoặc mô hình ngữ nghĩa API có thể giúp hệ thống dự đoán các lỗ hổng phức tạp, bao gồm chuỗi tấn công kết hợp, điều kiện đua, hoặc sai sót trong kiểm soát truy cập. Đồng thời, cần mở rộng nguồn dữ liệu huấn luyện AI từ nhiều nền tảng khác nhau (NVD, OSV, Exploit‑DB, GitHub Security Advisories) nhằm đảm bảo độ bao phủ rộng hơn và tính cập nhật liên tục.

Bên cạnh đó, việc tối ưu hóa hiệu năng là cần thiết để hệ thống hoạt động tốt trong môi trường doanh nghiệp hoặc các hệ thống lớn. Có thể nghiên cứu triển khai mô hình xử lý song song hoặc phân tán (distributed processing), cho phép thực hiện nhiều tác vụ phân tích cùng lúc, giảm đáng kể thời gian quét và xử lý dữ liệu.

Một hướng phát triển quan trọng khác là cải thiện năng lực gợi ý khai thác tự động. Hệ thống có thể bổ sung cơ chế kiểm thử an toàn giả lập (sandbox exploitation), cho phép AI thử nghiệm payload trong môi trường cô lập trước khi đề xuất cho người dùng. Điều này vừa tăng độ chính xác, vừa giảm rủi ro gây tác động không mong muốn lên hệ thống mục tiêu. Ngoài ra, việc tích hợp sâu với Metasploit, Burp Suite API hoặc các framework khai thác khác sẽ giúp công cụ tiến gần hơn đến khả năng khai thác bán tự động thực sự.

Cuối cùng, việc xây dựng hệ thống báo cáo chuyên nghiệp là cần thiết. Các tính năng như xuất báo cáo theo chuẩn OWASP, kèm hình ảnh, sơ đồ kiến trúc, mức độ rủi ro, khuyến nghị khắc phục, cùng khả năng đồng bộ hóa với SIEM/SOAR sẽ giúp công cụ phù hợp hơn với môi trường thực tế doanh nghiệp. Song song đó, có thể xây dựng thêm chức năng quản lý người dùng, phân quyền, lưu lịch sử quét, và giao diện trực quan hơn nhằm nâng cao trải nghiệm và tính ứng dụng lâu dài.

### Kết luận chương

Qua ba kịch bản thực nghiệm, có thể khẳng định rằng hệ thống a vận hành ổn định, tích hợp hiệu quả các công cụ quét như Nmap, Nuclei và Nikto, đồng thời chuyển đổi dữ liệu thô thành dạng chuẩn hóa, thống nhất, thuận tiện cho phân tích và lưu trữ bằng chứng. Module AI enrich và AI recursive cho thấy khả năng đọc hiểu dữ liệu đầu vào, nhận dạng dịch vụ, dự đoán lỗ hổng tiềm năng, gợi ý CVE hợp lý và hỗ trợ phát hiện endpoint động trên ứng dụng web, giúp tăng tốc quá trình kiểm thử và nâng cao độ chính xác.

Bên cạnh những hiệu quả rõ ràng, thực nghiệm cũng cho thấy một số hạn chế: độ chính xác của AI phụ thuộc nhiều vào chất lượng dữ liệu đầu vào; một số gợi ý CVE hoặc payload có thể chưa phù hợp hoàn toàn với ngữ cảnh thực tế; việc khai thác vẫn cần sự can thiệp thủ công trong nhiều trường hợp để xác minh.

Hướng phát triển của hệ thống trong tương lai tập trung vào nâng cao khả năng phân tích và dự đoán của AI, cải thiện chất lượng dữ liệu đầu vào, bổ sung tự động hóa sinh payload kiểm thử, mở rộng phạm vi quét tới các môi trường phức tạp hơn và tăng cường khả năng tích hợp các công cụ kiểm thử mới. Tổng thể, chương 3 đã chứng minh tính khả thi, hiệu quả và tiềm năng phát triển của hệ thống Pentest-Auto trong việc hỗ trợ kiểm thử an toàn hạ tầng và ứng dụng web một cách tự động hóa và thông minh.

Kết luận

Đồ án này trình bày nghiên cứu và thực nghiệm về thiết kế, triển khai và đánh giá một công cụ kiểm thử an toàn thông minh (Pentest-Auto) theo 3 chương chính: lý thuyết nền tảng, thiết kế, kỹ thuật, và thực nghiệm-đánh giá. Cụ thể:

Chương 1 đã hệ thống hóa bối cảnh và động lực của đề tài: khái niệm kiểm thử an toàn, vai trò các công cụ truyền thống (Nmap, Nikto, Nuclei) và xu hướng tích hợp trí tuệ nhân tạo để tăng tốc phân tích và enrich kết quả. Mục tiêu nghiên cứu được xác định rõ: xây dựng một hệ thống tự động hóa quét và hỗ trợ phân tích bằng AI.

Chương 2 mô tả kiến trúc và hiện thực hệ thống: một orchestrator trung tâm phối hợp collectors, scanners, các mô-đun AI và giao diện web; định dạng lưu trữ JSONL và SQLite cho metadata; các module (thu thập subdomain, quét Nmap, quét web, enrich AI, phân tích endpoint, giao diện) đã được hiện thực và mô tả chi tiết. Các quyết định thiết kế (JSONL, quản lý API key bằng file với cơ chế quay vòng, prompt bằng tiếng Việt, hạn chế payload non-destructive) tạo nền tảng rõ ràng cho việc vận hành và mở rộng.

Chương 3 chứng minh tính khả thi qua thực nghiệm: triển khai môi trường lab, thực hiện kịch bản quét IP (SMB có CVE) , kịch bản thu thập subdomain, kịch bản quét lỗ hổng web (endpoint detection + vulnerability suggestion. Hệ thống hoàn thành các chức năng cốt lõi: phát hiện cổng/dịch vụ, chuẩn hóa dịch vụ/phiên bản, gợi ý CVE (kèm msf\_module trong một số trường hợp), phát hiện endpoint và gợi ý lỗ hổng thông qua OpenAI cùng các bản phân tích của OpenAI với kết quả phân tích các công cụ quét Nuclei, Nikto.

Hệ thống còn tồn tại những hạn chế cần lưu ý. Độ tin cậy của kết quả AI phụ thuộc mạnh vào chất lượng banner/service (presence of version/CPE); khi thông tin đầu vào thiếu hoặc mơ hồ, kết quả AI có thể là gợi ý rộng hoặc sai lệch. Mô hình ngôn ngữ có khả năng tạo ra thông tin không chính xác (hallucination), do đó mọi gợi ý khai thác hoặc CVE cần bước xác minh bổ sung. Hệ thống cũng phụ thuộc vào API key và quota của nhà cung cấp AI; mặc dù có cơ chế quay vòng và loại key lỗi, vận hành dài hạn đòi hỏi quản lý key và chiến lược caching để giới hạn chi phí.

Về hiệu năng và vận hành, pipeline hiện thực thi tuần tự cho nhiều bước tốn thời gian (Nmap, web scanners, gọi AI), khiến thời gian xử lý tăng khi mở rộng số mục tiêu. Kiến trúc hiện tại thiếu cơ chế hàng đợi/worker để xử lý song song, và giao diện chưa có cơ chế xác thực/ủy quyền mạnh mẽ. Về an toàn, công cụ gửi payload thử nghiệm (được giới hạn ở dạng non‑destructive trong cấu hình hiện tại) nên chỉ vận hành trong môi trường lab được phép; chưa có sandbox riêng cho payload mang rủi ro cao, vì vậy cần chính sách vận hành nghiêm ngặt.

Để khắc phục những hạn chế và nâng cao tính trưởng thành của công cụ, cần các bước cải tiến cụ thể. Trước hết nên bổ sung bước xác thực tự động bằng tra cứu NVD/OSV để kiểm tra độ chính xác các id/chi tiết do AI gợi ý, và cải tiến tiền xử lý banner (CPE matching, fingerprinting) để cung cấp input chuẩn hơn cho mô hình. Về kiến trúc, triển khai hệ thống hàng đợi với worker (Celery/RQ) và caching cho kết quả AI giúp tăng thông lượng và giảm chi phí. Về bảo mật vận hành, việc chuyển lưu trữ API key sang secret manager, bổ sung logging/audit và cơ chế phân quyền cho giao diện người dùng là bắt buộc trước khi dùng công cụ trong môi trường tổ chức.

Tóm lại, đồ án đã đạt được mục tiêu kỹ thuật cơ bản: xây dựng được một nguyên mẫu tích hợp quét truyền thống và khả năng phân tích hỗ trợ bởi AI, kèm theo giao diện để vận hành và thu thập kết quả. Để chuyển từ nguyên mẫu nghiên cứu sang công cụ sẵn sàng sản xuất cần ưu tiên bổ sung cơ chế xác thực kết quả AI, mở rộng kiến trúc để xử lý song song, và tăng cường biện pháp an toàn khi thử nghiệm. Những hướng phát triển này tạo lộ trình thực tế và có thể thực hiện được để nâng cao độ tin cậy, hiệu năng và an toàn cho hệ thống trong các ứng dụng thực tế.

Tài liệu tham khảo

[1] Verizon Data Breach Investigations Report (DBIR) 2024. Link: https://www.verizon.com/business/resources/reports/2024-dbir-data-breach-investigations-report.pdf?utm\_source=chatgpt.com

[2] ENISA Threat Landscape 2024. Link: https://www.enisa.europa.eu/sites/ default/files/2024-11/ENISA%20Threat% 20Landscape%202024\_0.pdf

[3] Check Point Q1 2025 / 2025 Cyber Security Report (thống kê tăng cường tấn công, credential theft). Link: https://blog.checkpoint.com/research/q1-2025-global-cyber-attack-report-from-check-point-software-an-almost-50-surge-in-cyber-threat s -worldwide-with-a-rise-of-126-in-ransomware-attacks/?utm\_source=chatgpt.com

[4] Fortinet / các phân tích ngành về tự động hóa và AI trong tấn công (tóm tắt báo

chí / phân tích). Link: https://www.techradar.com/pro/security/ai-powering-a-dramatic-surge-in-cyberthreats-as-automated-scans-hit-36-000-perecond?utm\_sour ce=chatgpt.com

[5] OWASP Top Ten (để làm nền cho phần web application risks). Link: https://owasp.org/www-project-top-ten/?utm\_source=chatgpt.com

[6] Lockheed Martin. *Intelligence-Driven Computer Network Defense: Cyber Kill*

*Chain* (2011). Link: https://www.lockheedmartin.com/en-us/capabilities/cyber/ cyber-kill-chain.html

[7] World Economic Forum. *Global Cybersecurity Outlook 2025*. Geneva: World

Economic Forum, 2025. [Online].Link: https://reports.weforum.org/docs/WEF\_ Global\_Cybersecurity\_Outlook\_2025.pdf?utm\_source=chatgpt.com

[8] ENISA. *Good Practices for Security Testing* (2022). Link: <https://www.enisa.europa.eu/publications>

[9] ISO/IEC 27001:2013. *Information Security Management.* Link: <https://www.iso.org/standard/54534.html>

[10] PCI DSS. *Payment Card Industry Data Security Standard*. Link: https://www. pcisecuritystandards.org/pci\_security/

[11] ENISA Threat Landscape Reports. Link: https://www.enisa.europa.eu /topics/csirt-cert-services

[12] Lockheed Martin, *Cyber Kill Chain Framework*. Link: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[13] Bài hướng dẫn OSINT trên SANS Institute. Link: https://www.sans.org/cyber-security-courses/osint-gathering/

[14] Bài báo *Application of AI in Penetration Testing* (2025), arXiv. Link: https://arxiv.org/abs/2501.00001

[15] Gordon “Fyodor” Lyon. *Nmap Network Scanning* — Official Nmapdocumentation / book. Link: <https://nmap.org/book/>

[16] Sullo (Nikto). *Nikto web server scanner documentation & GitHub*. Link:[https://cirt.net/nikto/](https://cirt.net/nikto/?utm_source=chatgpt.com)

[17] ProjectDiscovery. *Nuclei documentation & GitHub (templates, overview)*. Link: <https://docs.projectdiscovery.io/opensource/nuclei/overview>

[18] crt.sh — Certificate Transparency search. Link: <https://crt.sh/>

[19] Rapid7. *Metasploit Framework Documentation / Overview*. Link: <https://docs.rapid7.com/metasploit/>

[20] PortSwigger. *Burp Suite Documentation (features, proxy, scanner)*. Link: <https://portswigger.net/burp/documentation>

[21] Tổng quan AI trong an ninh mạng — “A Comprehensive Review of AI’s Current Impact and Future Prospects in Cybersecurity”. Link: https://www.researchgate.net/publication/387915732\_A\_Comprehensive\_Review\_of\_AI%27s\_Current\_Impact\_and\_Future\_Prospects\_in\_Cybersecurity?utm\_source=chatgpt.com

[22] Deng, G. et al., “An LLM-empowered Automatic Penetration Testing Tool (PentestGPT)”, arXiv, 2023. Link: [https://arxiv.org/abs/2308.06782?utm\_source= chatgpt.com](https://arxiv.org/abs/2308.06782?utm_%20source=chatgpt.com)

[23] Kujanpää, K. “Automating Privilege Escalation with Deep Reinforcement”, arXiv/Conference paper (mô tả agent RL cho privilege escalation). <https://arxiv.org/pdf/2110.01362>

[24] Veracode/Tech industry reporting: nghiên cứu về lỗ hổng trong code do AI sinh ra (báo cáo ngành). Link: [https://www.techradar.com/pro/nearly-half-of-all-code-generated-by-ai-found-to-contain-security-flaws-even-big-llms-affected?utm \_source=chatgpt.com](https://www.techradar.com/pro/nearly-half-of-all-code-generated-by-ai-found-to-contain-security-flaws-even-big-llms-affected?utm_source=chatgpt.com)

[25] Fang, C., et al. “Large Language Models for Code Analysis” (USENIX/2024). (LLM trong phân tích mã nguồn và review bảo mật). Link: [https://www.usenix.org/system/files/usenixsecurity24-fang.pdf?utm\_source=chat gpt.com](https://www.usenix.org/system/files/usenixsecurity24-fang.pdf?utm_source=chat%20gpt.com)

Phụ lục

1. Link toàn bộ mã nguồn công cụ Pentest-Auto

Link: https://github.com/Huy-Ciao-hacker/Pentest-Auto

1. Kết quả quét bằng Nmap ở kịch bản 1 dưới dạng file json:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | {  "ip": "10.30.10.143",  "hostname": "",  "domain": "",  "nmap\_success": true,  "nmap\_stderr": "",  "parsed": {  "services": [  {  "port": "135",  "proto": "tcp",  "state": "open",  "service": "msrpc",  "product": "Microsoft Windows RPC",  "version": null,  "extrainfo": null,  "cpe": ["cpe:/o:microsoft:windows"]  },  {  "port": "139",  "proto": "tcp",  "state": "open",  "service": "netbios-ssn",  "product": "Microsoft Windows netbios-ssn",  "version": null,  "extrainfo": null,  "cpe": ["cpe:/o:microsoft:windows"]  },  {  "port": "445",  "proto": "tcp",  "state": "open",  "service": "microsoft-ds",  "product": "Windows 7 Professional 7601 Service Pack 1 microsoft-ds",  "version": null,  "extrainfo": "workgroup: WORKGROUP",  "cpe": ["cpe:/o:microsoft:windows"]  },  {  "port": "49152",  "proto": "tcp",  "state": "open",  "service": "msrpc",  "product": "Microsoft Windows RPC",  "version": null,  "extrainfo": null,  "cpe": ["cpe:/o:microsoft:windows"]  }  ]  },  "ts": "2025-11-14T10:29:40.993864-05:00"  } |

1. Kết quả AI phân tích từ file nmap\_results.json ở kịch bản 1 dưới dạng file json:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | {  "ip": "10.30.10.143:139",  "product": "Microsoft Windows netbios-ssn",  "id": "CVE-2020-0601",  "cve\_url": "https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-0601",  "exploit\_url": "Không có",  "msf\_module": "Không có",  "complexity": "thấp",  "short\_description": "Lỗ hổng này cho phép kẻ tấn công thực hiện tấn công giả mạo  chứng chỉ."  },  {  "ip": "10.30.10.143:139",  "product": "Microsoft Windows netbios-ssn",  "id": "CVE-2019-0708",  "cve\_url": "https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-0708",  "exploit\_url": "Không có",  "msf\_module": "Không có",  "complexity": "cao",  "short\_description": "Lỗ hổng này cho phép kẻ tấn công từ xa thực thi mã tùy ý trên hệ  thống bị ảnh hưởng."  },  {  "ip": "10.30.10.143:445",  "product": "Windows 7 Professional 7601 Service Pack 1 microsoft-ds",  "id": "CVE-2020-0601",  "cve\_url": "https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-0601",  "exploit\_url": "Không có",  "msf\_module": "exploit/windows/smb/ms17\_010\_eternalblue",  "complexity": "thấp",  "short\_description": "Lỗ hổng này cho phép kẻ tấn công thực hiện các cuộc tấn công  man-in-the-middle để giả mạo chứng chỉ."  },  {  "ip": "10.30.10.143:445",  "product": "Windows 7 Professional 7601 Service Pack 1 microsoft-ds",  "id": "CVE-2019-0708",  "cve\_url": "https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-0708",  "exploit\_url": "Không có",  "msf\_module": "exploit/windows/rdp/cve\_2019\_0708\_rdp",  "complexity": "thấp",  "short\_description": "Lỗ hổng này cho phép kẻ tấn công từ xa thực thi mã tùy ý trên hệ  thống bị ảnh hưởng."  } |

1. Kết quả quét các miền phụ cho miền example.com ở kịch bản 2 dưới dạng file json:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | {  "domain": "example.com",  "subdomains": [  {"ip": "125.235.4.59", "subdomain": "dev.example.com"},  {"ip": "125.235.4.59", "subdomain": "m.example.com"},  {"ip": "125.235.4.59", "subdomain": "products.example.com"},  {"ip": "125.235.4.59", "subdomain": "support.example.com"},  {"ip": "23.192.228.80", "subdomain": "example.com"},  {"ip": "23.192.228.84", "subdomain": "example.com"},  {"ip": "23.215.0.136", "subdomain": "example.com"},  {"ip": "23.215.0.138", "subdomain": "example.com"},  {"ip": "23.220.75.232", "subdomain": "example.com"},  {"ip": "23.220.75.245", "subdomain": "example.com"},  {"ip": "27.77.81.74", "subdomain": "www.example.com"},  {"ip": "27.77.81.82", "subdomain": "www.example.com"}  ]} |

1. Kết quả phân tích AI của Nikto cho miền example.com ở kịch bản 2 dưới dạng file json:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | {  "primary\_target": "http://example.com",  "findings": [  {  "path": "/",  "title": "Thiếu header X-Frame-Options",  "severity": "Medium",  "confidence": "High",  "description": "Thiếu Header X-Frame-Options, có thể tấn công clickjacking.",  "evidence": "Không có header X-Frame-Options.",  "impact": "Người dùng có thể bị tấn công clickjacking.",  "exploitability": "Có thể khai thác dễ dàng.",  "remediation": "Thêm header X-Frame-Options vào phản hồi HTTP.",  "references": "https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-  Frame-Options"  },  {  "path": "/",  "title": "Thiếu header X-Content-Type-Options",  "severity": "Medium",  "confidence": "High",  "description": "Header X-Content-Type-Options không được thiết lập, có thể cho  phép trình duyệt hiển thị nội dung theo cách khác với loại MIME.",  "evidence": "Không có header X-Content-Type-Options.",  "impact": "Có thể dẫn đến tấn công XSS.",  "exploitability": "Có thể khai thác dễ dàng.",  "remediation": "Thêm header X-Content-Type-Options vào phản hồi HTTP.",  "references": "https://www.netsparker.com/web-vulnerability-  scanner/vulnerabilities/missing-content-type-header/"  },  {  "path": "/",  "title": "Thay đổi banner server",  "severity": "Low",  "confidence": "Medium",  "description": "Banner server thay đổi từ AkamaiNetStorage sang AkamaiGHost.",  "evidence": "Banner server: AkamaiGHost.",  "impact": "Có thể tiết lộ thông tin về cấu hình server.",  "exploitability": "Không có khả năng khai thác trực tiếp.",  "remediation": "Giữ bí mật thông tin về server.",  "references": "Không có"  },  {  "path": "/",  "title": "Khả năng đăng nhập với thông tin mặc định",  "severity": "High",  "confidence": "High",  "description": "Có thể thử đăng nhập với thông tin mặc định 'admin' hoặc 'monitor'.",  "evidence": "Khuyến nghị thử đăng nhập với 'admin' hoặc 'monitor'.",  "impact": "Có thể dẫn đến truy cập trái phép vào hệ thống.",  "exploitability": "Có thể khai thác dễ dàng.",  "remediation": "Thay đổi thông tin đăng nhập mặc định và sử dụng mật khẩu mạnh.",  "references": "Không có"  }  ]  } |

1. Kết quả phân tích AI của Nuclei cho miền example.com ở kịch bản 2 dưới dạng file json:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | {  "primary\_target": "http://example.com",  "findings": [  {  "template": "azure-domain-tenant",  "severity": "info",  "confidence": "medium",  "evidence": "Phát hiện liên kết đến cấu hình OpenID của Microsoft cho tên miền  example.com.",  "description": "Tên miền này có thể liên kết với dịch vụ Azure, có thể dẫn đến rủi ro  bảo mật nếu không được cấu hình đúng.",  "remediation": "Kiểm tra và cấu hình lại các thiết lập bảo mật cho dịch vụ Azure liên  quan đến tên miền này."  },  {  "template": "external-service-interaction",  "severity": "info",  "confidence": "medium",  "evidence": "Có nhiều tương tác với dịch vụ bên ngoài từ http://example.com.",  "description": "Tương tác với dịch vụ bên ngoài có thể tạo ra rủi ro bảo mật nếu  không được kiểm soát.",  "remediation": "Đánh giá và giám sát các tương tác với dịch vụ bên ngoài để đảm bảo  an toàn."  },  {  "template": "waf-detect:akamai",  "severity": "info",  "confidence": "medium",  "evidence": "Phát hiện WAF của Akamai đang bảo vệ http://example.com.",  "description": "Sử dụng WAF có thể giúp bảo vệ ứng dụng web khỏi các cuộc tấn  công, nhưng cần đảm bảo cấu hình đúng.",  "remediation": "Kiểm tra cấu hình WAF để đảm bảo nó đang hoạt động hiệu quả."  },  {  "template": "request-based-interaction:dns",  "severity": "info",  "confidence": "medium",  "evidence": "Có tương tác dựa trên yêu cầu DNS từ http://example.com.",  "description": "Tương tác DNS có thể tiết lộ thông tin nhạy cảm nếu không được bảo  vệ.",  "remediation": "Đảm bảo rằng các yêu cầu DNS được bảo mật và không tiết lộ thông  tin nhạy cảm."  }  ]  } |