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

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(*Theo định hướng ứng dụng*)

HÀ NỘI - 2025

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

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC

TS. ĐINH TRƯỜNG DUY

HÀ NỘI - 2025

LỜI CAM ĐOAN

Tôi xin cam đoan các nghiên cứu trong đề án này là công trình nghiên cứu của cá nhân tôi, được thực hiện dưới sự hướng dẫn khoa học của thầy TS. Đinh Trường Duy, các số liệu và những kết luận nghiên cứu được trình bày trong luận văn này là hoàn toàn trung thực, do tôi nghiên cứu hoặc được tổng hợp từ nhiều nguồn tài liệu khác nhau. Các tài liệu tham chiếu có xuất xứ rõ ràng và được trích dẫn đầy đủ, hợp pháp.

Tôi xin hoàn toàn chịu trách nhiệm về lời cam đoan này.

Hà Nội, ngày 10 tháng 02 năm 2026

Học viên

Hà

Nguyễn Thị Thu Hà

LỜI CẢM ƠN

Trước tiên, em xin trân trọng cảm ơn các thầy cô giáo công tác trong Học viện Công nghệ Bưu chính viễn thông, nhất là Quý thầy cô, cán bộ, giảng viên Khoa Công nghệ thông tin, Khoa An toàn thông tin và Phòng Sau đại học đã truyền đạt cho em những kiến thức bổ ích trong suốt hai năm học vừa qua; giúp đỡ và tạo điều kiện cho em hoàn thành luận văn này.

Em xin gửi lời cảm ơn tới TS Đinh Trường Duy – Giảng viên Khoa An toàn thông tin đã giúp đỡ tận tình trong quá trình hướng dẫn, định hướng, nghiên cứu luận văn.

Bên cạnh đấy em xin gửi lời cảm tới Công ty Cổ phần Công nghệ An ninh Không gian mạng đã tạo điều kiện để em hoàn thành khoá học và đề tài luận văn.

Cuối cùng, em muốn gửi lời cảm ơn tới tất cả mọi người đã tạo những điều kiện tốt nhất để em hoàn thành luận văn tốt nghiệp này!

Hà Nội, ngày tháng năm 20

Học viên

Hà

Nguyễn Thị Thu Hà

MỤC LỤC

LỜI CAM ĐOAN i
LỜI CẢM ƠN ii
MỤC LỤC iii
DANH MỤC HÌNH VẼ v
DANH MỤC BẢNG vi
DANH MỤC TỪ VIẾT TẮT vii
MỞ ĐẦU 1
CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG GIÁM SÁT AN NINH MẠNG VÀ MÔ HÌNH TẠO SINH DỰA TRÊN TRUY XUẤT TĂNG CƯỜNG (RAG) 3
1.1. Tổng quan về SOC và hệ thống cảnh báo an ninh mạng 3
1.1.1. Khái niệm SOC 3
1.1.2. Chức năng của SOC 3
1.1.3. Hệ thống cảnh báo an ninh mạng 4
1.2. Khảo sát các phương pháp phân tích cảnh báo an ninh mạng hiện nay 6
1.3. Kiến trúc và đặc trưng của mô hình LLM 9
1.3.1. Khái niệm LLM 9
1.3.2. Kiến trúc LLM 9
1.3.3. Đặc trưng của mô hình LLM 11
1.4. Kiến trúc và nguyên lý hoạt động của mô hình RAG 13
1.4.1. Khái niệm RAG 13
1.4.2. Các thành phần của RAG 13
1.4.3. Kiến trúc và nguyên lý hoạt động của RAG 15
1.5. Ứng dụng mô hình RAG trong SOC 16
1.6. Khảo sát các giải pháp và nghiên cứu liên quan 17
1.7. Kết chương 19
CHƯƠNG 2: NGHIÊN CỨU VÀ ỨNG DỤNG MÔ HÌNH RAG TRONG PHÂN TÍCH CẢNH BÁO AN NINH MẠNG 20
2.1. Thiết kế kiến trúc mô hình RAG trong phân tích cảnh báo an ninh mạng 20
2.1.1. Yêu cầu kiến trúc 20

2.1.2. Thiết kế kiến trúc 20
2.2. Mô tả luồng dữ liệu và định nghĩa các chức năng của mô hình 27
2.2.1. Mô tả luồng dữ liệu 27
2.2.2. Định nghĩa các chức năng của mô hình 28
2.3. Triển khai mô hình trong môi trường thử nghiệm 31
2.3.1. Chuẩn bị môi trường và các thành phần thử nghiệm 31
2.3.2. Các bước triển khai 33
2.4. Kết chương 35
CHƯƠNG 3: THỬ NGHIỆM VÀ ĐÁNH GIÁ MÔ HÌNH 36
3.1. Mô tả mô hình thử nghiệm 36
3.1.1. Sơ đồ thử nghiệm 36
3.1.2. Quy trình thử nghiệm 36
3.1.3. Các chỉ số đánh giá 38
3.2. Kịch bản thử nghiệm 39
3.3. Nhận xét, phân tích kết quả thử nghiệm 50
3.4. Kết chương. 55
KẾT LUẬN 56
1. Nội dung đề án đã thực hiện 56
2. Tồn tại, hạn chế 56
3. Hướng phát triển 57
TÀI LIỆU THAM KHẢO 58

DANH MỤC HÌNH VẼ

Hình 1.1.3-1: Ví dụ về các thành phần trong cảnh báo an ninh mạng 5
Hình 1.3.2-1: Mô hình kiến trúc Transformer 10
Hình 1.3.3-1: Các đặc trưng của mô hình LLM 11
Hình 1.4.2-1: Sơ đồ kiến trúc RAG [8] 14
Hình 2.1.2-1: Sơ đồ tổng thể kiến trúc hệ thống RAG 21
Hình 2.1.2-2: Nền tảng bảo mật Splunk [19] 22
Hình 2.1.2-3: Mô tả hoạt động của Splunk 23
Hình 2.1.2-4: Cơ chế Tìm kiếm Vector trong Qdrant 24
Hình 2.2.1-1: Sơ đồ mô tả luồng dữ liệu 27
Hình 2.3.2-1: Sơ đồ các bước triển khai mô hình 33
Hình 3.1.1-1: Sơ đồ các thành phần thử nghiệm 36
Hình 3.1.2-1: Sơ đồ quy trình thử nghiệm 37
Hình 3.1.3-1: Log Brute Force trên Splunk 40
Hình 3.1.3-2: Kết quả chạy Rule Brute Force trên Splunk 41
Hình 3.1.3-3: Kết quả phân tích cảnh báo Brute Force 42
Hình 3.1.3-4: Log SQLi trên Splunk 44
Hình 3.1.3-5: Kết quả chạy Rule SQLi trên Splunk 45
Hình 3.1.3-6: Kết quả phân tích cảnh báo SQLi 47
Hình 3.1.3-7: Kết quả chạy Rule Privilege Escalation trên Splunk 48
Hình 3.1.3-8: Kết quả phân tích cảnh báo Privilege Escalation 50
Hình 3.1.3-1: Kết quả phân tích cảnh báo SQLi 51
Hình 3.1.3-2: Kết quả phân tích cảnh báo SQLi sử dụng Gemini 54
Hình 3.1.3-3: Kết quả phân tích mã hash sử dụng dữ liệu VirusTotal 55

DANH MỤC BẢNG

Bảng 3.1.3-1: Giá trị hiệu năng khi chạy hệ thống RAG 50
Bảng 3.1.3-2: So sánh đặc điểm vận hành giữa SOC truyền thống và SOC tích hợp RAG 52

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Thuật ngữ tiếng Anh	Thuật ngữ tiếng Việt
AI	Artificial Intelligence	Trí tuệ nhân tạo
CLI	Command Line Interface	Giao diện dòng lệnh
HEC	HTTP Event Collector	Phương thức thu thập dữ liệu dưới dạng HTTP/HTTPS vào Splunk
IDS/IPS	Intrusion Detection System/ Intrusion Prevention System	Hệ thống phát hiện/ngăn chặn xâm nhập
IOC	Indicators of Compromise	Chỉ số xâm phạm
LLM	Large Language Model	Mô hình ngôn ngữ lớn
ML	Machine Learning	Học máy
RAG	Retrieval-Augmented Generation	Mô hình tạo sinh dựa trên truy xuất tăng cường
SIEM	Security Information and Event Management	Giải pháp quản lý thông tin và sự kiện bảo mật
SOAR	Security Orchestration, Automation and Response	Giải pháp điều phối, tự động hóa và phản hồi bảo mật
SOC	Security Operations Center	Trung tâm điều hành an ninh mạng

MỞ ĐẦU

Tính cấp thiết của đề tài:

Trong bối cảnh vận hành các Trung tâm điều hành An ninh mạng hiện nay, một thách thức hiện hữu là sự bùng nổ về số lượng cảnh báo (alert fatigue). Các chuyên gia phân tích thường phải đối mặt với việc 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 thức 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 bùng nổ của trí tuệ nhân tạo tạo sinh (Generative AI), và sự kết hợp giữa mô hình ngôn ngữ lớn (LLM) cùng kỹ thuật truy xuất tăng cường (RAG) đã tạo ra bước đột phá 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 dữ liệu từ nhiều nguồn (Threat Intelligence, IOC feeds, MITRE ATT&CK…) để đưa ra đánh giá và giải thích đánh giá.

Hiện nay, các cuộc tấn công mạng nhằm vào hệ thống của doanh nghiệp và tổ chức tài chính ngày càng tăng về cả số lượng và quy mô, dẫn đến việc nghiên cứu và phát triển các giải pháp đảm bảo an ninh mạng trở nên cần thiết hơn. Xuất phát từ thực tiễn đó, đề tài này nghiên cứu sử dụng mã nguồn mở nhằm tự động việc 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 RAG
- Áp dụng hệ thống RAG vào phân tích cảnh báo an ninh mạng.
- Công cụ thử nghiệm (CLI hoặc giao diện demo) 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 SOC và mô hình RAG. Ứng dụng mô hình RAG vào phân tích cảnh báo an ninh mạng.

Đề án gồm 3 chương như sau:

Chương 1: Tổng quan về hệ thống giám sát an ninh mạng và mô hình tạo sinh dựa trên truy xuất tăng cường

Chương này tìm hiểu tổng quan về khái niệm, chức năng, các thành phần của hệ thống SOC, những thách thức đang phải đối mặt và ý nghĩa của việc phân tích các cảnh báo. Đồng thời tìm hiểu về kiến trúc và cách hoạt động của mô hình RAG.

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

Chương này thực hiện, xây dựng, triển khai và ứng dụng mô hình RAG trong phân tích cảnh báo an ninh mạng, bao gồm thiết kế kiến trúc, mô tả luồng dữ liệu và triển khai mô hình. Chương này đã đi sâu vào nền tảng ứng dụng kỹ thuật của mô hình RAG.

Chương 3: Thử nghiệm và đánh giá mô hình

Chương này thực hiện thử nghiệm mô hình, xây dựng các kịch bản thử nghiệm để phân tích các cảnh báo an ninh mạng để đánh giá kết quả mô hình mang lại.

CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG GIÁM SÁT AN NINH MẠNG VÀ MÔ HÌNH TẠO SINH DỰA TRÊN TRUY XUẤT TĂNG CƯỜNG (RAG)

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

1.1.1. Khái niệm SOC

Trước sự gia tăng tính phức tạp của các mối đe dọa trên không gian mạng, việc thiết lập Trung tâm Điều hành An ninh mạng (SOC) để đảm bảo an toàn thông tin liên tục đã trở thành yêu cầu thiết yếu đối với các tổ chức hiện đại. Được ví như "đầu não" an ninh của doanh nghiệp, SOC hoạt động độc lập với chức năng vận hành hạ tầng của phòng IT truyền thống, tập trung nguồn lực chuyên sâu vào ba trụ cột chính: phát hiện, phân tích và phản ứng sự cố. Hoạt động không ngừng nghỉ theo chế độ 24/7, đơn vị này kết hợp cả yếu tố con người, quy trình chuẩn hóa và công nghệ tiên tiến để tạo nên một hệ sinh thái bảo vệ toàn diện.

Thay vì chỉ được xem như một phòng giám sát kỹ thuật đơn thuần, SOC cần được hiểu là một hệ sinh thái vận hành bảo mật toàn diện.. Theo định nghĩa chuẩn hóa từ Viện SANS [1], hiệu quả của một SOC không nằm ở số lượng thiết bị phần cứng, mà phụ thuộc vào sự giao thoa chặt chẽ và cân bằng giữa ba trụ cột chiến lược là Con người, Quy trình và Công nghệ. Đội ngũ nhân sự (People) đóng vai trò nòng cốt với các nhà phân tích và kỹ sư ra quyết định dựa trên dữ liệu. Yếu tố này được hỗ trợ bởi các Quy trình vận hành (Process) gồm các tiêu chuẩn (SOP) và kịch bản ứng phó (Playbooks) nhằm đảm bảo tính nhất quán trong xử lý sự cố. Cuối cùng, nền tảng Công nghệ (Technology) với các giải pháp như SIEM, SOAR hay IDS/IPS đóng vai trò công cụ giúp thu thập dữ liệu và thực thi các quyết định bảo mật một cách tự động hoặc bán tự động [1].

1.1.2. Chức năng của SOC

Chức năng của SOC không chỉ dừng lại ở việc quan sát thụ động, mà là một chu trình hoạt động khép kín đảm bảo an toàn toàn diện cho hệ thống thông tin. Nền tảng của chu trình này là khả năng giám sát liên tục, trong đó SOC thực hiện việc theo dõi toàn bộ cơ sở hạ tầng công nghệ thông tin của tổ chức theo chế độ không ngừng nghỉ. Việc giám sát này không đơn thuần là thu thập dữ liệu, mà còn bao gồm quá trình chuẩn hóa log từ mọi điểm cuối, máy chủ và thiết bị mạng để loại bỏ các điểm mù trong hệ thống bảo mật.

Dựa trên dòng dữ liệu được giám sát, SOC triển khai các cơ chế phát hiện mối đe dọa tinh vi, kết hợp giữa các luật cảnh báo tĩnh đã được định nghĩa trước với các thuật toán

học máy động. Sự kết hợp này cho phép hệ thống không chỉ nhận diện các mẫu tấn công đã biết thông qua signature matching, mà còn có khả năng phát hiện các hành vi bất thường tinh vi như tấn công APT hay các lỗ hổng zero-day chưa từng xuất hiện. Khi một cảnh báo được kích hoạt, chức năng phân tích và điều tra sẽ được khởi động, trong đó các chuyên gia phân tích phải thực hiện quá trình phân loại ưu tiên để xác định đây là sự cố thực sự hay chỉ là báo động giả. Quá trình này đòi hỏi khả năng liên kết các sự kiện rời rạc, tra cứu thông tin từ các nguồn tình báo mối đe dọa và xác định mức độ ảnh hưởng tiềm tàng.

Sau khi xác nhận một sự cố thực sự, SOC chuyển sang giai đoạn phản ứng, thực hiện các giải pháp ngăn chặn thiệt hại, cô lập hệ thống bị ảnh hưởng và loại bỏ mối đe dọa khỏi môi trường mạng. Các hành động có thể bao gồm việc chặn địa chỉ IP độc hại, cô lập máy trạm bị nhiễm, xóa bỏ tệp tin nguy hiểm hoặc vô hiệu hóa các tài khoản bị xâm nhập. Cuối cùng, sau khi sự cố được kiểm soát, SOC hỗ trợ quá trình đưa hệ thống về hiện trạng bình thường và thực hiện đánh giá sau sự cố. Giai đoạn này không chỉ là khôi phục dịch vụ, mà còn để rút ra bài học, cập nhật dữ liệu về các kỹ thuật tấn công mới và tinh chỉnh các quy trình bảo mật để nâng cao khả năng phòng thủ trong tương lai.

1.1.3. Hệ thống cảnh báo an ninh mạng

Hệ thống cảnh báo an ninh mạng được ví như "trái tim" công nghệ của SOC, chịu trách nhiệm chuyển đổi hàng tỷ dòng log thô thành các cảnh báo có ý nghĩa để con người có thể xử lý. Trong kiến trúc hiện đại, hệ thống này thường được vận hành xoay quanh giải pháp Quản lý thông tin và sự kiện bảo mật.

1.1.3.1. Nguyên lý hoạt động

Hệ thống cảnh báo hoạt động theo quy trình xử lý dữ liệu nhiều tầng:

- Thu thập (Collection): Dữ liệu log được đẩy từ các nguồn (Firewall, Linux Server, Windows AD...) thông qua các Agent (như Splunk Forwarder, Syslog-ng) về hệ thống trung tâm.
- Chuẩn hóa (Normalization): Dữ liệu thô từ nhiều định dạng khác nhau được chuyển đổi về một định dạng chuẩn (ví dụ: Common Information Model - CIM) để dễ dàng truy vấn và so sánh.
- Tương quan (Correlation): Đây là bước quan trọng nhất. Hệ thống sử dụng các luật tương quan (Correlation Rules) để liên kết các sự kiện rời rạc thành một chuỗi hành vi.

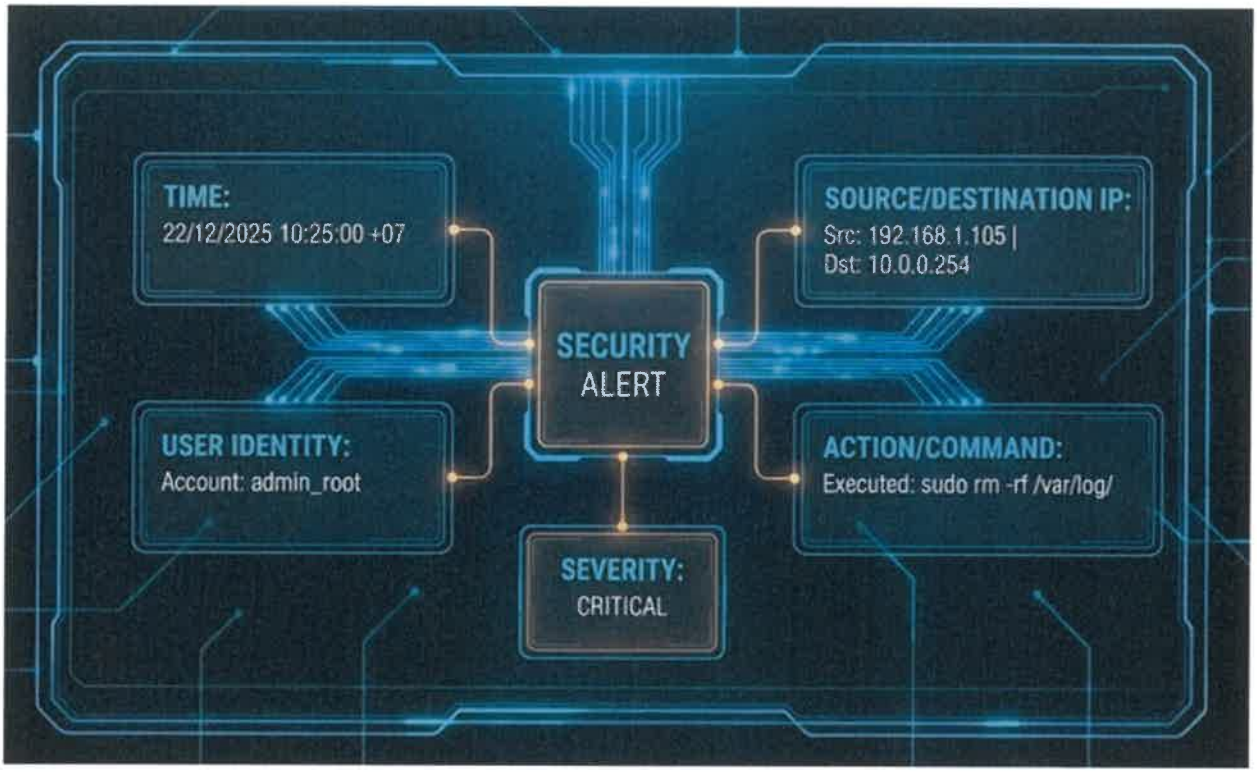
Ví dụ: Một lần đăng nhập thất bại là bình thường, nhưng 10 lần đăng nhập thất bại trong 1 phút từ cùng một IP, theo sau là một lần đăng nhập thành công, sẽ kích hoạt cảnh báo "Brute Force Attack Success".

- Cảnh báo (Alerting): Khi thỏa mãn điều kiện của luật, hệ thống sinh ra một bản tin cảnh báo (Alert) và gửi thông báo đến các nhà phân tích qua Dashboard, Email hoặc các kênh tích hợp.

1.1.3.2. Các thành phần dữ liệu trong cảnh báo

Một cảnh báo an ninh mạng tiêu chuẩn thường bao gồm các thông tin chỉ báo (Indicators of Compromise - IOCs) quan trọng như:

- Time: Thời gian xảy ra sự cố.
- Source/Destination IP: Địa chỉ IP nguồn tấn công và IP đích bị tấn công.
- User Identity: Tài khoản người dùng liên quan.
- Action/Command: Hành động hoặc câu lệnh được thực thi.
- Severity: Mức độ nghiêm trọng (Low, Medium, High, Critical).



Hình 1.1.3-1: Ví dụ về các thành phần trong cảnh báo an ninh mạng

1.1.3.3. Thách thức của hệ thống cảnh báo hiện nay

Dù đóng vai trò quan trọng, các hệ thống cảnh báo truyền thống (dựa trên SIEM Rule-based) đang gặp phải vấn đề lớn là "Quá tải cảnh báo" (Alert Fatigue). Do dựa trên các luật tĩnh, hệ thống thường sinh ra số lượng lớn cảnh báo dương tính giả (False Positive). Điều này khiến các chuyên gia bị quá tải, dẫn đến nguy cơ bỏ sót các cảnh báo thực sự

nguy hiểm hoặc phản ứng chậm trễ. Đây chính là động lực thúc đẩy việc ứng dụng các công nghệ mới như AI và Generative AI để hỗ trợ quá trình phân tích và sàng lọc cảnh báo, cũng là trọng tâm nghiên cứu của đề tài này.

1.2. Khảo sát các phương pháp phân tích cảnh báo an ninh mạng hiện nay

Tiến trình phát triển của an ninh mạng đã ghi nhận sự chuyển dịch mạnh mẽ trong phương pháp phân tích cảnh báo: từ các quy trình thủ công phụ thuộc hoàn toàn vào con người sang các hệ thống tự động hóa phức tạp ứng dụng Học máy (Machine Learning). Mỗi phương pháp tiếp cận đều tồn tại những ưu điểm và hạn chế nội tại; do đó, việc thấu hiểu bản chất của từng giai đoạn là cơ sở cốt lõi để đề xuất giải pháp tối ưu cho từng ngữ cảnh bảo mật cụ thể.

Phương pháp phân tích thủ công đại diện cho cách tiếp cận truyền thống và căn bản nhất, với con người giữ vai trò trung tâm và quyết định. Khi một cảnh báo xuất hiện, chuyên gia an ninh mạng phải thực hiện các thao tác phức tạp đòi hỏi cả kiến thức chuyên môn lẫn kinh nghiệm thực tế. Quá trình bắt đầu với việc phân tích các dòng log thô - những chuỗi ký tự dày đặc chứa đầy các mã số, địa chỉ IP và timestamp. Người phân tích phải có khả năng đọc hiểu những dòng log này, nhận diện các mẫu bất thường và trích xuất các thông tin then chốt. Tiếp theo, họ cần truy vấn cơ sở dữ liệu lịch sử để tìm hiểu xem địa chỉ IP liên quan hoặc tài khoản người dùng đã có hành vi đáng ngờ nào trước đó hay không. Công việc tra cứu không dừng lại ở dữ liệu nội bộ, mà còn mở rộng ra các nguồn tình báo mối đe dọa bên ngoài như VirusTotal hay AlienVault OTX để xác minh độ uy tín của các chỉ báo xâm phạm. Sau khi tổng hợp đủ thông tin, người phân tích phải đưa ra quyết định về cách xử lý dựa trên kinh nghiệm và hiểu biết của mình về môi trường mạng cụ thể của tổ chức.

Giá trị không thể phủ nhận của phương pháp này nằm ở khả năng tận dụng trực giác và kinh nghiệm tích lũy của con người. Một chuyên gia nhiều kinh nghiệm có thể nhận ra các dấu hiệu tinh tế mà máy móc bỏ qua, có khả năng đánh giá ngữ cảnh phức tạp và đưa ra những quyết định linh hoạt trong các tình huống chưa từng gặp. Con người vượt trội trong việc xử lý các trường hợp ngoại lệ, những tình huống không theo bất kỳ mẫu nào đã được lập trình sẵn. Tuy nhiên, phương pháp này cũng có những hạn chế làm giảm khả năng mở rộng. Hiệu quả của phân tích phụ thuộc nhiều vào trình độ và kinh nghiệm cá nhân của người thực hiện, tạo ra sự không đồng đều trong chất lượng xử lý giữa các chuyên gia khác nhau. Tốc độ xử lý bị giới hạn bởi khả năng nhận thức và thời gian làm việc của con người - một analyst không thể xử lý hàng nghìn cảnh báo mỗi ngày mà không bị kiệt sức. Hơn nữa, áp lực công việc liên tục và sự mệt mỏi dẫn đến nguy cơ cao về sai sót, bỏ sót các cảnh báo quan trọng hoặc đánh giá không chính xác mức độ nghiêm trọng. Một vấn đề

khác là tính nhất quán - hai chuyên gia khác nhau có thể đưa ra hai kết luận hoàn toàn khác nhau khi phân tích cùng một sự cố, gây khó khăn cho việc xây dựng các quy trình.

Để khắc phục những hạn chế về quy mô và tốc độ của phương pháp thủ công, cộng đồng an ninh mạng đã chuyển hướng sang các kỹ thuật thống kê và học máy, tập trung chủ yếu vào bài toán phát hiện các bất thường trong hành vi hệ thống. Tiền đề của phương pháp là: nếu chúng ta có thể xây dựng một mô hình toán học mô tả hành vi bình thường của hệ thống hoặc người dùng, thì bất kỳ sự lệch lạc đáng kể nào khỏi mô hình này đều có thể là dấu hiệu của một mối đe dọa. Quá trình bắt đầu bằng việc thu thập một lượng dữ liệu đủ để thiết lập một đường cơ sở, ví dụ như ghi nhận các mẫu truy cập mạng, lượng dữ liệu truyền tải, hoặc thời gian làm việc điển hình của người dùng trong giờ hành chính. Sau khi đường cơ sở được thiết lập, hệ thống liên tục so sánh các hoạt động hiện tại với mô hình này và đánh giá mức độ lệch lạc. Các thuật toán phổ biến như phân cụm K-Means có thể nhóm các hành vi tương tự nhau và đánh dấu những điểm dữ liệu nằm xa các cụm chính, trong khi các phương pháp như Isolation Forest chuyên phát hiện các điểm bất thường bằng cách đánh giá mức độ "cô lập" của chúng trong không gian đặc trưng.

Giá trị của phương pháp phát hiện bất thường (Anomaly Detection) nằm ở khả năng nhận diện các mối đe dọa mới – những cuộc tấn công không thể bị phát hiện bởi các cơ chế dựa trên chữ ký (signature-based). Đặc tính này đóng vai trò then chốt trong việc phòng thủ trước các cuộc khai thác lỗ hổng zero-day - những điểm yếu trong phần mềm mà ngay cả nhà phát triển cũng chưa biết đến, hoặc các hành vi lạm dụng nội bộ khi nhân viên sử dụng quyền truy cập hợp pháp của mình cho mục đích xấu. Hơn nữa, phương pháp này mang lại mức độ tự động hóa cao, giảm sự phụ thuộc vào việc phải viết và duy trì hàng nghìn luật phát hiện thủ công. Tuy nhiên, thực tế triển khai đã bộc lộ những thách thức đáng kể. Vấn đề nan giải nhất là tỷ lệ báo động giả cực kỳ cao - không phải mọi bất thường đều là tấn công. Một nhân viên làm việc muộn để hoàn thành deadline, một chiến dịch marketing làm tăng đột biến lưu lượng truy cập website, hay một thay đổi hợp lệ trong quy trình kinh doanh đều có thể được hệ thống hiểu nhầm là các cuộc tấn công. Điều này dẫn đến tình trạng nghịch lý: càng tinh chỉnh hệ thống để bắt được nhiều bất thường, càng sinh ra nhiều cảnh báo sai, nhưng nếu tăng ngưỡng để giảm nhiễu thì lại có nguy cơ bỏ sót các mối đe dọa thực sự. Một vấn đề khác là tính minh bạch - các mô hình học máy thường hoạt động như những hộp đen, đưa ra kết luận mà không giải thích rõ ràng lý do. Khi hệ thống báo rằng một hành vi là bất thường, chuyên gia phân tích thường không hiểu được căn cứ nào đã dẫn đến kết luận này, gây khó khăn khi điều tra tiếp theo và ra quyết định.

Phương pháp dựa trên luật đại diện cho cách tiếp cận trung dung, kết hợp khả năng tự động hóa của máy tính với kiến thức chuyên môn của con người. Phương pháp này sử dụng các quy tắc logic điều kiện được định nghĩa rõ ràng theo dạng "nếu-thì". Mỗi quy tắc

mô tả một điều kiện cụ thể cần giám sát và hành động sẽ thực hiện khi điều kiện đó được đáp ứng. Ví dụ, một quy tắc có thể phát biểu: "Nếu mã sự kiện là 4625 (đăng nhập Windows thất bại) VÀ số lượng sự kiện này lớn hơn năm lần trong vòng một phút từ cùng một địa chỉ IP, THÌ kích hoạt cảnh báo với thông điệp 'Phát hiện tấn công Brute Force'". Các quy tắc này xây dựng dựa trên các dấu hiệu nhận diện tấn công đã được biết đến, các kỹ thuật được mô tả trong framework MITRE ATT&CK, hoặc kinh nghiệm thực tế từ các sự cố đã xảy ra trước đó.

Điểm mạnh rõ ràng nhất của phương pháp dựa trên luật là độ chính xác cao khi đối phó với các mối đe dọa đã được nghiên cứu và hiểu rõ. Khi một kỹ thuật tấn công có dấu hiệu đặc trưng rõ ràng, việc viết một quy tắc để phát hiện nó là tương đối đơn giản và hiệu quả. Tính minh bạch của các quy tắc logic cũng là một lợi thế quan trọng - bất kỳ ai có kiến thức cơ bản đều có thể đọc hiểu quy tắc, biết chính xác điều kiện nào đang được giám sát và tại sao cảnh báo được kích hoạt. Điều này tạo điều kiện cho việc tùy chỉnh, bảo trì và giải thích kết quả. Về mặt hiệu năng, các quy tắc logic có thể được đánh chỉ mục và xử lý với tốc độ cao, đáp ứng yêu cầu về giám sát thời gian thực ngay cả khi xử lý hàng triệu sự kiện mỗi giây.

Hạn chế lớn của phương pháp này là sự bất lực hoàn toàn trước các cuộc tấn công mới chưa có trong cơ sở dữ liệu chữ ký. Một kỹ thuật tấn công sáng tạo hoặc một biến thể mới của mã độc có thể hoạt động công khai trong hệ thống mà không bị phát hiện, đơn giản vì không có quy tắc nào được viết để tìm kiếm nó. Vấn đề trở nên phức tạp hơn theo thời gian - khi các mối đe dọa mới liên tục xuất hiện, số lượng quy tắc tăng lên không ngừng. Một tổ chức trưởng thành có thể có hàng nghìn đến hàng chục nghìn quy tắc hoạt động, tạo ra gánh nặng quản lý khổng lồ. Các quy tắc phải được xem xét định kỳ, tinh chỉnh để giảm báo động giả, vô hiệu hóa khi không còn phù hợp, và đảm bảo không xung đột lẫn nhau. Quá trình này đòi hỏi nguồn lực đáng kể và đòi hỏi chuyên môn cao.

Nhìn lại toàn cảnh, có thể thấy rằng không có phương pháp đơn lẻ nào là hoàn hảo cho mọi tình huống. Mỗi cách tiếp cận đều có vùng hoạt động hiệu quả riêng và những điểm mù riêng. Xu hướng hiện đại trong lĩnh vực an ninh mạng là hướng tới các kiến trúc lai ghép, kết hợp điểm mạnh của nhiều phương pháp để bù đắp cho những điểm yếu của nhau. Trong thực tế đó, sự xuất hiện của các công nghệ Generative AI, đặc biệt là khả năng của mô hình LLM được tăng cường bởi cơ chế truy xuất thông tin, mở ra một chiều hướng mới. Các công nghệ này không chỉ có thể hỗ trợ việc phân tích và phát hiện, mà còn có khả năng giải thích các phát hiện của mình bằng ngôn ngữ tự nhiên, đưa ra ngữ cảnh từ nhiều nguồn thông tin khác nhau, và đề xuất các hành động cụ thể dựa trên kiến thức tích lũy. Đây chính là khoảng trống mà các phương pháp truyền thống đang thiếu và là động lực chính thúc đẩy hướng nghiên cứu của đề tài này.

1.3. Kiến trúc và đặc trưng của mô hình LLM

Trong những năm gần đây, sự phát triển vượt bậc của các mô hình ngôn ngữ lớn (Large Language Models – LLM) đã mở ra nhiều cơ hội mới trong lĩnh vực xử lý ngôn ngữ tự nhiên và AI. Các mô hình này, với khả năng hiểu và sinh ngôn ngữ tự nhiên ở mức độ gần với con người, đang dần được ứng dụng rộng rãi trong nhiều lĩnh vực, bao gồm chăm sóc khách hàng, hỗ trợ ra quyết định, và đặc biệt là an ninh mạng.

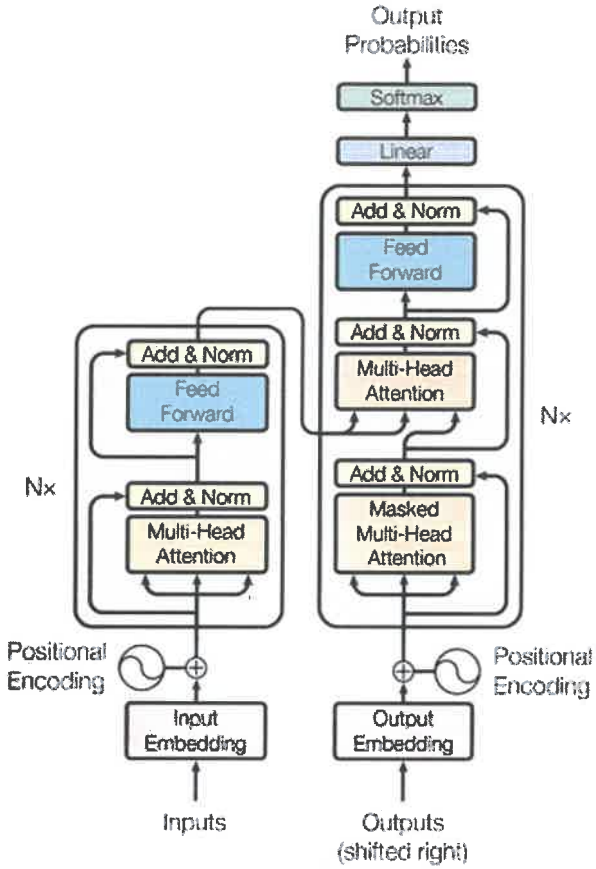
1.3.1. Khái niệm LLM

"Mô hình ngôn ngữ lớn là một loại mô hình học sâu (Deep Learning) được huấn luyện trên một tập dữ liệu văn bản khổng lồ (có thể lên tới hàng nghìn tỷ từ) nhằm hiểu, tóm tắt, tạo sinh và dự đoán nội dung văn bản mới" [2]. Về bản chất, LLM là các mô hình xác suất, hoạt động bằng cách dự đoán từ tiếp theo (token) trong một chuỗi dựa trên ngữ cảnh của các từ trước đó..

Khác với các mô hình NLP truyền thống thường được thiết kế cho các tác vụ cụ thể [3] (như phân loại văn bản hay dịch máy), LLM thuộc nhóm "Mô hình nền tảng" (Foundation Models). Điều này có nghĩa là một mô hình duy nhất sau khi được huấn luyện trước (Pre-training) có thể được tinh chỉnh (Fine-tuning) để thực hiện đa dạng các tác vụ khác nhau mà không cần xây dựng lại từ đầu. Các ví dụ điển hình bao gồm dòng mô hình GPT (Generative Pre-trained Transformer) của OpenAI, Llama của Meta hay Gemini của Google.

1.3.2. Kiến trúc LLM

Các LLM hiện nay đều được xây dựng dựa trên kiến trúc mạng nơ-ron Transformer, được Google giới thiệu lần đầu vào năm 2017 trong bài báo kinh điển "Attention Is All You Need" [4]. Kiến trúc Transformer đã thay thế các mạng hồi quy (RNN/LSTM) trước đây nhờ khả năng xử lý song song dữ liệu và nắm bắt sự phụ thuộc xa (long-range dependencies) trong văn bản tốt hơn.



Hình 1.3.2-1: Mô hình kiến trúc Transformer

Kiến trúc Transformer bao gồm hai thành phần chính:

- Bộ mã hóa (Encoder): Chịu trách nhiệm xử lý văn bản đầu vào để hiểu ngữ nghĩa.
- Bộ giải mã (Decoder): Chịu trách nhiệm sinh ra văn bản đầu ra.

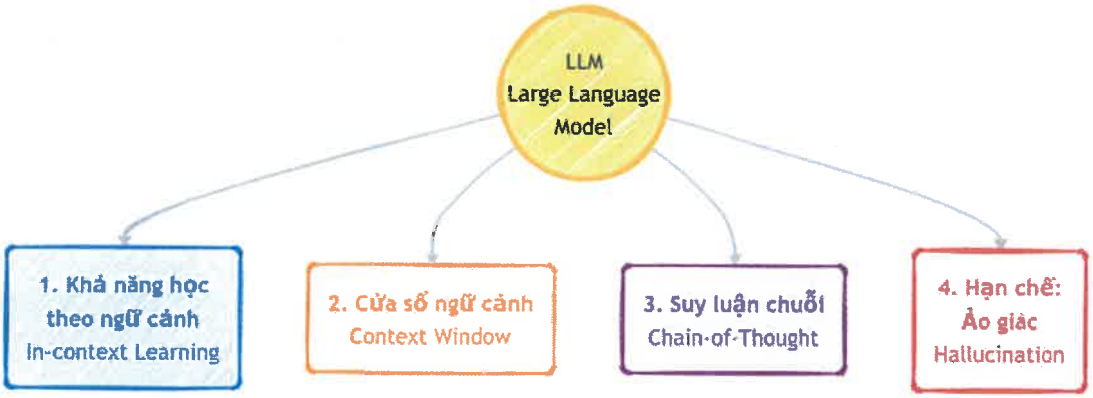
Tuy nhiên, phần lớn các LLM tạo sinh (Generative LLMs) được sử dụng hiện nay (như GPT-4, Llama 3) đều sử dụng kiến trúc Decoder-only.

Cơ chế cốt lõi tạo nên sức mạnh của Transformer là cơ chế Self-Attention (Sự chú ý bản thân). Cơ chế này cho phép mô hình đánh trọng số (weight) tầm quan trọng của các từ khác nhau trong một câu đối với từ đang được xử lý.

Ví dụ: Trong câu "Hệ thống phát hiện *mã độc* nhưng không thể xóa *nó*", cơ chế Attention giúp mô hình hiểu rằng từ "nó" đang ám chỉ "mã độc" chứ không phải "hệ thống". Khả năng này cực kỳ quan trọng trong việc phân tích các bản ghi log an ninh mạng phức tạp, nơi mối quan hệ giữa các thực thể (IP, User, File) thường nằm rải rác.

1.3.3. Đặc trưng của mô hình LLM

Để hiểu được lý do các mô hình ngôn ngữ lớn đang tạo ra cuộc cách mạng trong nhiều lĩnh vực ứng dụng, chúng ta cần đi sâu vào phân tích những đặc trưng độc đáo tạo nên sức mạnh cũng như những rủi ro tiềm ẩn của chúng. Những đặc trưng này [5] không chỉ quyết định khả năng ứng dụng mà còn định hình cách thức chúng ta cần thiết kế hệ thống để tận dụng tối đa lợi ích trong khi giảm thiểu những hạn chế.



Hình 1.3.3-1: Các đặc trưng của mô hình LLM

Đặc trưng đầu tiên đáng chú ý là khả năng học theo ngữ cảnh. So với các mô hình truyền thống đòi hỏi phải fine-tune với dữ liệu mới cho mỗi tác vụ khác nhau, các mô hình LLM có khả năng hiểu và thực hiện tác vụ chỉ dựa trên mô tả bằng ngôn ngữ tự nhiên hoặc một vài ví dụ minh họa. Khả năng này biểu hiện ở hai mức độ khác nhau. Ở mức cơ bản nhất, được gọi là học zero-shot, mô hình có thể phản ứng với các chỉ dẫn hoàn toàn mới mà nó chưa từng được huấn luyện cụ thể. Ví dụ, khi được yêu cầu "Hãy phân tích đoạn log sau đây và xác định xem có dấu hiệu của cuộc tấn công không", mô hình có thể đưa ra phân tích hợp lý mặc dù không được huấn luyện cho việc phân tích log bảo mật. Khả năng này xuất phát từ việc mô hình đã học được các mẫu ngôn ngữ và kiến thức chung trong quá trình huấn luyện trên kho dữ liệu khổng lồ. Ở mức tinh vi hơn, học few-shot cho phép chúng ta định hướng kết quả trả về của mô hình bằng việc cung cấp ví dụ mẫu ngay trong câu lệnh. Ví dụ, chúng ta có thể đưa cho mô hình hai ba ví dụ về cách phân tích các loại cảnh báo khác nhau theo chuẩn của tổ chức, sau đó yêu cầu nó phân tích một cảnh báo mới theo phong cách tương tự. Mô hình sẽ học được cấu trúc và phong cách mong muốn từ các ví dụ này và áp dụng vào nhiệm vụ mới. Trong ngữ cảnh của đề tài này, kỹ thuật few-shot trở nên vô cùng giá trị vì nó cho phép chúng ta điều chỉnh cách mô hình phân tích và trình bày thông tin mà không cần phải fine-tune trên một tập dữ liệu lớn về các sự cố bảo mật.

Đặc trưng thứ hai là khái niệm cửa sổ ngữ cảnh. Cửa sổ ngữ cảnh định nghĩa là lượng token tối đa mà mô hình có thể "ghi nhớ" và xử lý trong một lần tương tác. Một

token có thể là một từ, một phần của từ, hoặc một ký tự đặc biệt, và số lượng token này trực tiếp giới hạn lượng thông tin chúng ta có thể đưa vào mô hình. Trong những thế hệ đầu tiên của các mô hình ngôn ngữ lớn, giới hạn này khá chặt chẽ - mô hình GPT-3 ban đầu chỉ hỗ trợ hai nghìn token, tương đương khoảng một đến hai trang văn bản. Điều này tạo ra thách thức lớn trong nhiều ứng dụng thực tế, đặc biệt là trong phân tích bảo mật nơi các file log có thể chứa hàng nghìn dòng. Tuy nhiên, trong những năm gần đây công nghệ đã mang lại những bước nhảy vọt đáng kinh ngạc. Các mô hình thế hệ mới như Gemini 1.5 Pro hay Claude 3 có thể xử lý cửa sổ ngữ cảnh lên tới một triệu token, cho phép nạp toàn bộ các tài liệu kỹ thuật dài, nhiều file log cùng lúc, hoặc thậm chí toàn bộ codebase của một dự án để phân tích. Đối với đề tài này, khả năng này có ý nghĩa quan trọng vì nó cho phép hệ thống phân tích các cảnh báo phức tạp với đầy đủ ngữ cảnh từ nhiều nguồn khác nhau mà không cần phải cắt xén hoặc tóm tắt thông tin.

Khả năng suy luận chuỗi là đặc trưng thứ ba, đây là sự tiến bộ của các mô hình LLM so với các hệ thống AI trước đây. Thay vì chỉ đưa ra một câu trả lời cuối cùng, các mô hình này có khả năng chia nhỏ một vấn đề phức tạp thành các bước suy luận logic tuần tự, giống như cách một con người sẽ tiếp cận giải quyết vấn đề. Quá trình này thường được kích hoạt thông qua các prompt được thiết kế đặc biệt, yêu cầu mô hình "suy nghĩ từng bước" hoặc "giải thích lý do cho mỗi kết luận". Khi được hướng dẫn như vậy, mô hình sẽ không nhảy trực tiếp đến kết luận mà thay vào đó xây dựng một chuỗi lập luận có cấu trúc. Trong bối cảnh phân tích an ninh mạng, khả năng này trở nên vô cùng có giá trị. Thay vì chỉ đơn giản kết luận "Đây là một cuộc tấn công", mô hình có thể trình bày toàn bộ chuỗi suy luận: "Bước một: Quan sát thấy địa chỉ IP lạ đang thực hiện quét cổng trên dải mạng nội bộ, điều này cho thấy giai đoạn trinh sát. Bước hai: Sau khi quét, IP này tập trung vào cổng SSH của máy chủ quan trọng và thực hiện nhiều lần đăng nhập thất bại, đây là dấu hiệu của tấn công brute force. Bước ba: Một lần đăng nhập thành công xuất hiện, cho thấy kẻ tấn công đã chiếm được thông tin xác thực. Bước bốn: Ngay sau đó, các lệnh mã hóa file hàng loạt được thực thi với quyền root. Kết luận: Đây có khả năng cao là một cuộc tấn công ransomware đang trong giai đoạn thực thi." Chuỗi suy luận này không chỉ đưa ra kết luận mà còn giải thích rõ ràng căn cứ, giúp các chuyên gia phân tích hiểu được logic đằng sau và có thể xác minh hoặc điều chỉnh nếu cần.

Tuy nhiên, cùng với những khả năng ấn tượng, các mô hình LLM cũng mang trong mình một đặc trưng rủi ro được gọi là hiện tượng ảo giác. Đây có lẽ là thách thức lớn nhất và nguy hiểm nhất khi triển khai các mô hình này trong các ứng dụng quan trọng. Hiện tượng ảo giác xảy ra khi mô hình đưa ra các thông tin không chính xác, bịa đặt hoặc không có căn cứ, nhưng trình bày chúng với giọng điệu tự tin và thuyết phục đến mức người dùng có thể tin tưởng mà không kiểm chứng. Bản chất của vấn đề nằm ở cách thức hoạt động

của các mô hình này - chúng là những hệ thống xác suất được huấn luyện để dự đoán token tiếp theo dựa trên các token trước đó, chứ không phải là cơ sở dữ liệu sự thật tuyệt đối. Khi được hỏi về một thông tin mà mô hình không chắc chắn, thay vì thừa nhận sự thiếu hiểu biết, nó có thể "bịa" ra một câu trả lời có vẻ hợp lý dựa trên các mẫu ngôn ngữ đã học. Trong lĩnh vực an ninh mạng, điều này có thể biểu hiện dưới nhiều dạng nguy hiểm. Mô hình có thể bịa ra một mã lỗi không tồn tại để giải thích một thông báo hệ thống, trích dẫn một bài báo nghiên cứu bảo mật giả mạo với tên tác giả và năm xuất bản cụ thể, hoặc đưa ra các khuyến nghị xử lý dựa trên các quy trình không có thật. Một ví dụ cụ thể: khi được hỏi về cách khắc phục một lỗ hổng cụ thể, mô hình có thể đề xuất "Cập nhật lên phiên bản 2.4.7 của thư viện X", trong khi thực tế phiên bản này không bao giờ tồn tại. Điều đáng lo ngại là những thông tin sai lệch này thường được trình bày với các chi tiết cụ thể và độ tự tin cao, khiến chúng trở nên đặc biệt nguy hiểm.

Chính những đặc trưng trên, đặc biệt là việc kết hợp giữa khả năng mạnh mẽ và những rủi ro tiềm ẩn, đã dẫn đến sự ra đời của kiến trúc tăng cường truy xuất. Kiến trúc này được thiết kế để tận dụng những ưu điểm của mô hình LLM- khả năng hiểu ngữ cảnh phức tạp, suy luận logic và trả lời bằng văn bản tự nhiên - trong khi giảm thiểu đáng kể rủi ro ảo giác bằng cách buộc mô hình phải dựa vào các nguồn thông tin xác thực được cung cấp từ bên ngoài. Thay vì để mô hình tự do tạo ra thông tin từ bộ nhớ nội tại không hoàn hảo của nó, hệ thống RAG đảm bảo rằng mọi phát biểu đều được "dựa" vào các tài liệu, dữ liệu hoặc kiến thức thực tế đã được xác minh. Đây chính là cầu nối then chốt giúp chúng ta có thể tin tưởng triển khai các mô hình LLM trong các môi trường quan trọng như phân tích bảo mật.

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

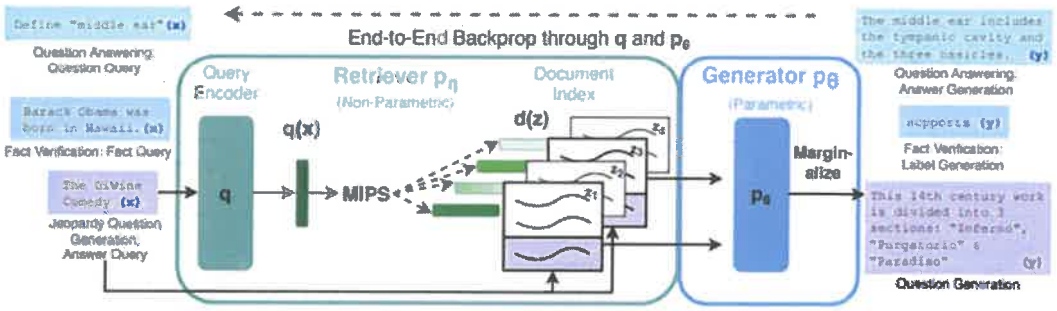
1.4.1. Khái niệm RAG

Retrieval-Augmented Generation (RAG) là một kỹ thuật được giới thiệu lần đầu bởi Patrick Lewis và các cộng sự (Facebook AI Research) vào năm 2020 trong bài báo "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" [6].

RAG là một kiến trúc lai (hybrid architecture) cho phép các mô hình LLM truy xuất và kết hợp thông tin mới [7]. Nói cách khác, RAG là sự kết hợp giữa khả năng tạo sinh ngôn ngữ của các mô hình LLM (parametric memory) với khả năng truy xuất thông tin chính xác từ các nguồn bên ngoài (non-parametric memory).

1.4.2. Các thành phần của RAG

Một hệ thống RAG hoàn chỉnh bao gồm ba thành phần hoạt động tương hỗ lẫn nhau:



Hình 1.4.2-1: Sơ đồ kiến trúc RAG [8]

a) Bộ truy xuất thông tin (Retriever) Đây là thành phần phụ trách tìm kiếm các thông tin có liên quan từ cơ sở dữ liệu hoặc từ các nguồn dữ liệu bên ngoài để bổ trợ cho câu hỏi của người dùng.

- Input: Câu truy vấn của người dùng (Query).
- Output: Một danh sách các đoạn văn bản (Chunks) có nội dung liên quan nhất (Top-K results).
- Chức năng: Tìm kiếm top-K đoạn văn bản (z) từ cơ sở dữ liệu có nội dung tương đồng nhất với câu truy vấn (x) của người dùng
- Công nghệ cốt lõi: Sử dụng các mô hình Bi-Encoder (như BERT) để mã hóa cả câu hỏi và tài liệu thành các vector n-chiều. Phép đo độ tương đồng thường dùng là Cosine Similarity hoặc Dot Product.

b) Bộ sinh ngôn ngữ (Generator) Đây là thành phần tạo ra câu trả lời sử dụng thông tin được truy xuất và dữ liệu cục bộ của mô hình. Hay nói cách khác, đây chính là mô hình ngôn ngữ lớn-LLM.

- Input: Một câu lệnh (Prompt) bao gồm câu hỏi của người dùng cộng với các thông tin ngữ cảnh do bộ truy xuất cung cấp.
- Output: Câu trả lời hoàn chỉnh bằng ngôn ngữ tự nhiên.
- Vai trò: LLM đóng vai trò như một bộ xử lý logic và tổng hợp thông tin. Nó không cần phải "nhớ" kiến thức, mà chỉ cần biết cách đọc hiểu thông tin được cung cấp và trình bày lại theo yêu cầu.

c) Cơ sở dữ liệu kiến thức (Knowledge Base): Là kho dữ liệu chứa các thông tin chuyên ngành mà chúng ta muốn hệ thống AI nắm bắt. Kho dữ liệu này bao gồm cơ sở dữ liệu có trong hệ thống và cả dữ liệu từ các nguồn bên ngoài. Trong đề tài này, cơ sở dữ liệu bao gồm:

- Khung kiến thức MITRE ATT&CK (chiến thuật, kỹ thuật tấn công).
- Các quy trình phản ứng sự cố chuẩn (Playbooks) của tổ chức.
- Dữ liệu IOC từ các nguồn Threat Intelligence.

1.4.3. Kiến trúc và nguyên lý hoạt động của RAG

Quy trình hoạt động của hệ thống RAG có thể chia thành hai giai đoạn chính: Giai đoạn nạp dữ liệu (Ingestion) và Giai đoạn truy vấn (Inference).

Giai đoạn 1: Nạp dữ liệu và Đánh chỉ mục (Ingestion & Indexing) Đây là quá trình chuẩn bị cơ sở dữ liệu, thường chạy định kỳ (offline):

1. Thu thập (Load): Tài liệu từ nhiều nguồn (PDF, Docx, HTML, API) được thu thập.
2. Chia nhỏ (Chunking): Các văn bản dài được chia thành các đoạn nhỏ (chunks) có độ dài phù hợp (ví dụ: 500-1000 tokens) để đảm bảo ngữ cảnh được bảo toàn và phù hợp với cửa sổ ngữ cảnh của LLM.
3. Mã hóa Vector (Embedding): Mỗi đoạn văn bản được đưa qua mô hình Embedding để chuyển đổi thành một vector nhiều chiều (ví dụ: 1536 chiều). Vector này đại diện cho ý nghĩa ngữ nghĩa của đoạn văn.
4. Lưu trữ (Storing): Các vector cùng với nội dung gốc được lưu vào Vector Database để phục vụ việc tìm kiếm nhanh.

Giai đoạn 2: Truy vấn và Tạo sinh (Retrieval & Generation) Đây là quy trình xảy ra khi người dùng tương tác với hệ thống (online):

1. Nhận câu hỏi: Người dùng (hoặc hệ thống SIEM) gửi một câu hỏi/cảnh báo (Query). Ví dụ: *IP 192.168.1.5 đang brute-force, tôi phải làm gì?"*
2. Mã hóa câu hỏi: Câu hỏi được chuyển đổi thành vector sử dụng cùng một mô hình Embedding như giai đoạn 1.
3. Tìm kiếm tương đồng (Similarity Search): Vector của câu hỏi được so sánh với hàng triệu vector trong cơ sở dữ liệu để tìm ra K đoạn văn bản gần nhất (có nội dung liên quan nhất).
4. Tăng cường ngữ cảnh (Augmentation): Các đoạn văn bản tìm được (Context) được ghép vào câu hỏi gốc để tạo thành một Prompt hoàn chỉnh.
    - *Prompt mẫu:* "Dựa trên tài liệu quy trình sau: [Nội dung quy trình xử lý Brute Force], hãy trả lời câu hỏi: [Câu hỏi của người dùng]".

5. Tạo câu trả lời (Generation): Prompt được gửi đến LLM. LLM đọc quy trình và sinh ra câu trả lời chính xác, có trích dẫn nguồn gốc.

Nhờ cơ chế này, RAG giúp giải quyết bài toán cập nhật kiến thức cho LLM mà không cần huấn luyện lại, đồng thời cung cấp khả năng giải thích (Explainability) cho các quyết định của AI trong môi trường SOC.

1.5. Ứng dụng mô hình RAG trong SOC

Việc tích hợp mô hình Retrieval-Augmented Generation (RAG) vào Trung tâm Điều hành An ninh mạng (Security Operations Center - SOC) đang mở ra những hướng tiếp cận đột phá trong việc giải quyết hai thách thức then chốt của ngành an ninh mạng hiện nay: tình trạng "quá tải cảnh báo" (alert fatigue) và tình trạng thiếu hụt nhân lực chuyên môn cao. Theo báo cáo của Ponemon Institute (2022), các nhóm SOC trung bình phải xử lý hơn 4.484 cảnh báo mỗi ngày, trong khi chỉ có đủ nguồn lực để điều tra khoảng 46% số cảnh báo này [9]. Đồng thời, (ISC) Cybersecurity Workforce Study (2023) chỉ ra rằng toàn cầu đang thiếu hụt khoảng 4 triệu chuyên gia an ninh mạng [10].

Tự động phân tích và phân loại cảnh báo (Automated Alert Triage): RAG có thể trở thành một trợ lý ảo "Tier 1 Analyst". Khi có cảnh báo mới từ SIEM, hệ thống RAG tự động truy xuất các thông tin từ cơ sở dữ liệu (như lịch sử cảnh báo, hồ sơ tài sản) để đánh giá mức độ nghiêm trọng. Dựa trên việc kết hợp thông tin được truy xuất với khả năng phân tích ngữ nghĩa của LLM, RAG có thể đánh giá sơ bộ mức độ nghiêm trọng, xác định độ tin cậy của cảnh báo và loại bỏ nhanh chóng các dương tính giả (False Positives) - vốn chiếm tới 30-50% tổng số cảnh báo theo nghiên cứu của Gartner [11]. Điều này làm giảm gánh nặng cho các chuyên gia con người, cho phép họ tập trung vào các mối đe dọa thực sự nghiêm trọng..

Hỗ trợ điều tra và săn tìm mối đe dọa (Investigation & Threat Hunting): Trong quá trình điều tra sự cố, các chuyên gia SOC thường phải tra cứu thủ công thông tin từ nhiều nguồn như: Các nền tảng Threat Intelligence (VirusTotal, AlienVault OTX, MISP, IBM X-Force); Cơ sở dữ liệu kỹ thuật chiến thuật tấn công (MITRE ATT&CK Framework); Tài liệu nội bộ về kiến trúc hạ tầng và chính sách bảo mật; Diễn đàn chuyên ngành và báo cáo nghiên cứu mới nhất;... để hiểu rõ về một mã độc hay một địa chỉ IP lạ. Quá trình này không chỉ tốn thời gian mà còn đòi hỏi kỹ năng cao trong việc tổng hợp và phân tích thông tin từ nhiều nguồn khác biệt. Theo nghiên cứu của IBM Security (2023), thời gian trung bình để xác định và phân tích một mối đe dọa (Mean Time To Detect and Analyze) là 204 ngày [12]. RAG giúp tự động hóa quá trình này bằng cách tổng hợp thông tin từ các nguồn

Threat Intelligence và trình bày dưới dạng báo cáo ngắn gọn, giúp rút ngắn thời gian điều tra (MTTR - Mean Time To Response) từ 25-40% [13].

Đề xuất quy trình xử lý (Recommendation): Một trong những ứng dụng có giá trị cao của RAG trong SOC là khả năng đề xuất quy trình xử lý phù hợp dựa trên loại hình tấn công được phát hiện. RAG có khả năng truy xuất quy trình xử lý chuẩn (Standard Operating Procedure - SOP) hoặc Playbook tương ứng từ kho tài liệu của tổ chức (local database). Hệ thống không chỉ đưa ra đường dẫn đến tài liệu mà còn có thể trích xuất các bước hành động cụ thể (ví dụ: "Bước 1: Cô lập máy chủ X", "Bước 2: Reset mật khẩu tài khoản Y") để hướng dẫn nhân viên trực vận hành.

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

Việc ứng dụng trí tuệ nhân tạo vào phân tích bảo mật không phải là hướng tiếp cận hoàn toàn mới. Trong những năm gần đây, cộng đồng nghiên cứu và các tổ chức công nghệ đã thực hiện nhiều nỗ lực nhằm tận dụng sức mạnh của học máy và xử lý ngôn ngữ tự nhiên để giải quyết bài toán quá tải cảnh báo trong môi trường SOC. Tuy nhiên, các giải pháp hiện có vẫn còn những khoảng trống cần được lấp đầy.

Trong những nghiên cứu tiên phong về ứng dụng RAG trong SOC là dự án Security Event Response Copilot (SERC), được công bố trên tạp chí Sensors (MDPI) tháng 1/2025. Nghiên cứu này tích hợp RAG với Wazuh - một nền tảng SIEM mã nguồn mở - để hỗ trợ phân tích viên SOC trong việc phản ứng và giảm thiểu các vi phạm bảo mật một cách hiệu quả hơn [14].

SERC bao gồm hai thành phần cốt lõi: (1) trích xuất dữ liệu sự kiện bảo mật sử dụng phương pháp RAG, và (2) hướng dẫn phản ứng sự cố dựa trên LLM. Hệ thống áp dụng phương pháp RAG để truy xuất thông tin ngữ cảnh cụ thể từ ba bộ sưu tập vector: kiến thức phản ứng sự cố tổng quát, khung MITRE ATT&CK, và khung NIST Cybersecurity Framework (CSF) [14].

Kiến trúc này cho phép hệ thống nhận diện chính xác các kỹ thuật đối nghịch (TTPs) liên quan đến sự kiện bảo mật, đồng thời tuân thủ các thực hành tốt nhất trong an ninh mạng thông qua việc so sánh với các khung tiêu chuẩn.

Một nghiên cứu khác được công bố trên arXiv (tháng 8/2025) đã phân tích 3.090 truy vấn được gửi đến GPT-4 trong 10 tháng (từ tháng 5/2023 đến tháng 3/2024) bởi 45 chuyên gia SOC tại eSentire Inc. - một công ty chuyên về managed detection and response 24/7 [15]. Nghiên cứu này chỉ ra cách các chuyên gia SOC thực tế sử dụng LLMs trong công việc hàng ngày.

Kết quả cho thấy các chuyên gia dùng LLMs như một công cụ on-demand để tạo ý nghĩa (sensemaking) và xây dựng ngữ cảnh, thay vì để đưa ra quyết định. Đa số các truy vấn liên quan đến việc giải thích dữ liệu telemetry mức thấp (như các lệnh) và cải thiện giao tiếp kỹ thuật thông qua các tương tác ngắn (1-3 lượt) [15]. Đáng chú ý, 93% truy vấn phù hợp với các năng lực an ninh mạng đã được thiết lập (NICE Framework).

Ngoài ra, nghiên cứu của Deng và cộng sự năm 2024 đã so sánh RAG với các phương pháp truyền thống trong việc phân loại cảnh báo bảo mật. Kết quả cho thấy RAG cải thiện độ chính xác phân loại lên 23% so với các hệ thống dựa trên luật thuần túy. Tuy nhiên, nghiên cứu này chủ yếu tập trung vào môi trường mạng doanh nghiệp quy mô lớn với nguồn lực dồi dào, chưa xem xét khả năng triển khai trên cơ sở hạ tầng hạn chế hoặc sử dụng các mô hình mã nguồn mở. Hơn nữa, nghiên cứu chưa phân tích sâu cách thức tích hợp thông tin từ nhiều nguồn như threat intelligence feeds và tài liệu nội bộ.

Rapid7 đã phát triển hệ thống AI Alert Triage cho nền tảng InsightIDR, đạt độ chính xác 99,93% trong việc phân loại cảnh báo tự động và xử lý 4,8 nghìn tỷ cảnh báo mỗi tuần cho khách hàng MDR (Managed Detection and Response), giúp tiết kiệm hàng trăm giờ công thủ công cho các chuyên gia SOC [16]. Tương tự, nghiên cứu của IBM (2025) cho thấy các tổ chức sử dụng hệ thống bảo mật được hỗ trợ bởi AI có thể giảm thiểu chi phí vi phạm dữ liệu gần 2 triệu USD và rút ngắn thời gian phản ứng 80 ngày [17]. Tuy nhiên, hầu hết các giải pháp thương mại như Rapid7 InsightIDR chủ yếu tập trung vào môi trường có quy mô lớn và nguồn lực dồi dào, chưa xem xét khả năng triển khai trên hạ tầng hạn chế hoặc dùng các mô hình mã nguồn mở chi phí thấp.

Asa và nhóm nghiên cứu đã đề xuất mô hình Self-RAG trong bài báo năm 2023 [18], tập trung vào khả năng tự đánh giá và cải thiện chất lượng câu trả lời thông qua cơ chế phản hồi. Mô hình này đạt được kết quả ấn tượng trong việc giảm thiểu hiện tượng ảo giác của LLM. Tuy nhiên, kiến trúc Self-RAG yêu cầu tài nguyên tính toán đáng kể và chưa được kiểm chứng cụ thể trong lĩnh vực phân tích sự cố bảo mật thời gian thực, nơi yêu cầu về độ trễ thấp là tối quan trọng.

NVIDIA đã phát triển framework Morpheus tích hợp RAG [19] cho phân tích bảo mật, tận dụng khả năng tăng tốc GPU của họ. Giải pháp này cho phép xử lý khối lượng lớn dữ liệu log trong thời gian thực với hiệu năng cao. Nhưng Morpheus yêu cầu đầu tư lớn vào phần cứng chuyên dụng của NVIDIA và khó khăn trong việc tùy biến cho các trường hợp sử dụng đặc thù của từng tổ chức. Chi phí triển khai cũng là rào cản đáng kể đối với các doanh nghiệp vừa và nhỏ.

Dự án Threat Intelligence RAG trên GitHub của CyberScienceLab [20] cung cấp một bằng chứng về việc kết hợp dữ liệu threat intelligence với LLM. Dự án này mang tính

minh họa cao và nói lên tính khả thi của ý tưởng. Tuy nhiên, nó chưa phát triển thành một hệ thống hoàn chỉnh có khả năng tích hợp với các nền tảng SIEM thương mại như Splunk hoặc QRadar. Thiếu vắng cơ chế xử lý cảnh báo tự động và quy trình phản hồi lại hệ thống giám sát cũng là những hạn chế rõ ràng.

Qua khảo sát các nghiên cứu và giải pháp hiện có, có thể nhận thấy một số khoảng trống quan trọng. Thứ nhất, phần lớn các nghiên cứu học thuật tập trung vào việc chứng minh hiệu quả của phương pháp trên tập dữ liệu chuẩn, nhưng ít đi vào chi tiết kiến trúc triển khai thực tế trong môi trường sản xuất. Thứ hai, các giải pháp thương mại thường là các hệ thống đóng, khó tùy biến và có chi phí cao. Thứ ba, các dự án mã nguồn mở chưa đủ khả thi để triển khai thực tế.

Đề tài này nhằm lấp đầy những khoảng trống trên bằng cách xây dựng một kiến trúc RAG hoàn chỉnh, có khả năng tích hợp với hệ thống SIEM phổ biến, sử dụng hoàn toàn công nghệ mã nguồn mở để đảm bảo tính minh bạch và khả năng tùy biến cao. Điểm đặc biệt là đề tài tập trung vào khả năng vận hành trên cơ sở hạ tầng hạn chế bằng cách sử dụng các mô hình LLM cục bộ được tối ưu hóa, đồng thời vẫn duy trì được chất lượng phân tích tốt thông qua cơ chế làm giàu dữ liệu từ nhiều nguồn khác nhau.

1.7. Kết chương

Chương 1 đã cung cấp một cái nhìn tổng quan và nền tảng lý thuyết vững chắc cho đề tài. Chương này đã đi sâu vào phân tích thực trạng của các hệ thống giám sát an ninh mạng hiện nay, chỉ ra những hạn chế của các phương pháp phân tích truyền thống. Đồng thời, chương này cũng đã giới thiệu về LLM và đặc biệt là kiến trúc RAG - giải pháp then chốt để kết hợp sức mạnh của AI tạo sinh với cơ sở dữ liệu chính xác.

CHƯƠNG 2: NGHIÊN CỨU VÀ ỨNG DỤNG MÔ HÌNH RAG TRONG PHÂN TÍCH CẢNH BÁO AN NINH MẠNG

2.1. Thiết kế kiến trúc mô hình RAG trong phân tích cảnh báo an ninh mạng

Dựa trên cơ sở lý thuyết về RAG và thực trạng các hệ thống SOC đã trình bày ở Chương 1, chương này đề xuất một kiến trúc hệ thống tích hợp nhằm tự động hóa quy trình phân tích cảnh báo, tận dụng sức mạnh của các mô hình ngôn ngữ lớn mã nguồn mở và cơ sở dữ liệu vector

2.1.1. Yêu cầu kiến trúc

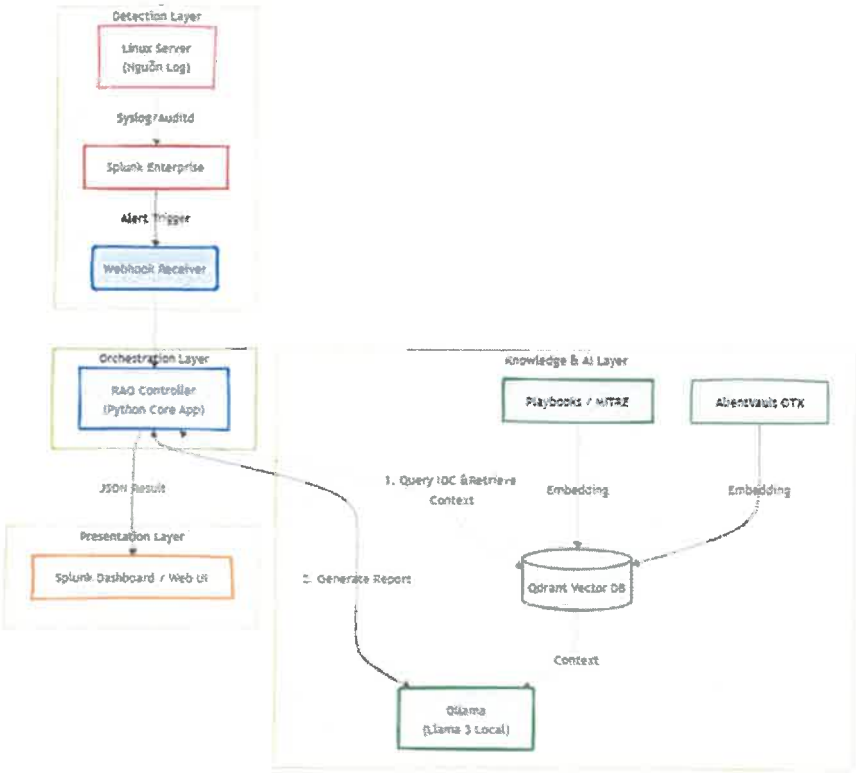
Để đảm bảo hệ thống hoạt động hiệu quả trong môi trường giám sát an ninh mạng thực tế, kiến trúc đề xuất cần thỏa mãn các yêu cầu phi chức năng sau:

a) Tính bảo mật dữ liệu (Data Privacy): Đây là yêu cầu tiên quyết. Các dữ liệu log và chi tiết cảnh báo chứa thông tin nhạy cảm của hạ tầng (IP, User, cấu hình hệ thống) hạn chế gửi đến các API công cộng (như ChatGPT). Do đó, kiến trúc ưu tiên sử dụng các LLM chạy cục bộ (Local LLMs).
b) Tính thời gian thực (Real-time capability): Hệ thống phải có khả năng tiếp nhận và xử lý cảnh báo ngay khi nó được sinh ra từ SIEM. Độ trễ xử lý (Latency) phải nằm trong mức chấp nhận được (ví dụ: ~ 60 giây/cảnh báo) để hỗ trợ phản ứng nhanh.
c) Tính chính xác và có căn cứ (Grounding): Các phân tích do AI sinh ra phải dựa trên các tài liệu quy trình và thông tin tình báo (Threat Intel) thực tế, hạn chế tối đa hiện tượng ảo giác (Hallucination).
d) Khả năng mở rộng (Scalability): Kiến trúc phải được thiết kế dạng module hóa, cho phép dễ dàng thay thế model LLM mới hoặc bổ sung thêm các nguồn dữ liệu mà không làm gián đoạn hệ thống.

2.1.2. Thiết kế kiến trúc

Dựa trên các yêu cầu trên và nền tảng công nghệ đã lựa chọn, kiến trúc tổng thể của hệ thống được thiết kế theo mô hình xử lý sự kiện (Event-driven architecture), bao gồm 4 tầng chính:

a) Sơ đồ kiến trúc tổng thể



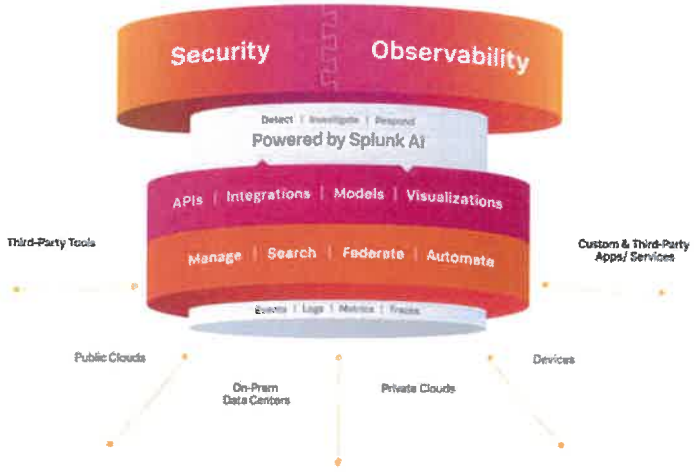
Hình 2.1.2-1: Sơ đồ tổng thể kiến trúc hệ thống RAG

b) Mô tả các thành phần kiến trúc

Kiến trúc của hệ thống được thiết kế theo mô hình phân tầng, trong đó mỗi tầng phụ trách một tập hợp chức năng cụ thể và tương tác với các tầng khác thông qua các định nghĩa rõ ràng. Cách tiếp cận này không chỉ tạo ra sự tách biệt giữa các chức năng khác nhau mà còn cho phép dễ dàng thay thế hoặc nâng cấp từng thành phần mà không ảnh hưởng đến hoạt động hệ thống. Kiến trúc bao gồm bốn tầng chính, mỗi tầng xây dựng dựa trên nền tảng của tầng bên dưới để tạo nên một hệ thống xử lý thông tin hoàn chỉnh.

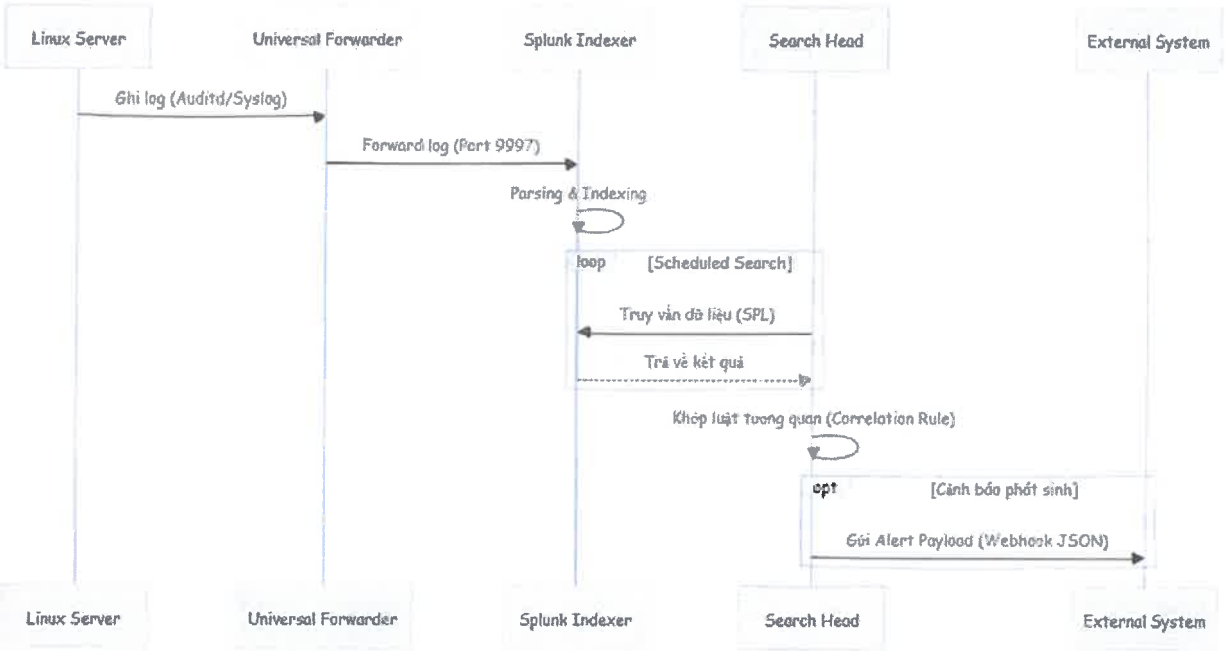
Tầng nền tảng đầu tiên, được gọi là tầng giám sát, đóng vai trò là điểm khởi đầu của toàn bộ quy trình xử lý. Đây là nơi dữ liệu thô được sinh ra và các mối đe dọa lần đầu tiên được phát hiện ở dạng sơ khai. Các máy chủ Linux trong môi trường production đại diện cho lớp tài sản cần được bảo vệ, đồng thời cũng là nguồn phát sinh dữ liệu quan trọng nhất. Trên mỗi máy chủ này, các agent ghi log hệ thống được cấu hình để ghi lại chi tiết mọi hoạt động đáng chú ý. Hệ thống auditd - một framework audit mạnh mẽ được tích hợp sẵn trong nhân Linux, được cấu hình để theo dõi các lệnh hệ thống nhạy cảm, các thay đổi đối với file cấu hình quan trọng, và các hoạt động liên quan đến quyền quản trị. Đồng thời, syslog thu thập các thông điệp từ mọi dịch vụ đang chạy, tạo nên một dòng sự kiện liên tục phản ánh trạng thái của hệ thống. Tất cả dữ liệu log này không được lưu trữ cục bộ lâu dài mà

được chuyển tiếp ngay lập tức đến hệ thống tập trung nhờ Splunk Universal Forwarder - một agent nhẹ được thiết kế để thu thập và truyền tải log với độ trễ tối thiểu và ảnh hưởng rất thấp đến hiệu năng của máy chủ production.



Hình 2.1.2-2: Nền tảng bảo mật Splunk [21]

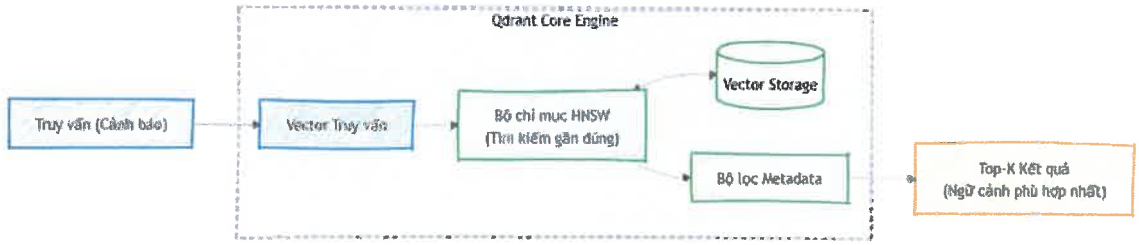
Splunk Enterprise đóng vai trò trung tâm trong tầng này với tư cách là hệ thống quản lý thông tin và sự kiện bảo mật. Nền tảng này không chỉ đơn thuần là một kho lưu trữ log, mà là một công cụ xử lý sự kiện phức tạp với khả năng phân tích thời gian thực [21]. Khi các Forwarder đẩy dữ liệu về, thành phần Indexer của Splunk tiếp nhận, phân tích cú pháp và lập chỉ mục cho từng sự kiện, biến chúng thành các bản ghi có cấu trúc có thể tìm kiếm với tốc độ cao. Splunk áp dụng các phép biến đổi dữ liệu để trích xuất các trường thông tin quan trọng, ánh xạ chúng vào mô hình dữ liệu chung, và làm giàu thêm bằng các metadata như thông tin địa lý của IP hoặc danh tính tài sản. Sau khi dữ liệu được chuẩn hóa, thành phần Search Head bắt đầu thực thi các tìm kiếm đã được lưu - đây chính là nơi các luật tương quan được định nghĩa. Các luật này được viết bằng ngôn ngữ truy vấn đặc thù của Splunk và thực hiện logic phát hiện phức tạp, từ các mẫu đơn giản như đếm số lần xuất hiện của một sự kiện, cho đến các phân tích phức tạp liên quan đến việc xâu chuỗi nhiều loại sự kiện khác nhau theo thời gian và không gian. Khi một luật phát hiện ra hành vi đáng ngờ - ví dụ như mười lần đăng nhập SSH thất bại liên tiếp theo sau bởi một lần thành công - cơ chế cảnh báo được kích hoạt. Splunk không chỉ ghi lại cảnh báo vào database nội bộ mà còn đóng gói tất cả thông tin ngữ cảnh liên quan vào một payload JSON có cấu trúc và gửi nó đến hệ thống phân tích thông qua webhook, một cơ chế callback HTTP cho phép tích hợp thời gian thực giữa các hệ thống.



Hình 2.1.2-3: Mô tả hoạt động của Splunk

Tầng thứ hai, tầng điều phối, đóng vai trò như trung gian thông minh kết nối giữa thế giới giám sát truyền thống và thế giới phân tích dựa trên trí tuệ nhân tạo. Trái tim của tầng này là ứng dụng RAG Controller, một service backend được phát triển bằng Python với framework FastAPI, được thiết kế để hoạt động như một điều phối luồng dữ liệu và logic xử lý. Khi webhook từ Splunk được gửi đến, RAG Controller tiếp nhận request HTTP POST chứa thông tin cảnh báo. Bước đầu tiên trong quy trình xử lý là xác thực nguồn gốc để đảm bảo request thực sự đến từ Splunk server đáng tin cậy chứ không phải từ một nguồn độc hại cố gắng gửi dữ liệu giả vào hệ thống. Sau khi xác thực thành công, controller phân tích cú pháp gói tin JSON để trích xuất các trường thông tin quan trọng. Quá trình này không chỉ đơn giản là đọc các trường có sẵn mà còn bao gồm logic xử lý phức tạp để chuẩn hóa dữ liệu - ví dụ như chuyển đổi timestamp về múi giờ chuẩn, validate và làm sạch các địa chỉ IP, hoặc trích xuất các chỉ báo xâm phạm từ các trường văn bản tự do. Sau khi parsing hoàn tất, controller không xử lý tuần tự mà khởi động song song nhiều luồng công việc: một luồng gửi các chỉ báo xâm phạm đến module làm giàu dữ liệu để tra cứu threat intelligence, một luồng khác chuẩn bị dữ liệu để truy vấn vector databas. Controller sử dụng cơ chế async/await của Python để quản lý các tác vụ bất đồng bộ này hiệu quả, đảm bảo hệ thống có thể xử lý nhiều cảnh báo đồng thời mà không bị block. Sau khi tất cả các module phụ trợ trả về kết quả, controller tổng hợp chúng lại, xây dựng một ngữ cảnh đầy đủ, và chuyển tiếp đến tầng xử lý AI để phân tích cuối cùng.

Tầng thứ ba, tầng dữ liệu và trí tuệ nhân tạo, là bộ não của toàn bộ hệ thống, nơi kiến thức được lưu trữ và suy luận được thực hiện. Thành phần đầu tiên và quan trọng nhất là cơ sở dữ liệu vector Qdrant, một hệ quản trị cơ sở dữ liệu chuyên biệt được tối ưu hóa cho việc lưu trữ và tìm kiếm vector embedding [22]. Khác với các cơ sở dữ liệu quan hệ truyền thống lưu trữ dữ liệu dạng bảng với các khóa và giá trị, Qdrant lưu trữ các vector - những mảng số thực đa chiều đại diện cho ý nghĩa ngữ nghĩa của văn bản. Qdrant đóng vai trò như bộ nhớ dài hạn của hệ thống, chứa đựng toàn bộ kiến thức mà mô hình ngôn ngữ cần để đưa ra phân tích chính xác. Kiến thức này được tổ chức thành nhiều collection khác nhau, mỗi collection phục vụ một mục đích riêng. Collection đầu tiên chứa các tài liệu nội bộ của tổ chức, bao gồm các quy trình xử lý sự cố được chuẩn hóa, các hướng dẫn mô tả từng bước cần thực hiện khi đối mặt với từng loại tấn công cụ thể, các chính sách bảo mật quy định ranh giới cho phép và không cho phép, cùng với các case study về các sự cố đã xảy ra trước đó và cách chúng được giải quyết. Collection thứ hai chứa dữ liệu từ các nguồn threat intelligence bên ngoài, đặc biệt là các chỉ báo xâm phạm từ AlienVault OTX - những địa chỉ IP đã được báo cáo trong các hoạt động độc hại, mã hash của file mã độc, tên miền được sử dụng trong các chiến dịch phishing, cùng với các mô tả chi tiết về các chiến thuật, kỹ thuật và quy trình của kẻ tấn công theo framework MITRE ATT&CK [23]. Mỗi đoạn văn bản trong các tài liệu này đều được chuyển đổi thành vector embedding thông qua các mô hình như all-MiniLM-L6-v2, và những vector này được đánh chỉ mục trong Qdrant cùng với metadata liên quan như nguồn gốc, timestamp và độ tin cậy.



Hình 2.1.2-4: Cơ chế Tìm kiếm Vector trong Qdrant

Cơ chế hoạt động của Qdrant dựa trên khái niệm tìm kiếm tương đồng trong không gian vector. Khi RAG Controller gửi một truy vấn - ví dụ như một mô tả về cảnh báo đang được phân tích - truy vấn này cũng được chuyển đổi thành vector bằng cùng mô hình embedding. Qdrant sau đó thực hiện một phép tính toán học phức tạp, so sánh vector truy vấn này với hàng triệu vector đã được lưu trữ để tìm ra những vector "gần nhất" trong không gian đa chiều. Khái niệm "gần" ở đây được định nghĩa bằng các metric như cosine similarity - hai vector gần nhau trong không gian này có nghĩa là hai đoạn văn bản tương ứng có ý nghĩa ngữ nghĩa tương tự nhau, ngay cả khi chúng không chia sẻ bất kỳ từ nào giống nhau. Ví dụ, một cảnh báo về "nhiều lần thử đăng nhập SSH thất bại" sẽ có vector

gần với các tài liệu về "tấn công brute force" hoặc "credential stuffing", mặc dù các thuật ngữ sử dụng khác nhau. Qdrant được thiết kế với các cấu trúc dữ liệu đặc biệt như HNSW graph để có thể thực hiện tìm kiếm này với độ phức tạp logarithmic, cho phép trả về kết quả trong vòng vài millisecond ngay cả khi tìm kiếm trong hàng triệu vector. Sau khi tìm được top-K vector gần nhất (thường là 3-5 kết quả), Qdrant trả về không chỉ vector mà còn cả văn bản gốc và metadata đi kèm, cung cấp ngữ cảnh đầy đủ cho mô hình ngôn ngữ.

Thành phần thứ hai trong tầng này là LLM, được vận hành thông qua nền tảng Ollama. Ollama là một framework mã nguồn mở được thiết kế để đơn giản hóa việc triển khai và chạy các mô hình ngôn ngữ lớn trên máy chủ cục bộ, loại bỏ sự phụ thuộc vào các dịch vụ cloud bên ngoài. Trong hệ thống của chúng ta, Ollama đang vận hành mô hình Llama 3, một trong những mô hình ngôn ngữ mã nguồn mở tiên tiến nhất hiện nay với khả năng hiểu ngữ cảnh phức tạp và tạo ra văn bản chất lượng cao. Vai trò của LLM trong kiến trúc RAG không phải là ghi nhớ kiến thức - đây là sai lầm phổ biến khi hiểu về các hệ thống này - mà là đóng vai trò như một engine suy luận và tổng hợp thông tin. Khi Qdrant trả về các đoạn văn bản liên quan, RAG Controller xây dựng một prompt có cấu trúc, bắt đầu bằng việc thiết lập vai trò và nhiệm vụ của mô hình: "Bạn là một chuyên gia phân tích bảo mật. Nhiệm vụ của bạn là phân tích cảnh báo sau đây và đưa ra đánh giá." Tiếp theo, prompt bao gồm tất cả ngữ cảnh đã thu thập được: log thô từ cảnh báo gốc, thông tin reputation về IP từ OTX, đoạn trích từ playbook xử lý loại tấn công tương ứng, và có thể cả các case study về sự cố tương tự trước đó. Cuối cùng, prompt đưa ra câu hỏi hoặc yêu cầu cụ thể: "Dựa trên thông tin trên, hãy phân tích mức độ nghiêm trọng và đề xuất các bước xử lý."

LLM xử lý prompt này bằng cách sử dụng cơ chế attention để xác định mối quan hệ giữa các phần thông tin khác nhau, sau đó sinh ra một phản hồi bằng cách dự đoán từng token một theo trình tự. Tuy nhiên, khác với khi LLM hoạt động độc lập và có thể "ảo giác", trong kiến trúc RAG này mô hình được "neo" vào các thông tin thực tế được cung cấp. Prompt được thiết kế một cách cẩn thận với các chỉ dẫn rõ ràng: "Chỉ sử dụng thông tin từ các tài liệu được cung cấp", "Nếu thông tin không đủ để trả lời, hãy nói rõ", "Trích dẫn nguồn cho mỗi khẳng định". Những ràng buộc này giảm đáng kể khả năng mô hình tự bịa đặt thông tin. Kết quả mà LLM sinh ra không phải là một đoạn văn ngắn gọn mà là một báo cáo phân tích có cấu trúc, bao gồm phần tóm tắt executive giải thích bản chất của sự cố bằng ngôn ngữ không chuyên môn, phần phân tích kỹ thuật đi sâu vào các chi tiết về cách thức tấn công hoạt động, đánh giá mức độ nghiêm trọng dựa trên các yếu tố như giá trị tài sản bị ảnh hưởng và khả năng lan rộng, và quan trọng nhất là danh sách các bước khuyến nghị xử lý được trích xuất trực tiếp từ playbook nội bộ.

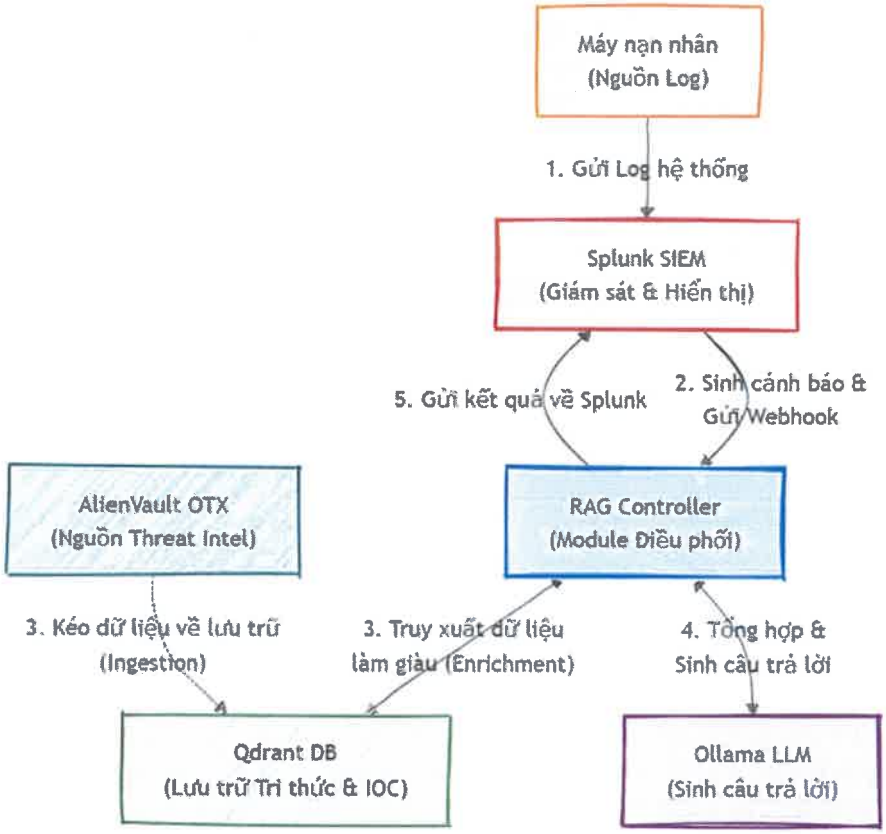
Thành phần cuối cùng trong tầng này là các nguồn dữ liệu bên ngoài, đặc biệt là AlienVault OTX, đóng vai trò như nguồn cấp dữ liệu liên tục cho hệ thống. OTX là một nền tảng chia sẻ threat intelligence cộng đồng, nơi các tổ chức và nhà nghiên cứu bảo mật trên toàn thế giới đóng góp thông tin về các mối đe dọa họ phát hiện. Hệ thống của chúng ta không truy cập OTX theo cách thủ công mà thông qua một quy trình tự động chạy ngầm được gọi là ingestion pipeline. Pipeline này hoạt động theo lịch trình định kỳ, ví dụ mỗi sáu giờ một lần, để tải về các pulse mới nhất - những bộ sưu tập chỉ báo xâm phạm được cập nhật liên tục. Sau khi tải về, pipeline thực hiện một loạt các bước xử lý: trích xuất các IOC từ định dạng JSON của OTX, chuẩn hóa các địa chỉ IP và domain về format nhất quán, lọc bỏ các chỉ báo đã hết hạn hoặc có độ tin cậy thấp, làm giàu thêm bằng cách tra cứu thông tin bổ sung từ các nguồn khác, chuyển đổi mô tả văn bản thành vector embedding, và cuối cùng cập nhật vào Qdrant thông qua các thao tác upsert - insert nếu chưa tồn tại, update nếu đã có. Quy trình này đảm bảo rằng cơ sở kiến thức của hệ thống luôn được cập nhật với thông tin mới nhất về các mối đe dọa đang hoạt động, một yếu tố cực kỳ quan trọng trong lĩnh vực an ninh mạng nơi bối cảnh đe dọa thay đổi hàng ngày.

Tầng cao nhất và là điểm kết thúc của chu trình xử lý là tầng tương tác, nơi kết quả phân tích được trình bày cho người dùng cuối - các chuyên gia vận hành SOC. Thiết kế của tầng này tuân theo nguyên tắc mang thông tin đến nơi người dùng đang làm việc thay vì buộc họ phải chuyển đổi giữa nhiều công cụ khác nhau. Sau khi LLM hoàn thành việc sinh ra báo cáo phân tích, RAG Controller không chỉ đơn giản trả về response mà còn thực hiện bước quan trọng là gửi ngược kết quả về Splunk, tạo nên một vòng lặp phản hồi khép kín. Việc này được thực hiện thông qua Splunk HTTP Event Collector, một dịch vụ cho phép các ứng dụng bên ngoài gửi dữ liệu vào Splunk như thể chúng là các sự kiện log thông thường. Báo cáo phân tích được đóng gói dưới dạng một sự kiện đặc biệt với sourcetype riêng, chứa đầy đủ thông tin: ID của cảnh báo gốc để có thể liên kết, toàn bộ nội dung phân tích, các tag về mức độ nghiêm trọng và loại tấn công, cùng với timestamp. Khi dữ liệu này được index trong Splunk, nó trở nên có thể tìm kiếm và tương quan với các sự kiện khác. Các dashboard được thiết kế đặc biệt sẽ hiển thị các cảnh báo cùng với phân tích AI tương ứng trong cùng một view, cho phép chuyên gia phân tích xem log gốc bên cạnh với đánh giá của AI để so sánh và xác minh. Họ có thể drill down vào các chi tiết, theo dõi timeline của sự cố, và quan trọng nhất là có thể hành động ngay lập tức dựa trên các khuyến nghị cụ thể mà hệ thống đưa ra. Interface này cũng cung cấp các control cho phép người dùng đánh giá chất lượng phân tích của AI - một cơ chế phản hồi có thể được sử dụng sau này để cải thiện hệ thống.

2.2. Mô tả luồng dữ liệu và định nghĩa các chức năng của mô hình

2.2.1. Mô tả luồng dữ liệu

Luồng xử lý của hệ thống được thực hiện tuần tự qua 5 bước, đảm bảo thông tin đầu vào cho LLM là đầy đủ và chính xác nhất.



Hình 2.2.1-1: Sơ đồ mô tả luồng dữ liệu

a) Bước 1: Nhận cảnh báo (Alert Ingestion)

- Hệ thống SIEM (Splunk) sau khi phát hiện các hành vi bất thường (như Brute Force SSH, Malware detection) sẽ kích hoạt cơ chế gửi thông báo.
- Dữ liệu cảnh báo được đóng gói dưới dạng bản tin JSON chứa các trường thông tin thô (src_ip, user, timestamp, raw_log) và được đẩy (Push) tới Endpoint của hệ thống RAG qua giao thức HTTP POST (Webhook).

b) Bước 2: Tiếp nhận và Chuẩn hóa (Parsing & Normalization)

- Module điều phối (RAG Controller) tiếp nhận yêu cầu từ Webhook.

- Hệ thống thực hiện xác thực nguồn gốc (để đảm bảo an toàn) và phân tích cú pháp (Parse) gói tin JSON.
- Các chỉ báo tấn công IOC quan trọng như địa chỉ IP nguồn, mã băm (Hash) tệp tin, tên miền độc hại được trích xuất và chuẩn hóa về định dạng danh sách (List of Dictionaries) để phục vụ các bước xử lý sau.

c) Bước 3: Làm giàu dữ liệu (Data Enrichment)

- Từ các IOC đã trích xuất, hệ thống tự động kết nối với các nguồn dữ liệu tình báo mối đe dọa (Threat Intelligence) như AlienVault OTX.
- Mục tiêu là thu thập thêm ngữ cảnh về đối tượng tấn công: IP này có nằm trong danh sách đen (Blacklist) không? Mã băm này thuộc về dòng mã độc nào? Mức độ uy tín (Reputation Score) là bao nhiêu?

d) Bước 4: Xử lý và Phân tích (Processing & RAG Analysis)

- Nhúng dữ liệu (Embedding): Nội dung cảnh báo và thông tin làm giàu được chuyển đổi thành vector số học sử dụng các mô hình ngôn ngữ nhỏ (như all-MiniLM-L6-v2, all-mpnet-base-v2).
- Truy xuất dữ liệu (Retrieval): Hệ thống thực hiện truy vấn tương đồng (Similarity Search) trong cơ sở dữ liệu vector (Qdrant) để tìm kiếm các quy trình xử lý hoặc kỹ thuật tấn công tương ứng với vector cảnh báo.
- Tổng hợp (Synthesis): LLM (Ollama) tiếp nhận toàn bộ ngữ cảnh (Log thô + Thông tin OTX + dữ liệu nội bộ) để thực hiện suy luận logic.

e) Bước 5: Trả kết quả (Response & Reporting)

- Mô hình sinh ra báo cáo phân tích ngôn ngữ tự nhiên, bao gồm: Tóm tắt sự cố, Đánh giá mức độ nghiêm trọng, và Hướng dẫn xử lý từng bước.
- Kết quả này được trả về cho người dùng qua giao diện Dashboard hoặc cập nhật ngược lại vào hệ thống quản lý sự cố (Ticketing System)

2.2.2. Định nghĩa các chức năng của mô hình

Hệ thống được thiết kế theo dạng chức năng (module) hóa để đảm bảo tính linh hoạt và khả năng mở rộng, bao gồm các thành phần xử lý chính sau:

a) Chức năng Thu thập và Làm giàu IOC (Data Ingestion & Enrichment):

Thành phần này chịu trách nhiệm thiết lập kênh giao tiếp API với các nền tảng chia sẻ thông tin mối đe dọa (Threat Intelligence), ví dụ như AlienVault OTX. Chức năng chính của module là hoạt động như một bộ lọc thông minh, tự động tải về các chỉ số xâm phạm (IOC) quan trọng theo lịch trình hoặc yêu cầu.

Giải pháp kỹ thuật : Sử dụng thư viện requests trong Python để gọi API OTX và xử lý dữ liệu trả về.

```
# Lấy IOC từ OTX
import requests
def fetch_otx_indicators(api_key):
    headers = {'X-OTX-API-KEY': api_key}
    response = requests.get("[https://otx.alienvault.com/api/v1/pulses/{pid}]
    return response.text
```

b) Chức năng Xử lý và Nhúng dữ liệu (Data Processing & Embedding):

Hệ thống phải có khả năng chuyển đổi dữ liệu văn bản (mô tả cảnh báo, thông tin IOC) thành dạng vector số học (Embeddings). Đồng thời hệ thống phải hỗ trợ các mô hình ngôn ngữ đa dạng (Tiếng Anh và Tiếng Việt) để tối ưu hóa khả năng hiểu ngữ nghĩa.

Giải pháp kỹ thuật: Sử dụng sentence-transformers để mã hóa văn bản.

```
from sentence_transformers import SentenceTransformer
# Khởi tạo model embedding nhẹ và nhanh
model = SentenceTransformer('all-MiniLM-L6-v2')
def embed_ioc_data(parsed_data):
 # Kết hợp Indicator và Description để tạo ngữ cảnh đầy đủ
 return [model.encode(f"{x['indicator']} {x['description']}") for x in parsed_data]
```

c) Chức năng Lưu trữ và Truy xuất dữ liệu (Storage & Retrieval):

Hệ thống phải lưu trữ được hàng triệu vector chỉ báo với tốc độ truy xuất mili-giây. Hệ thống phải cung cấp khả năng tìm kiếm tương đồng (Similarity Search) để tìm ra các IOC hoặc tài liệu nội bộ liên quan nhất đến cảnh báo hiện tại.

Giải pháp kỹ thuật: Sử dụng Qdrant làm Vector Database

```
from qdrant_client import QdrantClient
from qdrant_client.models import PointStruct
client = QdrantClient(":memory:") # Chạy in-memory cho thử nghiệm
def index_vectors(embeddings, metadata):
  points = [
    PointStruct(id=idx, vector=vec.tolist(), payload=meta)
    for idx, (meta, vec) in enumerate(zip(metadata, embeddings))
  ]
  client.upsert(collection_name="secops_kb", points=points)
```

d) Chức năng Phân tích và Sinh báo cáo (Analysis & Generation):

Hệ thống phải tích hợp được với các LLM chạy cục bộ (như Llama 3) thông qua Ollama. Hệ thống phải thực hiện được quy trình RAG: Truy vấn dữ liệu liên quan -> Tạo Prompt -> Sinh câu trả lời tự động dựa trên dữ liệu đã truy xuất.

Giải pháp kỹ thuật : Sử dụng LlamaIndex hoặc LangChain để điều phối quy trình RAG

```
from llama_index.core import VectorStoreIndex, Document
from llama_index.llms.ollama import Ollama
# 1. Truy xuất dữ liệu liên quan từ Vector Store
index = VectorStoreIndex.from_documents(documents)
retriever = index.as_retriever(similarity_top_k=3)
# 2. Kết hợp với LLM để sinh báo cáo
llm = Ollama(model="llama3", request_timeout=120.0)
def generate_analysis(query):
    context = retriever.retrieve(query)
    prompt = f"Dựa trên dữ liệu: {context}. Hãy phân tích cảnh báo: {query}"
    return llm.complete(prompt)
```

2.3. Triển khai mô hình trong môi trường thử nghiệm

2.3.1. Chuẩn bị môi trường và các thành phần thử nghiệm

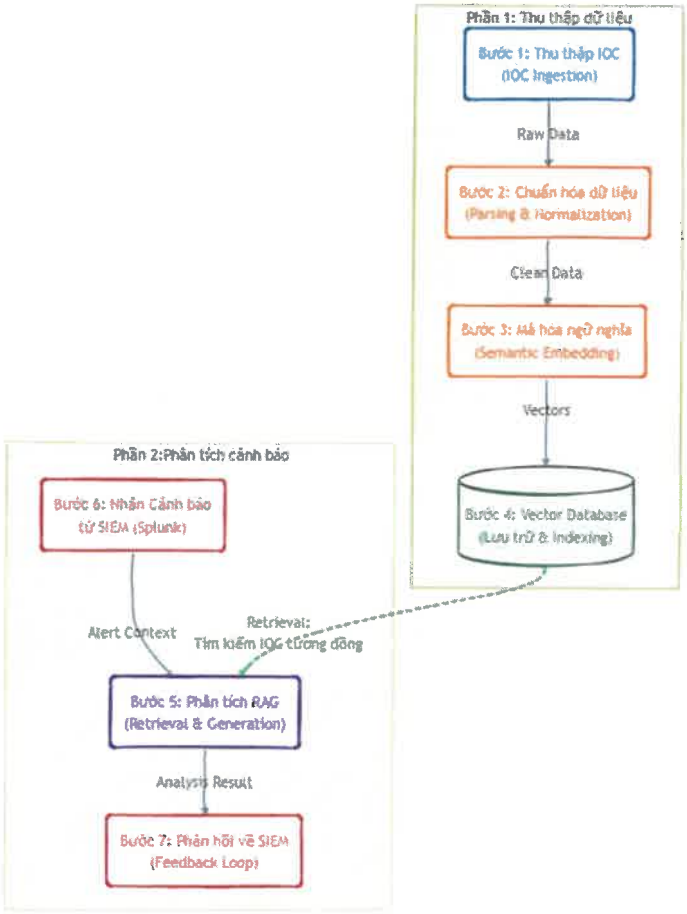
Môi trường thử nghiệm được thiết kế gồm 3 máy chủ độc lập, mô phỏng quy trình tấn công và phòng thủ thực tế.

- Máy chủ 1: Máy nạn nhân
    - Vai trò: Là đối tượng bị tấn công trong kịch bản thử nghiệm.
    - Hệ điều hành: Ubuntu Server 22.04 LTS.
    - Cấu hình giám sát:
        - Cài đặt auditd để ghi nhận các lệnh hệ thống.
        - Cài đặt Splunk Universal Forwarder: Cấu hình để đẩy log (/var/log/auth.log, /var/log/syslog) về máy chủ SIEM.
- Máy chủ 2: Máy chủ SIEM (Splunk Server)
    - Vai trò: Trung tâm thu thập log và phát hiện tấn công.

- Phần mềm: Splunk Enterprise 9.4.1.
- Chức năng:
  - Indexer: Nhận log từ máy nạn nhân và lưu trữ.
  - Search Head: Chạy các luật tìm kiếm (Saved Searches) để phát hiện hành vi bất thường.
  - Alerting: Khi luật được kích hoạt, gửi Webhook chứa thông tin cảnh báo sang máy chủ RAG.

- Máy chủ 3: Máy chủ RAG (Analysis Server)
  - Vai trò: Tiếp nhận cảnh báo và thực hiện phân tích.
  - Cấu hình phần cứng: Yêu cầu RAM tối thiểu 16GB (để chạy mô hình LLM).
  - Phần mềm & Công cụ:
    - RAG Controller: Ứng dụng Python (Flask/FastAPI) để nhận Webhook từ Splunk.
    - Ollama: Nền tảng chạy Local LLM (model llama3 hoặc mistral).
    - Qdrant: Cơ sở dữ liệu Vector lưu trữ.

2.3.2. Các bước triển khai



Hình 2.3.2-1: Sơ đồ các bước triển khai mô hình

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.
    1. Truy cập Settings > Searches, Reports, and Alerts.
    2. Thiết lập hành động kích hoạt (Trigger Action) là Webhook.
    3. Định tuyến tới địa chỉ API của Máy chủ RAG: http://ip_rag_server:5000/api/analyze.
    4. Payload gửi đi sẽ bao gồm toàn bộ thông tin thô của sự kiện log

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).

2.4. Kết chương

Chương 2 đã trình bày chi tiết về thiết kế và phương án triển khai hệ thống RAG cho phân tích cảnh báo an ninh mạng. Kiến trúc được đề xuất đảm bảo tính bảo mật dữ liệu nhờ sử dụng Local LLM (Ollama) và vector DB nội bộ (Qdrant), đồng thời vẫn tận dụng được nguồn dữ liệu bên ngoài qua OTX. Các luồng dữ liệu và chức năng module đã được định nghĩa rõ rang làm cơ sở thực hiện các kịch bản thử nghiệm và đánh giá hiệu quả sẽ được trình bày trong Chương 3.

CHƯƠNG 3: THỬ NGHIỆM VÀ ĐÁNH GIÁ MÔ HÌNH

3.1. Mô tả mô hình thử nghiệm

Để kiểm chứng tính khả thi và hiệu quả của giải pháp đề xuất, một môi trường thử nghiệm (Test) được xây dựng mô phỏng lại hệ thống mạng của một doanh nghiệp vừa và nhỏ, nơi các cuộc tấn công mạng diễn ra và được giám sát, xử lý theo thời gian thực

3.1.1. Sơ đồ thử nghiệm

Mô hình thử nghiệm được triển khai trên nền tảng ảo hóa (VMware ESXi) bao gồm 04 thành phần chính được kết nối trong cùng một mạng nội bộ (LAN) ảo.

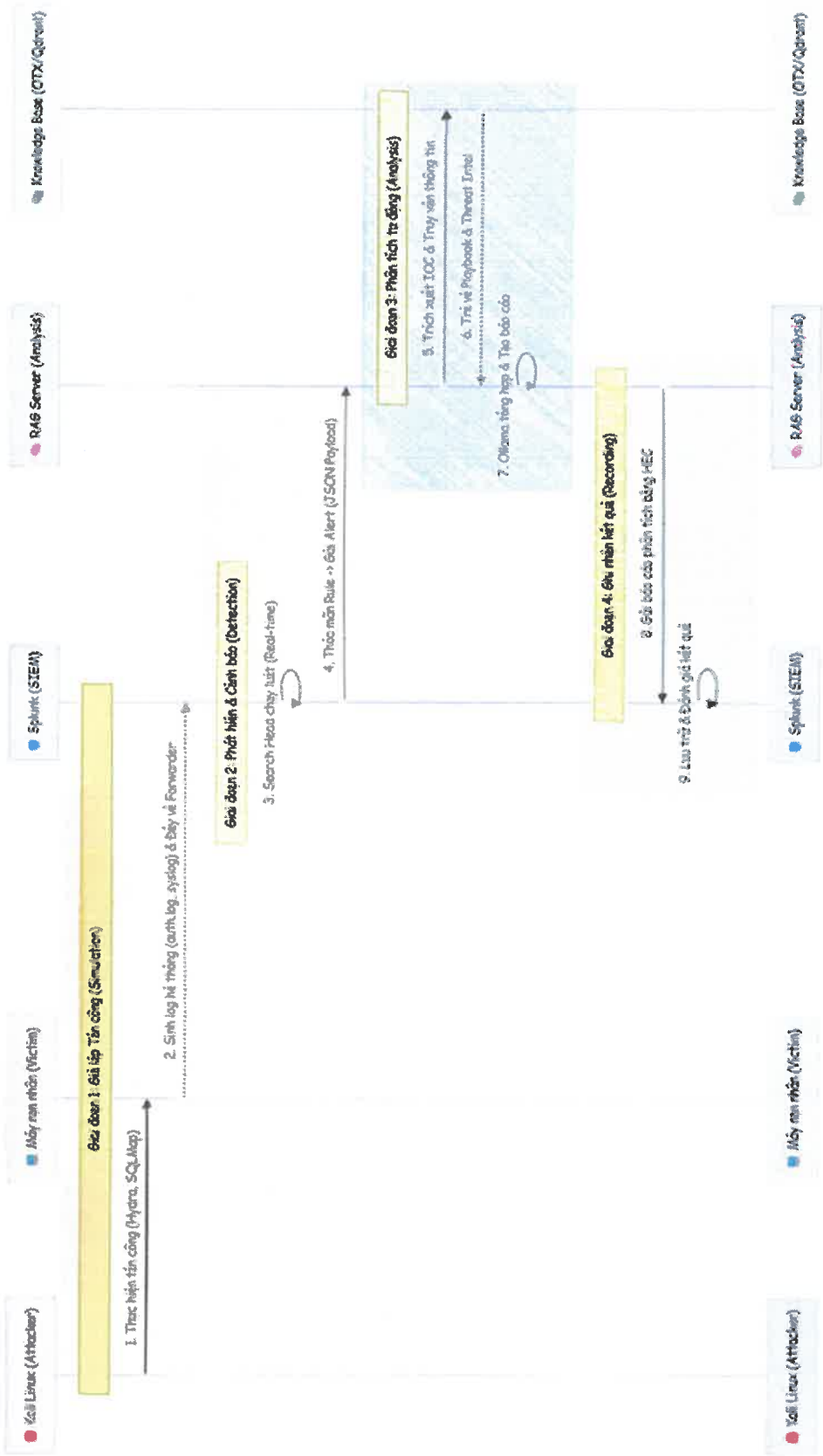


Hình 3.1.1-1: Sơ đồ các thành phần thử nghiệm

- Kẻ tấn công: Sử dụng Kali Linux, đóng vai trò là nguồn phát sinh các cuộc tấn công.
- Máy nạn nhân: Máy chủ Ubuntu Server 22.04 chạy các dịch vụ web (Apache/Nginx) và SSH. Máy này được cài đặt Splunk Universal Forwarder để đẩy log về SIEM.
- Hệ thống giám sát (SIEM Server): Chạy Splunk Enterprise, chịu trách nhiệm thu thập log, chuẩn hóa và phát hiện tấn công.
- Hệ thống phân tích (RAG Analysis Server): Chạy ứng dụng Python tích hợp LLM (ví dụ: Ollama) và Vector DB (ví dụ: Qdrant) để tiếp nhận cảnh báo từ SIEM và thực hiện phân tích

3.1.2. Quy trình thử nghiệm

Quá trình thử nghiệm được thực hiện lặp lại cho từng kịch bản tấn công theo 4 giai đoạn:



Hình 3.1.2-1: Sơ đồ quy trình thử nghiệm

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.
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.

3.1.3. Các chỉ số đánh giá

Hiệu quả hoạt động của mô hình đề xuất được đánh giá một cách toàn diện dựa trên hai khía cạnh chính là hiệu năng hệ thống (Performance) và chất lượng nội dung phân tích (Quality). Đối với nhóm chỉ số hiệu năng, thước đo quan trọng hàng đầu là Thời gian phản hồi trung bình (Mean Time To Response - MTTR), một chỉ số tiêu chuẩn trong quy trình xử lý sự cố an ninh mạng theo định nghĩa của NIST [24]. Trong phạm vi nghiên cứu này, MTTR được tính toán bằng thời gian chênh lệch từ khi hệ thống SIEM (Splunk) kích hoạt tín hiệu cảnh báo đến khi RAG Server hoàn tất quy trình phân tích và trả về kết quả cuối cùng. Bên cạnh yếu tố tốc độ, mức độ tiêu thụ tài nguyên phần cứng cũng được ghi nhận thông qua các thông số sử dụng CPU và RAM của RAG Server trong quá trình vận hành. Việc đo lường này nhằm đảm bảo tính khả thi và ổn định của hệ thống khi triển khai trên

các hạ tầng máy chủ có cấu hình giới hạn, tránh gây ảnh hưởng đến hiệu suất chung của toàn bộ hệ thống giám sát.

Song song với các chỉ số kỹ thuật, chất lượng nội dung do AI sinh ra được kiểm định nghiêm ngặt bởi các chuyên gia dựa trên khung đánh giá RAGAS dành cho các hệ thống thế hệ mới [25]. Tiêu chí đầu tiên được xem xét là Độ chính xác truy xuất (Retrieval Precision), phản ánh khả năng của hệ thống trong việc tìm kiếm các tài liệu từ cơ sở tri thức (Knowledge Base) thực sự liên quan đến ngữ cảnh của cảnh báo, đảm bảo giảm thiểu các thông tin nhiễu đầu vào. Tiếp theo, Độ tin cậy (Faithfulness) đóng vai trò lớn trong việc kiểm soát rủi ro về ảo giác (Hallucination) – một vấn đề cố hữu của các LLM đã được Ji và cộng sự cảnh báo trong các khảo sát gần đây [26]. Chỉ số này đảm bảo rằng các phân tích và kết luận của mô hình được xây dựng dựa trên dữ liệu ngữ cảnh được cung cấp mà không chứa các thông tin bịa đặt hoặc suy diễn sai lệch. Cuối cùng, Tính khả dụng (Actionability) được đánh giá dựa trên mức độ thực tiễn của các khuyến nghị xử lý [27], xác định xem các quy trình ứng phó do AI đề xuất có tuân thủ quy chuẩn an toàn thông tin và có thể thực thi ngay lập tức để ngăn chặn mối đe dọa hay không

3.2. Kịch bản thử nghiệm

Để đánh giá toàn diện, đề tài xây dựng 03 kịch bản tấn công phổ biến nhất nhắm vào máy chủ Linux.

3.2.1. Kịch bản 1: Tấn công dò mật khẩu SSH (SSH Brute Force)

Tấn công dò mật khẩu SSH là một trong những vector tấn công phổ biến nhất nhắm vào các máy chủ Linux được kết nối Internet. Dịch vụ SSH, mặc dù là giao thức quản trị từ xa được mã hóa và an toàn, vẫn trở thành mục tiêu hấp dẫn do tính chất đặc quyền của nó. Một khi kẻ tấn công chiếm được quyền truy cập SSH, chúng có thể thực hiện mọi thao tác quản trị trên hệ thống, từ đánh cắp dữ liệu nhạy cảm, cài đặt backdoor, cho đến sử dụng máy chủ làm bàn đạp cho các cuộc tấn công tiếp theo.

Phương pháp tấn công brute force dựa trên việc thử nghiệm có hệ thống một số lượng lớn các tổ hợp tên đăng nhập và mật khẩu. Kẻ tấn công thường bắt đầu với các tài khoản có tên phổ biến như root, admin, user hoặc ubuntu, sau đó thử nghiệm các mật khẩu từ các từ điển được biên soạn sẵn. Các từ điển này có thể chứa hàng triệu mật khẩu thông dụng được thu thập từ các vụ rò rỉ dữ liệu trước đó. Tính tự động hóa cao của phương pháp này cho phép kẻ tấn công thử nghiệm hàng nghìn lần trong thời gian ngắn, đặc biệt khi tấn công phân tán từ nhiều địa chỉ IP để tránh bị phát hiện.

Trong kịch bản thử nghiệm này, vai trò kẻ tấn công được đảm nhận bởi máy chạy Kali Linux, một bản phân phối được thiết kế đặc biệt cho kiểm thử bảo mật. Công cụ được lựa chọn là Hydra, một trong những công cụ tấn công brute force mạnh mẽ và linh hoạt. Hydra hỗ trợ nhiều giao thức khác nhau và có khả năng tùy chỉnh cao về số luồng song song, độ trễ giữa các lần thử và khả năng tiếp tục từ điểm dừng.

Máy chủ mục tiêu là một máy Ubuntu Server chạy dịch vụ OpenSSH với cấu hình mặc định. Máy chủ này đã được cài đặt Splunk Universal Forwarder để đẩy log xác thực từ tệp /var/log/auth.log về hệ thống SIEM trung tâm theo thời gian thực. Cấu hình này mô phỏng một máy chủ production điển hình trong môi trường doanh nghiệp, nơi việc giám sát log tập trung là yêu cầu bắt buộc.

Giai đoạn trinh sát bắt đầu bằng việc xác định xem dịch vụ SSH có đang mở trên máy mục tiêu hay không. Kẻ tấn công sử dụng công cụ quét cổng để xác nhận cổng 22 đang ở trạng thái listening. Sau khi xác định được mục tiêu, công cụ Hydra được khởi động với cấu hình tấn công tài khoản root sử dụng từ điển mật khẩu phổ biến. Số luồng song song được thiết lập ở mức vừa phải để tránh làm quá tải dịch vụ SSH, điều có thể dẫn đến việc kích hoạt các cơ chế bảo vệ tự động của hệ điều hành.

Trong quá trình tấn công, Hydra gửi liên tiếp các yêu cầu xác thực SSH với các mật khẩu khác nhau. Mỗi lần thử đăng nhập thất bại, hệ thống Linux sẽ ghi lại một dòng log với thông điệp "Failed password for root from [IP_address]" kèm theo timestamp chính xác. Nếu tấn công thành công, một thông điệp "Accepted password for root" sẽ được ghi nhận, cho phép kẻ tấn công thiết lập phiên SSH hợp lệ



Hình 3.1.3-1: Log Brute Force trên Splunk

Hoạt động brute force để lại những dấu vết đặc trưng rất dễ nhận biết trong log hệ thống. Tệp /var/log/auth.log sẽ chứa một chuỗi dài các sự kiện xác thực thất bại liên tiếp từ cùng một địa chỉ IP nguồn trong khoảng thời gian ngắn. Tần suất xuất hiện của các sự kiện này thường cao bất thường, có thể lên tới hàng chục hoặc hàng trăm lần trong vòng vài phút, hoàn toàn khác biệt so với hành vi đăng nhập bình thường của người dùng thật.

Ngoài ra, các lần thử thường nhắm vào các tên tài khoản hệ thống phổ biến, một dấu hiệu rõ ràng của tấn công tự động.

Trên hệ thống giám sát SIEM, Splunk Enterprise nhận được luồng sự kiện này theo thời gian thực. Một correlation rule đã được định nghĩa trước sẽ theo dõi số lượng sự kiện xác thực thất bại từ cùng một IP nguồn trong cửa sổ thời gian nhất định. Khi ngưỡng được thiết lập (ví dụ: 10 lần thất bại trong 60 giây) bị vi phạm, rule sẽ kích hoạt và tạo ra một bản tin cảnh báo mức độ nghiêm trọng cao. Bản tin này không chỉ chứa thông tin về số lượng lần thử và địa chỉ IP nguồn, mà còn bao gồm các đoạn log thô để phục vụ điều tra chi tiết.



Hình 3.1.3-2: Kết quả chạy Rule Brute Force trên Splunk

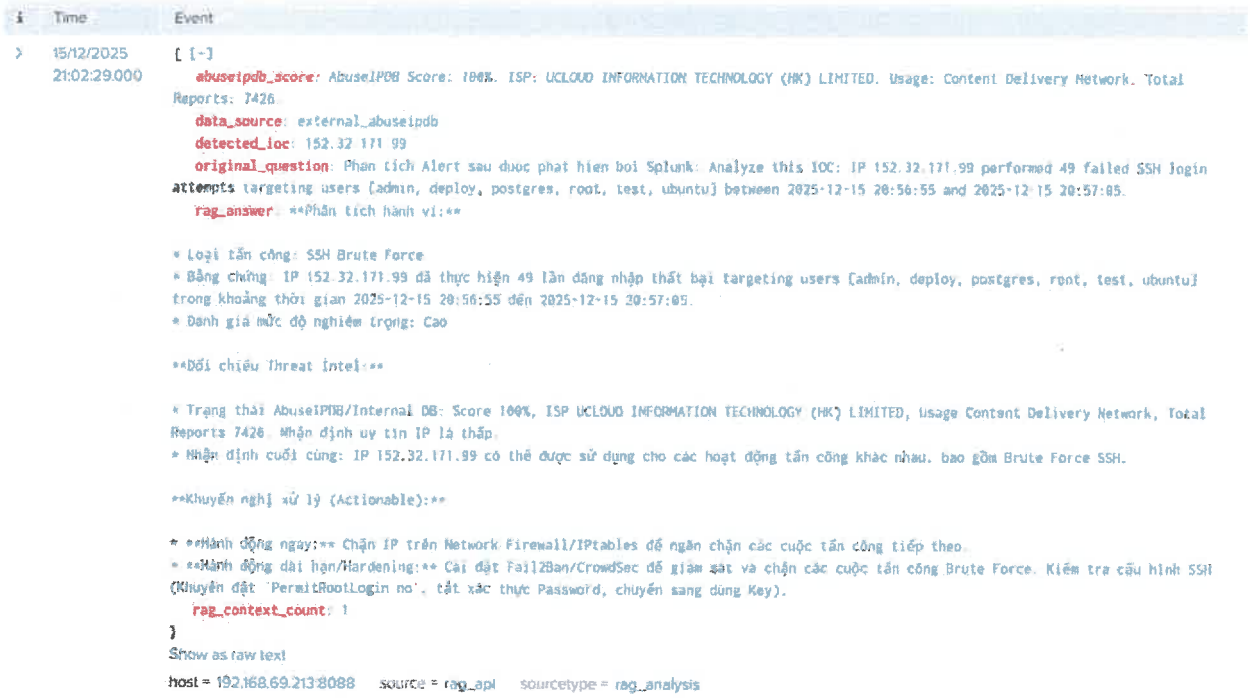
Khi hệ thống RAG nhận được cảnh báo brute force SSH từ SIEM, quá trình phân tích tự động sẽ được khởi động. Giai đoạn đầu tiên là trích xuất các chỉ báo xâm phạm quan trọng, trong trường hợp này là địa chỉ IP nguồn của kẻ tấn công và tài khoản đích bị nhắm tới. Địa chỉ IP này sẽ được gửi đến module làm giàu dữ liệu, nơi hệ thống tự động truy vấn các nguồn threat intelligence như AlienVault OTX hoặc AbuseIPDB để kiểm tra reputation của IP.

Nếu IP này đã được báo cáo trước đó trong các hoạt động độc hại, thông tin ngữ cảnh về lịch sử tấn công, quốc gia xuất xứ, và loại hình mối đe dọa liên quan sẻ được thu thập. Đồng thời, module RAG thực hiện tìm kiếm vector similarity trong cơ sở dữ liệu Qdrant để truy xuất các tài liệu quy trình xử lý liên quan. Trong trường hợp này, hệ thống cần tìm ra thông tin về "Xử lý tấn công Brute Force SSH".

Sau khi thu thập đủ ngữ cảnh, mô hình LLM sẽ tổng hợp tất cả thông tin này cùng với log gốc để sinh ra một báo cáo phân tích. Báo cáo này được kỳ vọng sẽ bao gồm các thành phần sau: phần tóm tắt sự cố mô tả ngắn gọn bản chất của cuộc tấn công, đánh giá mức độ nghiêm trọng dựa trên các yếu tố như số lần thử, reputation của IP nguồn và tài

khoản bị nhắm tới, phân tích kỹ thuật giải thích cơ chế hoạt động của tấn công brute force và tại sao nó nguy hiểm, và quan trọng nhất là các bước khuyến nghị xử lý cụ thể.

Các khuyến nghị này cần phải thực tế và có thể thực hiện ngay lập tức, bao gồm việc chặn địa chỉ IP nguồn trên tường lửa, kiểm tra và tăng cường cấu hình SSH như tắt tính năng PermitRootLogin, thực thi xác thực đa yếu tố, hoặc giới hạn số lần thử đăng nhập thất bại. Nếu có dấu hiệu cho thấy tấn công đã thành công, báo cáo cần đề xuất các biện pháp ứng cứu khẩn cấp như cô lập máy chủ, thay đổi ngay tất cả mật khẩu và tiến hành điều tra forensic để xác định mức độ xâm nhập.



Hình 3.1.3-3: Kết quả phân tích cảnh báo Brute Force

3.2.2. Kịch bản 2: Khai thác lỗ hổng Web (SQL Injection)

Tấn công SQL Injection đại diện cho một trong những lớp lỗ hổng nguy hiểm và phổ biến nhất trong lĩnh vực bảo mật ứng dụng web, liên tục xuất hiện trong danh sách OWASP Top 10 về các rủi ro bảo mật quan trọng nhất trong hơn hai thập kỷ. Sức mạnh đáng sợ của kỹ thuật này nằm ở khả năng biến một giao diện người dùng vô hại thành một cổng truy cập trái phép vào trái tim của hệ thống - cơ sở dữ liệu nơi lưu trữ tất cả thông tin quan trọng của tổ chức. Khác với các cuộc tấn công nhắm vào hạ tầng mạng hay hệ điều hành, SQL Injection khai thác logic của chính ứng dụng, biến những đoạn mã được thiết kế để xử lý dữ liệu người dùng thành vũ khí tấn công.

Động lực thúc đẩy kẻ tấn công theo đuổi lỗ hổng này là vô cùng đa dạng và đều mang tính nghiêm trọng cao. Mục tiêu chính thường là đánh cắp dữ liệu nhạy cảm - thông tin cá nhân của khách hàng, thông tin thẻ tín dụng, dữ liệu y tế được bảo vệ bởi luật, hoặc bí mật thương mại của doanh nghiệp. Trong nhiều trường hợp, kẻ tấn công không chỉ đọc dữ liệu mà còn có thể thao túng chúng, thay đổi giá cả trong hệ thống thương mại điện tử, điều chỉnh số dư tài khoản trong ứng dụng ngân hàng, hoặc thậm chí xóa toàn bộ cơ sở dữ liệu để tống tiền hoặc phá hoại. Một số cuộc tấn công tinh vi hơn sử dụng SQL Injection như bước đầu tiên trong một chuỗi tấn công phức tạp - sau khi chiếm quyền kiểm soát database, kẻ tấn công có thể leo thang đặc quyền lên database server, sau đó pivot sang các hệ thống khác trong mạng nội bộ. Nguy hiểm hơn nữa, nhiều hệ quản trị cơ sở dữ liệu hiện đại cung cấp các stored procedure cho phép thực thi lệnh hệ điều hành, biến một lỗ hổng injection thành một shell tương tác đầy đủ trên server.

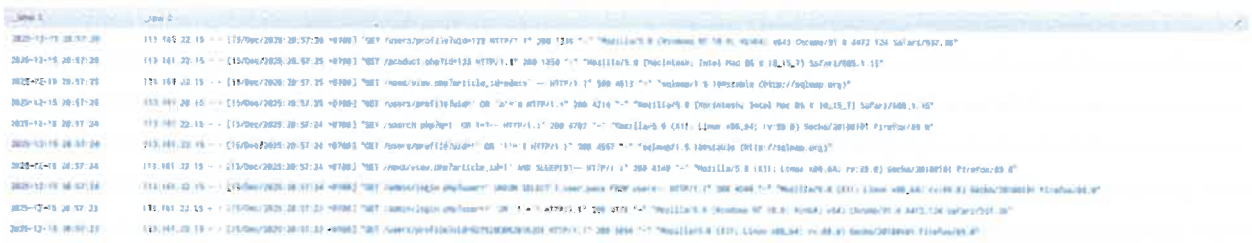
Lỗ hổng SQL Injection tồn tại do một nguyên nhân căn bản trong thiết kế ứng dụng: việc thiếu sự phân tách rõ ràng giữa dữ liệu và mã lệnh. Khi một ứng dụng web xây dựng câu truy vấn SQL bằng cách nối chuỗi trực tiếp dữ liệu đầu vào của người dùng mà không qua bất kỳ kiểm tra hay làm sạch nào, nó vô tình trao cho người dùng khả năng tác động đến cấu trúc logic của câu lệnh SQL. Ví dụ điển hình, một form đăng nhập có thể xây dựng truy vấn như: "SELECT * FROM users WHERE username='[input_user]' AND password='[input_pass]'". Nếu ứng dụng chỉ đơn giản thay thế input của người dùng vào vị trí này mà không kiểm tra, một kẻ tấn công có thể nhập vào trường username chuỗi "admin'--" để biến câu truy vấn thành "SELECT * FROM users WHERE username='admin'--' AND password='...'". Hai dấu gạch ngang trong SQL là ký hiệu comment, có nghĩa là mọi thứ sau nó sẽ bị bỏ qua, do đó điều kiện kiểm tra password bị vô hiệu hóa và kẻ tấn công có thể đăng nhập mà không cần biết mật khẩu. Đây chỉ là một trong rất nhiều biến thể của kỹ thuật này.

Trong kịch bản thử nghiệm này, vai trò kẻ tấn công một lần nữa được đảm nhận bởi máy chạy hệ điều hành Kali Linux, nhưng lần này công cụ được sử dụng là SQLMap - được coi là "vũ khí hạt nhân" trong lĩnh vực khai thác SQL Injection. SQLMap là một công cụ mã nguồn mở được phát triển bởi cộng đồng bảo mật quốc tế, tích hợp hàng trăm kỹ thuật injection khác nhau và có khả năng tự động hóa toàn bộ quy trình từ phát hiện lỗ hổng, xác định loại cơ sở dữ liệu, cho đến khai thác và trích xuất dữ liệu. Công cụ này không chỉ đơn giản thử nghiệm một vài payload cơ bản mà sử dụng các thuật toán thông minh để phát hiện các bộ lọc bảo vệ, tự động điều chỉnh chiến thuật tấn công, và thậm chí có thể bypass các web application firewall bằng cách obfuscate payload hoặc sử dụng các kỹ thuật evasion tinh vi.

Mục tiêu của cuộc tấn công là một máy chủ Ubuntu Server đang chạy một stack web điển hình bao gồm Apache HTTP Server làm web server, PHP làm ngôn ngữ xử lý backend, và MySQL làm hệ quản trị cơ sở dữ liệu. Trên server này, chúng tôi đã cố tình triển khai một ứng dụng web đơn giản có chứa lỗ hổng SQL Injection để phục vụ mục đích thử nghiệm. Ứng dụng này mô phỏng một trang web thương mại điện tử với chức năng xem chi tiết sản phẩm, trong đó parameter id trong URL được sử dụng để truy vấn thông tin sản phẩm từ database. Đoạn mã PHP xử lý request này được viết theo cách không an toàn: nó lấy trực tiếp giá trị của parameter id từ URL và nhúng vào câu truy vấn SQL mà không qua bất kỳ quá trình sanitization hay sử dụng prepared statement. Cấu hình này đại diện cho một lỗi lập trình phổ biến mà nhiều developer mắc phải, đặc biệt trong các ứng dụng legacy hoặc khi áp lực thời gian khiến họ bỏ qua các best practice về bảo mật.

Giống như trong kịch bản trước, máy chủ web này cũng được tích hợp với hệ thống giám sát tập trung. Splunk Universal Forwarder được cấu hình để theo dõi và đẩy các file log quan trọng về SIEM, bao gồm access log của Apache (thường nằm ở /var/log/apache2/access.log) ghi lại mọi HTTP request với đầy đủ thông tin về IP client, timestamp, URL được request, User-Agent string và response code, cùng với error log (/var/log/apache2/error.log) chứa các thông báo lỗi của web server và ứng dụng PHP. Ngoài ra, MySQL general query log cũng được bật để ghi lại tất cả các câu truy vấn SQL được thực thi, mặc dù trong môi trường production tính năng này thường bị tắt do ảnh hưởng đến hiệu năng, nhưng trong môi trường thử nghiệm nó cung cấp khả năng quan sát chi tiết quá trình tấn công.

Mặc dù SQL Injection tấn công vào tầng ứng dụng và database, nó vẫn để lại những dấu vết trong các log hệ thống mà một hệ thống giám sát được cấu hình đúng có thể phát hiện. Dấu hiệu rõ ràng nhất nằm trong access log của web server. Khi SQLMap hoạt động, nó gửi hàng chục đến hàng trăm request trong thời gian ngắn, tất cả đều nhắm vào cùng một endpoint nhưng với các giá trị parameter khác nhau. Những giá trị parameter này không phải là dữ liệu bình thường mà chứa các ký tự đặc biệt và cấu trúc SQL - dấu nháy đơn ('), dấu ngoặc kép ("), từ khóa SQL như UNION, SELECT, ORDER BY, comment syntax như -- hay #, và các hàm database đặc thù như CONCAT(), CHAR(), hoặc SLEEP()



Hình 3.1.3-4: Log SQLi trên Splunk

Từng dòng log này kể một phần của câu chuyện tấn công: payload đầu tiên với dấu nháy đơn gây ra lỗi server (500), các payload AND được sử dụng để test boolean-based injection, ORDER BY để xác định số column, và cuối cùng UNION để khai thác. Tần suất request cũng bất thường - một người dùng thật sẽ không truy cập cùng một trang hàng chục lần mỗi giây.

Thêm vào đó, User-Agent string trong các request thường tiết lộ công cụ đang được sử dụng. SQLMap theo mặc định sử dụng một User-Agent đặc trưng chứa chuỗi "sqlmap", mặc dù kẻ tấn công có kinh nghiệm có thể thay đổi điều này. Tuy nhiên, ngay cả khi User-Agent bị giả mạo, các đặc điểm của request pattern - thứ tự cụ thể của các payload, timing giữa các request, và cấu trúc của URL encoding - vẫn có thể được sử dụng để nhận diện công cụ tự động.

Trên hệ thống SIEM, Splunk đã được cấu hình với các correlation rule đặc biệt để phát hiện SQL injection. Một rule đơn giản có thể tìm kiếm các từ khóa SQL nguy hiểm trong URL: bất kỳ request nào có chứa "UNION.*SELECT", "OR 1=1", "' OR '1'='1", hoặc các pattern tương tự sẽ trigger cảnh báo. Rule phức tạp hơn sử dụng statistical analysis để phát hiện bất thường: nếu một IP nguồn đột ngột gửi nhiều request đến cùng một endpoint với error rate cao (nhiều response 500), đây có thể là dấu hiệu của SQL injection attempts. Rule có thể được tuned để giảm false positive bằng cách whitelist các IP đáng tin cậy hoặc yêu cầu multiple indicators phải xảy ra đồng thời.

Khi rule được trigger, Splunk tạo ra một bản tin cảnh báo với severity level High hoặc Critical, vì SQL injection có khả năng dẫn đến data breach nghiêm trọng. Cảnh báo bao gồm không chỉ thông tin cơ bản như source IP và timestamp, mà còn chứa các log entry thô cho thấy chính xác các payload đã được gửi. Bản tin này sau đó được đóng gói thành JSON và gửi qua webhook đến hệ thống RAG để phân tích sâu hơn



Hình 3.1.3-5: Kết quả chạy Rule SQLi trên Splunk

SQL Injection không giống như brute force đơn giản - nó đòi hỏi hiểu biết sâu về cách thức hoạt động của ứng dụng web, cấu trúc của các câu truy vấn SQL, và các kỹ thuật khai thác khác nhau. Hệ thống RAG được kỳ vọng phải thực hiện một phân tích nhiều tầng.

Tầng đầu tiên là nhận diện chính xác bản chất của cuộc tấn công. Hệ thống cần phải phân tích các thông tin trong URL, nhận ra các ký tự và từ khóa SQL đặc trưng, và phân loại đây là UNION-based SQL injection chứ không phải các loại khác như blind injection hay second-order injection. Việc nhận diện này không chỉ dựa vào việc khớp dữ liệu mà cần hiểu ngữ nghĩa - ví dụ phân biệt giữa một request hợp lệ có chứa từ "select" trong search query và một request độc hại có chứa "SELECT" như một phần của SQL.

Tầng thứ hai là phân tích công cụ và chiến thuật. Khi hệ thống phát hiện User-Agent chứa "sqlmap", nó nên có khả năng tra cứu trong knowledge base để tìm thông tin về công cụ này - đây là một công cụ tự động được sử dụng rộng rãi, có khả năng thực hiện nhiều kỹ thuật injection khác nhau, và thường được sử dụng bởi cả penetration tester hợp pháp lẫn attacker thực sự. Thông tin này giúp đánh giá mức độ nghiêm trọng - một cuộc tấn công sử dụng công cụ tự động tinh vi thường nguy hiểm hơn những lần thử nghiệm thủ công vô tình.

Tầng thứ ba là phân tích ngữ cảnh và đánh giá rủi ro. Hệ thống cần truy xuất thông tin về địa chỉ IP nguồn từ OTX hoặc AbuseIPDB. Trong kịch bản thử nghiệm của chúng tôi, IP là một địa chỉ ISP di động nội địa Việt Nam với reputation score bằng 0 - tức là "sạch" và chưa từng được báo cáo. Đây là một tình huống để kiểm tra khả năng suy luận của hệ thống. Một hệ thống đơn giản dựa trên reputation có thể đánh giá thấp mức độ nghiêm trọng vì IP có vẻ an toàn. Tuy nhiên, hệ thống RAG được kỳ vọng phải thể hiện reasoning phức tạp hơn: mặc dù IP chưa được báo cáo trước đó, hành vi tấn công là rõ ràng và không thể chối cãi - 48 lần thử với các dữ liệuSQL injection trong vài phút không thể là hành vi người dùng bình thường. Do đó, mức độ nghiêm trọng vẫn phải được đánh giá là High bất kể reputation của IP là 0. Khả năng này kết hợp các chỉ số và đưa ra đánh giá dựa trên hành vi thay vì chỉ số reputation - là điều phân biệt một AI system thông minh với một rule-based system đơn giản.

Tầng thứ tư là giải thích kỹ thuật. RAG nên giải thích cho người phân tích hiểu được cơ chế tấn công: UNION-based injection hoạt động bằng cách nối kết quả của một truy vấn độc hại với truy vấn hợp pháp của ứng dụng, cho phép attacker extract dữ liệu trực tiếp trong response HTML. Hệ thống nên chỉ ra tại sao kỹ thuật này nguy hiểm - nó có thể được sử dụng để dump toàn bộ database, đánh cắp mật khẩu, thông tin thẻ tín dụng, hoặc dữ liệu nhạy cảm khác. Giải thích này không chỉ nêu facts mà còn connect các facts lại với nhau để tạo thành một narrative có ý nghĩa.

Tầng cuối cùng là đề xuất hành động. Hệ thống RAG cần truy xuất dữ liệu về xử lý SQL injection từ Qdrant và extract các bước cụ thể. Đề xuất phải bao gồm cả hành động nay và hành dộng dài hạn. Cần: chặn IP nguồn tại WAF hoặc firewall để ngăn chặn tiếp tục khai thác, cô lập web server khỏi database server nếu có dấu hiệu database đã bị compromise. Dài hạn: rà soát mã nguồn của ứng dụng để xác thực và sửa điểm injection - cụ thể là tham số 'id' trong tệp vuln.php, triển khai các câu lệnh đã chuẩn bị sẵn (truy vấn được tham số hóa) thay vì nối chuỗi, triển khai hoặc điều chỉnh các quy tắc WAF để phát hiện và chặn các nỗ lực chèn SQL, tiến hành xem xét mã và kiểm tra thâm nhập cho toàn bộ ứng dụng để tìm các lỗi khác. Những khuyến nghị này nên được trích xuất trực tiếp từ dữ liệu nội bộ của tổ chức, đảm bảo chúng phù hợp với các chính sách đã được phê duyệt.



Hình 3.1.3-6: Kết quả phân tích cảnh báo SQLi

3.2.3. Kịch bản 3: Leo thang đặc quyền (Privilege Escalation)

Trong phân loại các kỹ thuật tấn công mạng, leo thang đặc quyền đặc biệt quan trọng vì nó đại diện cho ranh giới giữa một xâm nhập có giới hạn và một tấn công bảo mật toàn diện. Phần lớn các cuộc tấn công thành công không bắt đầu với việc kẻ tấn công có ngay quyền quản trị viên, thường bắt đầu với việc sử dụng quyền truy cập ở mức độ thấp - có thể là tài khoản người dùng thông thường thông qua lừa đảo, nhồi thông tin xác thực hoặc khai thác một lỗi web. Tại điểm này, kẻ tấn công đang tồn tại trong hệ thống nhưng bị giới hạn bởi quyền hạn chế. Leo thang đặc quyền là quá trình họ phá vỡ những điều bắt buộc để nâng cấp đặc quyền của mình lên mức cao nhất - quyền root trên Linux hoặc SYSTEM/Administrator trên Windows.

Động lực thúc leo thang đặc quyền được rõ ràng: với quyền root, kẻ tấn công trở thành "thần" trong hệ thống. Họ có thể đọc mọi tệp bất kể sự cho phép, sửa đổi bất kỳ cấu hình nào, cài đặt backdoor tồn tại lâu dài, ghi nhật ký và giám sát vô hiệu hóa, truy cập vào thông tin xác thực được lưu trữ và quan trọng nhất là xoay sang các hệ thống khác trong nội bộ mạng. Trong nhiều khuôn khổ tấn công như Lockheed Martin Cyber Kill Chain hay MITRE ATT&CK, leo thang đặc quyền được coi là một giai đoạn then chốt - nếu không qua được, kẻ tấn công sẽ bị mắc kẹt với những gì mà người dùng bình thường có thể làm được, làm hạn chế đáng kể hoạt động của cuộc tấn công.

Các phương pháp leo thang đặc quyền trên Linux rất đa dạng, mỗi phương pháp khai thác một loại sai sót khác nhau trong cấu hình hoặc thiết kế hệ thống. Một hướng phổ biến là lạm dụng cấu hình sudo sai - khi các quản trị viên cấu hình file /etc/sudoers không cẩn thận, cho phép user thông thường chạy lệnh với quyền root. Một hướng khác là khai thác SUID binaries - các tệp thực thi có bit SUID được set sẽ chạy với quyền của chủ sở hữu (thường là root) bất kể ai thực thi chúng. Nếu một binary như vậy có lỗ hổng hoặc có chức năng có thể bị lạm dụng, nó trở thành cầu thang để leo lên quyền root. Các phương pháp khác bao gồm khai thác kernel vulnerabilities, chạy cron jobs quyền cao, khai thác các dịch vụ systemd,... Kịch bản này tập trung vào loại đầu tiên – lạm dụng cấu hình sudo sai- vì đây là một trong những lỗi cấu hình phổ biến nhất trong thực tế.

Trên Splunk SIEM, một correlation rule được thiết kế để phát hiện leo thang đặc quyền. Quy tắc này phức tạp hơn các quy tắc trước vì cần phân biệt giữa các hành động quản trị hợp pháp và tấn công leo thang. Một approach là look for unusual combinations: nếu một user account mà normally không sử dụng sudo đột nhiên chạy sudo để get a shell, hoặc nếu sudo được chạy từ một unusual parent process (như một web server process), hoặc nếu có một sequence của escalation followed ngay bởi suspicious activities như reading /etc/shadow hay creating new users.



Hình 3.1.3-7: Kết quả chạy Rule Privilege Escalation trên Splunk

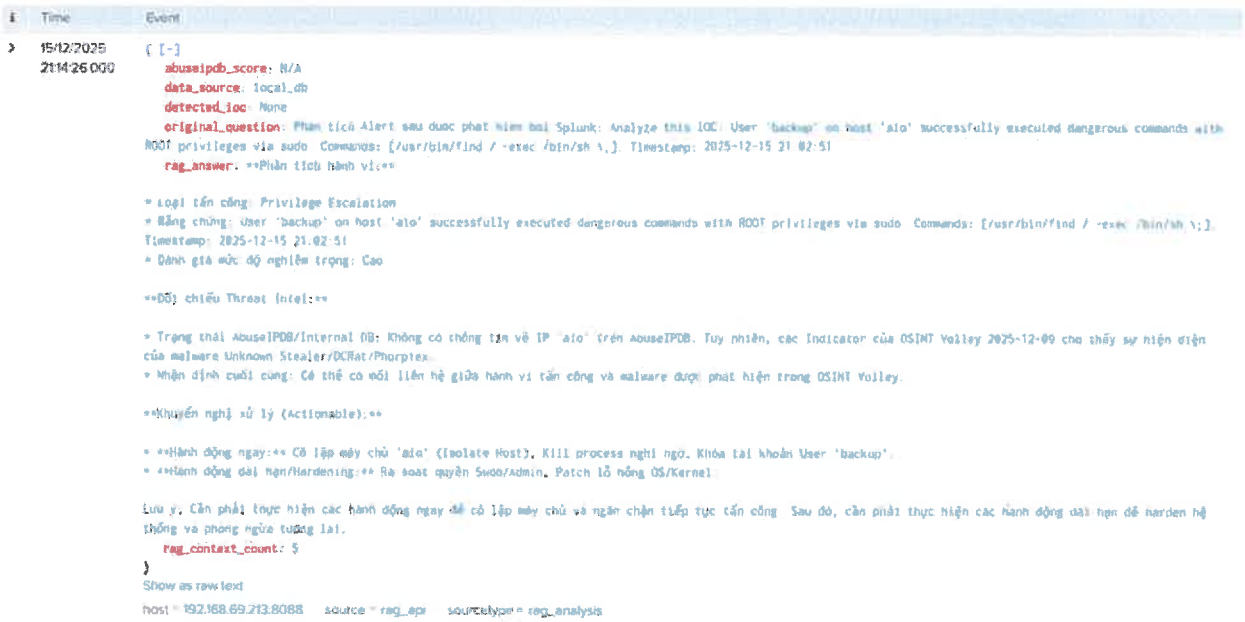
Leo thang đặc quyền là một trong những sự kiện bảo mật nghiêm trọng và thống RAG phải phản ánh mức độ nghiêm trọng của điều này. Khi nhận được cảnh báo, hệ thống phải thực hiện phân tích nhiều lớp.

Lớp đầu tiên là phân tích kỹ thuật của phương pháp leo thang. Hệ thống cần nhận ra rằng đây là sự leo thang dựa trên sudo khi cho phép thực thi bash shell. Cần đào tạo cơ sở dữ liệu để trích xuất thông tin về kỹ thuật cụ thể này - trong khung MITER ATT&CK, đây ánh xạ tới kỹ thuật T1548.003 "Sudo và Sudo Caching". Hệ thống nên giải thích tại sao cấu hình cho phép người dùng chạy bash với sudo là nguy hiểm: bash là một shell tương tác hoàn toàn, nghĩa là một khi kẻ tấn công có được quyền root, họ có thể làm bất cứ điều gì mà không cần khai thác thêm.

Lớp thứ hai là phân tích bối cảnh về người dùng. Hệ thống cần kiểm tra: người dùng backup có phải là quản trị viên hợp pháp không? Dựa vào tư cách thành viên nhóm và cách sử dụng sudo lịch sử, có thể xác định rằng backup là tài khoản nhân viên thông thường, không phải sysadmin. Đây là thông tin củng cố kết luận rằng đang xảy ra sự cố leo thang đặc quyền.

Lớp thứ ba là phân tích hành vi. Trong trường hợp này, việc đọc /etc/shadow và tạo người dùng mới là những dấu hiệu rõ ràng về mục đích độc hại vì đây không phải là các nhiệm vụ quản trị điển hình mà là hành vi của kẻ tấn công. Điều này giait thích cho mức độ nghiêm trọng là cao.

Về các khuyến nghị, hệ thống cần cung cấp cả hành động ngăn chặn ngay lập tức và biện pháp khắc phục lâu dài như: Khóa tài khoản user, rà soát lại file /etc/sudoers,…



Hình 3.1.3-8: Kết quả phân tích cảnh báo Privilege Escalation

3.3. Nhận xét, phân tích kết quả thử nghiệm

Sau khi thực hiện 3 kịch bản trên, kết quả thu được từ hệ thống RAG (sử dụng model OLlama-3) được tổng hợp như sau.

3.3.1. Đánh giá Hiệu năng hệ thống

Bảng 3.1.3-1: 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

Nhận xét:

- Thời gian phản hồi trung bình khoảng 40 giây. Đây là con số chấp nhận được đối với việc phân tích sâu (so với con người mất trung bình 10-15 phút để tra cứu thủ công).
- Ollama tiêu tốn tài nguyên khá lớn khi sinh câu trả lời (Inference), đòi hỏi máy chủ phải có cấu hình RAM tối thiểu 16GB để hoạt động ổn định.

3.3.2. Đánh giá Chất lượng kết quả đầu ra

Kết quả phân tích đầu ra của RAG:



Hình 3.1.3-1: Kết quả phân tích cảnh báo SQLi

Kết quả thực nghiệm trong hình cho thấy khả năng của mô hình trong việc xử lý ngữ cảnh (Contextual Understanding) và khả năng giải thích (Explainability). Cụ thể, khi tiếp nhận dữ liệu đầu vào bao gồm cảnh báo tấn công với tần suất cao (48 lần nỗ lực SQL Injection) và thông tin ngữ cảnh từ cơ sở dữ liệu bên ngoài (External Context) chỉ ra rằng địa chỉ IP nguồn "113.161.22.15" có điểm tín nhiệm "sạch" (AbuseIPDB Score 0%), hệ thống không đơn thuần dựa vào chỉ số tín nhiệm để bỏ qua cảnh báo. Thay vào đó, mô hình đã tổng hợp dữ liệu và đưa ra kết luận chính xác: mặc dù địa chỉ IP chưa nằm trong danh sách đen (blacklist), nhưng hành vi tấn công là rõ ràng và nghiêm trọng. Đồng thời, hệ thống cũng nhận diện chính xác công cụ tấn công tự động sqlmap thông qua phân tích chuỗi User-Agent, chứng minh khả năng hiểu sâu về kỹ thuật tấn công.

Trong kịch bản thực nghiệm SQL Injection, mặc dù kho tri thức chứa nhiều kỹ thuật tấn công web khác nhau, cơ chế tìm kiếm tương đồng (Similarity Search) đã trích xuất chính xác tài liệu về SQL Injection thay vì các tài liệu nhiễu như XSS hay Path Traversal. Hệ thống đạt độ chính xác cao khi ánh xạ thành công hành vi trong log thô với các kỹ thuật cụ thể trong khung MITRE ATT&CK và các quy trình vận hành. Việc đạt được độ chính xác truy xuất cao (nhờ mô hình embedding all-MiniLM-L6-v2) giúp giảm thiểu tối đa các thông tin không liên quan đưa vào prompt. Điều này không chỉ giúp tiết kiệm cửa sổ ngữ cảnh (Context Window) mà còn là yếu tố tiên quyết giúp LLM đưa ra phản hồi tập trung, tránh hiện tượng suy luận lan man hoặc nhầm lẫn giữa các loại hình sự cố khác nhau. Việc đảm bảo Độ chính xác truy xuất (Retrieval Precision) ở mức cao giúp hệ thống duy trì sự

cân bằng tốc độ xử lý và chất lượng phân tích. Kết quả cho thấy, dù chỉ trích xuất Top-K kết quả gần nhất (K=1 đến 3), hệ thống vẫn cung cấp đủ bằng chứng xác thực để mô hình LLM hoàn thiện báo cáo phân tích mà không gặp phải thông tin mâu thuẫn từ các tài liệu truy xuất sai

Về tính minh bạch và khả năng giải thích, câu trả lời của mô hình tuân thủ chặt chẽ cấu trúc tư duy logic (Chain-of-Thought), đi từ việc liệt kê các hành vi quan sát được, đối chiếu với dữ liệu tình báo bên ngoài, và cuối cùng là đưa ra kết luận tổng hợp. Điểm mạnh nổi bật nằm ở tính khả dụng (Actionability) của các khuyến nghị. Sau khi xác định chính xác bản chất cuộc tấn công là SQL Injection nhắm vào lớp ứng dụng, hệ thống đã đề xuất các giải pháp xử lý cụ thể và thiết thực như rà soát mã nguồn để kiểm tra dữ liệu đầu vào (Input Validation) và tinh chỉnh luật cho tường lửa ứng dụng web (Tuning WAF), thay vì chỉ đưa ra các cảnh báo chung chung.

Cuối cùng, đánh giá về độ tin cậy (Faithfulness) và hiện tượng ảo giác (Hallucination) cho thấy mô hình hoạt động ổn định và trung thực với dữ liệu nguồn. Các thông tin chi tiết về chỉ số IOC (Indicators of Compromise) như điểm AbuseIPDB (0%), nhà cung cấp dịch vụ (ISP: Vietnam Posts and Telecommunications Group), và loại hình sử dụng (Usage: Mobile ISP) được trích xuất nguyên vẹn từ cơ sở dữ liệu mà không có dấu hiệu bịa đặt. Tương tự, các thông số kỹ thuật quan trọng như số lượng yêu cầu (48 attempts), nội dung payload (OR 1=1), và công cụ sử dụng (sqlmap) đều được trích xuất chính xác tuyệt đối từ log gốc, đảm bảo tính toàn vẹn của dữ liệu trong báo cáo phân tích.

Việc tích hợp RAG đánh dấu bước chuyển dịch từ mô hình "phản ứng thụ động" sang "phân tích chủ động". Nếu như SOC truyền thống thường khiến chuyên gia bị quá tải bởi các cảnh báo dương tính giả (False Positives) do thiếu thông tin kiểm chứng, thì mô hình RAG hoạt động như một trợ lý ảo thông minh, giúp lọc nhiễu và cung cấp sẵn "kịch bản hành động". Điều này giải quyết trực tiếp bài toán thiếu hụt nhân sự chất lượng cao và giảm thiểu rủi ro sai sót do yếu tố con người trong quá trình trực vận hành căng thẳng.

Bảng 3.1.3-2: So sánh đặc điểm vận hành giữa SOC truyền thống và 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	Rút ngắn đáng kể nhờ khả năng tự động phân tích và đề

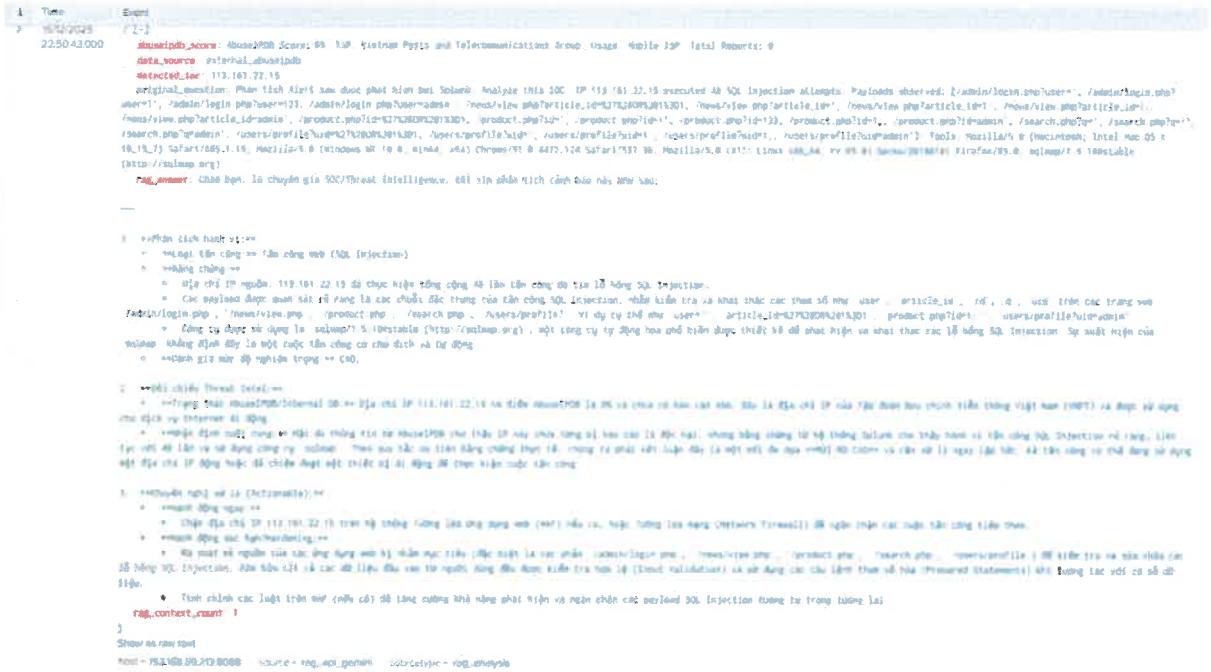
	kéo dài từ vài chục phút đến hàng giờ đối với các sự cố phức tạp.	xuất giải pháp tức thời. Chuyên gia chỉ cần xác nhận quyết định cuối cùng.
Khả năng giải thích	Hạn chế. Các cảnh báo thường chỉ hiển thị mã lỗi hoặc tên luật vi phạm mà không giải thích "tại sao" đây là mối đe dọa	Minh bạch và dễ hiểu: 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 mà không cần huấn luyện lại mô hình

3.3.3. Các kịch bản thử nghiệm mở rộng

Để chứng minh tính linh hoạt và khả năng mở rộng của kiến trúc RAG đề xuất, đề tài thực hiện thêm các kịch bản thay thế thành phần hệ thống.

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

- Mục đích: So sánh hiệu năng và chất lượng phân tích giữa mô hình mã nguồn mở chạy cục bộ (Ollama - Llama 3) và mô hình thương mại chạy trên Cloud (Google Gemini Pro).
- Cấu hình thay đổi:
    - Chuyển module LLM Interface trong RAG Controller để gọi API của Google Gemini thay vì Ollama endpoint.
    - Prompt giữ nguyên để đảm bảo tính công bằng.
- Kết quả:

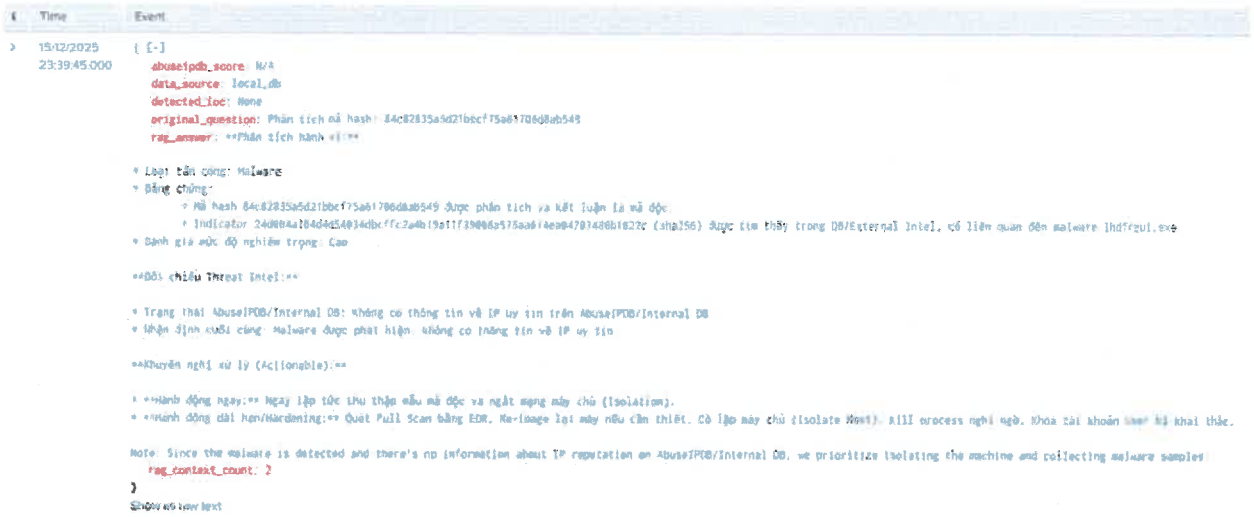


Hình 3.1.3-2: Kết quả phân tích cảnh báo SQLi sử dụng Gemini

- Google Gemini: Tốc độ phản hồi nhanh hơn (do hạ tầng Cloud mạnh mẽ), khả năng diễn đạt ngôn ngữ tự nhiên mượt mà hơn và ít lỗi ngữ pháp tiếng Việt hơn. Tuy nhiên, dữ liệu log phải gửi ra ngoài Internet (rủi ro bảo mật).
- Ollama (Llama 3): Tốc độ chậm hơn (phụ thuộc phần cứng máy chủ thử nghiệm), nhưng đảm bảo tuyệt đối dữ liệu không rời khỏi mạng nội bộ.

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

- Mục đích: Kiểm chứng khả năng tích hợp đa nguồn dữ liệu tri thức.
- Cấu hình thay đổi:
  - Thay thế nguồn dữ liệu từ AlienVault OTX sang VirusTotal.
  - Kịch bản: Kiểm tra một mã Hash của file mã độc (Malware hash).
- Kết quả:



Hình 3.1.3-3: Kết quả phân tích mã hash sử dụng dữ liệu VirusTotal

3.4. Kết chương.

Chương 3 đã trình bày quá trình thử nghiệm mô hình RAG trên môi trường Lab với 3 kịch bản tấn công thực tế. Kết quả cho thấy hệ thống hoạt động ổn định, đạt được mục tiêu về tự động hóa phân tích và giảm thiểu thời gian phản ứng sự cố.

Mặc dù vẫn còn một số hạn chế về tài nguyên phần cứng và khả năng xử lý các kịch bản quá phức tạp, nhưng những kết quả này đã chứng minh tính đúng đắn của hướng nghiên cứu: Kết hợp LLM và RAG là một cách khả thi và hiệu quả để nâng cao năng lực cho trung tâm SOC.

Các kết quả này là cơ sở để đưa ra kết luận và kiến nghị phát triển trong phần cuối cùng của đề án

KẾT LUẬN

1. Nội dung đề án đã thực hiện

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

- Về mặt lý thuyết:
    - Hệ thống 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)

TÀI LIỆU THAM KHẢO

[1] Institute, SANS, "Building a World-Class Security Operations Center: A Roadmap," 2011.

[2] "wikipedia," [Online]. Available: https://vi.wikipedia.org/wiki/Mô_hình_ngôn_ngữ_lớn.

[3] "wikipedia," [Online]. Available: https://vi.wikipedia.org/wiki/Mô_hình_ngôn_ngữ.

[4] A. Vaswani et al., "Attention Is All You Need," 2017.

[5] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, Jianfeng Gao, "Large Language Models: A Survey," *arXiv preprint arXiv:2402.06196,* 2024.

[6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel và D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems (NeurIPS),* 2020.

[7] "wikipedia," [Online]. Available: https://vi.wikipedia.org/wiki/Tạo_sinh_dựa_trên_truy_xuất_tăng_cường.

[8] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," 2020.

[9] "The Cost of Malware Containment Report," *Ponemon Institute,* 2022.

[10] "Cybersecurity Workforce Study: Building a Diverse and Capable Workforce," *(ISC)², 2023.*

[11] "Market Guide for Security Information and Event Management," *Gartner,* 2023.

[12] "Cost of a Data Breach Report 2023," *IBM Security,* 2023.

[13] Deng, L., Wang, H., & Chen, Y. , "Enhancing Security Alert Triage with Retrieval-Augmented Generation: A Comparative Study," *IEEE Transactions on Information Forensics and Security,* 2024.

[14] Kurnia, D., et al., "Enhancing Security Operations Center: Wazuh Security Event Response with Retrieval-Augmented-Generation-Driven Copilot," Vols. Sensors, 25(3), 870. MDPI, 2025.

[15] Singh, R., Tariq, S., Jalalvand, F., Baruwal Chhetri, M., Nepal, S., Paris, C., & Lochner, M, "LLMs in the SOC: An Empirical Study of Human-AI Collaboration in Security Operations Centres," p. arXiv:2508.18947, 2025.

[16] 2025. [Online]. Available: https://www.rapid7.com/blog/post/2025/04/29/insightidr-ai-alert-triage-automatically-classifies-alerts-with-99-93-accuracy/.

[17] 2025. [Online]. Available: https://www.dropzone.ai/resource-guide/alert-triage-guide-2025.

[18] T. Asa, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," *arXiv preprint arXiv:2401.14887,* 2023.

[19] "Augmenting Security Operations Centers with Accelerated Alert Triage and LLM Agents Using NVIDIA Morpheus," *NVIDIA Developer Blog,* 2024.

[20] CyberScienceLab, [Online]. Available: https://github.com/CyberScienceLab/Threat_Intelligence_Rag.

[21] [Online]. Available: https://www.splunk.com/.

[22] "Qdrant Documentation: Vector Database for the next generation of AI applications," *Qdrant Solutions GmbH,* 2024.

[23] "AlienVault Open Threat Exchange (OTX) API Documentation," *AT&T Cybersecurity,* 2023.

[24] Cichonski, P., Millar, T., Grance, T., & Scarfone, K., "Computer Security Incident Handling Guide," *National Institute of Standards and Technology,* 2012.

[25] Es, S., James, J., Espinosa-Anke, L., & Schockaert, S., "RAGAS: Automated Evaluation of Retrieval Augmented Generation," *arXiv preprint arXiv:2309.15217,* 2023.

[26] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... & Fung, P., "Survey of Hallucination in Natural Language Generation," *ACM Computing Surveys,* 2023.

[27] Ferrag, M. A., Battah, A., Tihanyi, N., et al., "Secure-LLM: A Comprehensive Survey on Large Language Models for Cybersecurity," *arXiv preprint arXiv:2401.xxxxx,* 2024.

[28] [Online]. Available: https://github.com/CyberScienceLab/Threat_Intelligence_Rag.

[29] [Online]. Available: https://docs.nvidia.com/morpheus/examples/llm/rag/README.html .

[30] [Online]. Available: https://www.linkedin.com/pulse/build-your-first-rag-system-jupyter-notebook-guide-kannan-venkat-uuage.

[31] [Online]. Available: https://towardsdatascience.com/how-i-built-my-first-rag-pipeline-6e178326e3c8/ .

[32] [Online]. Available: https://docs.openvino.ai/2023.3/notebooks/254-rag-chatbot-with-output..

[33] S. Iaroshev, "Evaluating Retrieval-Augmented Generation Models for Financial Question Answering," *Applied Sciences,* vol. 14, 2024.

[34] X. Chen, "Knowledge Graph-Augmented Retrieval-Augmented Generation," *Scientific Reports,* vol. 15, 2025.

[35] C.-C. Chang, C.-F. Li, C.-H. Lee, "Enhancing Low-Resource Minority Language Translation with LLMs and Retrieval-Augmented Generation for Cultural Nuances," *arXiv preprint arXiv:2505.10829,* 2025.

[36] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, "Retrieval-Augmented Generation for Large Language Models: A Survey,," *arXiv preprint arXiv:2312.10997,* 2023.

[37] "Large language models with retrieval-augmented generation enhance expert modelling of Bayesian network for clinical decision support," *J. Comput. Assist. Radiol. Surg,* 2025.

[38] "Retrieval augmented generation for 10 large language models and its generalizability in assessing medical fitness," *Digital Medicine,* 2025.

[39] Y. Huang và J. Huang, "A Survey on Retrieval-Augmented Text Generation for Large Language Models," *arXiv preprint arXiv:2404.10981,* 2024.

[40] Deng, L., Wang, H., & Chen, Y., "Enhancing Security Alert Triage with Retrieval-Augmented Generation: A Comparative Study," *IEEE Transactions on Information Forensics and Security,* 2024.

[41] "LLMs in the SOC: An Empirical Study of Human-AI Collaboration in Security Operations Centres," *arXiv.,* p. arXiv:2508.18947, 2025.

19% Tính tương đồng nói chung

Tổng cộng của tất cả các kết quả trùng khớp, bao gồm cả các nguồn trùng lặp, cho mỗi c...

Đã lọc khỏi Báo cáo

- Mục lục tham khảo
- Văn bản được trích dẫn
- Kết quả trùng khớp nhỏ (ít hơn 8 từ)

Nguồn hàng đầu

16% Nguồn Internet
5% Ấn bản
7% Bài tập được nộp (bài của học sinh)

Học viên	Người hướng dẫn khoa học
Hà	
Nguyễn Thị Thu Hà	TS. Đinh Trường Duy

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ
BƯU CHÍNH VIỄN THÔNG

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BÁO CÁO GIẢI TRÌNH
SỬA CHỮA, HOÀN THIỆN ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

Họ và tên học viên: Nguyễn Thị Thu Hà.....

Chuyên ngành: Hệ thống thông tin .....

Khóa: 2024 đợt I. .....

Tên đề tài: Nghiên cứu và ứng dụng mô hình RAG trong phân tích cảnh báo an ninh mạng

Người hướng dẫn khoa học: TS. Đinh Trường Duy.....

Ngày bảo vệ: 24/01/2026 .....

Các nội dung học viên đã sửa chữa, bổ sung trong đề án tốt nghiệp theo ý kiến đóng góp của Hội đồng chấm đề án tốt nghiệp:

TT	Ý kiến hội đồng	Sửa chữa của học viên
1	Bổ sung thêm khảo sát ứng dụng mô hình RAG trong SOC	Đã bổ sung khảo sát ứng dụng mô hình RAG trong SOC ở mục 1.6 trang 17,18
2	Chỉnh sửa đánh số hình ảnh, bảng biểu, rà soát lỗi chính tả	Đã chỉnh sửa đánh số các hình ảnh, bảng biểu trong bài. Đã rà soát các lỗi chính tả (ví dụ: trang 1, 10,..)
3	Thay đổi các hình ảnh bị mờ	Đã thay đổi các hình ảnh bị mờ (ví dụ: trang 54-55,…)
4	Bổ sung phần phân tích độ chính xác trong báo cáo	Đã bổ sung phân tích độ chính xác ở mục 3.3.2 trang 51-52.

Hà Nội, ngày 10... tháng 02 năm 2026

Ký xác nhận của

CHỦ TỊCH HỘI ĐỒNG CHẤM ĐỀ ÁN	THƯ KÝ HỘI ĐỒNG	NGƯỜI HƯỚNG DẪN KHOA HỌC	HỌC VIÊN
PGS.TS. Hoàng Xuân Dậu	TS. Đặng Hoàng Long	TS. Đinh Trường Duy	Nguyễn Thị Thu Hà

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BIÊN BẢN
HỌP HỘI ĐỒNG CHẤM ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

Căn cứ quyết định số Quyết định số 3299/QĐ-HV ngày 26 tháng 12 năm 2025 của Giám đốc Học viện Công nghệ Bưu chính Viễn thông về việc thành lập Hội đồng chấm đề án tốt nghiệp thạc sĩ. Hội đồng đã họp vào hồi ....8... giờ ..15.. phút, ngày 24 tháng 01 năm 2025 tại Học viện Công nghệ Bưu chính Viễn thông để chấm đề án tốt nghiệp thạc sĩ cho:

Học viên: **Nguyễn Thị Thu Hà**

Tên đề án tốt nghiệp: **Nghiên cứu và ứng dụng mô hình RAG trong phân tích cảnh báo an ninh mạng**

Ngành: **Hệ thống thông tin** Mã số: **8.48.01.04**

Các thành viên của Hội đồng chấm đề án tốt nghiệp có mặt: 05 / 05

TT	HỌ VÀ TÊN	TRÁCH NHIỆM TRONG HĐ	GHI CHÚ
1	**PGS.TS. Hoàng Xuân Dậu**	Chủ tịch	
2	**TS. Đặng Hoàng Long**	Thư ký	
3	**PGS.TS. Tạ Minh Thanh**	Phản biện 1	
4	**TS. Vũ Văn Thỏa**	Phản biện 2	
5	**TS. Trần Đăng Công**	Uỷ viên	

Các nội dung thực hiện:

1. Chủ tịch Hội đồng điều khiển buổi họp. Công bố quyết định của Giám đốc Học viện Công nghệ Bưu chính Viễn thông về việc thành lập Hội đồng chấm đề án tốt nghiệp thạc sĩ.
2. Người hướng dẫn khoa học hoặc thư ký đọc lý lịch khoa học và các điều kiện bảo vệ đề án tốt nghiệp của học viên (Có bản lý lịch khoa học và kết quả các môn học cao học của học viên kèm theo).
3. Học viên trình bày tóm tắt đề án tốt nghiệp.
4. Phản biện 1 đọc nhận xét (có văn bản kèm theo)
5. Phản biện 2 đọc nhận xét (có văn bản kèm theo)
6. Các câu hỏi của thành viên Hội đồng:

Đã làm bao nhiêu kịch bản tấn công trong Bảng 1.8

Trình bày thêm về các câu trúc CSDL

7. Trả lời của học viên:

Học viên chạy 3 thích bản 1 ngày và thực hiện từ 7 ngày. Không gian 5000 vector

Học viên đã trình bày rõ về CSDL

8. Thư ký đọc nhận xét về quá trình thực hiện đề án tốt nghiệp của học viên *(có văn bản kèm theo).*

9. Hội đồng họp riêng:

- Bầu Ban kiểm phiếu:
  1. Trưởng Ban kiểm phiếu: TS. Vũ Văn Thoá
  2. Uỷ viên Ban kiểm phiếu: PGS.TS. Tạ Minh Thanh
  3. Uỷ viên Ban kiểm phiếu: TS. Đặng Hoàng Long
- Hội đồng chấm đề án tốt nghiệp bằng bỏ phiếu kín.
- Ban kiểm phiếu làm việc:
- Trưởng Ban kiểm phiếu báo cáo kết quả kiểm phiếu *(có Biên bản họp Ban kiểm phiếu kèm theo)*
- **Điểm trung bình của đề án tốt nghiệp:** 8,6

Kết luận:

1. Các nội dung cần chỉnh sửa, hoàn thiện sau bảo vệ đề án tốt nghiệp:

Chất lượng báo cáo tốt. Bổ sung thêm thông tin khảo sát ứng dụng mã hình RAF trong SOC

Bổ sung phần phân tích độ chính xác trong báo cáo

2. Đề nghị Học viện công nhận (hoặc không) và cấp bằng (hoặc không) thạc sĩ cho học viên: Công nhận

3. Đề án tốt nghiệp có thể phát triển thành đề tài nghiên cứu cho NCS.....

Buổi làm việc kết thúc vào 21.00 cùng ngày.

Chủ tịch	**Thư ký**
PGS.TS. Hoàng Xuân Dậu	**TS. Đặng Hoàng Long**

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BẢN NHẬN XÉT ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

Tên đề tài đề án:	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
Tên học viên:	Nguyễn Thị Thu Hà
Họ tên người nhận xét:	Vũ Văn Thỏa
Học hàm, học vị:	Tiến sĩ **Chuyên ngành**: Toán học tính toán
Cơ quan công tác:	Học viện Công nghệ Bưu chính Viễn thông

NỘI DUNG NHẬN XÉT

I/ Cơ sở khoa học và thực tiễn, tính cấp thiết của đề tài

Hiện nay, cùng với sự triển khai mạnh mẽ của các dịch vụ trên nền tảng Internet, quy mô và độ phức tạp của các hệ thống thông tin cung cấp dịch vụ ngày càng gia tăng. Kèm theo đó, vấn đề an toàn và bảo mật thông tin cho các hệ thống có ý nghĩa hết sức quan trọng nhằm bảo vệ dữ liệu, ngăn chặn hiệu quả các hành vi xâm nhập và tấn công mạng, dịch vụ. Trong đó, trung tâm điều hành an ninh mạng (SOC) có vai trò hết sức quan trọng. Thời gian gần đây, tội phạm mạng cũng đã sử dụng công nghệ trí tuệ nhân tạo (AI) để tự động hóa tấn công mạng, dịch vụ. Điều đó dẫn đến hệ thống SOC có thể sản sinh ra hàng nghìn cảnh báo mỗi ngày, trong đó tỷ lệ báo động giả (False Positive) rất cao. Bộ phận phân tích thường bị quá tải, dẫn đến nguy cơ bỏ lỡ các cuộc tấn công thực sự. Bài toán ứng dụng RAG (Retrieval-Augmented Generation) giúp tự động sàng lọc, đối chiếu dữ liệu lịch sử để đưa ra nhận định sơ bộ, giảm tải áp lực cho bộ phận phân tích trong SOC được quan tâm nghiên cứu và triển khai thực tế.

Đề án với tiêu đề "Nghiên cứu và ứng dụng mô hình RAG trong phân tích cảnh báo an ninh mạng" của học viên Nguyễn Thị Thu Hà tập trung nghên cứu ứng dụng mô hình RAG trong phân tích cảnh báo an ninh mạng tại các SOC. Vì vậy, đề tài đề án có tính cấp thiết, có ý nghĩa khoa học và thực tiễn.

II/ Về nội dung chất lượng của đề án, những kết quả đạt được

Đề án được trình bày trong 60 trang bao gồm phần mở đầu, 3 chương nội dung, phần kết luận và danh mục các tài liệu tham khảo.

Chương 1 của đề án gồm 13 trang, trình bày tổng quan các kiến thức cơ sở về hệ thống giám sát an ninh mạng, hệ thống cảnh báo an ninh mạng và mô hình RAG. Đề án đã khảo sát tổng quan các phương pháp phân tích cảnh báo an ninh mạng hiện có và chỉ rõ những thách thức dẫn đến sự quá tải của hệ thống cảnh báo an ninh mạng. Đề án cũng khảo sát kiến trúc và đặc trưng của mô hình ngôn ngữ lớn (LLM) và mô hình RAG. Từ đó, nghiên cứu ứng dụng trong hệ thống SOC để giải quyết các vấn đề tồn tại trong hệ thống cảnh báo an ninh mạng. Tuy nhiên các công nghệ trình bày trong chương 1 còn thiên về mô tả, chưa có so sánh định lượng giữa RAG và các phương pháp SOC AI khác (UEBA, XDR, SOAR AI). Mặt khác, Chưa làm

rõ ranh giới "hỗ trợ phân tích" và "tự động ra quyết định" khi sử dụng mô hình RAG trong hệ thống cảnh báo an ninh mạng. Phần kết luận chương chưa có các nội dung liên kết với các chương tiếp theo.

Nội dung chính của chương 2 là nghiên cứu ứng dụng mô hình RAG trong phân tích cảnh báo an ninh mạng được trình bày trong 16 trang. Đề án đã thực hiện thiết kế kiến trúc mô hình RAG trong phân tích cảnh báo an ninh mạng, mô tả luồng dữ liệu và xác định các mô đun chức năng của mô hình. Từ đó, mô hình đề xuất giải pháp triển khai mô hình trong môi trường thử nghiệm. Tuy nhiên, mô hình mới dừng ở mức hỗ trợ ra quyết định về cảnh báo an ninh mạng, chưa có cơ chế phản hồi tự động (auto-response).

Chương 3 của đề án gồm 22 trang, trình bày các nội dung triển khai thử nghiệm mô hình cảnh báo an ninh mạng sử dụng sử dụng mô hình RAG và phân tích, đánh giá kết quả thử nghiệm. Đề án đã thực hiện ba kịch bản thử nghiệm chính và hai thử nghiệm mở rộng. Trên cơ sở đó, đề án đã đánh giá hiệu năng của mô hình và chất lượng kết quả phân tích đầu ra của RAG. Tuy nhiên, phần đánh giá về độ chính xác của các kết quả phân tích cảnh báo an ninh mạng chưa có định lượng rõ ràng. Do đó, các kết quả so sánh SOC truyền thống và SOC tích hợp RAG (trang 52) có độ tin cậy chưa cao. Trong đề án chưa có phân tích sâu rủi ro khi dữ liệu truy xuất không đầy đủ hoặc lỗi thời và chi phí vận hành khi triển khai thực tế đối với doanh nghiệp vừa và nhỏ.

Đề án được chia thành các chương mục rõ ràng và cơ bản bám sát các nội dung theo đề cương đã được phê duyệt. Tuy nhiên, đề án cần đánh số các hình và các bảng kèm theo thứ tự chương theo quy định để thuận tiện cho việc theo dõi. Phần tài liệu tham khảo cần bổ sung thêm các tài liệu về các nghiên cứu thực nghiệm gần đây (2023–2024) về GenAI trong SOC và trình bày theo đúng quy định.

Trong đề án còn có khá nhiều lỗi in ấn phải sửa chữa tại các trang 1, 10, 14, 25, 30, ... Một số hình vẽ còn mờ (trang 53, 54) cần hiệu chỉnh.

III/ Những vấn đề cần giải thích thêm

(1) Giải thích rõ hơn về bài toán phân tích cảnh báo an ninh mạng đặt ra trong đề án: đặc điểm của Input và các yêu cầu cần thiết cho Output?
(2) Nêu chi tiết cấu trúc của cơ sở dữ liệu dùng cho mô hình RAG, các đặc điểm của dữ liệu cảnh báo an ninh mạng?
(3) Giải thích rõ hơn tiêu chí đánh giá về độ chính xác của các thử nghiệm được trình bày trong chương 3 của đề án?

IV/ Kết luận

Tôi đồng ý cho học viên Nguyễn Thị Thu Hà được bảo vệ đề án với đề tài "Nghiên cứu và ứng dụng mô hình RAG trong phân tích cảnh báo an ninh mạng" trước Hội đồng chấm đề án tốt nghiệp thạc sĩ của Học viện. Tuy nhiên, để đảm bảo chất lượng đề án tốt nghiệp thạc sĩ chuyên ngành Hệ thống thông tin, học viên cần chỉnh sửa đề án và giải trình rõ trước Hội đồng các vấn đề được góp ý trên đây.

Ngày 19 tháng 01 năm 2026

Người nhận xét

TS Vũ Văn Thỏa

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

NHẬN XÉT CỦA CÁN BỘ PHẢN BIỆN

Luận văn tốt nghiệp cao học

Họ và tên cán bộ phản biện: TẠ MINH THANH

Công tác chuyên môn: Giảng dạy

Học hàm, học vị: PGS. TS

Đơn vị công tác: Bộ môn An toàn thông tin – Công nghệ mạng, Viện Công nghệ thông tin và Truyền thông, Học viện Kỹ thuật Quân sự.

Nhận xét luận văn tốt nghiệp cao học

Tên luận văn: Nghiên cứu và ứng dụng mô hình RAG trong phân tích cảnh báo an ninh mạng.

Học viên: Nguyễn Thị Thu Hà

Chuyên ngành: Hệ thống thông tin

Mã số: 8 48 01 04

Nội dung luận văn bao gồm phần mở đầu, 3 chương nội dung và phần kết luận. Chương 1 trình bày về tổng quan hệ thống giám sát an ninh mạng và mô hình tạo sinh dựa trên RAG. Chương 2 trình bày về nghiên cứu và ứng dụng mô hình RAG trong phân tích cảnh báo an ninh mạng. Chương 3 mô tả chi tiết thử nghiệm và đánh giá thực nghiệm cho ứng dụng RAG trong cảnh báo. Phần kết luận tổng kết các kết quả làm được và đánh giá các hướng phát triển tiếp theo.

Học viên tập trung vào bài toán ứng dụng công nghệ RAG trên các hệ thống SOC, đặc biệt là mô hình ngôn ngữ lớn được tăng cường bởi cơ chế truy xuất thông tin, mở ra xu hướng mới có khả năng tự động hoá nhằm giảm thiểu hiện tượng "quá tải cảnh báo", từ đó giảm số lượng dương tính giả. Đây là một đề tài có tính thực tiễn cao và cần thiết nghiên cứu để xây dựng hệ thống cảnh báo, giám sát an ninh mạng sử dụng mô hình RAG. Phần trình bày trong luận văn đã cho thấy

tác giả đã tập trung tìm hiểu nội dung lý thuyết liên quan đến bài toán và đề xuất được ứng dụng mô hình RAG trong SOC.

Một số lưu ý cần phải chỉnh sửa:

- Trong Chương 1, học viên cần có những khảo sát những kỹ thuật sử dụng trong SOC để phân tích các cảnh báo mạng hiện nay, từ đó làm rõ vấn đề có quá nhiều cảnh báo dương tính giả gây ảnh hưởng đến quản trị mạng. Bên cạnh đó, cần khảo sát các nghiên cứu hiện nay đang ứng dụng mô hình RAG trong SOC và chỉ ra tính ưu điểm trong việc phân tích hiệu quả thông tin cảnh báo mạng.
- Trong Chương 2, cần thể hiện rõ sơ đồ tổng thể giữa kiến trúc hệ thống RAG trong hệ thống SOC trong giải pháp đề xuất.
- Phân tích rõ vì sao ứng dụng công nghệ RAG trong SOC lại làm giảm thiểu cảnh báo dương tính giả như các hệ thống SOC sử dụng ML/DL hiện nay?
- Trong Chương 3, RAG trong SOC có khả năng sẽ bị nhầm lẫn trong những kịch bản như thế nào?

Kết luận: Mặc dù còn những thiếu sót trong trình bày và ý tưởng đề xuất còn đơn giản, tuy nhiên đây là một đề tài thực tiễn, nội dung luận văn thể hiện học viên đã có những cố gắng trong quá trình thực hiện, thái độ làm việc nghiêm túc.

Ý kiến giáo viên phản biện: ***Cho học viên bảo vệ trước hội đồng.***

Ngày 22 tháng 01 năm 2026

Giáo viên phản biện

PGS. TS. TẠ MINH THANH