HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



Nguyễn Thị Thu Hà

NGHIÊN CỨU VÀ ỨNG DỤNG MÔ HÌNH RAG TRONG PHÂN TÍCH CẢNH BÁO AN NINH MẠNG

CHUYÊN NGÀNH : HỆ THỐNG THÔNG TIN

MÃ SỐ: 8.48.01.04

TÓM TẮT ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

HÀ NỘI - 2025

Đề án tốt nghiệp được hoàn thành tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Người hướng dẫn khoa học: TS. ĐINH TRƯỜNG DUY

Phản biện 1: ……

Phản biện 2: ……..

Đề án tốt nghiệp sẽ được bảo vệ trước Hội đồng chấm đề án tốt nghiệp thạc sĩ tại Học viện Công nghệ Bưu chính Viễn thông

Vào lúc: ..... giờ ..... ngày ..... tháng .... năm .....

Có thể tìm hiểu đề án tốt nghiệp tại:

- Thư viện của Học viện Công nghệ Bưu chính Viễn thông.

MỞ ĐẦU

Tính cấp thiết của đề tài

Trong bối cảnh vận hành các Trung tâm Điều hành An ninh mạng (SOC) hiện đại, một thách thức hiện hữu là sự bùng nổ về số lượng cảnh báo (alert fatigue). Các chuyên gia phân tích thường xuyên đối mặt với áp lực xử lý hàng nghìn log mỗi ngày, trong khi các thông tin ngữ cảnh cần thiết để ra quyết định lại nằm rải rác và thiếu tính liên kết. Hơn nữa, các phương pháp phát hiện dựa trên luật (rule-based) truyền thống đang bộc lộ điểm yếu trước các kỹ thuật tấn công tinh vi, ẩn danh hoặc chưa từng biết đến.

Sự xuất hiện và phát triển trí tuệ nhân tạo tạo sinh (Generative AI), đặc biệt là sự kết hợp giữa mô hình ngôn ngữ lớn (LLM) cùng với mô hình tạo sinh dựa trên truy xuất tăng cường (Retrieval-Augmented Generation - RAG) đã mở ra một cách tiếp cận mới trong việc xử lý log, phân tích ngữ cảnh và tự động hoá phân loại rủi ro. Việc tích hợp LLM vào RAG không chỉ giúp phân tích cảnh báo với ngữ cảnh cụ thể mà còn có thể truy xuất thông tin từ nhiều nguồn (Threat Intelligence, IOC feeds, MITRE ATT&CK…) để đưa ra đánh giá và giải thích đánh giá.

Xuất phát từ thực tiễn đó, đề tài này tập trung nghiên cứu giải pháp tích hợp các công cụ mã nguồn mở nhằm tự động hóa quy trình phân tích, đánh giá cảnh báo, giúp nâng cao tốc độ và độ chính xác trong phản ứng sự cố an ninh mạng.

Các kết quả đạt được

Các kết quả dự kiến bao gồm: Nắm được khái quát về hệ thống giám sát an ninh mạng và mô hình tạo sinh dựa trên truy xuất tăng cường (RAG); Áp dụng hệ thống RAG vào phân tích cảnh báo an ninh mạng; Công cụ thử nghiệm cho phép nhập cảnh báo và nhận phản hồi phân tích từ mô hình.

Mục tiêu nghiên cứu của đề tài

Mục tiêu chính của đề tài là nghiên cứu hệ thống giám sát an ninh mạng và mô hình tạo sinh dựa trên truy xuất tăng cường (RAG). Ứng dụng mô hình tạo sinh dựa trên truy xuất tăng cường (RAG) vào phân tích cảnh báo an ninh mạng.

Cấu trúc đề án

Đề án gồm 3 chương:

Chương 1 tổng quan về hệ thống giám sát an ninh mạng và mô hình RAG;

Chương 2 nghiên cứu và ứng dụng mô hình RAG trong phân tích cảnh báo an ninh mạng;

Chương 3 thử nghiệm và đánh giá mô hình.

CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG GIÁM SÁT AN NINH MẠNG VÀ MÔ HÌNH RAG

1.1. Tổng quan về SOC và hệ thống cảnh báo an ninh mạng

Trung tâm Điều hành An ninh mạng (SOC) đóng vai trò như trung tâm thần kinh của hệ thống bảo mật doanh nghiệp. Khác với các phòng IT truyền thống, đơn vị này được tối ưu hóa riêng cho việc phát hiện, phân tích và phản ứng với các mối đe dọa an ninh. Theo định nghĩa từ Viện SANS, hiệu quả của SOC phụ thuộc vào sự cân bằng giữa ba trụ cột chiến lược: Con người, Quy trình và Công nghệ.

Chức năng của SOC bao gồm chu trình khép kín: giám sát liên tục toàn bộ hạ tầng IT; phát hiện mối đe dọa kết hợp giữa luật cảnh báo và thuật toán học máy; phân tích và điều tra để phân loại cảnh báo; phản ứng sự cố với các biện pháp ngăn chặn và cô lập; và phục hồi hệ thống cùng đánh giá sau sự cố.

Hệ thống cảnh báo an ninh mạng hoạt động theo quy trình nhiều tầng: Thu thập log từ các nguồn; Chuẩn hóa về định dạng chung; Tương quan các sự kiện thành chuỗi hành vi; và Cảnh báo khi thỏa điều kiện luật. Thách thức lớn nhất hiện nay là "Quá tải cảnh báo" (Alert Fatigue) do số lượng dương tính giả cao, dẫn đến nguy cơ bỏ sót các mối đe dọa thực sự.

1.2. Khảo sát các phương pháp phân tích cảnh báo

Phương pháp phân tích đã trải qua nhiều giai đoạn tiến hóa. Phương pháp thủ công dựa hoàn toàn vào con người, có ưu điểm là tận dụng được trực giác và kinh nghiệm nhưng hạn chế về tốc độ và khả năng mở rộng. Phương pháp dựa trên thống kê và học máy có thể phát hiện mối đe dọa chưa biết nhưng gặp vấn đề về tỷ lệ dương tính giả cao và thiếu khả năng giải thích. Phương pháp dựa trên luật có độ chính xác cao với các mối đe dọa đã biết nhưng bất lực trước các cuộc tấn công mới.

Xu hướng hiện đại là kết hợp các phương pháp và sử dụng công nghệ AI tạo sinh, đặc biệt là mô hình ngôn ngữ lớn được tăng cường bởi cơ chế truy xuất thông tin, mở ra chiều hướng mới có khả năng tự động hóa, giải thích và đề xuất hành động cụ thể.

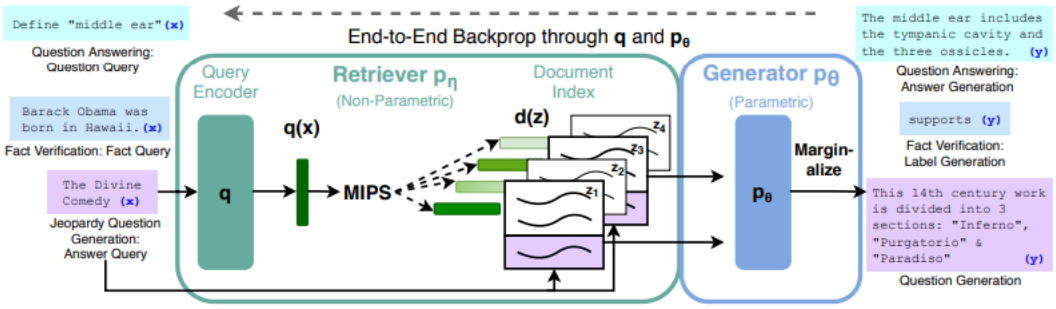
1.3. Kiến trúc và đặc trưng của mô hình LLM

Mô hình ngôn ngữ lớn là mô hình học sâu được huấn luyện trên tập dữ liệu khổng lồ, hoạt động như mô hình xác suất dự đoán token tiếp theo dựa trên ngữ cảnh. Khác với mô hình NLP truyền thống, LLM thuộc nhóm "Mô hình nền tảng" có thể fine-tune cho nhiều tác vụ khác nhau.

Hầu hết LLM được xây dựng trên kiến trúc Transformer với cơ chế Self-Attention cho phép mô hình đánh trọng số tầm quan trọng của các từ trong câu. Các đặc trưng quan trọng bao gồm: Khả năng học theo ngữ cảnh (zero-shot và few-shot learning); Cửa sổ ngữ cảnh định nghĩa lượng token có thể xử lý; Khả năng suy luận chuỗi chia nhỏ vấn đề phức tạp thành các bước logic; và Hiện tượng ảo giác - rủi ro khi mô hình tự tin đưa ra thông tin không chính xác.

1.4. Kiến trúc và nguyên lý hoạt động của mô hình RAG

RAG là kiến trúc lai cho phép LLM truy xuất và kết hợp thông tin mới, kết hợp sức mạnh tạo sinh ngôn ngữ với khả năng truy xuất thông tin chính xác từ nguồn bên ngoài. Hệ thống RAG bao gồm ba thành phần: Bộ truy xuất (Retriever) tìm kiếm thông tin liên quan từ cơ sở dữ liệu; Bộ sinh (Generator) là LLM tạo câu trả lời; và Cơ sở dữ liệu kiến thức chứa thông tin chuyên ngành.



Hình 1: Sơ đồ kiến trúc RAG

Quy trình hoạt động gồm hai giai đoạn: Giai đoạn nạp dữ liệu (Thu thập, Chia nhỏ, Mã hóa Vector, Lưu trữ); và Giai đoạn truy vấn (Nhận câu hỏi, Mã hóa, Tìm kiếm tương đồng, Tăng cường ngữ cảnh, Tạo câu trả lời). Cơ chế này giúp giải quyết bài toán cập nhật kiến thức cho LLM mà không cần huấn luyện lại.

1.5. Ứng dụng mô hình RAG trong SOC

RAG giải quyết hai thách thức then chốt: quá tải cảnh báo và thiếu hụt nhân lực. Theo Ponemon Institute (2022), SOC trung bình xử lý hơn 4.484 cảnh báo mỗi ngày nhưng chỉ điều tra được 46%. Ứng dụng chính bao gồm: Tự động phân tích và phân loại cảnh báo, loại bỏ 30-50% dương tính giả; Hỗ trợ điều tra tổng hợp thông tin từ nhiều nguồn, rút ngắn thời gian 25-40%; và Đề xuất quy trình xử lý cụ thể từ tài liệu nội bộ.

1.6. Khảo sát các giải pháp và nghiên cứu liên quan

Nghiên cứu của Deng và cộng sự (2024) cho thấy RAG cải thiện độ chính xác 23% nhưng chưa đi sâu tích hợp nhiều nguồn dữ liệu. Rapid7 đạt độ chính xác 99,93% và

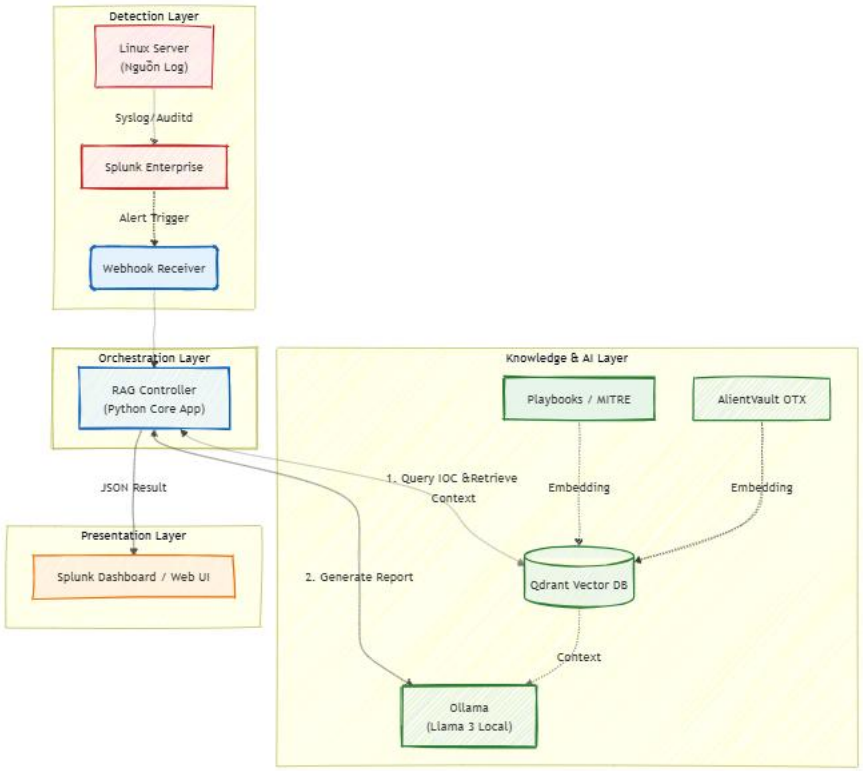
xử lý 4,8 nghìn tỷ cảnh báo/tuần. IBM (2025) chỉ ra AI giảm chi phí vi phạm 2 triệu USD và rút ngắn phản ứng 80 ngày. Self-RAG giảm ảo giác nhưng yêu cầu tài nguyên lớn. NVIDIA Morpheus mạnh nhưng đòi hỏi phần cứng đắt. Các dự án mã nguồn mở hiện còn thiếu khả năng tích hợp thực tế.

Khoảng trống: Các nghiên cứu thiếu chi tiết triển khai thực tế, giải pháp thương mại chi phí cao và khó tùy biến, dự án mã nguồn mở chưa trưởng thành. Đề tài lấp đầy bằng kiến trúc hoàn chỉnh, tích hợp SIEM, sử dụng công nghệ mã nguồn mở, vận hành trên hạ tầng hạn chế với LLM cục bộ.

CHƯƠNG 2: NGHIÊN CỨU VÀ ỨNG DỤNG MÔ HÌNH RAG

2.1. Thiết kế kiến trúc mô hình RAG

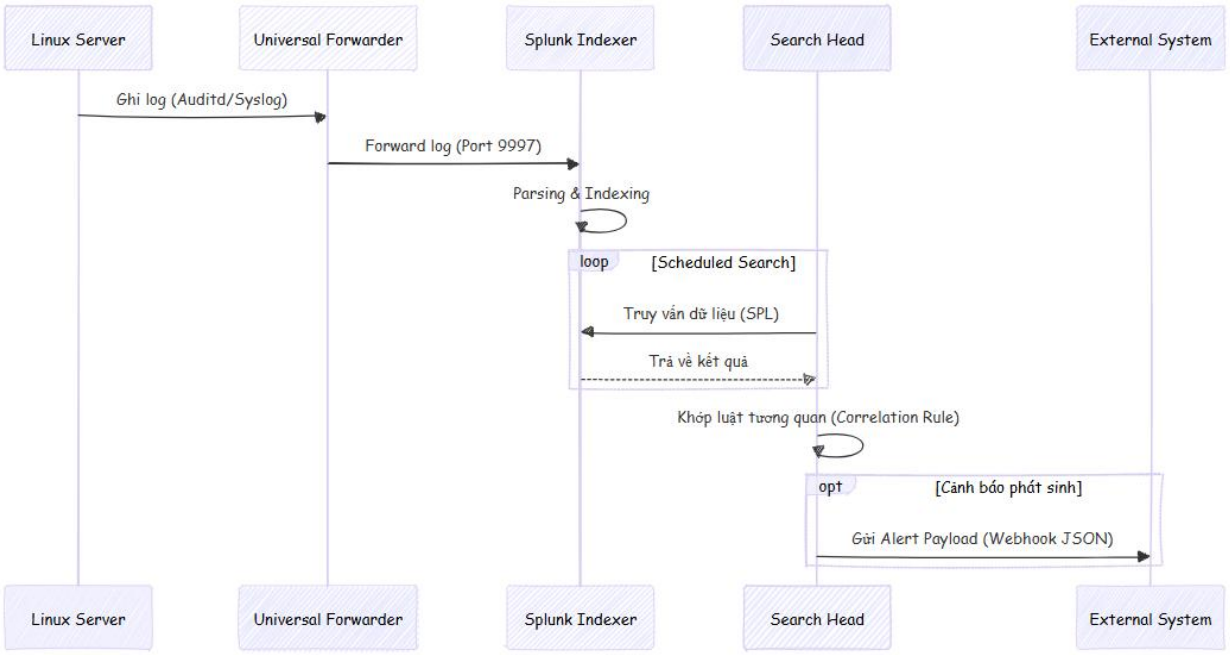
Yêu cầu kiến trúc bao gồm: Tính bảo mật dữ liệu ưu tiên LLM cục bộ; Tính thời gian thực độ trễ ~60 giây/cảnh báo; Tính chính xác dựa trên tài liệu thực tế hạn chế ảo giác; và Khả năng mở rộng thiết kế module hóa.



Hình 2: Sơ đồ tổng thể kiến trúc hệ thống RAG

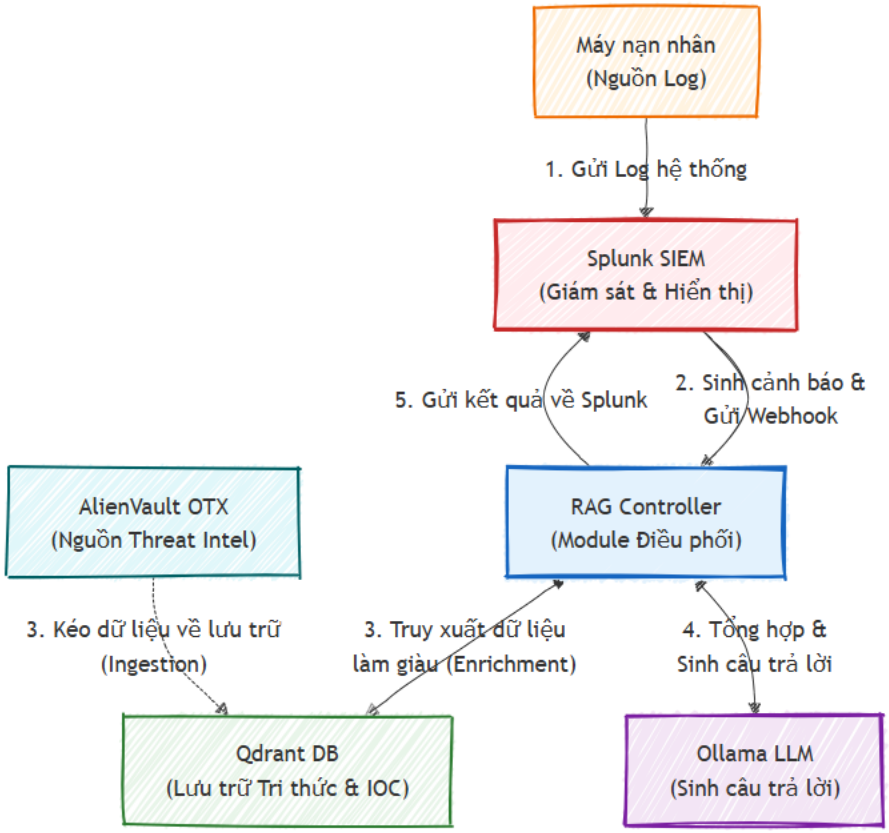
Kiến trúc gồm 4 tầng:

- Tầng 1 - Giám sát với Linux Server và Splunk Enterprise thu thập log, phát hiện tấn công, gửi webhook.
- Tầng 2 – Điều phối với RAG Controller (Python/FastAPI) tiếp nhận, parsing, xử lý song song bất đồng bộ.
- Tầng 3 - Dữ liệu & AI với Qdrant lưu trữ vector từ tài liệu nội bộ và threat intelligence; LLM (Ollama/Llama 3) tổng hợp và sinh báo cáo; AlienVault OTX cung cấp IOC.
- Tầng 4 - Tương tác với Splunk Dashboard hiển thị kết quả qua HTTP Event Collector.



Hình 3: Mô tả hoạt động của Splunk

2.2. Mô tả luồng dữ liệu và chức năng



Hình 4: Sơ đồ mô tả luồng dữ liệu

Luồng xử lý 5 bước:

(1) Nhận cảnh báo từ SIEM qua webhook JSON;

(2) Parsing và chuẩn hóa trích xuất IOC;

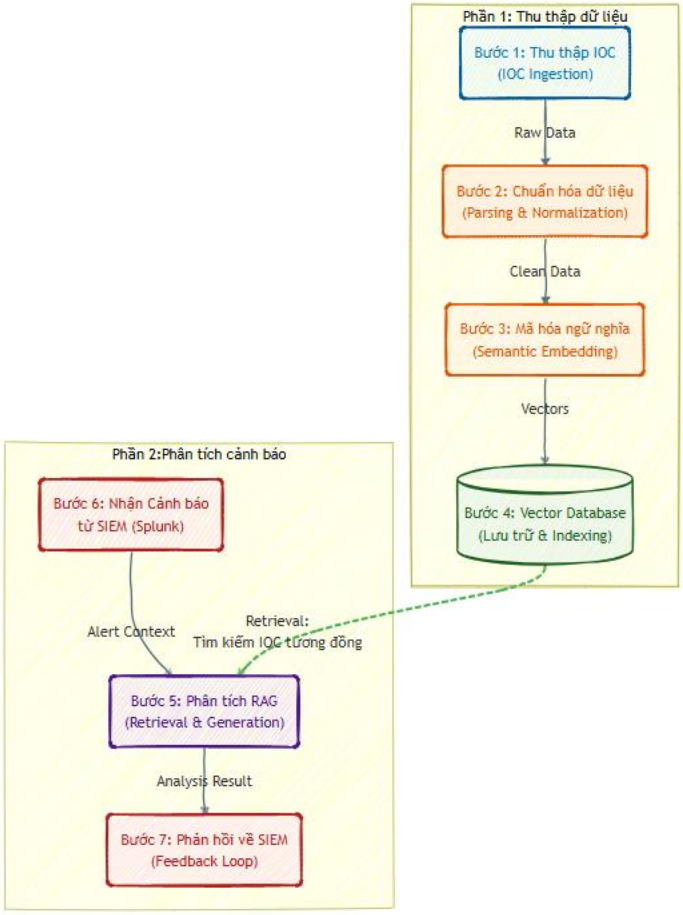
(3) Làm giàu dữ liệu từ OTX về reputation;

(4) Xử lý RAG: embedding, similarity search trong Qdrant, LLM tổng hợp;

(5) Trả kết quả báo cáo và khuyến nghị.

Các chức năng module: Thu thập và làm giàu IOC qua API OTX; Xử lý và nhúng dữ liệu bằng sentence-transformers; Lưu trữ và truy xuất trong Qdrant; Phân tích và sinh báo cáo qua LlamaIndex/LangChain với Ollama.

2.3. Triển khai trong môi trường thử nghiệm



Hình 5: Sơ đồ các bước triển khai mô hình

Môi trường gồm 3 máy chủ: Máy nạn nhân (Ubuntu Server với auditd, Splunk Forwarder); Máy SIEM (Splunk Enterprise 9.4.1); Máy RAG (Python/FastAPI, Ollama, Qdrant, RAM ≥16GB).

Các bước triển khai:

Bước 1: Thu thập dữ liệu chỉ báo tấn công (IOC Ingestion)

Giai đoạn đầu tiên tập trung vào việc thiết lập kết nối với các nguồn Threat Intelligence uy tín nhằm xây dựng cơ sở dữ liệu về các mối đe dọa hiện hữu.

- Mục đích: Tự động hóa việc cập nhật danh sách các địa chỉ IP, mã băm (Hash) tệp tin và tên miền độc hại từ nền tảng AlienVault OTX hoặc MISP.
- Phương pháp thực hiện: Sử dụng module Python để gọi API RESTful của nhà cung cấp dịch vụ. Dữ liệu thô trả về (định dạng JSON hoặc CSV) sẽ được lưu trữ tạm thời để phục vụ xử lý.

Bước 2: Chuẩn hóa và Phân tích cú pháp dữ liệu (Data Parsing & Normalization)

Dữ liệu thô sau khi thu thập cần được chuyển đổi về định dạng cấu trúc thống nhất để hệ thống có thể xử lý đồng bộ.

- Mục đích: Trích xuất các trường thông tin cốt lõi bao gồm: Loại chỉ báo (Type), Giá trị (Value), Mô tả (Description) và Thời điểm phát hiện (Timestamp).
- Phương pháp thực hiện: Sử dụng các thư viện xử lý chuỗi và định dạng dữ liệu (csv, json) để chuyển đổi dữ liệu thô thành danh sách các đối tượng (List of Dictionaries).

Bước 3: Mã hóa ngữ nghĩa (Semantic Embedding)

Đây là bước quan trọng để chuyển đổi dữ liệu văn bản sang dạng vector số học, cho phép thực hiện các phép toán tìm kiếm tương đồng về sau.

- Mục đích: Biểu diễn ngữ nghĩa của các chỉ báo tấn công và mô tả đi kèm dưới dạng vector đa chiều.
- Phương pháp thực hiện: Sử dụng các mô hình Transformer đã được huấn luyện trước (Pre-trained Models) như all-MiniLM-L6-v2. Mô hình này cân bằng tốt giữa tốc độ và độ chính xác.

Bước 4: Đánh chỉ mục vào Cơ sở dữ liệu Vector (Vector Indexing)

Các vector sau khi tạo ra cần được lưu trữ vào một cơ sở dữ liệu chuyên dụng để tối ưu hóa tốc độ truy vấn.

- Mục đích: Xây dựng kho dữ liệu số (Knowledge Base) có khả năng truy xuất thông tin trong thời gian thực (mili-giây).
- Phương pháp thực hiện: Sử dụng Qdrant làm Vector Database. Quy trình bao gồm việc khởi tạo bộ sưu tập (Collection), định nghĩa kích thước vector và thực hiện thao tác chèn (Upsert) dữ liệu.

Bước 5: Phân tích cảnh báo sử dụng RAG (RAG-based Analysis)

Đây là bước cốt lõi của hệ thống, nơi sức mạnh của LLM và cơ sở dữ liệu được kết hợp.

- Mục đích: Tự động phân tích nội dung cảnh báo từ SIEM, đối chiếu với cơ sở dữ liệu vector để tìm thông tin liên quan, và tổng hợp thành báo cáo hoàn chỉnh.
- Phương pháp thực hiện: Khi nhận được cảnh báo, hệ thống sẽ: (1) Vector hóa nội dung cảnh báo, (2) Truy vấn Qdrant để lấy ngữ cảnh (Context), (3) Gửi Prompt chứa ngữ cảnh tới Ollama để sinh câu trả lời

Bước 6: Tích hợp cơ chế cảnh báo từ SIEM (Splunk Alert Integration)

Thiết lập cơ chế tự động chuyển tiếp thông tin từ tầng giám sát sang tầng phân tích.

- Mục đích: Đảm bảo mọi cảnh báo mức độ cao (High Severity) đều được tự động gửi đi phân tích ngay lập tức.
- Phương pháp thực hiện: Cấu hình Webhook trên Splunk Enterprise.

Bước 7: Phản hồi kết quả về SIEM (Feedback Loop Integration)

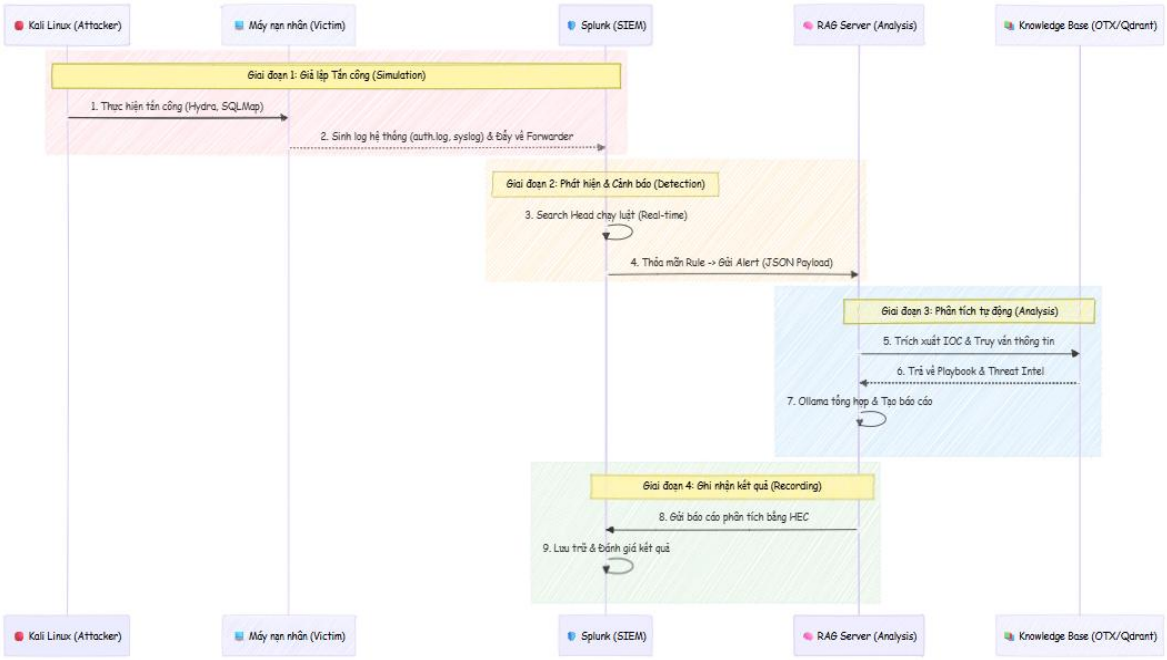
Khép kín quy trình bằng cách gửi ngược kết quả phân tích về lại giao diện giám sát trung tâm.

- Mục đích: Giúp chuyên gia SOC xem được kết quả phân tích của AI ngay trên Dashboard của Splunk mà không cần chuyển đổi công cụ.

  Phương pháp thực hiện: Sử dụng dịch vụ Splunk HTTP Event Collector (HEC).

CHƯƠNG 3: THỬ NGHIỆM VÀ ĐÁNH GIÁ MÔ HÌNH

3.1. Mô tả mô hình thử nghiệm



Hình 6: sơ đồ mô hình thử nghiệm

Môi trường trên VMware ESXi gồm: Kẻ tấn công (Kali Linux), Máy nạn nhân (Ubuntu Server 22.04), SIEM Server (Splunk Enterprise), RAG Analysis Server (Python/Ollama/Qdrant).

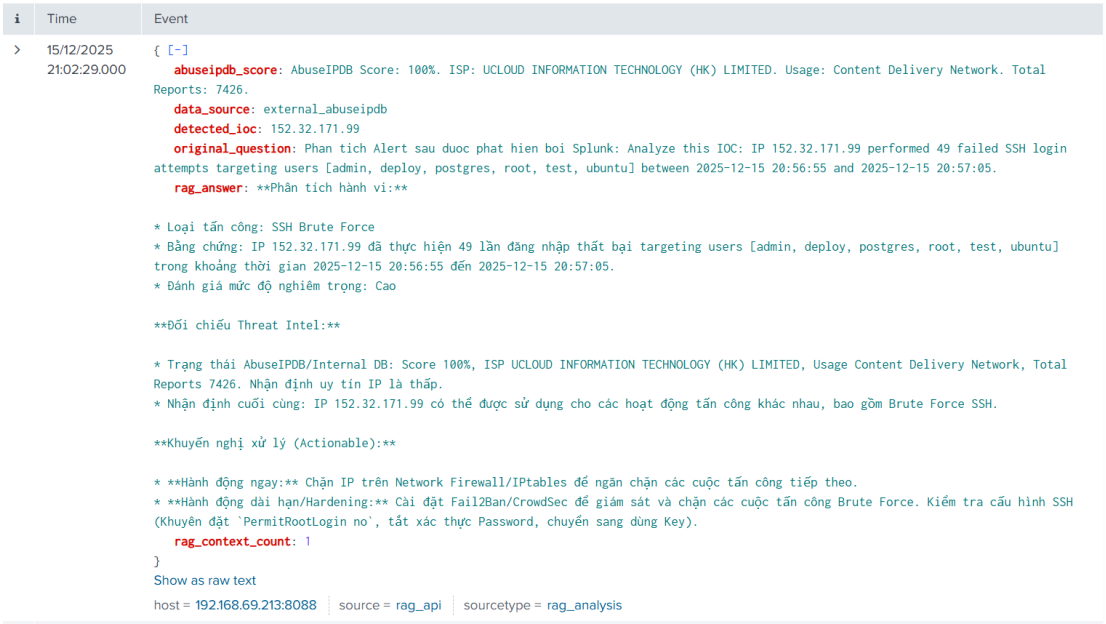
1. Giai đoạn 1: Giả lập tấn công (Simulation)
    - Sử dụng các công cụ trên Kali Linux (Hydra, SQLMap, Metasploit) để thực hiện hành vi tấn công vào máy nạn nhân.
    - Mục tiêu: Tạo ra các dấu hiệu (signatures) trong log hệ thống (auth.log, syslog, access.log).
2. Giai đoạn 2: Phát hiện và Cảnh báo (Detection)
    - Splunk Forwarder đẩy log về Indexer.
    - Splunk Search Head chạy các luật tìm kiếm thời gian thực (Real-time searches).
    - Khi điều kiện luật thỏa mãn (ví dụ: > 10 lần đăng nhập sai), Splunk kích hoạt Alert và gửi payload JSON sang RAG Server.

3. Giai đoạn 3: Phân tích tự động (Automated Analysis)
    - RAG Server nhận cảnh báo, trích xuất IOC (IP, User).
    - Thực hiện truy vấn làm giàu dữ liệu từ OTX và tìm kiếm Playbook từ Qdrant.
    - Ollama tổng hợp thông tin và sinh ra báo cáo phân tích.
4. Giai đoạn 4: Ghi nhận kết quả (Recording)
    - Kết quả phân tích được gửi ngược lại Splunk (qua HEC) và lưu trữ tại Splunk.
    - Đánh giá kết quả.

Các chỉ số đánh giá: Hiệu năng (MTTR theo NIST, tài nguyên CPU/RAM); Chất lượng theo RAGAS (Retrieval Precision, Faithfulness chống ảo giác, Actionability).

3.2. Kịch bản thử nghiệm

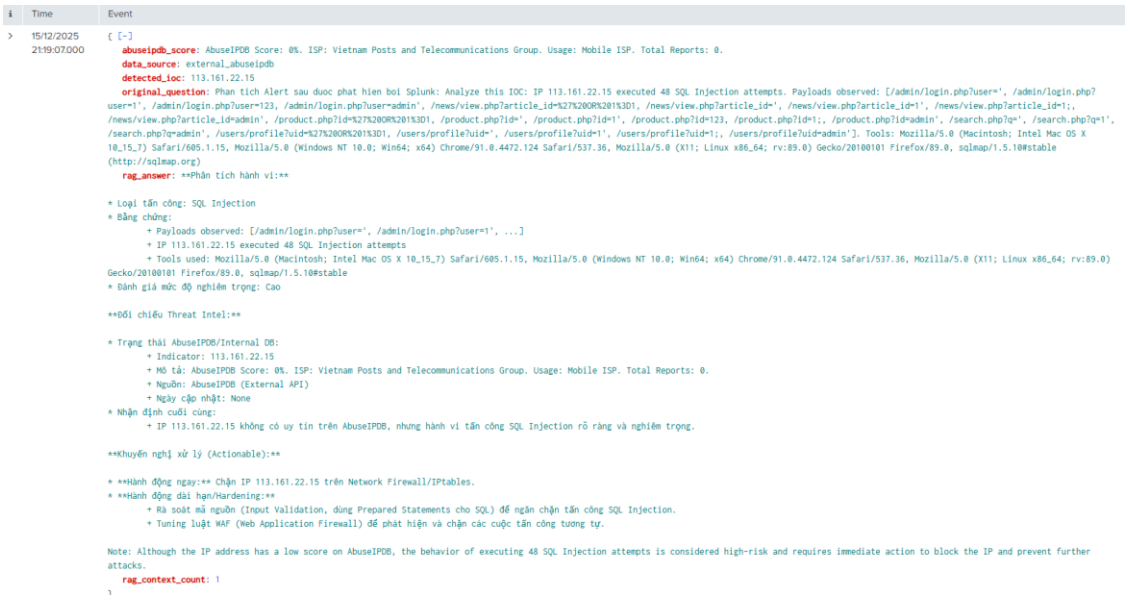
Kịch bản 1: SSH Brute Force - Sử dụng Hydra tấn công SSH với từ điển mật khẩu. Dấu vết: chuỗi "Failed password" liên tiếp trong auth.log. Splunk rule kích hoạt khi >10 lần thất bại/60s. RAG truy xuất IOC từ OTX, playbook từ Qdrant, đề xuất: chặn IP, tắt PermitRootLogin, MFA.



Hình 7: Kết quả phân tích cảnh báo Brute Force

Kịch bản 2: SQL Injection - SQLMap khai thác lỗ hổng UNION-based injection. Dấu vết: các ký tự SQL đặc biệt (UNION, SELECT, OR 1=1) trong access.log, User-Agent "sqlmap". Rule phát hiện pattern và high error rate. RAG nhận diện công cụ, giải

thích cơ chế UNION-based, đề xuất: chặn IP, rà soát mã nguồn, prepared statements, tune WAF.



Hình 8: Kết quả phân tích cảnh báo SQLi

Kịch bản 3: Privilege Escalation - Lạm dụng sudo misconfiguration (user backup có thể chạy /bin/bash với NOPASSWD). Dấu vết: audit log ghi "sudo /bin/bash", theo sau đọc /etc/shadow, tạo user mới. Rule phát hiện unusual sudo usage + suspicious post-escalation activities. RAG ánh xạ MITRE T1548.003, phân tích bối cảnh user, đề xuất: disable account, fix sudoers, reset passwords, forensic investigation.



Hình 9: Kết quả phân tích cảnh báo Privilege Escalation

3.3. Nhận xét, phân tích kết quả

Bảng 1:Giá trị hiệu năng khi chạy hệ thống RAG

Kịch bản	**Thời gian truy xuất (Qdrant)**	**Thời gian sinh (Ollama)**	**Tổng thời gian (MTTR)**	**CPU Usage (Max)**	**RAM Usage**
SSH Brute Force	26s	14.5s	40,5s	85%	14.2 GB
SQL Injection	23s	15.1s	38,1s	90%	15.5 GB
Privilege Escalation	27s	16.4s	43,4s	92%	14.7 GB

Hiệu năng: MTTR ~40s chấp nhận được (so với 10-15 phút thủ công). Ollama yêu cầu RAM tối thiểu 16GB.

Chất lượng: Khả năng ngữ cảnh tốt (tổng hợp alert + OTX score 0% → kết luận chính xác dựa hành vi không phải reputation). Nhận diện công cụ sqlmap qua User-Agent. Tính minh bạch với cấu trúc Chain-of-Thought logic. Actionability cao với khuyến nghị cụ thể (Input Validation, Tune WAF). Không ảo giác - dữ liệu IOC chính xác từ AbuseIPDB và từ log gốc.

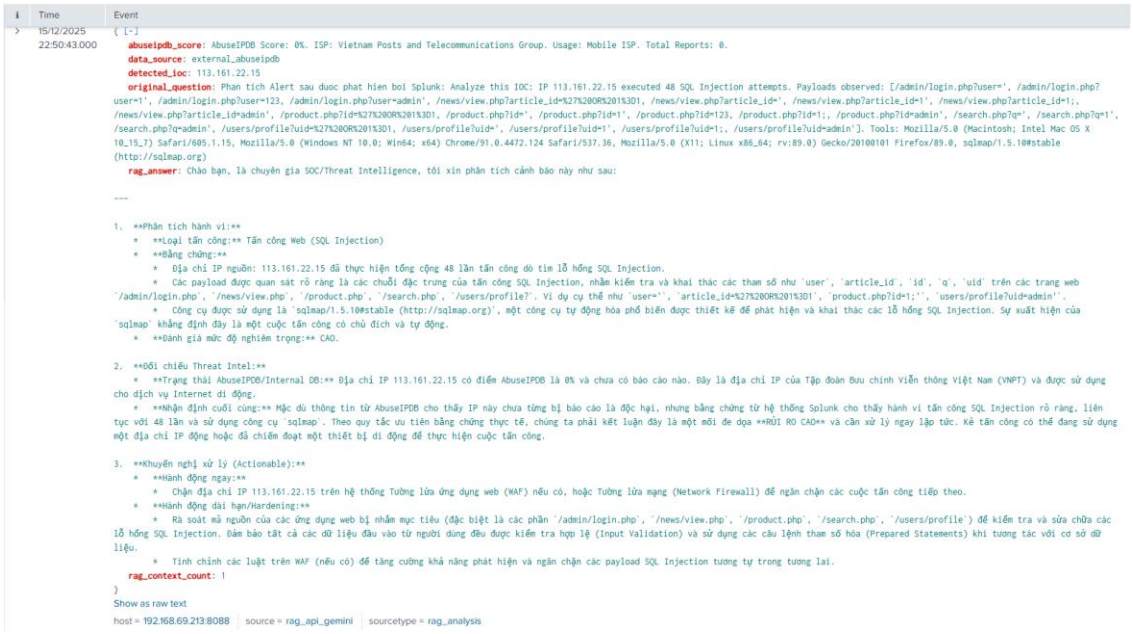
Bảng 2: So sánh SOC truyền thống với SOC tích hợp RAG

Tiêu chí so sánh	**SOC Truyền thống**	**SOC tích hợp RAG**
Khả năng ngữ cảnh hóa	Thông tin rời rạc (chỉ có IP, mã Hash, Log thô). Chuyên gia phải tự tra cứu thủ công từ nhiều nguồn bên ngoài để hiểu bối cảnh.	Tự động làm giàu ngữ cảnh: Hệ thống tự động truy xuất thông tin liên quan từ cơ sở dữ liệu để ghép nối với log sự cố
Thời gian phản hồi	Phụ thuộc hoàn toàn vào tốc độ thao tác của con người. Thường kéo dài từ vài chục phút đến hàng giờ đối với các sự cố phức tạp.	Rút ngắn đáng kể nhờ khả năng tự động phân tích và đề xuất giải pháp tức thời. Chuyên gia chỉ cần xác nhận quyết định cuối cùng.
Khả năng giải thích	Hạn chế. Các cảnh báo thường chỉ hiển thị mã lỗi hoặc tên luật vi phạm mà không giải thích "tại sao" đây là mối đe dọa	Minh bạch và dễ hiểu: LLM sinh ra báo cáo bằng ngôn ngữ tự nhiên, giải thích rõ ràng

		chuỗi diễn biến tấn công và lý do đưa ra kết luận
Cập nhật kiến thức	Cứng nhắc. Việc cập nhật luật mới hoặc quy trình xử lý đòi hỏi cấu hình thủ công và mất nhiều thời gian kiểm thử	Linh hoạt: Kiến thức mới (IOC mới, quy trình mới) chỉ cần được nạp vào cơ sở dữ liệu Vector là hệ thống có thể sử dụng ngay lập tức mà không cần huấn luyện lại mô hình

3.4. Thử nghiệm mở rộng

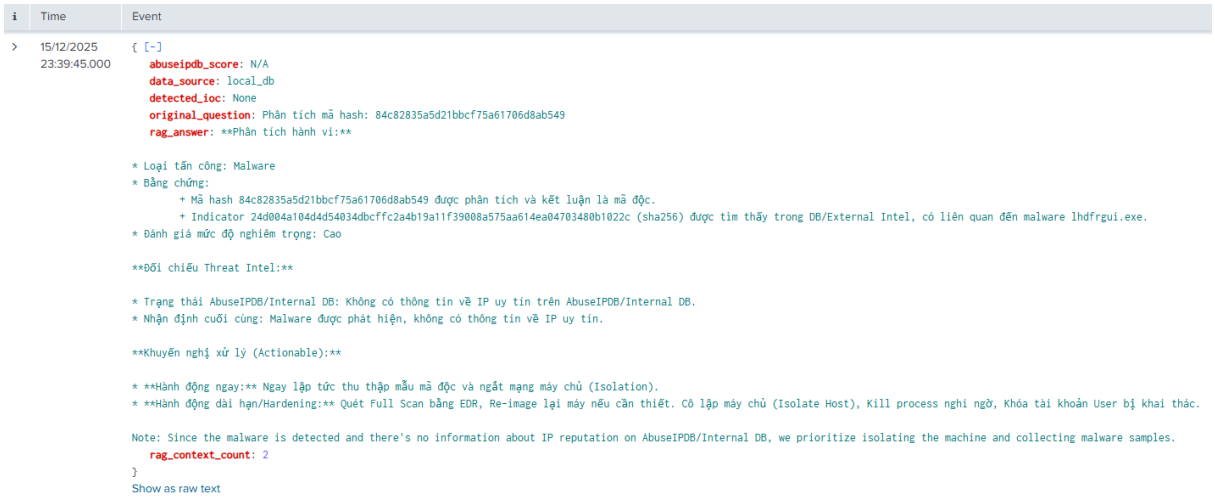
- Thử nghiệm thay thế mô hình LLM (Ollama vs Google Gemini)



Hình 10: Kết quả phân tích cảnh báo SQLi sử dụng Gemini

Nhận xét: Gemini nhanh hơn, thể hiện ngôn ngữ tự nhiên tốt hơn nhưng rủi ro bảo mật (gửi Internet)

- Thử nghiệm thay đổi nguồn Threat Intelligence (OTX vs VirusTotal)



i	Time	Event
>	15/12/2025 23:39:45.000	{ [-] abuseipdb_score: N/A data_source: local_db detected_ioc: None original_question: Phân tích mã hash: 84c82835a5d21bbcf75a61706d8ab549 rag_answer: **Phân tích hành vi:** * Loại tấn công: Malware * Bằng chứng: + Mã hash 84c82835a5d21bbcf75a61706d8ab549 được phân tích và kết luận là mã độc. + Indicator 24d004a104d4d54034dbcffc2a4b19a11f39008a575aa614ea04703480b1022c (sha256) được tìm thấy trong DB/External Intel, có liên quan đến malware lhdfrgui.exe. * Đánh giá mức độ nghiêm trọng: Cao **Đối chiếu Threat Intel:** * Trạng thái AbuseIPDB/Internal DB: Không có thông tin về IP uy tín trên AbuseIPDB/Internal DB. * Nhận định cuối cùng: Malware được phát hiện, không có thông tin về IP uy tín. **Khuyến nghị xử lý (Actionable):** * **Hành động ngay:** Ngay lập tức thu thập mẫu mã độc và ngắt mạng máy chủ (Isolation). * **Hành động dài hạn/Hardening:** Quét Full Scan bằng EDR, Re-image lại máy nếu cần thiết. Cô lập máy chủ (Isolate Host), Kill process nghi ngờ, Khóa tài khoản User bị khai thác. Note: Since the malware is detected and there's no information about IP reputation on AbuseIPDB/Internal DB, we prioritize isolating the machine and collecting malware samples. rag_context_count: 2 } Show as raw text

Bảng 3: Kết quả phân tích mã hash sử dụng dữ liệu VirusTotal

Nhận xét: VirusTotal thay OTX hoạt động tốt, kiểm chứng được khả năng tích hợp đa nguồn dữ liệu tri thức

KẾT LUẬN

1. Nội dung đã thực hiện

Đề án đã tập trung giải quyết bài toán quá tải cảnh báo tại các trung tâm SOC bằng cách ứng dụng công nghệ RAG. Sau thời gian nghiên cứu và triển khai, đề án đã đạt được các kết quả cụ thể như sau:

- Về mặt lý thuyết:
    - Hệ thống hóa được cơ sở lý luận về quy trình vận hành SOC, các phương pháp phân tích cảnh báo truyền thống và hạn chế của chúng.
    - Mô tả kiến trúc và nguyên lý hoạt động của LLM và kỹ thuật RAG.
- Về mặt thực tiễn:
    - Xây dựng thành công kiến trúc hệ thống giám sát và phân tích cảnh báo tự động gồm 3 thành phần chính: Máy nạn nhân, Hệ thống giám sát (Splunk SIEM) và Hệ thống phân tích RAG bằng các công nghệ mã nguồn mở:
        - Ollama: Vận hành mô hình ngôn ngữ Llama 3 cục bộ, đảm bảo an toàn dữ liệu.
        - Qdrant: Lưu trữ và truy xuất vector dữ liệu.
        - AlienVault OTX: Tự động làm giàu dữ liệu cảnh báo.

Thực hiện kiểm thử thành công trên 03 kịch bản tấn công phổ biến (SSH Brute Force, SQL Injection, Privilege Escalation). Kết quả cho thấy hệ thống giúp rút ngắn thời gian phân tích trung bình xuống còn dưới 60 giây/cảnh báo, đồng thời cung cấp các hướng dẫn xử lý chính xác, có căn cứ.

2. Tồn tại, hạn chế

Bên cạnh những kết quả đạt được, đề án vẫn còn tồn tại một số hạn chế cần khắc phục trong các giai đoạn phát triển tiếp theo:

- Yêu cầu tài nguyên phần cứng: Việc vận hành các mô hình LLM cục bộ (như Llama 3) đòi hỏi tài nguyên tính toán lớn (RAM > 16GB, khuyến nghị có GPU). Điều này có thể gây khó khăn khi triển khai trên diện rộng hoặc trên các máy chủ cấu hình thấp.
- Độ trễ trong trường hợp cao tải: Mặc dù thời gian phản hồi ~40 giây là chấp nhận được cho phân tích sâu, nhưng trong trường hợp hệ thống chịu các cuộc tấn công

dồn dập (như DDoS hoặc quét cổng quy mô lớn), hàng đợi xử lý của RAG có thể bị quá tải, dẫn đến độ trễ tăng cao.

Khả năng xử lý các tấn công phức tạp: Với các kỹ thuật tấn công tinh vi sử dụng mã hóa (Obfuscation) hoặc các biến thể Zero-day hoàn toàn mới chưa có trong cơ sở dữ liệu OTX/MITRE, khả năng phân tích của mô hình vẫn còn hạn chế và phụ thuộc nhiều vào chất lượng của dữ liệu huấn luyện.

3. Hướng phát triển

Dựa trên kết quả nghiên cứu và các hạn chế đã nhận diện, hướng phát triển tiếp theo của đề tài sẽ tập trung vào các nội dung:

- Tối ưu hóa mô hình: Nghiên cứu áp dụng các kỹ thuật lượng tử hóa (Quantization) để giảm kích thước và yêu cầu tài nguyên của mô hình LLM, giúp hệ thống hoạt động nhẹ nhàng hơn mà không làm giảm độ chính xác.
- Tích hợp cơ chế phản hồi (Feedback Loop): Xây dựng tính năng cho phép chuyên gia SOC đánh giá chất lượng câu trả lời của AI (Like/Dislike). Dữ liệu này sẽ được dùng để tinh chỉnh lại cơ sở dữ liệu vector, giúp hệ thống "thông minh" hơn theo thời gian.

Mở rộng sang Tự động hóa phản ứng (SOAR): Hiện tại hệ thống mới dừng lại ở mức "Đề xuất xử lý". Hướng phát triển tiếp theo là cho phép RAG tương tác trực tiếp với tường lửa hoặc Active Directory để thực thi các hành động ngăn chặn (như chặn IP, khóa tài khoản) khi độ tin cậy của cảnh báo đạt ngưỡng an toàn, tiến tới mô hình SOC tự hành (Autonomous SOC).