ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

TRẦN TUẤN ANH
NGUYỄN KHẮC HẬU

ĐỒ ÁN CHUYÊN NGÀNH

PHÁT TRIỂN PHƯƠNG PHÁP SỬ DỤNG LLM HỖ TRỢ PHÂN TÍCH MÃ ĐỘC TỪ DỮ LIỆU NHỊ PHÂN CÓ CẤU TRÚC VÀ NGỮ NGHĨA

DEVELOPING A METHOD USING LLMS TO SUPPORT MALWARE ANALYSIS FROM STRUCTURED AND SEMANTIC BINARY DATA

NGÀNH AN TOÀN THÔNG TIN

TP. Hồ Chí Minh, Tháng 7, 2025

ĐẠI HỌC QUỐC GIA HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

TRẦN TUẤN ANH - 22520080
NGUYỄN KHẮC HẬU - 22520410

ĐỒ ÁN CHUYÊN NGÀNH

PHÁT TRIỂN PHƯƠNG PHÁP SỬ DỤNG LLM HỖ TRỢ PHÂN TÍCH MÃ ĐỘC TỪ DỮ LIỆU NHỊ PHÂN CÓ CẤU TRÚC VÀ NGỮ NGHĨA

DEVELOPING A METHOD USING LLMS TO SUPPORT MALWARE ANALYSIS FROM STRUCTURED AND SEMANTIC BINARY DATA

NGÀNH AN TOÀN THÔNG TIN

GIẢNG VIÊN HƯỚNG DẪN:
ThS. Đỗ Thị Thu Hiền

TP.Hồ Chí Minh - Tháng 7, 2025

LỜI CẢM ƠN

Trong quá trình nghiên cứu và hoàn thành đồ án, nhóm em đã được cô Đỗ Thị Thu Hiền tận tình hướng dẫn và hỗ trợ. Nhóm xin bày tỏ lời cảm ơn tới thầy, sự chỉ dẫn và góp ý của thầy đã giúp chúng em hoàn thiện tốt nội dung đồ án.

Chúng em cũng chân thành cảm ơn các quý thầy cô trường Đại học Công nghệ Thông tin - ĐHQG TP.HCM, đặc biệt là các thầy cô trong khoa Mạng máy tính và truyền thông đã tạo điều kiện cho bọn em thực hiện và vận dụng kiến thức vào việc triển khai đồ án trong môn học Đồ án chuyên ngành.

TRẦN TUẤN ANH **NGUYỄN KHẮC HẬU**

MỤC LỤC

LỜI CẢM ƠN i
MỤC LỤC ii
DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT iv
DANH MỤC CÁC HÌNH VẼ v
DANH MỤC CÁC BẢNG BIỂU v
MỞ ĐẦU 1

CHƯƠNG 1. TỔNG QUAN **2**
1.1 Giới thiệu vấn đề 2
1.2 Giới thiệu những nghiên cứu liên quan 3
1.2.1 Sử dụng LLM để phát hiện mối đe dọa mạng 3
1.2.2 Sử dụng LLM cho trích xuất TTP từ mã độc 4
1.3 Tính mới và sáng tạo 5
1.4 Mục tiêu, đối tượng, và phạm vi nghiên cứu 5
1.4.1 Mục tiêu nghiên cứu 5
1.4.2 Đối tượng nghiên cứu 5
1.4.3 Phạm vi nghiên cứu 6
1.4.4 Cấu trúc đồ án chuyên ngành 6

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT **8**
2.1 Chiến thuật, kỹ thuật và quy trình - TTP 8
2.1.1 Chiến thuật (Tactics) 8
2.1.2 Kỹ thuật (Techniques) 9
2.1.3 Quy trình (Procedures) 10
2.2 Mô hình Ngôn ngữ lớn 10
2.2.1 Tổng quan 10

2.2.2 Hiện trạng Ứng dụng LLM vào Phân tích Mã độc 12
2.3 Retrieval-Augmented Generation 12
2.3.1 Định nghĩa và Nguyên tắc hoạt động 12
2.3.2 Kiến trúc và Luồng hoạt động 13

CHƯƠNG 3. PHƯƠNG PHÁP LUẬN VÀ THIẾT KẾ HỆ THỐNG 16
3.1 Phương pháp luận tổng quan 16
3.2 Thiết kế hệ thống 17
3.2.1 Tổng quan kiến trúc hệ thống 17
3.2.2 Chi tiết kiến trúc hệ thống 17

CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNH GIÁ 28
4.1 Thiết lập thực nghiệm 28
4.1.1 Môi trường 28
4.1.2 Tập dữ liệu 28
4.1.3 Xây dựng cơ sở dữ liệu vector cho RAG 31
4.1.4 Tạo báo cáo phân tích tĩnh và động từ mã độc 32
4.2 Kết quả thực nghiệm 33
4.2.1 Độ chính xác của hệ thống (Câu hỏi 1) 34
4.2.2 Thời gian xử lý của hệ thống (Câu hỏi 2) 36
4.2.3 Hiệu suất hệ thống trên các LLM khác nhau (Câu hỏi 3) . 37
4.3 Triển khai Giao diện Web 37

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 40
5.1 Kết luận 40
5.2 Hướng phát triển 41

TÀI LIỆU THAM KHẢO 43

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

LLM	Large Language Model - Mô hình Ngôn ngữ lớn
RAG	Retrieval-Augmented Generation
TTP	Tactics, Techniques, Procedures
APT	Advanced Persistent Threat
API	Application Programming Interface
IOC	Indicators of Comprise
SOC	Security Operations Center
CR	Coverage Rate
MRR	Mean Reciprocal Rank
PE	Portable Executable

DANH MỤC CÁC HÌNH VẼ

Hình 2.1 Các mô hình Ngôn ngữ lớn 11
Hình 2.2 Kiến trúc và Luồng hoạt động của RAG 13

Hình 3.1 Kiến trúc hệ thống 17
Hình 3.2 Khối đầu vào 18
Hình 3.3 Khối Tiền xử lý và Trích xuất Đặc trưng 19
Hình 3.4 Khối Tăng cường Ngữ cảnh 20
Hình 3.5 Ví dụ kết quả trả về của luồng Threat Intelligence 22
Hình 3.6 Khối suy luận trung tâm 22
Hình 3.7 Khối đầu ra 26

Hình 4.1 Giao diện của trang chứa thông tin của các nhóm APT . . . 30
Hình 4.2 Thông tin về các kỹ thuật mà nhóm APT1 sử dụng 30
Hình 4.3 Kết quả trả về sau khi truy vấn một mẫu mã độc 31
Hình 4.4 Một phần của tệp dữ liệu thu thập được 31
Hình 4.5 Giao diện khi bắt đầu của trang Web 38
Hình 4.6 Hệ thống đang tiến hành trích xuất 38
Hình 4.7 Kết quả sau khi trích xuất 39

DANH MỤC CÁC BẢNG BIỂU

Bảng 4.1 Bảng thống kê số lượng mẫu mã độc nhóm sử dụng từ APT Malware Dataset. 29
Bảng 4.2 Bảng thống kê tập dữ liệu sử dụng để thực nghiệm câu hỏi 1 và 2. 33
Bảng 4.3 Bảng kết quả so sánh độ chính xác sinh TTP (LLM sử dụng là Gemini 2.0 Flash). 34
Bảng 4.4 Bảng kết quả so sánh độ chính xác sinh danh sách nhóm APT đề xuất (LLM sử dụng là Gemini 2.0 Flash). 35
Bảng 4.5 Bảng kết quả so sánh thời gian xử lý (LLM sử dụng là Gemini 2.0 Flash). 36
Bảng 4.6 Bảng kết quả so sánh hiệu suất trên các LLM khác nhau. . 37

TÓM TẮT ĐỒ ÁN

Trong thời gian gần đây, các cuộc tấn công mạng APT ngày càng phổ biến và tinh vi, với mã độc nhị phân là công cụ tấn công quan trọng tạo ra khối lượng dữ liệu lớn, đa dạng và khó diễn giải, đòi hỏi chuyên gia kinh nghiệm để tổng hợp thành báo cáo ngữ nghĩa rõ ràng. Cùng với đó, sự phát triển vượt bậc của các mô hình ngôn ngữ lớn (LLM) đã mở ra nhiều hướng nghiên cứu mới trong lĩnh vực an toàn thông tin, đặc biệt là trong phân tích mã độc. Khác với các hệ thống phát hiện mã độc truyền thống vốn dựa trên đặc trưng thủ công và học máy cổ điển, mô hình LLM có khả năng hiểu ngữ cảnh sâu, xử lý dữ liệu nhị phân theo hướng ngữ nghĩa, từ đó hỗ trợ phân tích hành vi mã độc chính xác hơn.

Đồng thời, khung kiến thức MITRE ATT&CK ngày càng được ứng dụng phổ biến như một chuẩn tham chiếu nhằm xác định và phân loại các chiến thuật và kỹ thuật tấn công (TTP). Một số nghiên cứu gần đây đã khai thác LLM để sinh ra TTP từ mã độc thông dịch. Tuy nhiên, số lượng nghiên cứu áp dụng ý tưởng tương tự lên mã độc nhị phân vẫn còn ít.

Từ những phân tích hiện có, trong nghiên cứu này, chúng tôi đề xuất một phương pháp sử dụng LLM nhằm tự động hóa quy trình trích xuất các nhóm chiến thuật, kỹ thuật và thủ tục tấn công (TTP) từ dữ liệu phân tích tĩnh và động của mã độc nhị phân. Ngoài ra, hệ thống còn có khả năng suy luận và đề xuất nhóm APT khả nghi có liên quan dựa trên tập mẫu TTP được phát hiện, góp phần hỗ trợ quá trình điều tra và phản ứng sự cố.

CHƯƠNG 1. TỔNG QUAN

Chương này giới thiệu về vấn đề và các nghiên cứu liên quan. Đồng thời, trong chương này chúng tôi cũng trình bày phạm vi và cấu trúc của Đồ án.

1.1. Giới thiệu vấn đề

Ngày nay, khi các cuộc tấn công mạng có chủ đích Advanced Persistent Threats (APT) ngày càng phổ biến và tinh vi, mã độc nhị phân đã trở thành công cụ tấn công quan trọng được các nhóm đối tượng nguy hiểm sử dụng để xâm nhập, chiếm quyền điều khiển và duy trì hiện diện lâu dài trong hệ thống mục tiêu. Báo cáo của AV-TEST cho thấy mỗi ngày có hơn 450.000 mẫu mã độc mới được phát hiện, trong đó một tỷ lệ lớn là các biến thể nhị phân được đóng gói, làm rối và biến hình liên tục [6]. Trong bối cảnh đó, yêu cầu phân tích nhanh và chính xác mã độc nhị phân không chỉ đóng vai trò phát hiện, mà còn là cơ sở để xác định kỹ thuật và chiến thuật tấn công (TTPs), từ đó truy vết và dự đoán APT groups tiềm ẩn đứng sau.

Hiện nay, quy trình phân tích mã độc nhị phân trong thực tế chủ yếu dựa trên sự kết hợp giữa phân tích tĩnh (static analysis) và phân tích động (dynamic analysis). Phân tích tĩnh cho phép trích xuất thông tin về chuỗi API, chữ ký, chuỗi mã nhị phân và các chỉ báo tấn công tiềm năng, trong khi phân tích động quan sát hành vi thực thi trong sandbox để phát hiện các hoạt động đáng ngờ. Tuy nhiên, hai phương pháp này thường tạo ra khối lượng dữ liệu rất lớn, đa dạng định dạng và khó diễn giải, đòi hỏi các chuyên gia có kinh nghiệm cao mới tổng hợp được thành báo cáo ngữ nghĩa rõ ràng. Hơn nữa, việc thủ công liên kết thông tin giữa kết quả phân tích và cơ sở tri thức như MITRE ATT&CK [5] để rút ra TTPs đặc trưng vẫn tiêu tốn rất nhiều thời gian và dễ xảy ra sai sót.

Trong khi đó, các nghiên cứu về mô hình ngôn ngữ lớn (LLM – Large Language Models) thời gian gần đây đã đạt được nhiều thành tựu nổi bật, đặc biệt trong khả năng xử lý ngữ nghĩa tự nhiên, trích xuất thông tin và sinh văn bản từ dữ liệu đầu vào phức tạp. Điển hình, công trình GENTTP đã chứng minh việc sử dụng LLM để trích xuất TTP từ mã độc thông dịch (như gói Python độc hại) đạt độ chính xác cao trên hơn 3.700 mẫu malware PyPI [18]. Tuy nhiên, việc ứng dụng LLM trong bối cảnh mã độc nhị phân – vốn phức tạp hơn rất nhiều về mặt cấu trúc dữ liệu và hành vi runtime – hiện vẫn chưa được nghiên cứu và khai thác đầy đủ.

Hơn nữa, các công trình trước đây phần lớn mới dừng lại ở việc phân loại mã độc hoặc sinh nhãn nhị phân (malicious/benign), chưa có giải pháp nào có thể tự động tổng hợp dữ liệu từ phân tích tĩnh – động thành TTPs hoàn chỉnh, đưa ra đề xuất các APT groups liên quan. Điều này tạo ra khoảng trống nghiên cứu đáng kể khi nhu cầu điều tra, truy vết tác nhân đe dọa trên quy mô lớn ngày càng cấp thiết, nhất là trong môi trường hệ thống doanh nghiệp và hạ tầng trọng yếu.

Từ những phân tích trên, trong nghiên cứu này, chúng tôi đề xuất một phương pháp sử dụng LLM để tự động hóa quá trình trích xuất TTP từ dữ liệu phân tích tĩnh – động của mã độc nhị phân, đồng thời suy luận và đề xuất nhóm APT khả nghi liên quan dựa trên các mẫu TTP.

1.2. Giới thiệu những nghiên cứu liên quan

1.2.1. Sử dụng LLM để phát hiện mối đe dọa mạng

Các mô hình ngôn ngữ lớn (Large Language Models – LLMs) ngày càng được ứng dụng rộng rãi trong lĩnh vực an ninh mạng nhờ khả năng xử lý ngôn ngữ tự nhiên, phân tích ngữ nghĩa và học chuyển tiếp mạnh mẽ. LLM hỗ trợ hiệu quả trong nhiều nhiệm vụ bảo mật như:

- **Phân tích mã độc:** LLM có thể phân tích mã nhị phân, opcode hoặc biểu diễn ảnh (ví dụ PE bitmap) để phát hiện các loại mã độc. [11] [9].
- **Sinh mã độc mô phỏng:** LLM có thể được khai thác để sinh ra mã độc mẫu hoặc biến thể dùng trong huấn luyện mô hình phát hiện. Điều này giúp mô phỏng kịch bản tấn công thực tế và tăng cường khả năng phát hiện các mối đe dọa mới [13] [10].
- **Trích xuất thông tin mối đe dọa (CTI):**LLMs được ứng dụng để trích xuất chỉ báo tấn công (IoC), thực thể đe dọa và kỹ thuật từ các báo cáo CTI phi cấu trúc, từ đó ánh xạ sang khung ATT&CK hoặc CWE [7].
- **Phát hiện tấn công dạng văn bản:** Các mô hình LLM đã được sử dụng để phân loại nội dung tin nhắn, email phishing [12] [15].
- **Tích hợp vào hệ thống phát hiện xâm nhập (IDS):** LLM được tích hợp trong các hệ thống phát hiện xâm nhập để xử lý log, luồng mạng và sự kiện hệ thống, giúp phát hiện hành vi bất thường và hỗ trợ phản hồi tự động. [17].

Tổng thể, các nghiên cứu cho thấy LLM đóng vai trò ngày càng quan trọng trong việc nâng cao khả năng phát hiện, phân tích và phòng thủ trước các mối đe dọa an ninh mạng hiện đại.

1.2.2. Sử dụng LLM cho trích xuất TTP từ mã độc

Tận dụng tiềm năng như đã nói ở trên của LLM để giải quyết khó khăn trong việc phân tích và hiểu hành vi của mã độc, một nghiên cứu dã đề xuất phương pháp GENTTP [18] được đề xuất cho phép tự động sinh TTP từ mã độc thông dịch (interpreted) hay gói phần mềm (package) độc hại mà không cần dữ liệu huấn luyện (zero-shot), bằng cách phân tích metadata và mã nguồn rồi sử dụng kỹ thuật prompt engineering để sinh ra các chuỗi hành vi tấn công. Thử nghiệm

trên hơn 3,700 gói mã độc PyPI cho thấy GENTTP đạt độ chính xác cao (trên 90%) trong việc trích xuất và mô tả TTPs.

1.3. Tính mới và sáng tạo

Hiện tại không có nhiều nghiên cứu hướng đến việc áp dụng ý tưởng của GENTTP vào mã độc nhị phân nên trong đồ án này nhóm sẽ đề xuất một hệ thống thực hiện việc trích xuất TTP từ những báo cáo phân tích tĩnh và động để hỗ trợ phân tích mã độc nhị phân. Bên cạnh đó trong hệ thống của chúng tôi, không chỉ đơn thuần sử dụng LLM để trích xuất TTP mà còn sử dụng thêm Retrieval-Augmented Generation (RAG) và thông tin từ Threat Intelligence để tăng tính chính xác. Đồng thời, từ những TTP trích xuất được, chúng tôi sẽ đề xuất những nhóm tấn công APT có liên quan đến mã độc nhị phân đang phân tích.

1.4. Mục tiêu, đối tượng, và phạm vi nghiên cứu

1.4.1. Mục tiêu nghiên cứu

Ứng dụng LLM để trích xuất TTP từ báo cáo phân tích tĩnh và động để hỗ trợ phân tích mã độc nhị phân, đông thời đề xuất nhóm APT liên quan.

1.4.2. Đối tượng nghiên cứu

Đối tượng nghiên cứu:

- Mã độc nhị phân
- Mô hình ngôn ngữ lớn
- Kiến trúc RAG

1.4.3. Phạm vi nghiên cứu

Đề tài tập trung phát triển phương pháp sử dụng mô hình ngôn ngữ lớn (LLM) kết hợp với RAG để trích xuất TTP từ kết quả phân tích mã độc nhị phân và đề xuất nhóm APT tương ứng với TTP. Phạm vi cụ thể như sau:

- **Loại mã độc:** Các mẫu mã độc thực thi phổ biến như PE (Windows).
- **Dữ liệu đầu vào:**
  - **Dữ liệu có cấu trúc:** API call sequence, system call trace.
  - **Dữ liệu có ngữ nghĩa:** nhật ký hành vi (sandbox log), mô tả kỹ thuật, báo cáo phân tích.
- **Vai trò của LLM:** Tự động nhận diện hành vi, ánh xạ theo khung MITRE ATT&CK, và sinh TTP tương ứng.
- **Giới hạn:**
  - Không xử lý mã độc đã bị packer hoặc obfuscation quá mạnh.
  - Không xây dựng hệ thống phát hiện mã độc từ đầu.
- **Ứng dụng:** Hỗ trợ phân tích mã độc chuyên sâu, phục vụ điều tra số và tình báo mối đe dọa (Threat Intelligence) trong hệ thống SOC.

1.4.4. Cấu trúc đồ án chuyên ngành

Chúng tôi xin trình bày nội dung của Đồ án theo cấu trúc như sau:

- Chương 1: Giới thiệu tổng quan về đề tài của Đồ án và những nghiên cứu liên quan.
- Chương 2: Trình bày cơ sở lý thuyết liên quan đến đề tài.
- Chương 3: Trình bày phương pháp luận và thiết kế hệ thống cho phương pháp đề xuất.

- Chương 4: Trình bày thực nghiệm và đánh giá.
- Chương 5: Kết luận và hướng phát triển của đề tài.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Chương này trình bày cơ sở lý thuyết của nghiên cứu: Bao gồm chiến thuật, kỹ thuật và quy trình (TTP), mô hình Ngôn ngữ lớn và kiến trúc RAG.

2.1. Chiến thuật, kỹ thuật và quy trình - TTP

Chiến thuật, kỹ thuật và quy trình hay TTP là cách thức mà các tác nhân đe dọa sử dụng để xâm nhập, kiểm soát và duy trì hoạt động trong hệ thống mục tiêu. TTP là nền tảng để hiểu hành vi mã độc, hỗ trợ phân tích mối đe dọa và phản ứng sự cố. Khung chuẩn MITRE ATT&CK giúp chuẩn hóa và phân loại TTP thành: Chiến thuật (Tactics), Kỹ thuật (Techniques) và Quy trình (Procedures).

2.1.1. Chiến thuật (Tactics)

Chiến thuật của một tác nhân đe dọa mô tả cách thức mà họ hành động trong các giai đoạn khác nhau của chuỗi tấn công mạng (cyber kill chain). Các giai đoạn này bao gồm:

- Trinh sát (Reconnaissance)
- Phân phối và khai thác (Delivery and exploitation)
- Hành động dựa trên các mục tiêu (Acting on the objectives)

Mức độ khó khăn trong việc quy kết một mối đe dọa tiềm ẩn cho một chiến dịch cụ thể phụ thuộc vào mức độ mới mẻ và tinh vi của cuộc tấn công. Nếu các chỉ báo mối đe dọa cho thấy các mẫu tấn công phổ biến - chẳng hạn như

tấn công từ chối dịch vụ (DDoS) - thì các giai đoạn tiếp theo của chiến thuật tấn công có thể được dự đoán bằng cách thu thập dữ liệu về các yếu tố như:

- Điểm xâm nhập ban đầu
- Các nút hoặc thông tin xác thực đã bị xâm phạm

Một chiến thuật tấn công tinh vi tập trung vào việc duy trì sự ẩn mình, chỉ thực hiện những thay đổi nhỏ trong mạng bị xâm nhập cho đến khi tải trọng độc hại được triển khai hoặc dữ liệu quan trọng bị trích xuất và gửi về máy chủ điều khiển từ xa (command and control server).

2.1.2. Kỹ thuật (Techniques)

Các kỹ thuật là những hành động mà các tác nhân đe dọa thực hiện nhằm gây ra nhiều hậu quả nghiêm trọng, chẳng hạn như:

- Xâm nhập vào mạng.
- Thiết lập các trung tâm điều khiển và chỉ huy.
- Di chuyển ngang trong mạng mà không để lại dấu vết.
- Phát tán phần mềm độc hại trên nhiều vị trí mạng phân tán.
- Thiết lập quyền kiểm soát để thực hiện các thay đổi hạ tầng và truyền dữ liệu mà không thể truy vết.

Những kỹ thuật này thường mang tính chất tổng quát và có thể được áp dụng cho bất kỳ chiến dịch tấn công mạng nào. Kỹ thuật có thể không chỉ rõ công nghệ cụ thể, mà chỉ tập trung vào phương pháp của chiến dịch và hướng dẫn trình tự các hành động liên quan.

Ví dụ, một chiến thuật lừa đảo nhắm mục tiêu (spear phishing) dựa trên kỹ thuật xã hội có thể được sử dụng để đánh lừa người dùng không cảnh giác nhấp vào một liên kết, từ đó tải xuống mã độc vào máy tính và đánh cắp thông tin

đăng nhập. Kỹ thuật này có thể được thiết kế đặc biệt nhằm tấn công một nhóm người dùng hạn chế để làm cho cuộc tấn công kỹ thuật xã hội trở nên thuyết phục hơn.

2.1.3. Quy trình (Procedures)

Các quy trình là phần mô tả chi tiết về cách thức các chiến thuật được thực thi thông qua việc lựa chọn các kỹ thuật và một tập hợp các hành động có thể thực hiện được, được thiết kế cẩn thận và chính xác.

Những hành động này thường được tùy chỉnh cao và quá trình thực hiện được tài liệu hóa đầy đủ để các tác nhân đe dọa có thể làm theo chính xác theo đặc tả. Các hành động này có xu hướng phức tạp nhưng được lặp đi lặp lại thường xuyên.

2.2. Mô hình Ngôn ngữ lớn

2.2.1. Tổng quan

Mô hình Ngôn ngữ Lớn (Large Language Model - LLM) là một loại mô hình học sâu (deep learning) được đặc trưng bởi quy mô khổng lồ, với hàng tỷ tham số (parameters). Chúng được huấn luyện trên một kho dữ liệu văn bản cực kỳ lớn từ Internet và các nguồn khác, cho phép chúng học được các mẫu câu, ngữ pháp, kiến thức và khả năng suy luận phức tạp của ngôn ngữ tự nhiên.

Kiến trúc Transformer là kiến trúc cốt lõi sau hầu hết các LLM hiện đại. Thành công của kiến trúc này đến từ việc loại bỏ hoàn toàn cấu trúc hồi quy và thay thế bằng một cơ chế hiệu quả hơn.

- **Cơ chế Self-Attention (Tự chú ý):** Đây được coi là "trái tim" của Transformer. Khi xử lý một từ trong câu, cơ chế self-attention cho phép mô hình cân nhắc và đánh giá mức độ quan trọng của tất cả các từ khác trong cùng một câu. Điều này giúp mô hình nắm bắt được các mối quan hệ ngữ nghĩa

Hình 2.1: *Các mô hình Ngôn ngữ lớn*

phức tạp và các phụ thuộc xa giữa các từ, một khả năng mà RNN và LSTM gặp nhiều khó khăn.

Ví dụ, trong câu *"Con mèo ngồi trên tấm thảm, nó đang ngủ"*, cơ chế *attention* giúp mô hình hiểu rằng "nó" đang ám chỉ đến "con mèo" chứ không phải "tấm thảm".

- **Mô hình Encoder-Decoder:** Kiến trúc Transformer ban đầu bao gồm hai phần chính: *Encoder* (bộ mã hóa) để "đọc" và hiểu chuỗi đầu vào, và *Decoder* (bộ giải mã) để tạo ra chuỗi đầu ra. Các mô hình sau này đã được chuyên biệt hóa:
  - **Encoder-only (ví dụ: BERT):** Rất mạnh trong các tác vụ yêu cầu sự hiểu biết sâu sắc về ngữ cảnh của toàn bộ câu, như phân loại văn bản, nhận dạng thực thể,...
  - **Decoder-only (ví dụ: GPT):** Được tối ưu hóa cho các tác vụ sinh văn bản (*text generation*), như viết tiếp một câu chuyện, trả lời câu hỏi, hoặc tóm tắt văn bản. Hầu hết các LLM hiện đại đều dựa trên kiến trúc này.

2.2.2. Hiện trạng Ứng dụng LLM vào Phân tích Mã độc

Việc ứng dụng LLM trong lĩnh vực này đã có những bước tiến đáng kể:

- **Giai đoạn đầu:** Các nghiên cứu tập trung vào các mô hình dựa trên BERT (như CyberBERT), được tiền huấn luyện trên dữ liệu an ninh mạng, để thực hiện các tác vụ như phân loại họ malware hoặc nhận dạng các chỉ số tấn công (IoCs).
- **Giai đoạn hiện tại:** Các nghiên cứu đang khám phá khả năng của các LLM lớn hơn (như GPT-4, Gemini) để thực hiện các tác vụ phức tạp hơn như:
  - Tóm tắt các báo cáo phân tích mối đe dọa.
  - Tự động hóa phân tích.
  - Viết Rule và Gợi ý Phòng thủ.
- **Hạn chế hiện tại:** Các nghiên cứu vẫn đối mặt với những thách thức về độ chính xác, khả năng xử lý dữ liệu nhị phân thô, và đặc biệt là hiện tượng "ảo giác" (hallucination) - tức là LLM có thể tự tạo ra thông tin không có thật.

2.3. Retrieval-Augmented Generation

2.3.1. Định nghĩa và Nguyên tắc hoạt động

Để giải quyết các vấn đề về việc tạo ra thông tin không chính xác (hallucination) trong các LLM và khả năng hạn chế trong việc truy cập kiến thức mới nhất, Retrieval-Augmented Generation - RAG đã được giới thiệu, là một kiến trúc tiên tiến được thiết kế để nâng cao hiệu suất của LLM bằng cách kết hợp chúng với một cơ sở dữ liệu tri thức bên ngoài. Thay vì chỉ dựa vào kiến thức đã được trong quá trình huấn luyện, RAG cho phép LLM truy cập và sử dụng thông tin cập nhật, chuyên biệt và đáng tin cậy tại thời điểm sinh ra câu trả lời.

Quy trình cốt lõi của RAG tuân theo ba bước chính:

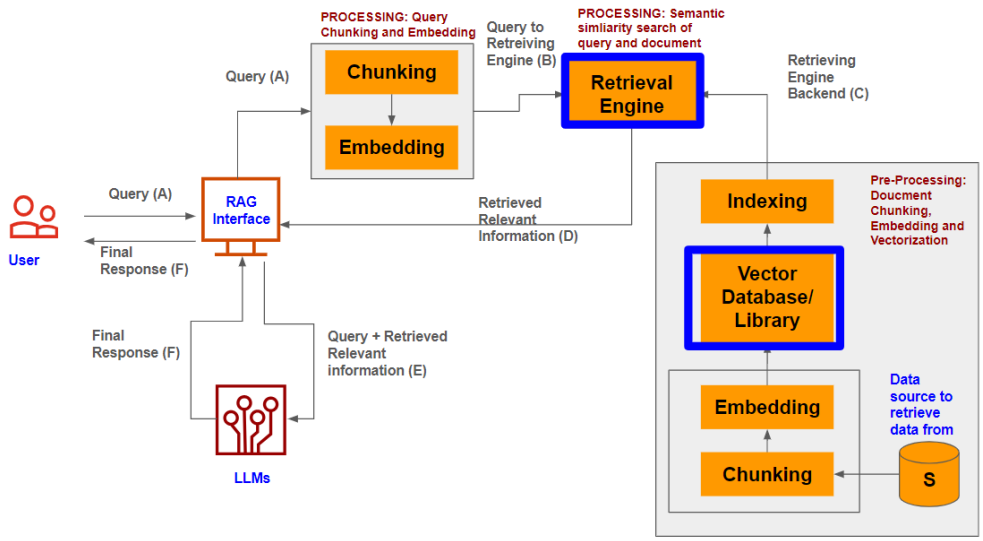
1. **Retrieve (Truy vấn)**: Khi nhận được một câu hỏi hoặc yêu cầu (*prompt*) từ người dùng, hệ thống không gửi nó trực tiếp đến LLM. Thay vào đó, nó sử dụng yêu cầu này để tìm kiếm trong một cơ sở dữ liệu tri thức đã được chuẩn bị trước (thường là một *vector database*) nhằm tìm ra những thông tin liên quan nhất.

2. **Augment (Bổ sung)**: Các thông tin liên quan được tìm thấy sẽ được tự động chèn vào câu lệnh ban đầu, tạo thành một "prompt đã được bổ sung ngữ cảnh" (*augmented prompt*).

3. **Generate (Sinh)**: Prompt đã được bổ sung này sau đó mới được gửi đến LLM. Với ngữ cảnh đầy đủ và chính xác, LLM có thể tạo ra một câu trả lời chất lượng cao, bám sát vào sự thật và giảm thiểu đáng kể các lỗi suy diễn.

2.3.2. Kiến trúc và Luồng hoạt động

Một hệ thống RAG điển hình bao gồm hai giai đoạn chính:



Hình 2.2: *Kiến trúc và Luồng hoạt động của RAG*

- **Giai đoạn Indexing (Lập chỉ mục):** Đây là quá trình chuẩn bị cơ sở dữ

liệu tri thức.

1. **Tải và Phân đoạn Dữ liệu (Load & Chunk):** Các tài liệu (ví dụ: mô tả các TTP từ MITRE, các bài phân tích malware) được tải và chia thành các đoạn nhỏ (chunks) để dễ dàng xử lý và tìm kiếm.
2. **Tạo Embeddings (Vector hóa):** Mỗi chunk dữ liệu văn bản được chuyển đổi thành một vector số học (embedding) bằng một mô hình embedding chuyên dụng (ví dụ: text-embedding-004 của Google). Các vector này biểu diễn ngữ nghĩa của văn bản, trong đó các đoạn văn bản có ý nghĩa tương tự sẽ có các vector nằm gần nhau trong không gian đa chiều.
3. **Lưu trữ vào Vector Database:** Các vector và nội dung văn bản gốc tương ứng được lưu trữ vào một cơ sở dữ liệu vector như ChromaDB, Pinecone, hoặc Milvus. Cơ sở dữ liệu này được tối ưu hóa cho việc tìm kiếm tương đồng ngữ nghĩa cực kỳ nhanh chóng.

- **Giai đoạn Retrieval và Generation (Truy vấn và sinh):** Đây là quá trình xử lý khi có một yêu cầu mới.

1. **Truy vấn của Người dùng:** Đầu vào là một đoạn mô tả hoặc một câu hỏi từ người dùng.
2. **Vector hóa Truy vấn:** Hành vi mới này được chuyển đổi thành một vector truy vấn bằng cùng một mô hình embedding.
3. **Tìm kiếm Tương đồng:** Hệ thống truy vấn vào Vector Store bằng vector này để tìm ra k đoạn văn bản (chunks) có vector gần nhất (liên quan nhất về mặt ngữ nghĩa).
4. **Tạo Prompt Bổ sung:** Các đoạn văn bản liên quan này được kết hợp với câu hỏi ban đầu để tạo thành một prompt hoàn chỉnh, giàu ngữ cảnh.

5. **Sinh Kết quả bởi LLM:** Prompt cuối cùng được gửi đến LLM để tạo ra câu trả lời cuối cùng.

CHƯƠNG 3. PHƯƠNG PHÁP LUẬN VÀ THIẾT KẾ HỆ THỐNG

Ở chương này trình bày chi tiết về phương pháp luận nghiên cứu và kiến trúc hệ thống được đề xuất để hỗ trợ phân tích mã độc. Phương pháp cốt lõi là xây dựng một pipeline tự động, kết hợp các kỹ thuật trích xuất đặc trưng, truy vấn cơ sở tri thức, và suy luận của LLM để ánh xạ các hành vi của mã độc sang các TTP tương ứng trong bộ khung MITRE ATT&CK.

3.1. Phương pháp luận tổng quan

Chúng tôi có tham khảo kiến trúc GENTTP của nhóm Zhang [18]. Qua đó, nhóm nhận thấy mô hình này sinh TTP chỉ dựa vào suy luận nội tại của LLM. Điều này khi áp dụng trong phân tích mã độc nhị phân - vốn phân tích phức tạp hơn mã độc thông dịch - sẽ dễ gây sinh ra TTP sai hoặc không tồn tại.

Phương pháp nhóm đề xuất là một hệ thống được thiết kế để khắc phục các điểm yếu của LLM như hiện tượng "ảo giác" (hallucination) và thiếu kiến thức cập nhật. Thay vì chỉ dựa vào khả năng suy luận nội tại của LLM, hệ thống sẽ ràng buộc các phán đoán của mô hình vào hai nguồn dữ liệu đã được kiểm chứng:

1. **Cơ sở tri thức nội bộ (Internal Knowledge Base):** Sử dụng kiến trúc Retrieval-Augmented Generation (RAG) để truy vấn một cơ sở dữ liệu vector chứa các cặp {hành vi đã biết} $\longrightarrow$ {TTP tương ứng}. Điều này cung cấp cho LLM các ví dụ tương tự để tham chiếu.

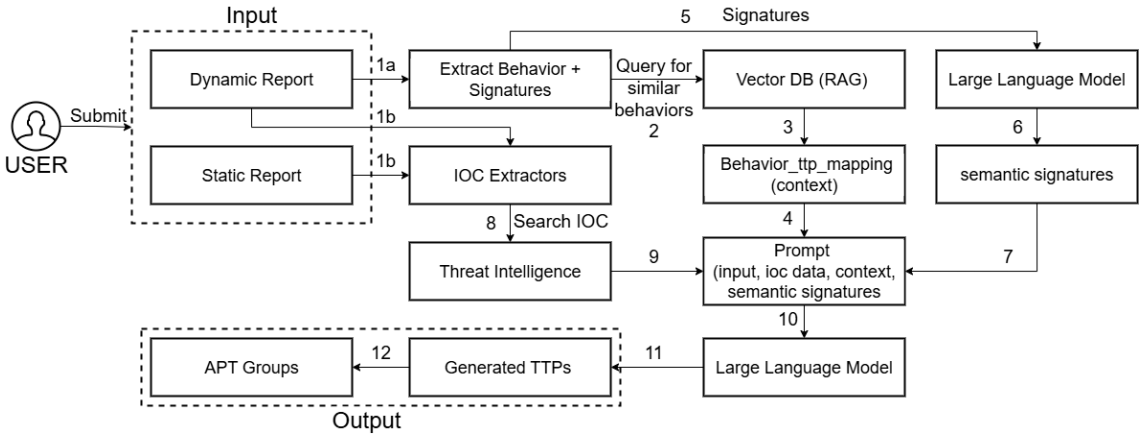
2. **Dữ liệu Tình báo Mối đe dọa bên ngoài (External Threat Intelligence):** Các chỉ số tấn công (IoCs) được trích xuất từ báo cáo phân tích

tĩnh và động sau đó được truy vấn trên các nền tảng tình báo đe dọa để làm giàu thêm ngữ cảnh về mã độc.

3.2. Thiết kế hệ thống

3.2.1. Tổng quan kiến trúc hệ thống

Với phương pháp luận đã trình bày ở mục 3.1, kiến trúc hệ thống của phương pháp như sau:



Hình 3.1: *Kiến trúc hệ thống*

Luồng xử lý tổng thể sẽ lấy đầu vào là các báo cáo phân tích tĩnh và động, tiền xử lý chúng để trích xuất hành vi, chữ ký hành vi và IoCs, làm giàu thông tin bằng ba luồng tăng cường ngữ cảnh, và cuối cùng tổng hợp tất cả thông tin vào một prompt chi tiết để LLM đưa ra kết quả cuối cùng.

3.2.2. Chi tiết kiến trúc hệ thống

3.2.2.1. Khối Đầu vào

Là khối gồm những dữ liệu thô mà người dùng đưa vào, dữ liệu này gồm 2 thành phần:

- **Báo cáo Phân tích động (Dynamic Report):** Đây là kết quả từ việc

thực thi mã độc trong môi trường an toàn (sandbox). Dữ liệu bao gồm:

  – Chuỗi các lời gọi API.

  – Các thay đổi trên hệ thống file.

  – Các thay đổi trên Registry.

  – Lưu lượng mạng được tạo ra.

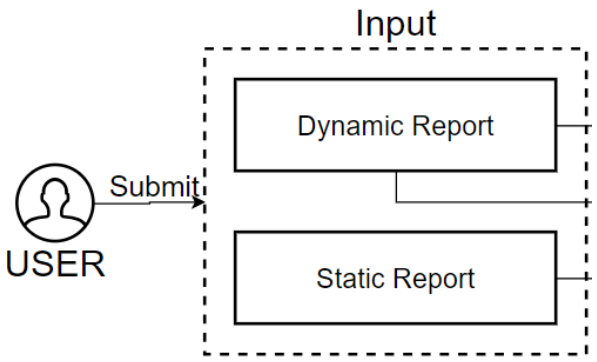
- **Báo cáo Phân tích tĩnh (Static Report):** Đây là kết quả từ việc phân tích file nhị phân mà không cần thực thi. Dữ liệu bao gồm:

  – Mã hash của mã độc nhị phân.

  – Các strings đáng ngờ.

  – Các API đáng ngờ.

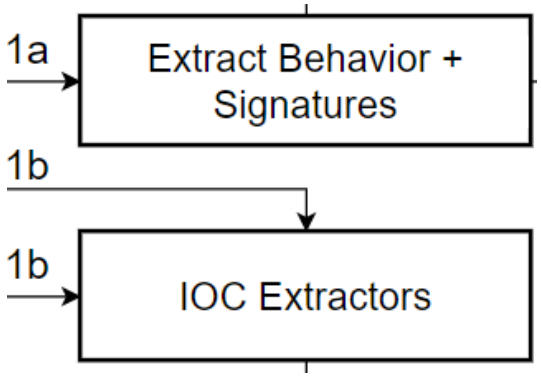
  – Các dấu hiệu của việc mã độc bị nén.



Hình 3.2: *Khối đầu vào*

3.2.2.2. Khối Tiền xử lý và Trích xuất Đặc trưng

Khối này chịu trách nhiệm chuyển đổi dữ liệu thô từ các báo cáo thành các đặc trưng có cấu trúc và ngữ nghĩa.



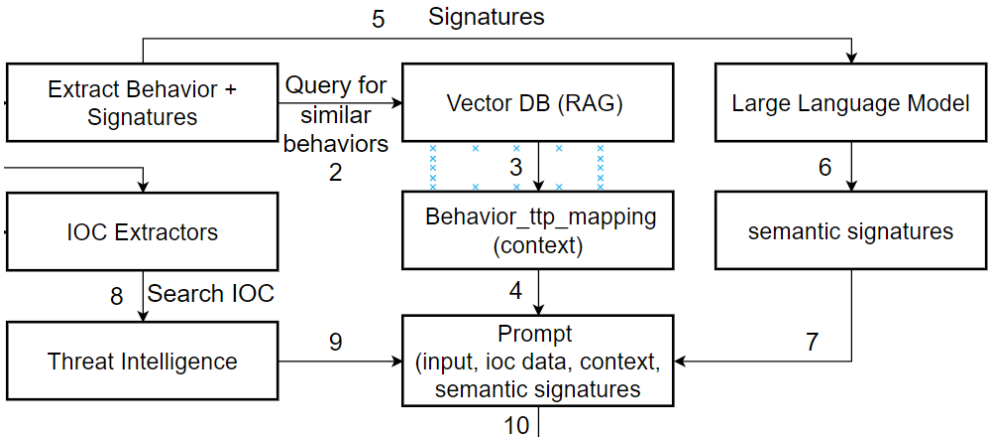
Hình 3.3: *Khối Tiền xử lý và Trích xuất Đặc trưng*

Extract Behavior + Signatures (Bước 1a): Với dữ liệu được đưa vào là các thông tin phân tích động, khối này có tác dụng phân tích cú pháp các log từ phân tích động để nhận diện các hành vi, các chữ ký hành vi (signatures) như tên API, section đáng ngờ, các dấu hiệu của kỹ thuật làm rối được phát hiện.

IOC Extractors (Bước 1b): Sử dụng các biểu thức chính quy (regular expressions) trong thư viện iocseacher [8] để tự động trích xuất các chỉ số tấn công (IoC) như: địa chỉ IP, tên miền, url từ cả báo cáo phân tích tĩnh và động.

3.2.2.3. Khối Tăng cường Ngữ cảnh

Khối này chịu trách nhiệm cung cấp thêm ví dụ, làm giàu thêm thông tin và ngữ cảnh về mã độc để cung cấp cho LLM bên cạnh đầu vào là báo cáo của phân tích tĩnh và phân tích động.



Hình 3.4: *Khối Tăng cường Ngữ cảnh*

1. **Retrieval-Augmented Generation (RAG) (Bước 2, 3, 4):** Nhóm đã tham khảo nghiên cứu của Rajapaksha [16] để xây dựng luồng này. Vector DB là một cơ sở dữ liệu vector được thiết kế sẵn, lưu trữ các vector embedding của hàng ngàn mô tả hành vi, đã được ánh xạ với TTP tương ứng. Các mô tả hành vi thu được từ khối Tiền xử lý và Trích xuất Đặc trưng sẽ được chuyển đổi thành vector embedding. Sau đó, hệ thống thực hiện truy vấn trên Vector DB để tìm ra ba hành vi đã biết có ngữ nghĩa gần nhất. Những hành vi không tìm được kết quả tương tự sẽ bị loại bỏ, nhằm hạn chế các mô tả không liên quan gây nhiễu quá trình suy luận của mô hình ngôn ngữ lớn (LLM) và tăng tốc độ xử lý của hệ thống. Kết quả của bước này là một tập hợp các mô tả hành vi có liên quan đến TTP của mẫu mã độc, kèm theo các ví dụ tương đồng đã được ánh xạ TTP trước đó. Đây chính là RAG context (Behavior_mapping_ttp).

2. **Tạo chữ ký hành vi có ngữ nghĩa (Bước 5, 6, 7):** Các chữ ký hành vi (signatures) được trích xuất sẽ gắn với mark hay đánh dấu hành vi, tuy nhiên phần lớn những mark của một chữ ký hành vi sẽ có sự tương đồng và số lượng lớn. Vì vậy, nhóm đã sử dụng một LLM ở luồng này để tóm tắt lại thành những mô tả có ngữ nghĩa và ngắn gọn để tránh cho việc prompt xử lý ở khối Suy luận Trung tâm quá dài giới hạn LLM có thể đáp ứng.

Prompt nhóm sử dụng trong luồng này:

You are a JSON-to-Insight transformer for malware dynamic analysis. Given a malware analysis report in JSON format, extract and output a new JSON object with the following structure:
{
"behavior": <string, taken directly from the "description" field>,
"evidence": <string or list of strings, semantically summarized from the "marks" field without redundancy>
}
Instructions:
- Always extract the exact text from the "description" field and place it as the value of the "behavior" key.
- For "evidence":
Review all entries in the "marks" array.
Summarize each entry into a short semantic description.
If multiple entries are semantically similar, include only one.
If they are distinct, output them as a list of short meaningful evidence strings.
Now convert this JSON:
{signatures}
Return only the output JSON object as described.
Response:

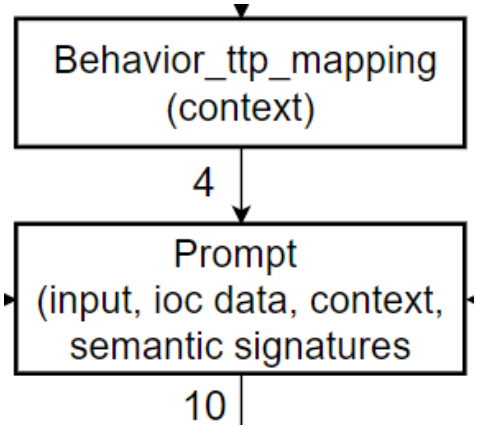
3. **Threat Intelligence (Bước 8, 9):** Các IoC trích xuất từ khối Tiền xử lý và Trích xuất Đặc trưng được sử dụng để truy vấn tự động vào các nền tảng Threat Intelligence, cụ thể nhóm đã sử dụng nền tảng ThreatFox [2]. Kết quả trả về là thông tin về danh tiếng của IoC như hình 3.5.

```
[{'ioc_desc': 'Domain that is used for botnet Command&control (C&C)',
  'malware': 'unknown_loader',
  'malware_alias': None,
  'threat_type': 'botnet_cc',
  'threat_type_desc': 'Indicator that identifies a botnet command&control server (C&C)'}]
```


Hình 3.5: *Ví dụ kết quả trả về của luồng Threat Intelligence*

3.2.2.4. Khối Suy luận Trung tâm

Khối này đóng vai trò như bộ não của hệ thống, thực hiện suy luận dựa trên các dữ liệu đã được xử lý và tăng cường ngữ cảnh nhằm trích xuất chính xác các kỹ thuật tấn công (TTP) theo khung MITRE ATT&CK.



Hình 3.6: *Khối suy luận trung tâm*

Quá trình tạo Prompt đầu vào (Bước 10): Prompt được xây dựng từ các luồng dữ liệu sau:

- **Vai trò:** "Bạn là chuyên gia phân tích mã độc."
- **Nhiệm vụ:** Trích xuất danh sách kỹ thuật tấn công hợp lệ từ các nguồn phân tích đầu vào.
- **Nguồn dữ liệu:**
  - `static_analysis`: Các chuỗi và API đáng ngờ, dấu hiệu packer.

  - `dynamic_analysis`: Chữ ký hành vi khi thực thi đã được tóm tắt, hành vi có liên quan, các ánh xạ TTP từ sandbox.
  - `iocs`: dữ liệu IoC từ Threat Intelligence.
  - `behavior_ttp_mapping`: kết quả của luồng RAG ở khối Khối Tăng cường Ngữ cảnh.

- **Chuỗi suy luận:**
  - Đánh giá từng hành vi để ánh xạ với TTP theo khung MITRE ATT&CK tương ứng.
  - Giải thích mục đích hoặc ảnh hưởng của hành vi.
  - Trích xuất mã kỹ thuật (Technique ID).
  - Gắn kèm các bằng chứng cụ thể từ static, dynamic hoặc IoCs.
  - Chỉ giữ các ánh xạ có độ tin cậy cao, tránh trường hợp suy luận yếu.
- **Định dạng đầu ra:** Dưới dạng JSON, gồm các trường:
  - `technique_id`: Mã định danh chính thức của một kỹ thuật tấn công cụ thể mà tác nhân đe dọa sử dụng trong quá trình xâm nhập hoặc khai thác hệ thống.
  - `evidence`: Mảng chuỗi chứa bằng chứng.
  - `reasoning`: Giải thích lý do ánh xạ.

Prompt nhóm sử dụng:

You are a cybersecurity analyst specialized in malware behavior analysis. Your task is to extract only valid MITRE ATT&CK Tactics, Techniques, and Procedures (TTPs) used by a suspicious file, based on multiple sources of analysis. Use the following input sources:

- static_analysis: Static characteristics such as suspicious strings, structurally detect packer presence, suspicious imports.
- dynamic_analysis: Runtime signatures behavior, some behaviors is assigned TTP by sandbox.
- iocs: Indicators of Compromise such as IPs, domains, urls and their information from Threat Intelligence.
- behavior_ttp_mapping: A list of pre-annotated behaviors with candidate TTPs from RAG. You can use this for reference only, but do not copy or cite evidence from this source.

Reasoning Process (Chain-of-thought):

For each observed behavior or pattern:

- Determine whether it corresponds to a known MITRE ATT&CK technique.
- Justify the mapping by explaining the intent or effect of the behavior.
- Extract the correct MITRE ATT&CK technique ID.
- Identify exact evidence strings from either static_analysis, dynamic_analysis, or iocs only.

- Do not use any evidence from behavior_ttp_mapping for citations.
- Only include mappings that are confident and valid. Avoid weak or speculative mappings.
Response only following output format (JSON) and do not add anything else:
Output Format (JSON array):

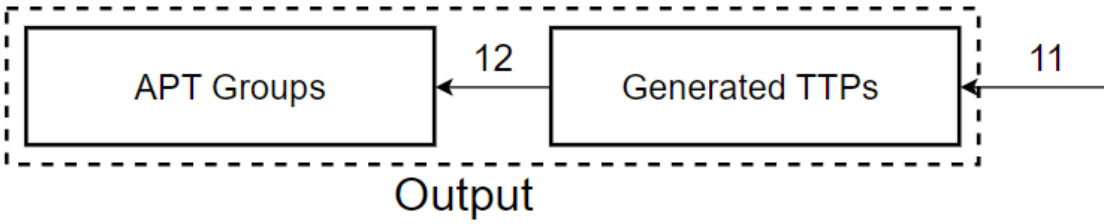
```
{
  "technique_id": "Txxxx",
  "evidence": [
    "Exact evidence string 1 from static/dynamic/IoC",
    "Exact evidence string 2 from static/dynamic/IoC"
  ],
  "reasoning": "Explanation of why this evidence supports the mapping to the technique."
}
```

Note:
- Merge multiple supporting behaviors into a single TTP mapping when appropriate.
- Avoid duplicate TTP entries.
- Only use official MITRE ATT&CK techniques.
Begin your analysis using the provided inputs:
static_analysis: {input}
dynamic_analysis: {input}
iocs: {input}
behavior_ttp_mapping: {input}
Response: {Answer}

Large Language Model (LLM) (Bước 11): Tiếp nhận prompt ở trên sau đó thực hiện phân tích để suy luận ra các TTP có khả năng cao xảy ra.

3.2.2.5. Khối Đầu ra

Khối này đóng vai trò ghi nhận và phản ánh kết quả cuối cùng từ quá trình suy luận của mô hình ngôn ngữ lớn (LLM), sử dụng prompt đã được cấu hình từ Khối Suy luận Trung tâm.



Hình 3.7: *Khối đầu ra*

Dữ liệu đầu ra TTP: LLM trả về danh sách các TTP đã được xác định một cách hợp lệ, dựa trên luồng phân tích hành vi mã độc và các chỉ số liên quan. Mỗi kỹ thuật được gắn với mã định danh chuẩn (`technique_id`), các chuỗi bằng chứng cụ thể và lời giải thích logic thể hiện chuỗi suy luận của mô hình. Kết quả này có cấu trúc rõ ràng để phục vụ các bước xử lý tiếp theo.

Ánh xạ nhóm APT (Bước 12): Tập TTP đầu ra sẽ được đối chiếu với cơ sở dữ liệu MITRE ATT&CK về các nhóm tấn công APT. Hệ thống sử dụng thuật toán (1) so khớp kỹ thuật để tính toán điểm tương đồng giữa tập kỹ thuật quan sát và tập kỹ thuật của từng nhóm APT đã biết. Kết quả trả về là danh sách nhóm APT tiềm năng, được xếp hạng theo độ phù hợp hỗ trợ nhà phân tích xác định kẻ tấn công đứng sau mã độc.

Lưu ý: Kết quả trả về từ LLM có thể chứa các ánh xạ chưa hoàn toàn trùng khớp với thực tế hoặc thiếu thông tin ở đầu ra. Việc xác thực lại bởi người dùng là cần thiết để đảm bảo độ chính xác trong các tình huống phản ứng hoặc điều tra sâu.

Thuật toán 1 Ánh xạ nhóm APT từ tập TTPs đầu ra

Input: $TTPs$, Danh sách các TTP được LLM sinh ra.
Output: APT_ranked, Danh sách xếp hạng các nhóm APT có liên quan.
Tải cơ sở dữ liệu các kỹ thuật mà nhóm APT đã biết.
Khởi tạo một counter rỗng `count`.
for *ttp* in $TTPs$ **do**
    for *group* mapped from *ttp* **do**
        Tăng $count[group]$ lên 1.
    end for
end for
if *count* rỗng **then**
    return danh sách rỗng.
end if
Sắp xếp cặp $(group, score)$ giảm dần theo điểm và tăng dần theo tên.
Khởi tạo $APT\_ranked = [\ ]$, $rank = 1$, $prev\_score = \text{None}$.
for $(group, score)$ in sorted list **do**
    if $score \neq prev\_score$ **then**
        $rank \leftarrow$ current index.
        $prev\_score \leftarrow score$
    end if
    Thêm $(group, score, rank)$ vào APT_ranked.
end for
return APT_ranked

CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Ở chương này chúng tôi tiến tạo môi trường, cài đặt và đưa ra các tiêu chí đánh giá về mức độ hiệu quả của hệ thống.

4.1. Thiết lập thực nghiệm

4.1.1. Môi trường

Hệ thống của chúng tôi được xây dựng và đánh giá trên môi trường Kaggle với cấu hình:

- RAM 29 GB
- GPU T4 15 GB x2
- Disk 57.6GB

Ngôn ngữ lập trình sử dụng là Python 3.11.

Các thư viện sử dụng: transformers, sentence-transformers, chromadb, google-generativeai, torch, bitsandbytes, iocsearcher.

4.1.2. Tập dữ liệu

4.1.2.1. Malware Bazaar

Tập dữ liệu Malware Bazaar là là một dự án từ abuse.ch với mục tiêu chia sẻ các mẫu phần mềm độc hại [1]. Nhóm sử dụng 10449 mẫu mã độc để xây dựng Vector DB cho RAG.

4.1.2.2. APT Malware Dataset

Dựa trên tập dữ liệu được công bố trên github [3], nhóm đã lọc và sử dụng để đánh giá hệ thống với các mẫu mã độc được gắn nhãn nhóm APT như sau:

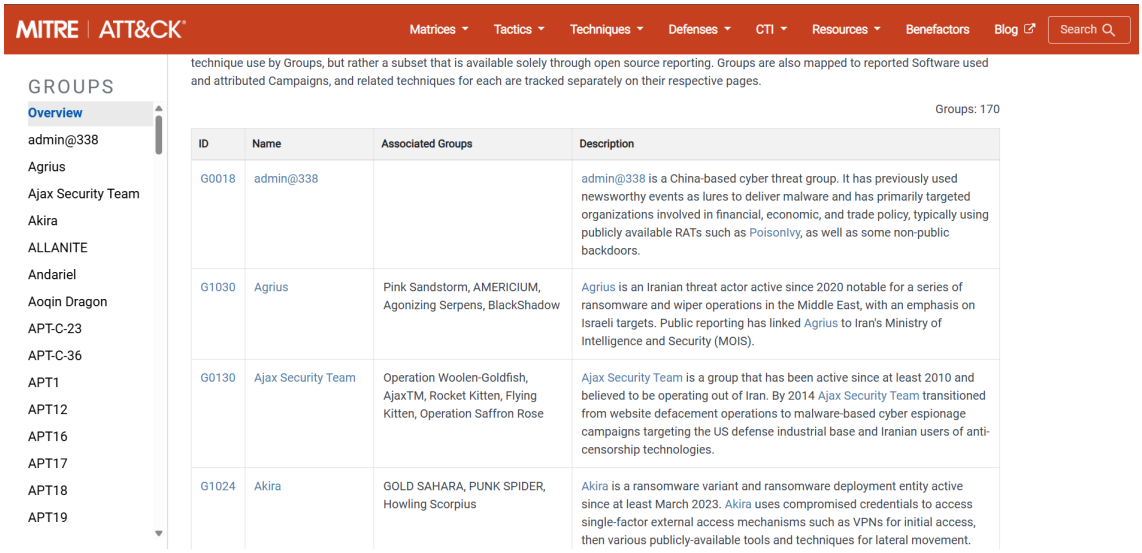
Nhóm APT	Số mẫu mã độc
APT1	28
APT19	18
APT28	42
APT29	22
APT30	11
DarkHotel	27
Equation Group	10
Gorgon Group	9
Winnti	17
Tổng	184

Bảng 4.1: *Bảng thống kê số lượng mẫu mã độc nhóm sử dụng từ APT Malware Dataset.*

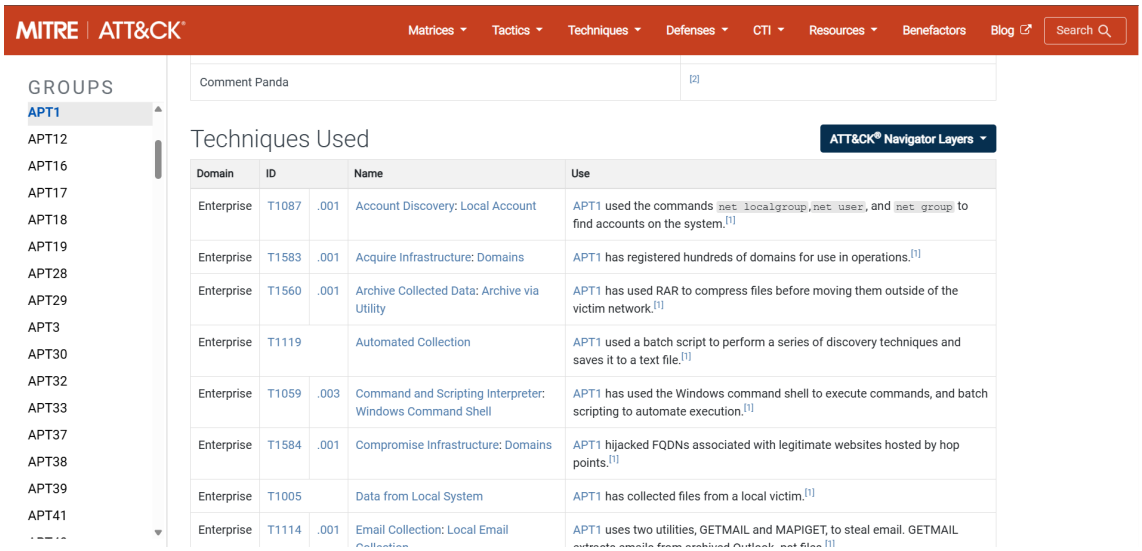
4.1.2.3. MITRE ATT&CK

Dữ liệu dùng để so khớp danh sách TTP sinh ra từ LLM và suy ra danh sách xếp hạng các nhóm APT có liên quan, nhóm tổng hợp từ trang chính thức của MITRE ATT&CK [`https://attack.mitre.org/groups`].

Trang này cung cấp thông tin về các nhóm APT gồm Group ID, mô tả, tên và đặc biệt là kỹ thuật mà các nhóm này sử dụng.



Hình 4.1: *Giao diện của trang chứa thông tin của các nhóm APT*

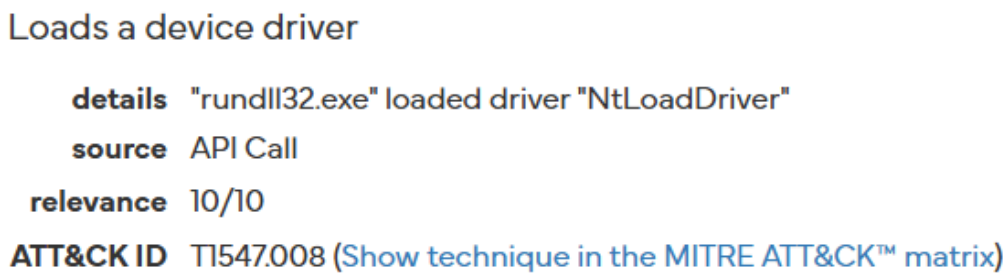


Hình 4.2: *Thông tin về các kỹ thuật mà nhóm APT1 sử dụng*

Như trên hình 4.2, MITRE ATT&CK có liệt kê các kỹ thuật mà nhóm APT1 sử dụng, chúng tôi sẽ dựa vào đó và tổng hợp lại các kỹ thuật mà nhóm APT sử dụng. Kết quả sẽ có một tập dữ liệu về các kỹ thuật ánh xạ với nhóm APT sử dụng kỹ thuật đó có dạng Technique ID: [Group 1, Group 2].

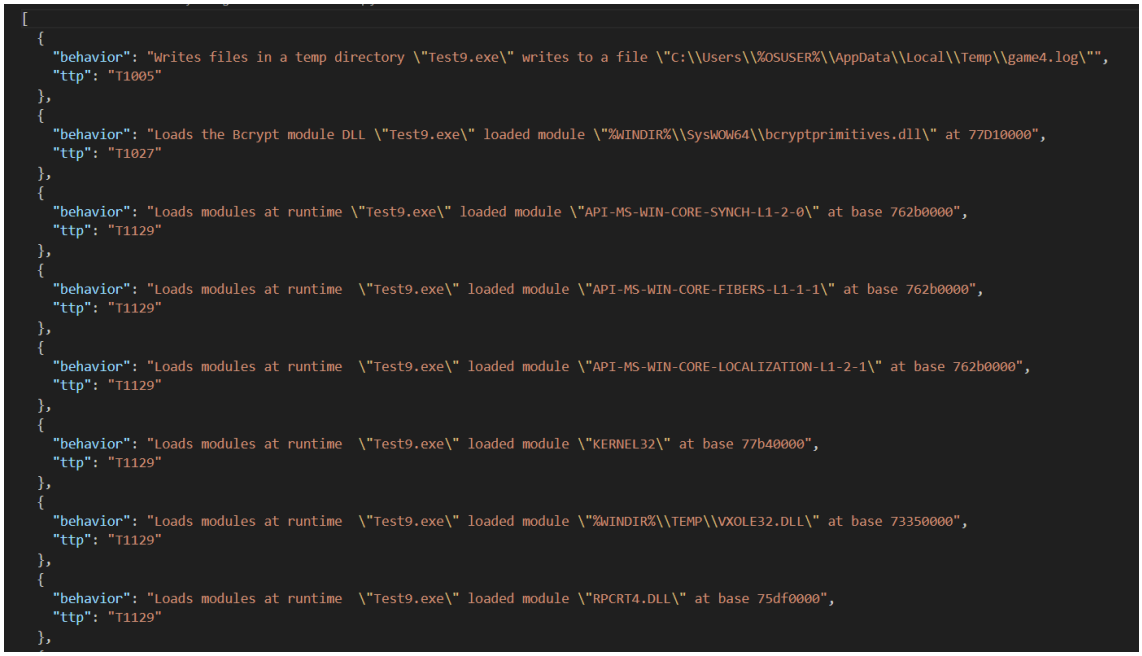
4.1.3. Xây dựng cơ sở dữ liệu vector cho RAG

Thu thập dữ liệu: Với mỗi mẫu mã độc trong tập dữ liệu lấy từ Malware Bazaar, chúng tôi sẽ thực hiện truy vấn thông tin trên Hybrid Analysis - một nền tảng Threat Intelligence sử dụng mã hash. Kết quả truy vấn sẽ có chứa các hành vi của mã độc đã được ánh xạ với TTP tương ứng, như trong hình 4.3.



Hình 4.3: *Kết quả trả về sau khi truy vấn một mẫu mã độc*

Kết quả sẽ có được một tệp chứa dữ liệu về mô tả hành vi và TTP tương ứng của các mẫu mã độc trong Malware Bazaar dataset.



Hình 4.4: *Một phần của tệp dữ liệu thu thập được*

Tiền xử lý dữ liệu và embedding: Dữ liệu được tải từ tệp lên sẽ được chia thành các khối 16 cặp hành vi - ttp. Mô tả hành vi sẽ được embedding sử dụng mô hình BAAI/bge-base-en-v1.5 [4], với mỗi cặp hành vi - ttp sau khi tiền xử lý và embedding có dạng như sau:

- id: định danh duy nhất.
- document: mô tả hành vi gốc
- metadata: TTP tương ứng với mô tả hành vi.
- embedding: embedding vector.

Lưu dữ liệu vào cơ sở dữ liệu vector: Ở bước này chúng tôi sử dụng thư viện chromadb để lưu trữ vector, văn bản gốc, và metadata để hỗ trợ truy vấn trong hệ thống.

4.1.4. Tạo báo cáo phân tích tĩnh và động từ mã độc

Đối với việc phân tích tĩnh mã độc, chúng tôi sử dụng công cụ Manalyze [14]. Lệnh chạy phân tích sẽ như sau:

```
manalyze <malware_filepath> --plugins=strings,packer,imports\
--dump=summary --hashes  -o  json >  <output_path>
```

Lệnh trên sẽ tạo ra một báo cáo với những thông tin gồm: những chuỗi nghi ngờ trong tệp PE, những API nghi ngờ trong bảng imports, phát hiện các dấu hiệu cho thấy tệp bị nén, mã hash, thông tin về tệp PE.

Đối với việc phân tích động, chúng tôi sử dụng cuckoo sandbox, thực hiện tải từng tệp mã độc và lấy báo cáo về kết quả phân tích trong đường dẫn *.cuckoo/storage/analyses/latest/reports/report.json*. Báo cáo sẽ gồm cả việc phân tích tĩnh như thông tin về tệp PE, các bảng import, export, strings, header, section,... và phân tích động như các giao tiếp mạng, chữ ký hành vi, hành vi mã độc thực hiện.

4.2. Kết quả thực nghiệm

Ở mục này, nhóm sẽ trình bày các kết quả thực nghiệm và đưa ra đánh giá.

Nhóm tập trung trả lời ba câu hỏi sau:

Câu hỏi 1: Độ chính xác của hệ thống trong việc sinh TTP và ánh xạ nhóm APT từ báo cáo phân tích tĩnh và động là bao nhiêu?

Câu hỏi 2: Thời gian xử lý trung bình cho một mẫu từ lúc nhập báo cáo đến khi sinh kết quả là bao lâu, có đủ nhanh để áp dụng trong các tình huống thực tế cần phản hồi nhanh không?

Câu hỏi 3: Hệ thống sẽ hoạt động với hiệu suất như nào khi áp dụng những LLM khác nhau?

Lưu ý: Khi thực nghiệm câu hỏi 1 và 2, việc bị giới hạn tài nguyên như request API miễn phí và các mẫu mã độc có số lượng hành vi lớn (khoảng vài nghìn đến chục nghìn hành vi) dẫn đến thời gian thực nghiệm kéo dài đối với phương pháp chỉ dùng LLM. Để giải quyết, chúng tôi đã lọc ra tập dữ liệu phụ từ APT Malware Dataset, chỉ lấy những mẫu mã độc có số hành vi sau khi trích xuất báo cáo ít hơn hoặc bằng 200 hành vi. Sau khi lọc chúng tôi có tập dữ liệu như sau:

Nhóm APT	Số mẫu mã độc
APT1	3
APT19	9
APT28	11
APT29	8
APT30	2
DarkHotel	5
Gorgon Group	1
Winnti	3
Tổng	42

Bảng 4.2: *Bảng thống kê tập dữ liệu sử dụng để thực nghiệm câu hỏi 1 và 2.*

Đối với câu hỏi 3, nhóm thực nghiệm bình thường với APT Malware Dataset.

4.2.1. Độ chính xác của hệ thống (Câu hỏi 1)

Độ chính xác của hệ thống sẽ được đánh giá trên hai kết quả: Tập TTP sinh ra và danh sách nhóm APT đề xuất.

Đối với tập TTP sinh ra, chúng tôi đề xuất sử dụng chỉ số Coverage Rate (CR) để đánh giá. Chỉ số này đo lường tỷ lệ các TTP ground truth được hệ thống tái tạo thành công, tức là tỷ lệ bao phủ của kết quả sinh ra so với tập TTP ground truth, công thức được định nghĩa như sau:

$$CR = \frac{Set_g \cap Set_r}{Set_r} \tag{4.1}$$

trong đó, Set_g là tập TTP được sinh ra từ hệ thống và Set_r là tập TTP ground truth, cách thu thập tập ground truth sẽ tương tự với cách thu thập dữ liệu để tạo Vector DB ở 4.1.3.

Phương pháp	CR
Chỉ dùng LLM	0.1093
LLM + RAG + Threat Intelligence	0.12

Bảng 4.3: *Bảng kết quả so sánh độ chính xác sinh TTP (LLM sử dụng là Gemini 2.0 Flash).*

Nhìn vào bảng, có thể thấy phương pháp đề xuất (LLM + RAG + Threat Intelligence) đã cho thấy sự cải thiện về chỉ số CR từ 0.1093 lên 0.12 so với chỉ dùng LLM. Mặc dù mức tăng không quá lớn, điều này đã xác thực giả thuyết cốt lõi của đề tài: việc tăng cường ngữ cảnh thông qua RAG và Threat Intelligence thực sự có tác động tích cực, giúp LLM đưa ra các phán đoán có cơ sở hơn.

Chỉ số CR còn thấp chỉ 12%. Điều này có nghĩa là hệ thống đã bỏ sót đến 88% các TTP trong tập dữ liệu thực nghiệm. Nguyên nhân có thể đến từ các yếu tố sau:

- Cơ sở dữ liệu RAG được xây dựng từ 10449 mẫu có thể chưa đủ lớn và đa dạng để bao phủ hết các hành vi phức tạp trong tập APT Malware Dataset;
- Chất lượng trích xuất hành vi từ báo cáo động có thể chưa đủ "ngữ nghĩa" để tìm thấy các ví dụ tương đồng hiệu quả trong Vector DB.

Đối với danh sách nhóm APT đề xuất, chúng tôi sử dụng chỉ số Mean Reciprocal Rank (MRR). MRR là một thước đo phổ biến được áp dụng rộng rãi trong các bài toán truy xuất thông tin và hệ thống khuyến nghị, đặc biệt phù hợp trong bối cảnh mà mỗi truy vấn có một hoặc một số câu trả lời đúng, và mục tiêu là xếp hạng các câu trả lời này càng cao càng tốt trong danh sách kết quả. Chỉ số này phản ánh trực tiếp mức độ thuận tiện và hiệu quả của hệ thống đối với người sử dụng: càng ít phải kiểm tra nhiều nhóm APT, người dùng càng nhanh chóng nhận diện được tác nhân đe dọa phù hợp. Với đặc trưng của bài toán, trong đó độ chính xác của thứ hạng nhóm APT quan trọng hơn là chỉ trả về danh sách không thứ tự, MRR cung cấp một thước đo rõ ràng, dễ hiểu và nhấn mạnh tính ưu tiên của các dự đoán đúng nằm ở những vị trí đầu tiên. Công thức tính sẽ như sau:

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i} \tag{4.2}$$

trong đó, N là tổng số mẫu mã độc trong tập dữ liệu dùng để đánh giá, $rank_i$ là xếp hạng nhóm APT ground truth trong danh sách đầu ra của mẫu mã độc thứ i.

Phương pháp	MRR
Chỉ dùng LLM	0.1736
LLM + RAG + Threat Intelligence	0.2246

Bảng 4.4: *Bảng kết quả so sánh độ chính xác sinh danh sách nhóm APT đề xuất (LLM sử dụng là Gemini 2.0 Flash).*

Tương tự như việc sinh TTP, chỉ số MRR đã tăng từ 0.1736 lên 0.2246, khẳng định được giá trị của việc làm giàu ngữ cảnh.

Giá trị MRR là 0.2246 cho thấy trung bình, câu trả lời đúng về nhóm APT nằm ở khoảng vị trí thứ 4 hoặc 5 trong danh sách gợi ý. Đây là một kết quả khả quan. Tuy nhiên, hiệu suất này bị ảnh hưởng trực tiếp bởi kết quả sinh TTP còn thấp ở trên. Khi tập TTP đầu vào không đầy đủ, thuật toán so khớp sẽ khó có thể phân biệt và ưu tiên chính xác các nhóm APT.

4.2.2. Thời gian xử lý của hệ thống (Câu hỏi 2)

Thời gian xử lý trung bình cho một mẫu mã độc, từ lúc nhận báo cáo đến khi trả về kết quả cuối cùng, được đo lường và thống kê trong bảng sau:

Phương pháp	Thời gian xử lý
Chỉ dùng LLM	151.56 (s)
LLM + RAG + Threat Intelligence	139.95 (s)

Bảng 4.5: *Bảng kết quả so sánh thời gian xử lý (LLM sử dụng là Gemini 2.0 Flash).*

Kết quả so sánh cho thấy thời gian xử lý của phương pháp LLM + RAG + Threat Intelligence nhanh hơn khoảng 11 giây so với chỉ dùng LLM mặc dù xử lý nhiều bước hơn. Có thể lý giải việc này như sau: tích hợp RAG sẽ giúp loại bỏ những thông tin nhiễu không liên quan, cùng với một prompt đã được bổ sung ngữ cảnh đã giúp LLM hiểu vấn đề và đưa ra câu trả lời nhanh hơn.

Thời gian xử lý trung bình khoảng 140 giây (hơn 2 phút) cho mỗi mẫu vẫn còn khá cao cho các tác vụ cần phản ứng tức thời trong một Trung tâm Điều hành An ninh (SOC). Tuy nhiên, tốc độ này hoàn toàn phù hợp cho các kịch bản phân tích sâu phải mất nhiều giờ để phân tích và viết báo cáo mã độc.

4.2.3. Hiệu suất hệ thống trên các LLM khác nhau (Câu hỏi 3)

Chúng tôi chọn ra 3 mô hình để đánh giá:

- Llama 3.1 8B Instruct.
- Gemini 2.0 Flash.
- Mistral 7B Instruct v0.3.

Kết quả đánh giá được thống kê trong bảng sau:

LLM	CR	MRR	Thời gian xử lý
Gemini 2.0 Flash	0.1234	0.0695	382.1 (s)
Llama 3.1 8B Instruct	0.1307	0.0623	1474.27 (s)
Mistral 7B Instruct v0.3	0.1135	0.0728	547.62 (s)

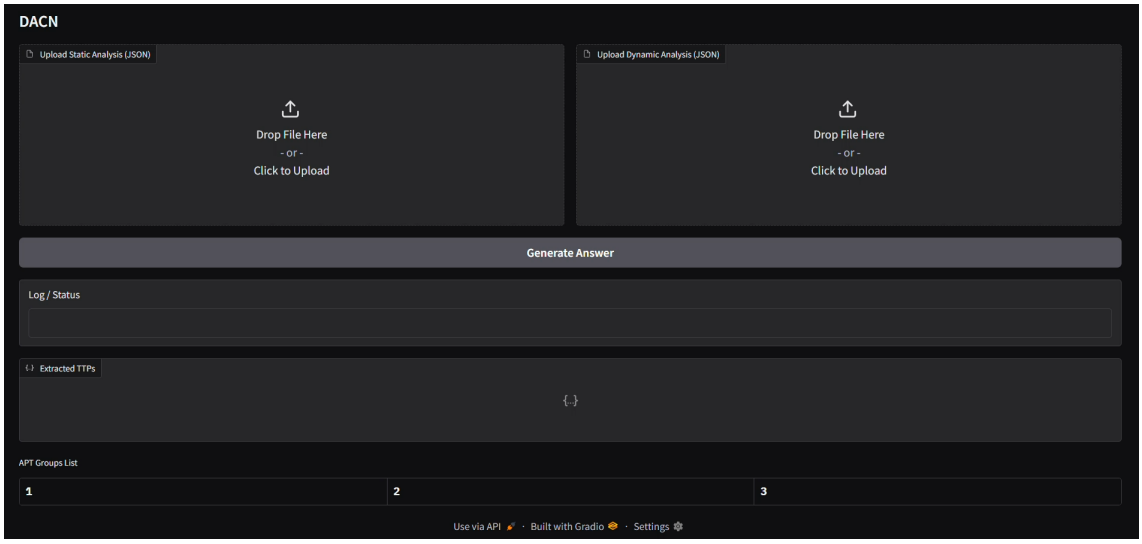
Bảng 4.6: *Bảng kết quả so sánh hiệu suất trên các LLM khác nhau.*

Hiệu suất giữa các mô hình không chênh lệch nhiều chứng tỏ hệ thống thích nghi khá tốt trên các mô hình khác nhau. Tuy nhiên hiệu suất còn thấp, cho thấy việc trích xuất TTP từ mã độc nhị phân là một bài toán chuyên biệt và phức tạp, đòi hỏi khả năng suy luận sâu. Các mô hình LLM dùng trong các tác vụ thông thường chưa đủ năng lực để giải quyết hiệu quả bài toán này.

4.3. Triển khai Giao diện Web

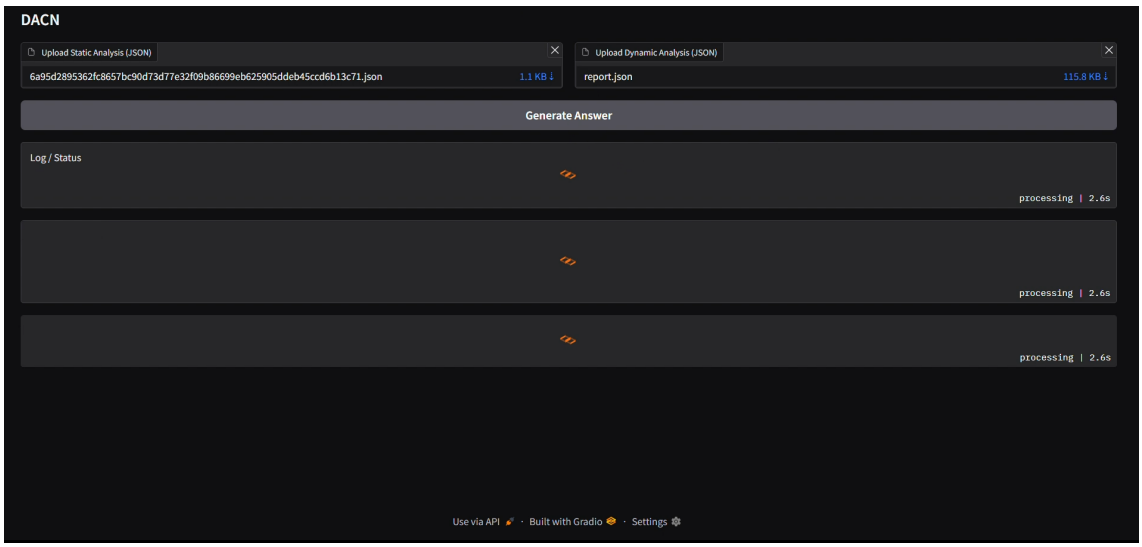
Để minh họa tính khả thi và cung cấp một phương thức tương tác trực quan với hệ thống đã được thiết kế, nhóm đã phát triển một ứng dụng web sử dụng thư viện gradio cho phép người dùng dễ dàng tải lên các báo cáo, theo dõi quá trình phân tích và nhận kết quả một cách tường minh.

Dưới đây là giao diện của web:

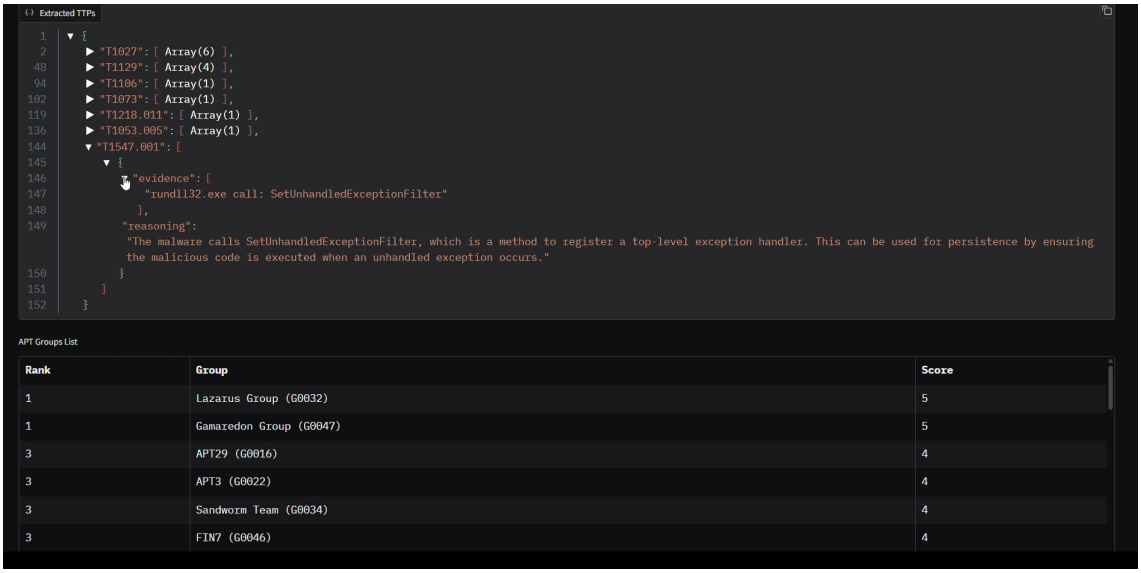


Hình 4.5: *Giao diện khi bắt đầu của trang Web*

Để bắt đầu, người dùng sẽ tải lên các báo cáo phân tích tĩnh và động vào hai ô là Upload Static Analysis (JSON) và Upload Dynamic Analysis (JSON). Sau đó, bấm nút Generate Answer để bắt đầu quá trình trích xuất.



Hình 4.6: *Hệ thống đang tiến hành trích xuất*



Hình 4.7: *Kết quả sau khi trích xuất*

Sau khi chạy xong, kết quả sẽ hiển thị hai phần, phần ở trên sẽ là những TTP mà hệ thống trích xuất được và phần ở dưới sẽ là danh sách các nhóm APT đề xuất.

Video Demo của nhóm: Video

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Ở chương này, chúng tôi đưa ra những kết luận về nghiên cứu, những hạn chế, và đồng thời đưa ra hướng cải thiện và phát triển.

5.1. Kết luận

Qua quá tình nghiên cứu và triển khai, đồ án đã chứng minh tiềm năng to lớn của việc áp dụng mô hình Ngôn ngữ lớn (LLM) trong công tác phân tích mã độc nhị phân. Hệ thống đã thành công trích xuất các TTP với dữ liệu là các báo cáo phân tích tĩnh và động của tệp mã độc.

Bên cạnh đó, nghiên cứu của chúng tôi cũng đạt được một số kết quả như sau:

- Tích hợp được RAG và các nguồn dữ liệu Threat Intelligence giúp hệ thống nâng cao độ chính xác.
- Áp dụng được kỹ thuật phân tích và sinh TTP tự động dựa trên dữ liệu phân tích tĩnh và động của tệp mã độc, hỗ trợ nhận diện nhóm APT theo TTP.
- Sử dụng nhiều mô hình khác nhau để đưa ra đánh giá về việc sinh TTP, ánh xạ nhóm APT và thời gian xử lý của hệ thống trong môi trường thực tế.

Từ các kết quả đạt được, có thể khẳng định rằng việc tích hợp mô hình LLM kết hợp RAG và dữ liệu Threat Intelligence là một hướng đi đầy triển vọng trong tự động hóa phân tích mã độc và nhận diện nhóm APT hiệu quả trong thực tiễn.

Tuy nhiên, vẫn tồn tại một số hạn chế ở hệ thống. Dưới đây là các điểm cần lưu ý:

- **Độ chính xác còn phụ thuộc vào chất lượng của báo cáo đầu vào**: Nếu tệp báo cáo mã độc không đầy đủ hoặc có sự sai lệch, việc sinh TTP và phân loại nhóm APT sẽ thiếu chính xác.
- **Chưa hỗ trợ xử lý mẫu mã độc từ nhiều nguồn động thực tế**: Việc xử lý chủ yếu dựa trên báo cáo, chưa tích hợp khả năng tự động thu thập và phân tích từ sandbox hoặc honeypot.
- **Chưa khai thác phản hồi từ chuyên gia**: Hệ thống hiện chưa tích hợp cơ chế phản hồi để chuyên gia có thể góp ý và điều chỉnh kết quả sinh ra, khiến quá trình cải tiến còn thủ công và gián đoạn.

5.2. Hướng phát triển

Để giải quyết các hạn chế hiện tại và mở rộng khả năng của hệ thống, chúng tôi đề xuất các hướng nghiên cứu và phát triển trong tương lai:

- **Tự động cập nhật và mở rộng cơ sở dữ liệu RAG:** Xây dựng một module có khả năng tự động thu thập, phân tích các báo cáo mối đe dọa mới từ các blog an ninh mạng, các nền tảng chia sẻ mẫu vật để liên tục cập nhật và làm giàu cho cơ sở dữ liệu RAG. Điều này giúp hệ thống luôn cập nhật với các biến thể mã độc mới.
- **Fine-tuning một mô hình chuyên biệt:** Thay vì sử dụng các LLM pretrained, một hướng đi tiềm năng là fine-tune một mô hình mã nguồn mở trên một bộ dữ liệu lớn gồm các báo cáo phân tích và TTP tương ứng. Điều này giúp nâng cao độ chính xác của hệ thống.
- **Tích hợp phản hồi từ chuyên gia:** Thiết kế một giao diện cho phép các nhà phân tích đánh giá và sửa lỗi kết quả của hệ thống. Phản hồi này (ví

dụ: "ánh xạ TTP này đúng/sai") sẽ được sử dụng để tự động cải thiện cơ sở dữ liệu RAG hoặc làm dữ liệu cho các vòng fine-tuning tiếp theo, tạo ra một chu trình học hỏi và cải tiến liên tục.

.

TÀI LIỆU THAM KHẢO

Tiếng Anh:

[1] Abuse.ch, *Malware Bazaar Dataset*, URL: https://datalake.abuse.ch/malware-bazaar/daily/.

[2] Abuse.ch, *ThreatFox*, URL: https://threatfox.abuse.ch/.

[3] *APT Malware Dataset*, URL: https://github.com/cyber-research/APTMalware.git.

[4] Beijing Academy of Artificial Intelligence, *BAAI/bge-base-en-v1.5 model*, URL: https://huggingface.co/BAAI/bge-base-en-v1.5.

[5] MITRE ATT&CK, *Globally-accessible knowledge base of adversary tactics and techniques based on real-world observations*, URL: https://attack.mitre.org/.

[6] AV-TEST Institute, *Malware Statistics - AV-TEST*, Accessed: 2025-07-02, 2024, URL: https://www.av-test.org/en/statistics/malware/.

[7] Markus Bayer, Tobias Frey, and Christian Reuter (2023), "Multi-level fine-tuning, data augmentation, and few-shot learning for specialized cyber threat intelligence", *Computers & Security*, 134, p. 103430.

[8] Juan Caballero et al. (2023), "The rise of goodfatr: A novel accuracy comparison methodology for indicator extraction tools", *Future Generation Computer Systems*, 144, pp. 74–89.

[9] Denız Demırcı, Cengiz Acarturk, et al. (2022), "Static malware detection using stacked BiLSTM and GPT-2", *IEEE Access*, 10, pp. 58488–58502.

[10] Dharani Devadiga et al., "Gleam: Gan and llm for evasive adversarial malware", in: *2023 14th International Conference on Information and Communication Technology Convergence (ICTC)*, IEEE, 2023, pp. 53–58.

[11] Yun Gao et al., "Malware detection using attributed CFG generated by pre-trained language model with graph isomorphism network", in: *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, IEEE, 2022, pp. 1495–1501.

[12] Abdallah Ghourabi and Manar Alohaly (2023), "Enhancing spam message classification and detection using transformer-based embedding and ensemble learning", *Sensors*, 23 (8), p. 3861.

[13] James Lee Hu, Mohammadreza Ebrahimi, and Hsinchun Chen, "Single-shot black-box adversarial attacks against malware detectors: A causal language model approach", in: *2021 IEEE International Conference on Intelligence and Security Informatics (ISI)*, IEEE, 2021, pp. 1–6.

[14] JusticeRage, *Manalyze: A static analyzer for PE executables*, 2015, URL: `https://github.com/JusticeRage/Manalyze`.

[15] Daniel Nahmias et al. (2024), "Prompted contextual vectors for spear-phishing detection", *arXiv preprint arXiv:2402.08309*.

[16] Sampath Rajapaksha, Ruby Rani, and Erisa Karafili, "A rag-based question-answering solution for cyber-attack investigation and attribution", in: *European Symposium on Research in Computer Security*, Springer, 2024, pp. 238–256.

[17] Febrian Setianto et al. (2021), "Gpt-2c: A gpt-2 parser for cowrie honeypot logs", *arXiv preprint arXiv:2109.06595*.

[18] Ying Zhang et al. (2024), "Tactics, Techniques, and Procedures (TTPs) in Interpreted Malware: A Zero-Shot Generation with Large Language Models", *arXiv preprint arXiv:2407.08532*.