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

PHẠM BÁ TÍN
TRẦN TẤN TÀI

KHÓA LUẬN TỐT NGHIỆP

NGHIÊN CỨU MÔ HÌNH PHÁT HIỆN TẤN CÔNG APT DỰA TRÊN ĐỒ THỊ NGUỒN GỐC

A STUDY ON APT ATTACK DETECTION USING PROVENANCE GRAPH DATA

CỬ NHÂN NGÀNH AN TOÀN THÔNG TIN

TP. HỒ CHÍ MINH, 2024

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

PHẠM BÁ TÍN - 20522016
TRẦN TẤN TÀI - 20521862

KHÓA LUẬN TỐT NGHIỆP

NGHIÊN CỨU MÔ HÌNH PHÁT HIỆN TẤN CÔNG APT DỰA TRÊN ĐỒ THỊ NGUỒN GỐC

A STUDY ON APT ATTACK DETECTION USING PROVENANCE GRAPH DATA

CỬ NHÂN NGÀNH AN TOÀN THÔNG TIN

GIẢNG VIÊN HƯỚNG DẪN:
ThS. ĐỖ THỊ THU HIỀN
ThS. BÙI THANH BÌNH

TP. Hồ Chí Minh - 2024

Thông tin hội đồng bảo vệ khóa luận

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số . . . . . .
ngày . . . . . của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

1. Chủ tịch: . . . . . .
2. Thư ký: . . . . . .
3. Ủy viên: . . . . . .

LỜI CẢM ƠN

Để hoàn thành khóa luận tốt nghiệp, chúng tôi đã nhận được nhiều sự định hướng, sự giúp đỡ nhiệt tình, các ý kiến đóng góp quý báu cùng những lời động viên, khích lệ từ các thầy cô, anh chị và bạn bè tại trường Đại học Công nghệ Thông tin - ĐHQG TP.HCM.

Chúng tôi xin chân thành cảm ơn Ban giám hiệu Trường Đại học Công nghệ Thông tin - ĐHQG TP.HCM đã tạo điều kiện và môi trường học tập, nghiên cứu tốt nhất. Chúng tôi rất biết ơn và gửi lời cảm ơn đến các quý thầy cô, anh chị công tác tại khoa Mạng máy tính và Truyền thông nói chung và Phòng thí nghiệm An toàn Thông tin - InSecLab nói riêng, đã giúp đỡ, nhiệt tình giảng dạy và truyền đạt những kiến thức trong suốt thời gian qua.

Bên cạnh đó, chúng tôi cũng xin bày tỏ lòng biết ơn sâu sắc và lời cảm ơn chân thành tới cô Đỗ Thị Thu Hiền, thầy Bùi Thanh Bình, đặc biệt là thầy Phan Thế Duy, thầy Nghi Hoàng Khoa và anh Ngô Đức Hoàng Sơn đã tận tình trực tiếp hướng dẫn, nhận xét, góp ý và giúp đỡ trong suốt quá trình hoàn thiện khóa luận tốt nghiệp.

Với điều kiện về thời gian và kinh nghiệm còn hạn chế, cũng như sự rộng lớn về lượng kiến thức, khóa luận này không thể tránh khỏi những thiếu sót. Chúng tôi rất mong nhận được sự cảm thông, góp ý và chỉ bảo của quý thầy cô, anh chị. Điều này sẽ giúp nhóm có điều kiện bổ sung, nâng cao nhận thức và ngày càng trở nên hoàn thiện hơn.

Phạm Bá Tín **Trần Tấn Tài**

Mục lục

Danh sách hình vẽ

Danh sách bảng

Danh mục từ viết tắt

APT	**A**dvanced **P**ersistent **T**hreat
IDS	**I**ntrusion **D**etection **S**ystem
PIDS	**P**rovenance-based **I**ntrusion **D**etection **S**ystem
GNN	**G**raph **N**eural **N**etwork
MLP	**M**ulti**l**ayer **P**erceptron **S**ystem
RE	**R**econstruction **E**rror
IDF	**I**nverse **D**ocument **F**requency
GCN	**G**raph **C**onvolutional **N**etwork
GAT	**G**raph **A**ttention **N**etwork
HAN	**H**eterogeneous Graph **A**ttention **N**etwork
GAE	**G**raph **A**ttention **E**mbedding
GRU	**G**ated **R**ecurrent **U**nit

Danh mục từ tạm dịch

APT	Mối đe dọa tinh vi liên tục
IDS	Hệ thống phát hiện xâm nhập
PIDS	Hệ thống phát hiện xâm nhập dựa trên nguồn gốc
systen call	lời gọi hệ thống
deep learning	học sâu
GNN	Mạng nơ-ron đồ thị
hideen layer	lớp ẩn
backdoor	cửa sau
time windows	cửa sổ thời gian
time windows queue	hàng đợi cửa sổ thời gian
Signature-based Detection	Phát hiện dựa trên chữ ký
Anomaly-based Detection	Phát hiện dựa trên bất thường

TÓM TẮT KHOÁ LUẬN

Trong thời đại khoa học - công nghệ ngày càng phát triển và dần trở nên phổ biến với cuộc sống, nhu cầu sử dụng hệ thống thông tin cho cá nhân, tổ chức và doanh nghiệp cũng tăng lên. Tuy nhiên, đi kèm với đó là những thách thức về sự an toàn khi đối mặt với các cuộc tấn công mạng với nhiều mục đích khác nhau. Điều này dẫn đến sự cần thiết của việc đảm bảo an ninh của hệ thống, phát hiện sớm và giảm thiểu ở mức thấp nhất thiệt hại do các cuộc tấn công gây ra.

Những năm gần đây, các hệ thống phát hiện xâm nhập đã không ngừng phát triển và cải tiến để giảm thiểu rủi ro về an toàn thông tin. Bên cạnh đó, tấn công APT cũng gia tăng nhanh chóng cả về số lượng và mức độ nghiêm trọng, ngày càng trở nên tinh vi và khó nắm bắt hơn. Sau khi tìm hiểu về đồ thị nguồn gốc, chúng tôi nhận thấy đây là một hướng đi có nhiều tiềm năng. Vì vậy, trong khóa luận này, chúng tôi hướng tới việc nghiên cứu, đề xuất giải pháp và xây dựng một hệ thống phát hiện xâm nhập dựa trên đồ thị nguồn gốc để phát hiện tấn công APT. Sau cùng, chúng tôi đánh giá mô hình trên các môi trường thử nghiệm và đề xuất một số hướng phát triển để cải thiện mô hình trong trong lai.

Chương 1

TỔNG QUAN ĐỀ TÀI

Tóm tắt chương

Trong chương này, chúng tôi xin giới thiệu tóm tắt vấn đề và các nghiên cứu liên quan đến bài toán xây dựng mô hình phát hiện xâm nhập. Đồng thời, chúng tôi cũng trình bày mục tiêu, phạm vi và cấu trúc của khóa luận.

1.1 Lý do chọn đề tài

Trong bối cảnh các Mối đe dọa tinh vi liên tục (Advanced Persistent Threat - APT) ngày càng trở nên tinh vi, nó đã và đang trở thành một trong những mối đe dọa lớn đối với môi trường mạng máy tính hiện đại [15]. Các biện pháp phòng thủ như Hệ thống phát hiện xâm nhập (Intrusion Detection System - IDS) không đủ khả năng để chống lại toàn bộ các cuộc tấn công. Kẻ tấn công liên tục thay đổi cách thức tấn công cũng như lẫn trốn; trong khi đó, hầu hết các IDS dựa trên chữ ký để phát hiện, điều này làm cho nó dễ dàng bị qua mặt.

Để nâng cao khả năng chống lại các cuộc tấn công, lĩnh vực an ninh mạng có hướng di chuyển sang sử dụng dữ liệu nguồn gốc. Việc phân tích đồ thị nguồn gốc giúp nắm bắt mối quan hệ phức tạp giữa các thực thể trong hệ thống/mạng máy tính. Hệ thống phát hiện xâm nhập dựa trên đồ thị nguồn gốc (Provenance-based Intrusion Detection System - PIDS) sử dụng các thông tin này để phát hiện các cuộc tấn công APT tinh vi. Tuy nhiên, công việc này đối mặt với 4 thử thách chính, bao gồm:

1. Phạm vi: Liệu PIDS có thể phát hiện các cuộc tấn công hiện đại mà xâm nhập qua các ranh giới ứng dụng không?
2. Tính bất khả tri tấn công: Liệu PIDS có thể phát hiện các cuộc tấn công mới mà không cần kiến thức trước về các đặc điểm tấn công không?
3. Tính kịp thời: Liệu PIDS có thể giám sát hiệu quả các hệ thống chủ khi chúng hoạt động không?
4. Khả năng tái cấu trúc tấn công: Liệu PIDS có thể chắt lọc hoạt động tấn công từ các đồ thị nguồn gốc lớn để các quản trị viên hệ thống có thể dễ dàng hiểu và nhanh chóng phản ứng với sự xâm nhập hệ thống không?

Mặc dù các PIDS có khả năng phát hiện khá tốt nhưng cũng còn một số hạn chế. Đối với các hệ thống dựa trên chữ ký, các phương pháp dự đoán hoặc dấu vết tấn công đã biết có thể bị trốn tránh khi những kẻ tấn công điều chỉnh mô hình của chúng. Một số hệ thống chọn cách xây dựng một đồ thị nguồn gốc duy nhất cho toàn bộ hệ thống từ nhật ký, tuy nhiên điều này làm cho việc chi phí xử lý đầu vào lớn và số lượng cảnh báo sai cũng tăng lên. Một số hệ thống được xây dựng dựa trên các ảnh chụp nhanh bất thường có độ chi tiết thấp vì các nhà phân tích phải phân tích tất cả các thực thể/tương tác trong các ảnh chụp nhanh bất thường. Mặt khác, các hệ thống này cung cấp ít thông tin để giúp người quản trị có thể nắm bắt những gì thực sự xảy ra trong hệ thống của họ trong các cuộc tấn công. Một số xây dựng mô hình phát hiện các điểm bất thường ở cấp độ biểu đồ. Để hỗ trợ điều tra tấn công chi tiết hơn, nó xếp hạng các nút biểu đồ dựa trên mức độ bất thường của chúng. Vì vậy, công tác điều tra sau phát hiện vẫn tốn nhiều công sức.

Bên cạnh đó, mô hình học sâu (deep learning) có khả năng tự học và trích xuất các đặc trưng phức tạp từ dữ liệu đầu vào mà không cần phụ thuộc vào việc xác định trước các đặc trưng đó. Mô hình học sâu thường được áp dụng trong các nhiệm vụ như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên và dự báo chuỗi thời gian. Các mô hình học sâu thường có kiến trúc mạng nơ-ron sâu với nhiều lớp ẩn (hidden layers) giúp mô hình hóa các mối quan hệ phức tạp hơn và khám phá được những đặc trưng ẩn sâu trong dữ liệu. Với số lượng lớp ẩn lớn, mô hình học sâu có khả năng biểu diễn các hàm phức tạp và thực hiện các tác vụ phân

loại và dự đoán chính xác. Do đó có thể áp dụng các mô hình học sâu trong các hệ thống giúp tăng khả năng phát hiện tấn công APT.

Từ những điều trên, chúng tôi nhận thấy việc xây dựng mô hình phát hiện tấn công APT là một nhu cầu cần thiết. Vì vậy, chúng tôi muốn xây dựng được một mô hình có thể phát hiện tấn APT hiệu quả, nhằm tiết kiệm thời gian và chi phí, đáp ứng được nhu cầu về nguồn nhân lực an toàn thông tin hiện nay.

1.2 Phương pháp nghiên cứu

Trong nghiên cứu này, chúng tôi đã tập trung vào việc tìm hiểu và khai thác khái niệm, cách thức hoạt động của các cuộc tấn công APT (Advanced Persistent Threats), cùng với kiến thức về đồ thị nguồn gốc và cách sử dụng đồ thị nguồn gốc trong việc phát hiện các cuộc tấn công APT. Các cuộc tấn công APT thường rất phức tạp, kéo dài và khó phát hiện, vì vậy việc hiểu rõ bản chất và phương thức tấn công là bước đầu tiên quan trọng để xây dựng một mô hình phát hiện hiệu quả.

Chúng tôi đã tiến hành nghiên cứu sâu rộng về đồ thị nguồn gốc, một công cụ mạnh mẽ trong việc biểu diễn mối quan hệ và sự tương tác giữa các thực thể trong mạng. Đồ thị nguồn gốc giúp chúng tôi mô hình hóa các sự kiện, hành vi và luồng dữ liệu trong hệ thống, từ đó nhận diện các mẫu hành vi bất thường có thể là dấu hiệu của các cuộc tấn công APT. Việc sử dụng đồ thị nguồn gốc cho phép chúng tôi tạo ra một bức tranh tổng thể về hoạt động trong hệ thống, giúp phát hiện các hành vi tấn công tinh vi mà các phương pháp truyền thống có thể bỏ qua.

Dựa trên hiểu biết này, chúng tôi đã xây dựng một mô hình phát hiện tấn công APT dựa trên đồ thị nguồn gốc. Mô hình này không chỉ tập trung vào việc phát hiện các hành vi bất thường mà còn lập biểu đồ tóm tắt các cuộc tấn công từ các hàng đợi cửa sổ thời gian. Chúng tôi sử dụng phương pháp học sâu GNN (Graph Neural Networks) để khai thác cấu trúc và động học của đồ thị nguồn gốc. GNN là một công cụ mạnh mẽ trong việc học các biểu diễn từ đồ thị, cho phép mô hình học được các đặc trưng phức tạp từ dữ liệu.

Trong quá trình phát triển mô hình, chúng tôi đã xem xét và áp dụng một số thuật toán phù hợp để tối ưu hóa hiệu quả phát hiện. Các thuật toán này bao gồm các kỹ thuật chuyển đổi và cập nhật biểu diễn của đỉnh trong đồ thị, sử dụng các lớp TransformerConv trong mô hình Graph Attention Embedding (GAE) và các lớp SAGEConv trong mô hình GraphSAGE. Các lớp này giúp mô hình học được các đặc trưng phức tạp từ dữ liệu đồ thị và cải thiện hiệu quả phát hiện tấn công.

Để đánh giá hiệu năng và độ chính xác của mô hình, chúng tôi đã thực hiện các thử nghiệm khác nhau trên các bộ dữ liệu CADETS và THEIA. Các thử nghiệm này không chỉ giúp chúng tôi đo lường hiệu suất của mô hình mà còn cung cấp những thông tin quan trọng để điều chỉnh và cải tiến mô hình. Chúng tôi đã sử dụng các chỉ số như True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN), Precision, Recall, Accuracy và AUC để đánh giá hiệu suất của mô hình. Kết quả thử nghiệm cho thấy mô hình GraphSAGE có hiệu suất vượt trội hơn so với mô hình GAE, đặc biệt là trong việc phát hiện các cuộc tấn công mà không bỏ sót bất kỳ trường hợp dương tính nào.

1.3 Mục tiêu nghiên cứu

Trong khóa luận này, chúng tôi tập trung vào hai mục tiêu chính, được xây dựng dựa trên nhu cầu cấp thiết trong việc phát hiện và phòng chống các cuộc tấn công APT (Advanced Persistent Threats), cũng như ứng dụng các công nghệ tiên tiến trong lĩnh vực học máy và học sâu. Cụ thể, các mục tiêu này được chi tiết như sau:

1. **Nghiên cứu xây dựng mô hình phát hiện tấn công APT dựa trên đồ thị nguồn gốc**: Mục tiêu đầu tiên của chúng tôi là nghiên cứu và xây dựng một mô hình phát hiện tấn công APT dựa trên đồ thị nguồn gốc. Đồ thị nguồn gốc là một công cụ mạnh mẽ trong việc biểu diễn các mối quan hệ và sự tương tác giữa các thực thể trong mạng. Việc sử dụng đồ thị nguồn gốc cho phép chúng tôi mô hình hóa các sự kiện, hành vi và luồng dữ liệu trong hệ thống, từ đó nhận diện các mẫu hành vi bất thường có thể là dấu hiệu của các cuộc tấn công APT.

2. **Ứng dụng một số mô hình học máy, học sâu nhằm tăng hiệu quả phát hiện**: Mục tiêu thứ hai của chúng tôi là ứng dụng các mô hình học máy và học sâu để tăng hiệu quả phát hiện các cuộc tấn công APT. Học máy và học sâu là các công nghệ tiên tiến trong lĩnh vực trí tuệ nhân tạo, có khả năng học và nhận diện các mẫu phức tạp từ dữ liệu. Việc ứng dụng các mô hình này giúp chúng tôi cải thiện độ chính xác và hiệu suất của mô hình phát hiện.

1.4 Phạm vi và Đối tượng nghiên cứu

Trong khóa luận này, chúng tôi tập trung nghiên cứu và xây dựng mô hình phát hiện xâm nhập dựa trên mô hình Kairos, một mô hình được giới thiệu và phân tích trong bài báo của Cheng et al. (2024) [4]. Mô hình Kairos được thiết kế để phát hiện các cuộc tấn công APT (Advanced Persistent Threats) thông qua việc sử dụng đồ thị nguồn gốc, giúp mô hình hóa và phân tích các sự kiện và mối quan hệ trong hệ thống.

1.4.1 Phạm vi nghiên cứu

1. Tìm hiểu mô hình Kairos:
    - Nghiên cứu chi tiết về cấu trúc, cơ chế hoạt động và cách thức áp dụng của mô hình Kairos. Điều này bao gồm việc tìm hiểu các thành phần chính của mô hình, cách thức nó xử lý dữ liệu nhật ký, và cách nó sử dụng đồ thị nguồn gốc để phát hiện các hoạt động bất thường trong hệ thống.
    - Phân tích các ưu điểm và hạn chế của mô hình Kairos so với các phương pháp phát hiện xâm nhập truyền thống khác.
2. Sử dụng đồ thị nguồn gốc từ tệp nhật ký:
    - Thu thập và tiền xử lý dữ liệu nhật ký từ các bộ dữ liệu công khai. Chúng tôi sẽ sử dụng các bộ dữ liệu đã được công bố và sẵn có để đảm bảo tính khách quan và khả năng so sánh với các nghiên cứu khác.

- Xây dựng đồ thị nguồn gốc từ các tệp nhật ký này. Đồ thị nguồn gốc sẽ giúp chúng tôi biểu diễn các mối quan hệ giữa các sự kiện và thực thể trong hệ thống, từ đó nhận diện các mẫu hành vi bất thường có thể là dấu hiệu của các cuộc tấn công APT.

3. Xây dựng và triển khai mô hình phát hiện xâm nhập:

   - Phát triển mô hình phát hiện xâm nhập dựa trên đồ thị nguồn gốc và áp dụng các kỹ thuật học sâu như Graph Neural Networks (GNNs). Chúng tôi sẽ triển khai các mô hình như Graph Attention Embedding (GAE) và GraphSAGE, đã được chứng minh là hiệu quả trong việc xử lý dữ liệu đồ thị.
   - Tích hợp các mô hình này với dữ liệu nhật ký đã được xử lý để xây dựng hệ thống phát hiện xâm nhập hoàn chỉnh.

4. Đánh giá và so sánh mô hình:

   - Thực hiện các thử nghiệm khác nhau để đánh giá hiệu suất của mô hình phát hiện xâm nhập. Chúng tôi sẽ sử dụng các chỉ số đánh giá như True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN), Precision, Recall, Accuracy và AUC để đánh giá độ chính xác và hiệu suất của mô hình.
   - So sánh mô hình phát hiện xâm nhập của chúng tôi với các mô hình tương tự khác để xác định những cải tiến và tối ưu hóa.

1.4.2 Đối tượng nghiên cứu

1. Các cuộc tấn công APT:

   - Các phương thức và kỹ thuật được sử dụng trong các cuộc tấn công APT. Chúng tôi sẽ tập trung vào việc phân tích các mẫu hành vi của các cuộc tấn công này để hiểu rõ hơn về cách thức chúng hoạt động và các dấu hiệu nhận biết.
   - Các đặc điểm và hành vi bất thường trong hệ thống mà có thể là dấu hiệu của các cuộc tấn công APT.

2. Dữ liệu nhật ký và đồ thị nguồn gốc:

    - Các tệp nhật ký từ các bộ dữ liệu công khai. Chúng tôi sẽ sử dụng dữ liệu từ các nguồn như CADETS và THEIA, vốn đã được sử dụng rộng rãi trong cộng đồng nghiên cứu an ninh mạng.
    - Đồ thị nguồn gốc được xây dựng từ dữ liệu nhật ký này. Đồ thị nguồn gốc sẽ giúp chúng tôi mô hình hóa các mối quan hệ và sự tương tác trong hệ thống, từ đó nhận diện các hành vi bất thường.

3. Mô hình phát hiện xâm nhập:

    - Các mô hình học máy và học sâu được áp dụng trong việc phát hiện xâm nhập, đặc biệt là các mô hình như Graph Attention Embedding (GAE) và GraphSAGE.
    - Các phương pháp và kỹ thuật để tối ưu hóa và cải thiện hiệu suất của các mô hình này.

Bằng việc tập trung vào các đối tượng và phạm vi nghiên cứu như trên, chúng tôi hy vọng sẽ phát triển được một mô hình phát hiện xâm nhập hiệu quả, có khả năng ứng dụng trong thực tế và góp phần nâng cao khả năng bảo mật cho các hệ thống mạng.

1.5 Cấu trúc Khóa luận tốt nghiệp

Nội dung khóa luận được tổ chức theo cấu trúc 5 chương như sau:

- Chương 1: TỔNG QUAN ĐỀ TÀI
  Trình bày khái quát định hướng nghiên cứu, mục tiêu, phạm vi và cấu trúc của khóa luận.

- Chương 2: CƠ SỞ LÝ THUYẾT
  Trình bày các định nghĩa, khái niệm, cơ sở lý thuyết cũng như kiến thức nền tảng để thực hiện khóa luận. Bên cạnh đó, chúng tôi cũng trình bày sơ lược một số công trình liên quan đến đề tài và hướng nghiên cứu.

- Chương 3: PHƯƠNG PHÁP THỰC HIỆN
  Trình bày những nội dung chính về phương pháp thực hiện và mô hình được sử dụng.
- Chương 4: HIỆN THỰC, ĐÁNH GIÁ VÀ THẢO LUẬN
  Đề cập đến quá trình hiện thực hóa phương pháp đề cập ở Chương 3. Sau đó trình bày phương pháp thực nghiệm, đánh giá kết quả và một số thảo luận.
- Chương 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN
  Đưa ra kết luận về đề tài, đề xuất một số hướng phát triển mở rộng cho các nghiên cứu trong tương lai.

Chương 2

CƠ SỞ LÝ THUYẾT

Tóm tắt chương

Trong chương này, chúng tôi sẽ trình bày cơ sở lý thuyết cần thiết của khóa luận, bao gồm: Tấn công APT; Hệ thống phát hiện xâm nhập; Đồ thị nguồn gốc; Mô hình học máy và Tóm tắt về những công trình nghiên cứu liên quan.

2.1 Tấn công APT

2.1.1 Tổng quan

APT (Advanced Persistent Threat hay Mối đe dọa tinh vi liên tục) thường được thực hiện bởi một nhóm kẻ tấn công tiên tiến được tài trợ tốt bởi một tổ chức hoặc chính phủ để thu thập thông tin quan trọng về tổ chức hoặc chính phủ mục tiêu của họ [15]. APT là một thuật ngữ quân sự được chuyển đổi vào ngữ cảnh an ninh thông tin, thường chỉ các cuộc tấn công do các quốc gia tiến hành. APT được định nghĩa bởi sự kết hợp của ba từ, đó là: Advanced, Persistent và Threat.

- **Advanced - tinh vi:** Kẻ tấn công APT thường được tài trợ tốt và có quyền truy cập vào các công cụ và phương pháp tiên tiến cần thiết để thực hiện cuộc tấn công APT. Các phương pháp tiên tiến này bao gồm việc sử dụng nhiều vector tấn công để triển khai cũng như duy trì cuộc tấn công.
- **Persistent - liên tục:** Kẻ tấn công APT có độ quyết đoán và kiên trì cao, họ không bao giờ từ bỏ. Một khi họ xâm nhập vào hệ thống, họ cố gắng duy trì ẩn mình trong hệ thống càng lâu càng tốt. Họ lập kế hoạch sử dụng nhiều

kỹ thuật né tránh để tránh phát hiện từ phía hệ thống phát hiện xâm nhập của mục tiêu. Họ thực hiện "chậm và thấp" để tăng tỷ lệ thành công của họ.

- **Threat - đe dọa:** thường là mất thông tin nhạy cảm hoặc cản trở các thành phần hoặc nhiệm vụ quan trọng. Đây là những mối đe dọa đang ngày càng tăng với nhiều thực thể quốc gia và tổ chức có hệ thống bảo vệ cao bảo vệ nhiệm vụ và/hoặc dữ liệu của họ.

2.1.2 Phân biệt giữa các mối đe dọa truyền thống và APTs

Chúng tôi tóm tắt sự khác biệt giữa các mối đe dọa truyền thống và APT đối với một số thuộc tính tấn công tại bảng 2.1. Các đặc điểm phân biệt của tấn công APT gồm:

- **Mục tiêu cụ thể và rõ ràng:** Các cuộc tấn công APT là những cuộc tấn công nhắm mục tiêu cao, luôn có một mục tiêu rõ ràng. Các mục tiêu thường là chính phủ hoặc các tổ chức sở hữu giá trị tài sản trí tuệ đáng kể. Dựa trên số lượng các cuộc tấn công APT được FireEye phát hiện vào năm 2013 [8], mười ngành công nghiệp bị nhắm mục tiêu nhiều nhất là giáo dục, tài chính, công nghệ cao, chính phủ, tư vấn, năng lượng, hóa chất, viễn thông, chăm sóc sức khỏe, và hàng không vũ trụ. Cũng theo báo cáo quý 1 năm 2023 của Kaspersky [14], các cuộc tấn APT tiếp tục mở rộng phạm vi tấn công như cơ quan nhà nước, ngành hàng không, năng lượng, sản xuất, bất động sản, tài chính, viễn thông, nghiên cứu khoa học, công nghệ thông tin và trò chơi. Trong khi các cuộc tấn công truyền thống lan truyền rộng rãi để tăng cơ hội thành công và tối đa hóa thu hoạch, một cuộc tấn công APT chỉ tập trung vào các mục tiêu đã được xác định trước, giới hạn phạm vi tấn công của nó. Về mục tiêu tấn công, APT thường tìm kiếm các tài sản kỹ thuật số mang lại lợi thế cạnh tranh hoặc lợi ích chiến lược, như dữ liệu an ninh quốc gia, tài sản trí tuệ, bí mật thương mại,... trong khi các mối đe dọa truyền thống chủ yếu tìm kiếm thông tin cá nhân như dữ liệu thẻ tín dụng hoặc thông tin có giá trị chung để tạo ra lợi nhuận tài chính.

- **Những kẻ tấn công có tổ chức cao và nhiều nguồn lực:** Những kẻ đứng sau các cuộc APT thường là một nhóm hacker có kỹ năng, làm việc theo

cách phối hợp. Họ có thể làm việc trong đơn vị mạng của chính phủ/quân đội, hoặc được thuê làm lính đánh thuê mạng bởi các chính phủ và công ty tư nhân. Họ có nhiều nguồn lực từ cả khía cạnh tài chính và kỹ thuật. Điều này cung cấp cho họ khả năng làm việc trong thời gian dài, có quyền truy cập bằng cách phát triển hoặc mua các lỗ hổng zero-day và các công cụ tấn công. Khi họ được nhà nước tài trợ, họ thậm chí có thể hoạt động với sự hỗ trợ của quân đội hoặc tình báo nhà nước.

- **Chiến dịch dài hạn với những nỗ lực lặp đi lặp lại:** Một cuộc tấn công APT thường là một chiến dịch dài hạn, có thể không bị phát hiện trong mạng của mục tiêu trong nhiều tháng hoặc thậm chí là nhiều năm. Các tác nhân APT kiên trì tấn công mục tiêu và liên tục điều chỉnh nỗ lực của mình để hoàn thành công việc khi một nỗ lực trước đó thất bại. Điều này khác với các mối đe dọa truyền thống, vì những kẻ tấn công truyền thống thường nhắm vào một loạt các nạn nhân, và họ sẽ chuyển ngay sang một mục tiêu ít bảo mật hơn nếu không thể xâm nhập vào mục tiêu ban đầu.

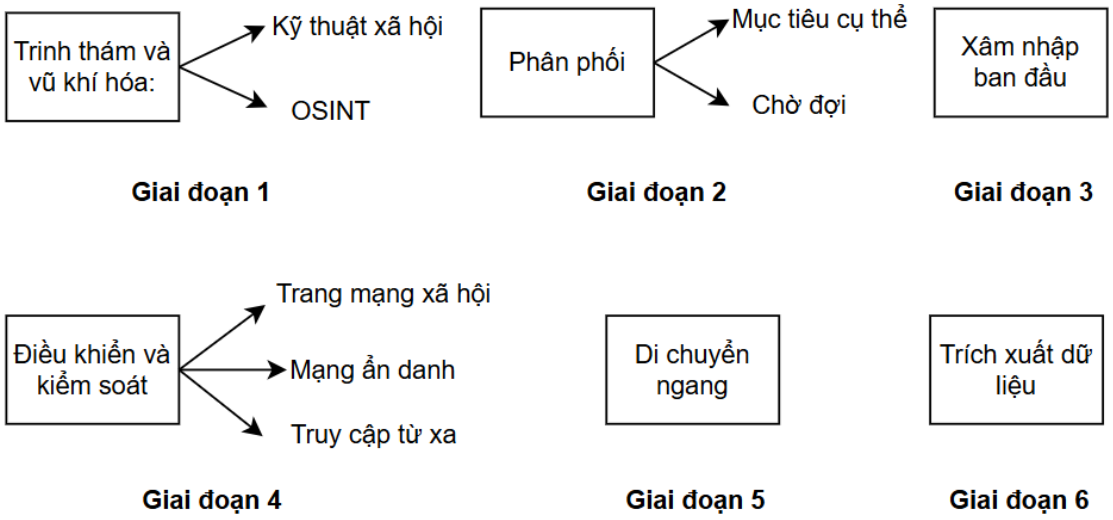
- **Kỹ thuật lén lút và né tránh:** Các cuộc tấn công APT là lén lút, có khả năng không bị phát hiện, ẩn mình trong lưu lượng mạng của doanh nghiệp, và tương tác vừa đủ để đạt được các mục tiêu đã xác định. Ví dụ, các tác nhân APT có thể sử dụng các lỗ hổng zero-day để tránh phát hiện dựa trên chữ ký, và mã hóa để làm rối lưu lượng mạng. Điều này khác với các cuộc tấn công truyền thống, nơi những kẻ tấn công thường sử dụng chiến thuật "Đập và Lấy" khiến người bảo vệ nhận ra.

BẢNG 2.1: Bảng tóm tắt khác biệt giữa tấn công truyền thống và tấn công APT [3]

	Tấn công truyền thống	**Tấn công APT**
Kẻ tấn công	Cá nhân, đơn vị đơn lẻ	Các nhóm tổ chức cao, tinh vi, quyết tâm và nguồn lực tốt
Mục tiêu	Khó xác định, thường là các hệ thống cá nhân	Các tổ chức cụ thể, các cơ quan chính phủ, các doanh nghiệp thương mại
Mục đích	Lợi ích tài chính, chứng tỏ khả năng	Lợi thế cạnh tranh, lợi ích chiến lược
Phương pháp tiếp cận	Thực hiện một lần, thời gian ngắn hạn	Các nỗ lực lặp đi lặp lại, ẩn nấp và chậm rãi, thích nghi để chống lại phòng thủ, thời gian dài hạn

2.1.3 Một số cách thức tấn công APT

Để đạt được mục tiêu được giao, những kẻ tấn công phải trải qua nhiều giai đoạn tấn công ở các hình thức khác nhau trong khi vẫn giữ được khả năng trốn tránh không bị phát hiện. Các giai đoạn này bao gồm việc có được quyền truy cập, quét mạng nội bộ và di chuyển từ một hệ thống này sang hệ thống khác trong mạng để đạt đến hệ thống mục tiêu và thực hiện hoạt động tổn hại của họ. Sau hoạt động tổn hại, những kẻ tấn công có thể chọn ở lại để tiếp tục các hoạt động độc hại trên các hệ thống khác trong mạng hoặc rời khỏi hệ thống sau khi dọn sạch; tùy thuộc vào yêu cầu từ nguồn tài trợ. Những giai đoạn đa dạng này thường bao gồm việc xâm nhập vào một trong các hệ thống trong mạng và sau đó thực hiện các bước nâng quyền cần thiết để đạt đến hệ thống mục tiêu, tiếp theo là truy cập vào các hệ thống nhạy cảm và gửi trạng thái/thông tin qua kết nối Internet đến trung tâm điều khiển và kiểm soát của kẻ tấn công. Hình 2.1 mô tả các giai đoạn thường gặp của một cuộc tấn công APT.



HÌNH 2.1: Một số giai đoạn của tấn công APT

2.2 Hệ thống phát hiện xâm nhập

2.2.1 Tổng quan

Hệ thống Phát hiện Xâm nhập (Intrusion Detection System - IDS) là một thành phần thiết yếu trong bảo mật mạng, đóng vai trò như một lớp phòng thủ để giám sát và phân tích các hoạt động trong hệ thống và mạng máy tính. IDS có nhiệm vụ phát hiện các hành vi đáng ngờ hoặc trái phép, giúp ngăn chặn và giảm thiểu thiệt hại từ các cuộc tấn công mạng.

IDS có khả năng phân tích các sự kiện xảy ra trong hệ thống, xác định các dấu hiệu bất thường hoặc các mẫu hành vi có thể là dấu hiệu của một cuộc tấn công mạng. Hệ thống này không chỉ giới hạn trong việc phát hiện mà còn cung cấp thông tin chi tiết để hỗ trợ các quản trị viên trong việc điều tra và phản ứng kịp thời với các sự cố bảo mật.

2.2.2 Phân loại IDS dựa trên nguồn dữ liệu

IDS được chia thành hai loại chính: Hệ thống Phát hiện xâm nhập dựa trên máy chủ (Host-based Intrusion Detection System - HIDS) và Hệ thống phát hiện xâm

nhập dựa trên mạng (Network-based Intrusion Detection System - NIDS).

2.2.3 Hệ thống phát hiện xâm nhập dựa trên máy chủ (HIDS)

HIDS hoạt động trên các thiết bị cá nhân như máy tính hoặc máy chủ, giám sát và phân tích các hoạt động nội bộ của hệ thống. HIDS tập trung vào việc kiểm tra các tệp nhật ký, cấu hình hệ thống, và các hoạt động của người dùng để phát hiện các hành vi bất thường. Các lợi ích của HIDS bao gồm:

- **Giám sát các thay đổi tệp:** HIDS có thể phát hiện các thay đổi trái phép trong các tệp quan trọng, giúp ngăn chặn các hành động phá hoại.
- **Phân tích hành vi người dùng:** Giám sát các hoạt động đăng nhập và đăng xuất, cùng với các hành vi của người dùng để phát hiện các hoạt động đáng ngờ.
- **Kiểm tra tính toàn vẹn của hệ thống:** HIDS theo dõi và xác nhận tính toàn vẹn của các tệp và cấu hình hệ thống, đảm bảo không có sự can thiệp trái phép.

2.2.4 Hệ thống Phát hiện Xâm nhập Dựa trên Mạng (NIDS)

NIDS giám sát lưu lượng mạng để phát hiện các cuộc tấn công vào hệ thống. NIDS hoạt động bằng cách phân tích các gói dữ liệu trên mạng, tìm kiếm các mẫu dữ liệu bất thường hoặc dấu hiệu của các cuộc tấn công. Các lợi ích của NIDS bao gồm:

- **Giám sát lưu lượng mạng:** Phát hiện các cuộc tấn công từ chối dịch vụ (DoS), các hoạt động quét mạng, và các cuộc tấn công vào giao thức mạng.
- **Phân tích mẫu dữ liệu:** Sử dụng các thuật toán để tìm kiếm các mẫu dữ liệu bất thường, giúp phát hiện các hành vi xâm nhập.
- **Khả năng mở rộng:** NIDS có thể giám sát toàn bộ mạng, giúp bảo vệ nhiều thiết bị cùng một lúc.

2.2.5 Phương pháp phát hiện xâm nhập

Có hai phương pháp chính để phát hiện xâm nhập: Phát hiện dựa trên chữ ký và Phát hiện dựa trên hành vi.

2.2.5.1 Phát hiện dựa trên chữ ký (Signature-based Detection)

Phương pháp này sử dụng cơ sở dữ liệu chứa các mẫu tấn công đã biết, được gọi là chữ ký, để so sánh với các hoạt động hiện tại trong hệ thống. Khi một hoạt động khớp với một chữ ký, IDS sẽ phát hiện và cảnh báo về cuộc tấn công. Ưu điểm của phương pháp này bao gồm:

- **Phát hiện nhanh:** Chữ ký cho phép phát hiện nhanh chóng các cuộc tấn công đã biết.
- **Độ chính xác cao:** Các chữ ký được thiết kế để phát hiện chính xác các mẫu tấn công cụ thể.

Tuy nhiên, phương pháp này cũng có nhược điểm:

- **Khả năng hạn chế với các cuộc tấn công mới:** Chữ ký chỉ có thể phát hiện các cuộc tấn công đã biết, không hiệu quả với các mẫu tấn công mới hoặc chưa được phát hiện.
- **Cập nhật liên tục:** Yêu cầu cập nhật thường xuyên cơ sở dữ liệu chữ ký để đảm bảo khả năng phát hiện.

2.2.5.2 Phát hiện dựa trên hành vi (Anomaly-based Detection)

Phương pháp này xây dựng mô hình hành vi bình thường của hệ thống hoặc người dùng và giám sát các hoạt động để phát hiện các hành vi bất thường. Khi một hoạt động vượt ra ngoài mô hình bình thường, IDS sẽ coi đó là một dấu hiệu của xâm nhập. Ưu điểm của phương pháp này bao gồm:

- **Phát hiện các cuộc tấn công mới:** Có khả năng phát hiện các mẫu tấn công mới hoặc chưa được biết đến.
- **Giám sát toàn diện:** Có thể phát hiện các hành vi bất thường trong toàn bộ hệ thống.

Nhược điểm của phương pháp này bao gồm:

- **Tỷ lệ báo động giả cao:** Các hoạt động hợp pháp nhưng không phổ biến có thể bị coi là bất thường.
- **Yêu cầu học máy:** Có thể phát hiện các hành vi bất thường trong toàn bộ hệ thống.

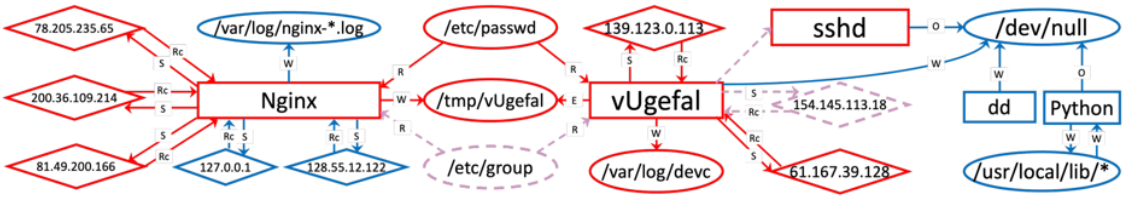
2.3 Đồ thị nguồn gốc

Nguồn gốc dữ liệu cấp hệ thống ghi lại các luồng dữ liệu giữa các đối tượng cấp hạt nhân, (ví dụ: tiến trình, tệp, socket). Nguồn gốc dữ liệu có thể được biểu diễn dưới dạng biểu đồ có hướng, được gọi là biểu đồ xuất xứ, trong đó các nút biểu thị các đối tượng cấp hạt nhân và các cạnh biểu thị các loại tương tác khác nhau (tức là các mối quan hệ phụ thuộc) giữa các đối tượng này.

Trong đồ thị nguồn gốc có chứa:

- **Các nút:** biểu diễn các đối tượng.
- **Các cạnh:** biểu diễn tương tác (mối quan hệ phụ thuộc) giữa các đối tượng. Những tương tác này thường là kết quả của các lời gọi hệ thống (system call).

Hình 2.2 là một ví dụ về đồ thị nguồn gốc. Trong đó, hình chữ nhật, hình oval, hình vuông đại diện tương ứng cho quy trình, tệp tin và socket. Các cạnh: R = Đọc, W = Viết, O = Mở, S = Gửi, Rc = Nhận, C = Sao chép và E = Thực thi. Các nút và cạnh đậm thể hiện lại cuộc tấn công. Các nút và cạnh màu hồng đứt đoạn là các hoạt động liên quan đến tấn công đã bỏ sót. Các nút và cạnh màu xanh dương là các hoạt động không được đề cập một cách rõ ràng trong định nghĩa đúng của cuộc tấn công nhưng được mô hình đưa vào.



HÌNH 2.2: Ví dụ về đồ thị nguồn gốc

Đồ thị nguồn gốc không chỉ hữu ích trong việc phát hiện xâm nhập mà còn trong việc phân tích nguyên nhân gốc rễ, xác định chính xác các bước tấn công và các thành phần bị ảnh hưởng. Điều này giúp cung cấp một cái nhìn toàn diện về an ninh hệ thống và hỗ trợ trong việc xây dựng các biện pháp phòng thủ hiệu quả.

Các nghiên cứu gần đây đã chỉ ra rằng việc sử dụng biểu đồ nguồn gốc có thể nâng cao hiệu quả phát hiện và phản ứng với các cuộc tấn công mạng. Các công nghệ mới như học máy và trí tuệ nhân tạo cũng đang được tích hợp để tự động hóa và cải thiện quá trình phân tích biểu đồ nguồn gốc, giúp hệ thống phát hiện xâm nhập trở nên mạnh mẽ và linh hoạt hơn.

2.4 Mô hình học máy

2.4.1 Tổng quan

Các mô hình học máy truyền thống ngày càng phổ biến và được ứng dụng rộng rãi trong các bài toán phân loại và dự đoán. Dựa trên các thuộc tính tĩnh đã được xác định từ trước và được trích xuất từ đối tượng, cùng với các quy luật được học từ dữ liệu huấn luyện, các mô hình học máy truyền thống có khả năng phân loại với tỉ lệ chính xác cao và tốc độ xử lý nhanh.

Bằng cách tận dụng các thuộc tính tĩnh để mô tả đối tượng hoặc dữ liệu đầu vào, các thuộc tính này có thể là các đặc trưng quan trọng, thông tin định danh, thông tin số liệu và nhiều thuộc tính khác. Các mô hình học máy sẽ học được các quy luật và mối quan hệ giữa các thuộc tính được học từ dữ liệu. Mô hình học máy truyền thống có nhiều ưu điểm. Đầu tiên, chúng dễ hiểu và có thể giải

thích một cách rõ ràng. Điều này giúp người dùng và các chuyên gia trong lĩnh vực liên quan dễ dàng áp dụng và giải thích kết quả của mô hình. Thứ hai, các mô hình học máy truyền thống có khả năng xử lý cả dữ liệu số và dữ liệu hạng mục. Chúng có khả năng làm việc với các tập dữ liệu lớn và xử lý nhiễu tương đối tốt. Có thể kể đến một số mô hình phổ biến thông dụng như Decition Tree, SVM, Logistic Regression,...

Dựa trên phương pháp, học máy có thể được chia thành các loại: Học máy giám sát, Học máy không giám sát, Học máy bán giám sát và Học máy tăng cường.

- **Học máy giám sát:** là quá trình học từ dữ liệu được gán nhãn trước đó. Dữ liệu đầu vào bao gồm các đặc trưng của đối tượng cần được dự đoán. Mục tiêu của học máy giám sát là phân loại đối tượng hoặc dự đoán giá trị của đầu ra (output) dựa trên các đặc trưng đó.
- **Học máy không giám sát:** là quá trình học từ dữ liệu không được gán nhãn. Quá trình này nhằm giúp tìm ra các mối liên hệ, mô hình hoặc cấu trúc trong dữ liệu để dễ dàng phân loại.
- **Học máy bán giám sát:** là quá trình học từ dữ liệu được gán một phần nhãn và một phần không được gán nhãn. Mục tiêu của quá trình này là học từ các dữ liệu được gán nhãn để dự đoán các dữ liệu không được gán nhãn.
- **Học máy tăng cường:** là quá trình học từ kinh nghiệm tự do của một hệ thống (agent) được bao quanh bởi môi trường và mục tiêu phải tìm cách tối đa hóa phần thưởng (reward) từ môi trường sau mỗi hành động (action).

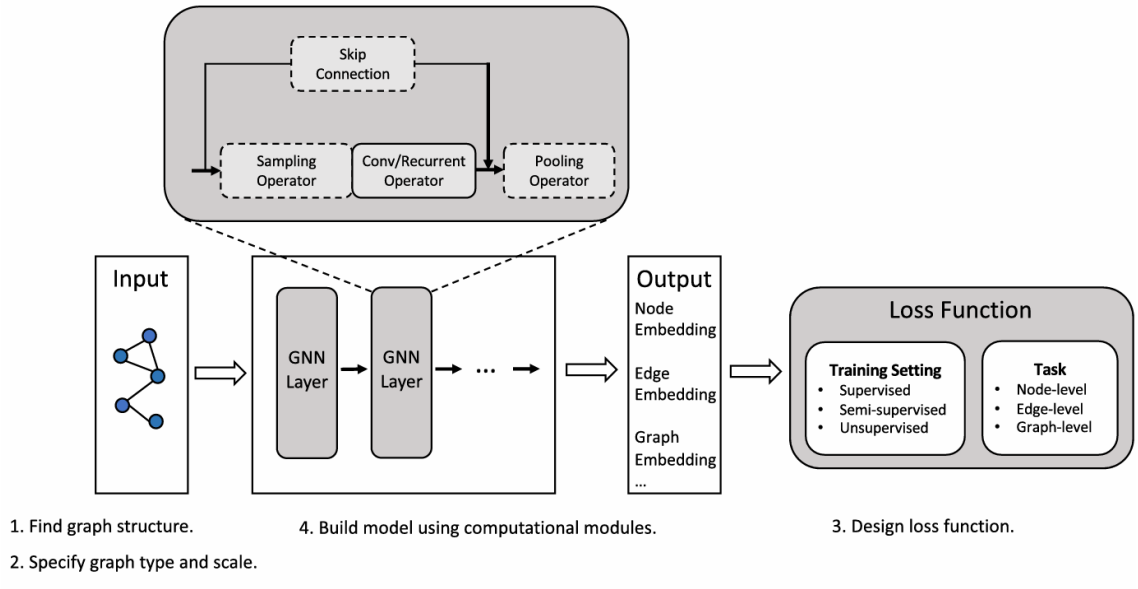
Tuy nhiên, mô hình học máy truyền thống có một số hạn chế. Với dữ liệu có mức độ phức tạp và mối quan hệ phi tuyến, các mô hình truyền thống có thể không hiệu quả và không thể mô hình hóa các quy luật phức tạp.

2.4.2 Mô hình học sâu

Mô hình học sâu (deep learning) có khả năng tự học và trích xuất các đặc trưng phức tạp từ dữ liệu đầu vào mà không cần phụ thuộc vào việc xác định trước các đặc trưng đó. Mô hình học sâu thường được áp dụng trong các nhiệm vụ như nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên và dự báo chuỗi thời gian.

Các mô hình học sâu thường có kiến trúc mạng nơ-ron sâu với nhiều lớp ẩn (hidden layers) giúp mô hình hóa các mối quan hệ phức tạp hơn và khám phá được những đặc trưng ẩn sâu trong dữ liệu. Với số lượng lớp ẩn lớn, mô hình học sâu có khả năng biểu diễn các hàm phức tạp và thực hiện các tác vụ phân loại và dự đoán chính xác.

Mô hình học sâu được sử dụng trong khóa luận là mạng nơ-ron đồ thị (graph neural network - GNN), là một loại mô hình học máy được thiết kế đặc biệt để làm việc với dữ liệu đồ thị. Hình 2.3 mô tả tổng quan kiến trúc của một mô hình GNN. GNN hoạt động bằng cách truyền thông tin qua các đỉnh và cạnh trong đồ thị. Mô hình học thông qua việc cập nhật và kết hợp thông tin từ các hàng xóm của mỗi đỉnh, cho phép nắm bắt thông tin cấu trúc và tương tác giữa các đối tượng trong đồ thị. Một trong những đặc điểm đáng chú ý của GNN là khả năng tích hợp thông tin từ cả đặc trưng của các đỉnh và cấu trúc đồ thị. Điều này cho phép GNN học mô hình phức tạp và biểu diễn các mối quan hệ phức tạp giữa các đối tượng trong đồ thị. GNN đã chứng tỏ được hiệu quả trong nhiều nhiệm vụ, bao gồm phân loại đồ thị, phân loại nút, dự đoán liên kết và nhúng đồ thị.



HÌNH 2.3: Hình ảnh tổng quan GNN [24]

2.4.3 Mạng nơ-rơn đồ thị cho phát hiện xâm nhập

Đối với công việc học biểu diễn đồ thị, cách tiếp cận chung là tìm một hàm ánh xạ để chiếu các nút thành một vector nhúng có kích thước cố định [2]. Sau đó, tập trung vào 2 kỹ thuật chính, đó là:

- **Học đặc tính (Inductive learning):** huấn luyện trên một tập hợp các đồ thị và sau đó dự đoán nhãn trên các đồ thị mới, được tạo thành từ các nút và cạnh không được nhìn thấy trong quá trình huấn luyện. Điều này giúp tổng quát hóa trên các mạng doanh nghiệp mới hoặc các máy chủ mới và có khả năng thực hiện suy luận trong các tình huống mới dựa trên trọng số đã được huấn luyện trước.

- **Học chuyển giao (Transductive learning):** dự đoán nhãn từ các nút và cạnh đã được nhìn thấy trong quá trình huấn luyện. Việc này có thể được huấn luyện trên một mạng cụ thể với một tập hợp cố định các máy chủ, hoặc trên một máy chủ cụ thể. Tuy nhiền cần phải được huấn luyện lại hoàn toàn nếu đồ thị thay đổi.

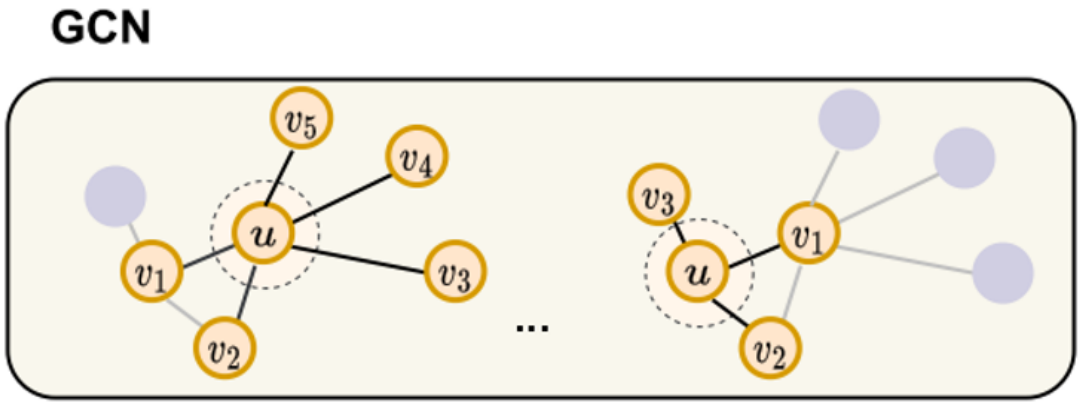
2 kỹ thuật random-walk thường được áp dụng:

- **Deep-walk:** Bắt đầu từ một nút xuất phát, sau đó chọn một nút láng giềng ngẫu nhiên để tiếp tục random-walk, theo một phân phối đồng đều. Các embedding của các nút được tạo ra bằng cách cố gắng tái tạo, trong khi các nút gần nhau có xu hướng xuất hiện cùng nhau trong nhiều random walk và biểu diễn vector chiều thấp của chúng có xu hướng tương tự nhau trong không gian ẩn. Điều này tạo ra thách thức đó là khó áp dụng trong một môi trường đặc trưng khi họ không chia sẻ các tham số giữa các nút, khiến phụ thuộc nhiều vào các siêu tham số và có xu hướng ưu tiên thông tin về sự gần kề hơn so với thông tin cấu trúc.

- **node2vec:** Cải tiến so với DeepWalk, trong đó sử dụng Breadth First Search (BFS) và Depth First Search (DFS) lấy cấu trúc cục bộ và toàn cục. Nó cho phép điều chỉnh các siêu tham số để kiểm soát cách di chuyển trong không gian đồ thị.

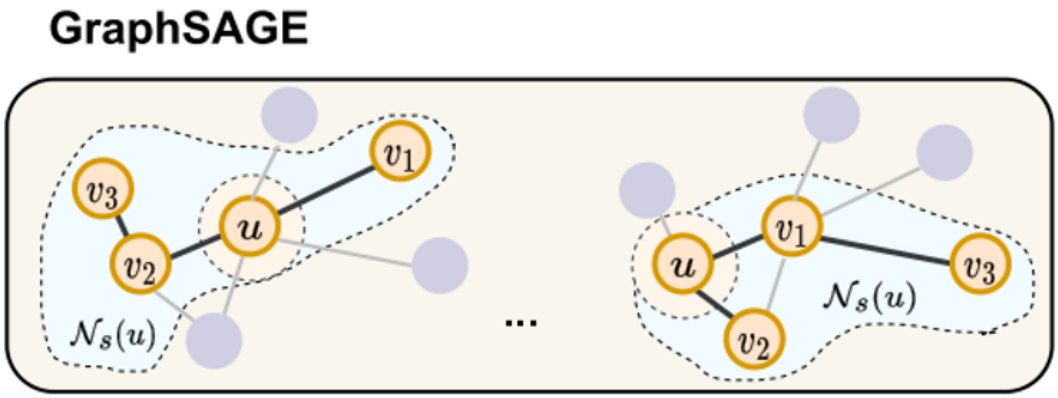
Một số mô hình GNN có thể áp dụng cho bài toán phát hiện xâm nhập:

- **Graph Convolutional Network (GCN) 2.4:** Mạng đồ thị tích chập, trong đó, các biểu diễn của nút được học từ các đặc trưng của nút cùng với cấu trúc của đồ thị. Mô hình yêu cầu lưu trữ toàn bộ ma trận kề với các tính năng tương ứng vào bộ nhớ, khiến mô hình này không thể sử dụng được trên các đồ thị rất lớn.



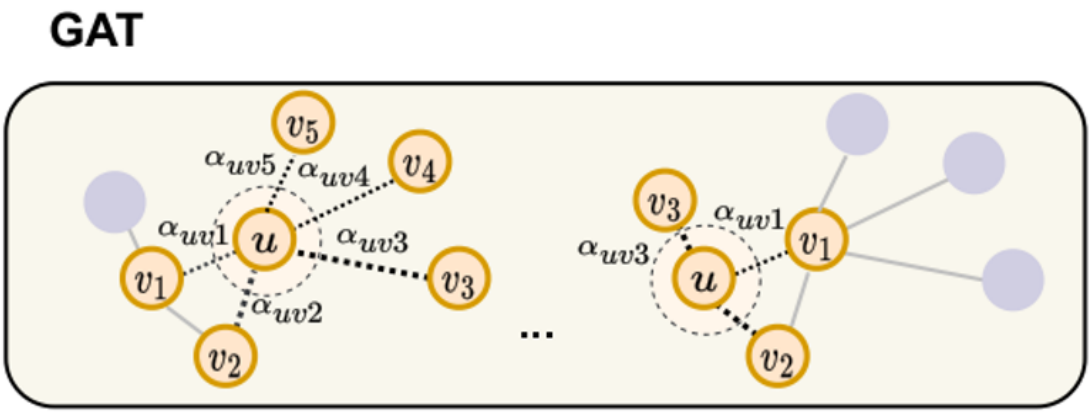
HÌNH 2.4: Hình ảnh mô tả GCN

- **Graph Sample and Aggregate (GraphSAGE) 2.5:** lấy mẫu một số lượng nút cố định từ hàng xóm của một nút cụ thể, thay vì sử dụng toàn bộ hàng xóm. Mô hình phù hợp huấn luyện theo mini-batch và huấn luyện đặc trưng, khiến cho mô hình này phù hợp với các đồ thị lớn có cấu trúc thay đổi.



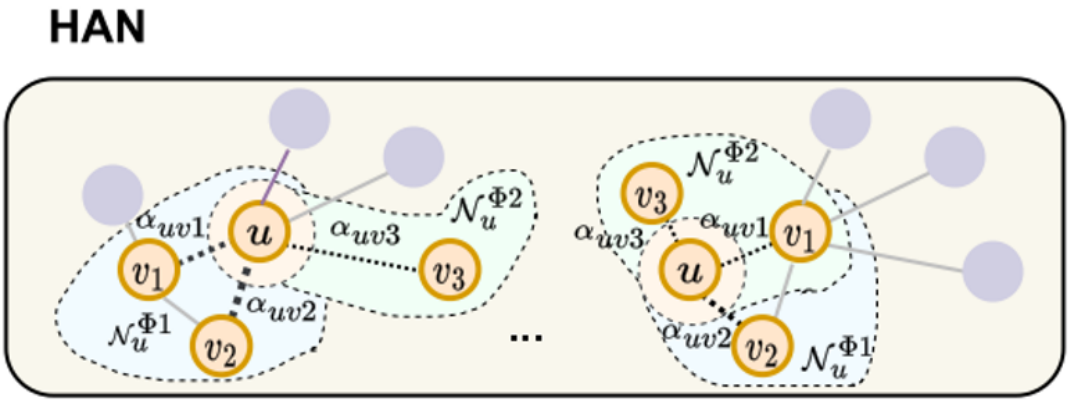
HÌNH 2.5: Hình ảnh mô tả GraphSAGE

- **Graph Attention Network (GAT) 2.6:** Tương tự như GCN, nhưng GAT quan tâm đến các nút lân cận phù hợp nhất. Mô hình có thể nắm bắt được nhiều mối quan hệ chi tiết hơn giữa các nút, dẫn đến giao tiếp tốt hơn và tổng hợp thông tin chất lượng cao hơn.



HÌNH 2.6: Hình ảnh mô tả GAT

- **Heterogeneous Graph Attention Network (HAN) 2.7:** sử dụng meta-path, là chuỗi các loại nút và cạnh nắm bắt ngữ nghĩa cụ thể trong đồ thị, do đó thường được áp dụng trong các biểu đồ không đồng nhất do thành phần không đồng nhất và khả năng trích xuất ngữ nghĩa mạnh mẽ của chúng.



HÌNH 2.7: Hình ảnh mô tả HAN

Tuy mô hình học sâu có khả năng mô hình hóa dữ liệu phức tạp và đạt được kết quả ấn tượng trong nhiều lĩnh vực, nhưng nó cũng đòi hỏi một lượng lớn dữ liệu huấn luyện và tài nguyên tính toán cao hơn so với các mô hình học máy truyền thống. Đồng thời, việc hiểu và giải thích quyết định của mô hình học sâu cũng trở nên phức tạp hơn rất nhiều.

2.5 Các công trình nghiên cứu liên quan

2.5.1 Một số thách thức

Công việc phát hiện xâm nhập gần đây sử dụng đồ thị nguồn gốc để chống lại sự xâm nhập hệ thống ngày càng tinh vi, đặc biệt là tấn công APT [3]. Các cuộc tấn công APT xâm nhập hệ thống mục tiêu một cách lén lút và duy trì sự hiện diện trong các máy chủ nạn nhân trong thời gian dài, thể hiện một kiểu tấn công "chậm và âm thầm". Trong suốt vòng đời của APT, kẻ tấn công thường sử dụng các khai thác zero-day khác nhau, một số trong đó thậm chí có thể được tùy chỉnh cho các hệ thống nạn nhân mục tiêu [16].

Vì những đặc điểm này, các PIDS hiện tại buộc phải đưa ra những sự đánh đổi. Chúng tôi nêu lại một số thách thức của công việc, trong đó xem xét 4 khía cạnh chính xuất hiện ở lượng lớn công trình nghiên cứu lĩnh vực này.

- **Tính không phụ thuộc vào kiểu tấn công:** Kiểu tấn công "chậm và âm thầm" của APTs làm cho việc phát hiện dựa trên sự bất thường trở nên khó khăn, vì hoạt động tấn công có thể ẩn trong một lượng lớn hoạt động lành mạnh và xuất hiện giống với hành vi bình thường nếu ngữ cảnh thực thi không được xem xét đầy đủ [11]. Ví dụ, trong kịch bản với dataset CADETS_E3, trong số 10.1 triệu cạnh, chỉ xác định được khoảng 1248 cạnh liên quan đến cuộc tấn công, chiếm chỉ 0.012% tổng số nhật ký; hay đối với dataset THEIA_E3, số cạnh tấn công chỉ chiếm 0.01% so với tổng số cạnh là 32.4 triệu cạnh. Để vượt qua thách thức này, các PIDSes như Holmes [17] và RapSheet [12] sử dụng kiến thức tình báo mối đe dọa hiện có để thủ công tạo ra các quy tắc ghép đồ thị chỉ ra sự hiện diện của APT. Tuy nhiên, khi các khai thác mới xuất hiện, chúng phải liên tục cập nhật cơ sở kiến thức của

mình, bao gồm các quy tắc bổ sung. Theo cấu trúc này, chúng sẽ luôn tụt hậu so với các đối thủ tinh vi thực hiện các cuộc tấn công chưa từng được biết đến.

- **Tái cấu trúc tấn công:** Các PIDS như Unicorn [11] và ThreaTrace [20] sử dụng phương pháp dựa trên sự bất thường để phát hiện hoạt động hệ thống lệch đáng kể so với hành vi lành mạnh đã biết. Mặc dù chúng không yêu cầu kiến thức trước về các đặc điểm của APT (không giống như Holmes), phát hiện của chúng cung cấp rất ít thông tin để giúp các quản trị viên hệ thống hiểu rõ cuộc tấn công. Kết quả là, cuộc điều tra pháp chứng theo sau thường liên quan đến việc kiểm tra thủ công kéo dài các đồ thị nguồn gốc lớn. Ví dụ, Unicorn giảm một đồ thị nguồn gốc xuống một vector đặc trưng gọn nhẹ để mô hình hóa hành vi hệ thống, nhưng một vector đặc trưng bất thường tương ứng với toàn bộ đồ thị nguồn gốc. Còn ThreaTrace, chỉ định rõ các nút bất thường có thể liên quan đến cuộc tấn công. Mặc dù những nút này có thể hữu ích ngay cả khi hoạt động độc hại hòa lẫn với các điểm khởi đầu, các quản trị viên hệ thống vẫn cần theo dõi thủ công hàng ngàn cạnh để hiểu toàn bộ câu chuyện tấn công. ThreaTrace nhận ra hạn chế này và thừa nhận khoảng cách giữa việc phát hiện dựa trên sự bất thường và tái cấu trúc cuộc tấn công.

- **Phạm vi:** Các PIDS như Winnower [12] xây dựng các mẫu đồ thị lành tính để làm nổi bật các đồ thị con bất thường không phù hợp với các mẫu này. Mặc dù điều này giúp phân tích pháp chứng, nhưng không phù hợp để phát hiện APT, vì nó không thể mở rộng đến các đồ thị lớn. Thay vào đó, Winnower tập trung vào phạm vi ứng dụng và phân tích các đồ thị nguồn gốc nhỏ hơn nhiều so với những đồ thị có thể mô tả hoạt động toàn hệ thống thực tế dưới sự tác động của các cuộc tấn công APT. Vì vậy, phải chạy ít nhất nhiều phiên bản của Winnower nhắm mục tiêu vào các ứng dụng khác nhau như Firefox, mail,...) để có thể phát hiện. Trong thực tế, một máy trạm có thể chạy nhiều chục ứng dụng, tất cả phải được giám sát riêng lẻ bởi Winnower, vì không biết trước ứng dụng nào sẽ liên quan đến APT. Tuy nhiên ngay cả khi đó thì cũng không rõ liệu phương pháp tiếp cận tách biệt, tập trung vào ứng dụng của Winnower có hiệu quả hay không. Điều này là do luồng

thông tin giữa các quy trình rất quan trọng để phát hiện APTs [11], nhưng Winnower không nhận biết điều này. Giống như Winnower, SIGL [10] giới hạn phát hiện của mình vào các bất thường trong quá trình cài đặt phần mềm; do đó, nó cũng không thể phân tích một đồ thị nguồn gốc có hàng triệu cạnh. Hơn nữa, giống như ThreaTrace, SIGL chỉ định rõ các nút bất thường, do đó không thể tái cấu trúc hoạt động tấn công.

- **Tính kịp thời:** Việc phát hiện APT và phân tích pháp chứng kịp thời rất quan trọng để nhanh chóng xác định cuộc tấn công và thực hiện các biện pháp khắc phục. Các PIDS như Poirot [18] ghép các chữ ký đồ thị phức tạp, mỗi chữ ký mô tả hành vi của một chương trình phần mềm độc hại cụ thể. Điều này giúp hiểu rõ mối đe dọa nhanh chóng sau khi mối đe dọa được ghép. Tuy nhiên, ngay cả khi bỏ qua vấn đề về tính không phụ thuộc vào kiểu tấn công, quá trình ghép của Poirot diễn ra chậm và do đó không phù hợp để phát hiện thời gian thực. Có hai lý do chính cho việc này. *Thứ nhất*, Poirot mất vài phút để tìm kiếm mỗi chữ ký trong một đồ thị nguồn gốc. Do đó, phương pháp này không thể mở rộng khi số lượng chữ ký tăng lên. *Thứ hai*, việc ghép chỉ thành công nếu một chương trình phần mềm độc hại thể hiện đầy đủ hành vi của nó như mô tả trong chữ ký. Vì vậy, Poirot phải lặp đi lặp lại việc cố gắng ghép các chữ ký đồ thị tương tự khi đồ thị phát triển theo thời gian, điều này càng làm trầm trọng thêm vấn đề mở rộng.

2.5.2 Các công trình nghiên cứu gần đây

Mặc dù các hệ thống phát hiện xâm nhập dựa trên nguồn gốc (PIDS) có khả năng phát hiện khá tốt nhưng cũng còn một số hạn chế. Như đã trình bày, các hệ thống buộc phải đưa ra những sự đánh đổi ít nhất một trong bốn khía cạnh. Chúng tôi xin tóm tắt và nêu một số hạn chế của các hệ thống này.

Đối với các hệ thống dựa trên chữ ký, các phương pháp dự đoán hoặc dấu vết tấn công đã biết có thể bị trốn tránh khi những kẻ tấn công điều chỉnh mô hình của chúng [1] [17]. Một số hệ thống chọn cách xây dựng một đồ thị nguồn gốc duy nhất cho toàn bộ hệ thống từ nhật ký [10] [13], tuy nhiên điều này làm cho việc chi phí xử lý đầu vào lớn và số lượng cảnh báo sai cũng tăng lên. Một số hệ thống được xây dựng dựa trên các ảnh chụp nhanh bất thường có độ chi tiết

thấp vì các nhà phân tích phải phân tích tất cả các thực thể/tương tác trong các ảnh chụp nhanh bất thường. Mặt khác, các hệ thống này cung cấp ít thông tin để giúp người quản trị có thể nắm bắt những gì thực sự xảy ra trong hệ thống của họ trong các cuộc tấn công [20] [11] [23]. Yang và cộng sự [22] xây dựng mô hình phát hiện các điểm bất thường ở cấp độ biểu đồ. Để hỗ trợ điều tra tấn công chi tiết hơn, nó xếp hạng các nút biểu đồ dựa trên mức độ bất thường của chúng, tương tự như [10]. Vì vậy, công tác điều tra sau phát hiện vẫn tốn nhiều công sức. Bên cạnh đó, ProGrapher [22] là mã nguồn đóng và báo cáo độ chính xác phát hiện tổng thể chưa quá vượt trội.

Chương 3

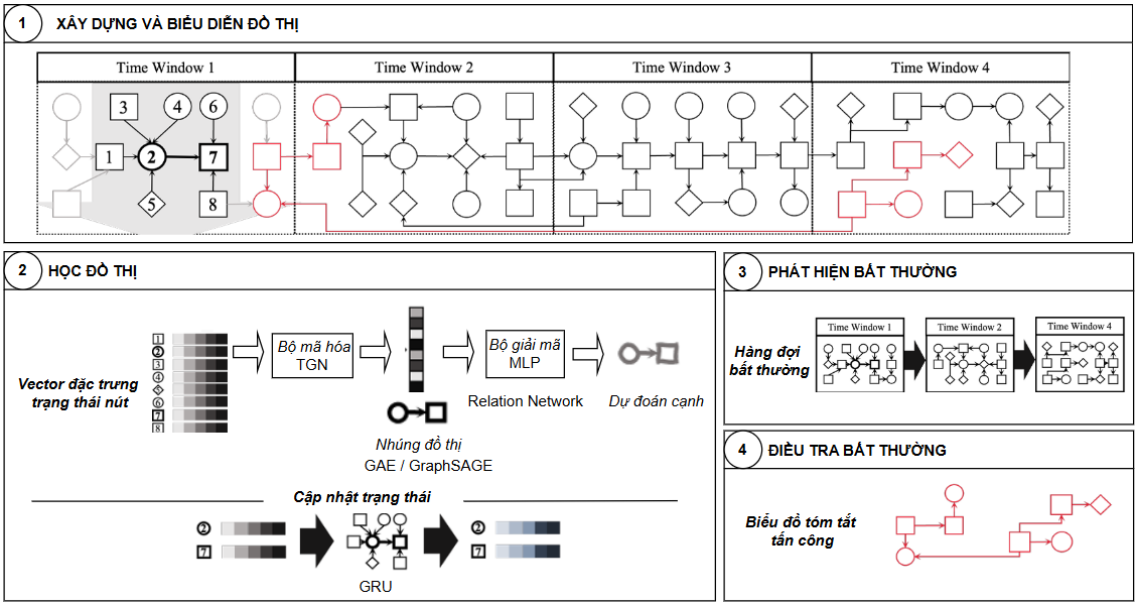
PHƯƠNG PHÁP THỰC HIỆN

Tóm tắt chương

Trong chương này chúng tôi giới thiệu mô hình, dựa trên đồ thị nguồn gốc để phát hiện tấn công APT. Hệ thống biểu diễn các hệ thống và sự kiện từ dữ liệu ghi nhật ký dưới dạng đồ thị, phân loại và mã hóa các đặc điểm để bảo vệ tính toàn vẹn và tăng cường bảo mật. Mô hình sử dụng kỹ thuật học đồ thị mã hóa - giải mã và hai phương pháp nhúng, GraphSAGE và GAE, để trích xuất và học đặc trưng từ đồ thị. Bằng cách cập nhật trạng thái nút thông qua mô hình GRU, mô hình giúp phát hiện và phân tích các hành vi không bình thường trong các cuộc tấn công APT, cung cấp nền tảng quan trọng cho an ninh mạng và phân tích dữ liệu đồ thị.

3.1 Kiến trúc tổng quát

Ở chương này chúng tôi sẽ trình bày kiến trúc mô hình phát hiện tấn công APT dựa trên đồ thị nguồn gốc.



HÌNH 3.1: Hình ảnh mô tả kiến trúc tổng quát của mô hình

Hệ thống phát hiện xâm nhập dựa trên bất thường và điều tra vụ tấn công dựa trên mô hình có tên Kairos [4]. Trong đó, tận dụng học sâu đồ thị hiện đại và khám phá cộng đồng thông qua các phụ thuộc nguyên nhân trong đồ thị nguồn gốc để phát hiện hành vi hệ thống bất thường mà không cần biết trước bất kỳ đặc tính tấn công cụ thể nào; và liên kết các bất thường phát hiện dựa trên luồng thông tin giữa các đối tượng. Sau cùng, cung cấp biểu đồ tóm tắt ngắn gọn để hỗ trợ phân tích hậu cần của con người trong chuỗi xử lý.

3.2 Xây dựng và biểu diễn đồ thị

Mô hình xây dựng biểu đồ nguồn gốc toàn bộ hệ thống từ dữ liệu thu thập từ các cơ sở hạ tầng ghi nhật ký. Trong đó, xem xét ba loại đối tượng và bảy loại tương tác. Mô hình chuyển đổi mỗi sự kiện thành một cạnh hướng, được đánh dấu thời gian, trong đó nút nguồn đại diện cho chủ thể của sự kiện và nút đích đại diện cho đối tượng đang được thực hiện.

Mô hình mã hóa đặc điểm của một nút bằng cách sử dụng một kỹ thuật băm đặc trưng phân cấp dựa trên các thuộc tính của nút. Băm đặc trưng phân cấp chuyển các vectơ đầu vào có chiều cao vào không gian đặc trưng có chiều thấp hơn trong

BẢNG 3.1: Bảng đối tượng và tương tác của hệ thống

Đối tượng	**Tương tác**	**Thuộc tính của đối tượng**
Tiến trình	Clone	Đường dẫn image
Tệp	Read, Write, Open, Exec	Đường dẫn tệp
Socket	Send, Receive	IP/port nguồn/đích

khi giữ nguyên tính tương đồng phân cấp giữa đầu vào ban đầu. Do đó, hai tệp nằm trong cùng một thư mục cha (ví dụ /var/log/wdev và /var/log/xdev) được ánh xạ gần nhau hơn trong không gian đặc trưng so với một tệp ở một thư mục khác (ví dụ /home/admin/profile).

Để thực hiện băm đặc trưng phân cấp, mô hình mã hóa thuộc tính của một nút nhiều lần, mỗi lần ở một cấp độ phân cấp khác nhau. Ví dụ: đối với một nút tệp với đường dẫn /home/admin/clean, mô hình tạo ba chuỗi con của thuộc tính đường dẫn: /home, /home/admin và /home/admin/clean; đối với một nút socket với địa chỉ IP 161.116.88.72, tạo ra bốn chuỗi con: 161, 161.116, 161.116.88 và 161.116.88.72.

Băm đặc trưng phân cấp giả định rằng hai thực thể cấp hạt nhân có ý nghĩa tương tự sẽ có các đặc điểm phân cấp tương tự. Có một khả năng thường xuyên xảy ra đó là kẻ tấn công cố gắng thay đổi thuộc tính của một thực thể để tránh phát hiện. Tuy nhiên, việc học đồ thị sẽ cập nhật những vector đặc trưng ban đầu này dựa trên tương đồng thời gian và cấu trúc, làm cho điều này khó có thể thay đổi và vượt qua mô hình.

3.3 Học đồ thị

Việc tính năng hóa nút chỉ thu thập thông tin về thuộc tính của các thực thể hệ thống mà không xem xét bất kỳ mối quan hệ cấu trúc (tức là tương tác giữa một thực thể và các thực thể khác) hoặc thời gian (tức là chuỗi sự kiện liên quan đến một thực thể) giữa các thực thể cá nhân và phần còn lại của đồ thị nguồn hốc. Điều này không may, vì đồ thị xuất xứ đang phát triển chính nó, mô tả hành vi động của một hệ thống, rõ ràng thể hiện các mối quan hệ như vậy. Quan trọng hơn, những mối quan hệ này cung cấp thông tin ngữ cảnh phong phú, cho phép

chúng ta mô hình hóa hành vi hệ thống cơ bản (tốt) và phân biệt sự bất thường so với cơ sở.

Ví dụ: Quá trình tiêm mã dẫn đến việc thực thi mã tùy ý trong không gian địa chỉ của một quy trình hợp lệ. Trong khi việc thực thi độc hại được ẩn dưới quy trình hợp lệ (tức là thuộc tính của quy trình vẫn giữ nguyên), dưới sự ảnh hưởng của kẻ tấn công, quy trình bị chiếm đó sẽ thể hiện các tương tác phổ biến khác biệt từ hoạt động bình thường của nó (như truy cập các tài nguyên hệ thống đặc quyền mà quy trình thông thường không cần). Những tương tác này được phản ánh như là những mối quan hệ cấu trúc bất thường trong đồ thị xuất xứ.

Thông tin thời gian có thể làm sáng tỏ những sự khác biệt hành vi; những khác biệt này khó, nếu không thể, xác định nếu chỉ nhìn vào các bản chụp tĩnh của một đồ thị xuất xứ động. Ví dụ: Một cuộc tấn công DDoS nhanh chóng làm cho một hệ thống mục tiêu bị quá tải với một lượng lớn kết nối mạng có thể dẫn đến cùng một cấu trúc đồ thị như một hệ thống không bị tấn công xử lý cùng một lượng kết nối trong một khoảng thời gian hợp lý. Nếu không tính đến mối quan hệ thời gian, sẽ gây khó khăn trong việc phát hiện cuộc tấn công bằng cách so sánh chỉ cấu trúc đồ thị.

Mô hình học cả mối quan hệ thời gian và cấu trúc trong đồ thị xuất xứ. Việc học đồ thị theo mô hình mã hóa - giải mã. Khi một cạnh mới xuất hiện trong đồ thị tại thời điểm t, bộ mã hóa nhúng vào một biểu diễn tiềm ẩn dựa trên trạng thái của khu vực lân cận của nó ngay trước thời điểm t. Nói cách khác, biểu diễn cạnh tóm lược các đặc trưng của đồ thị. Sau đó, bộ giải mã nhận biểu diễn cạnh từ bộ mã hóa và dự đoán loại của cạnh dưới dạng một phân phối xác suất, tức là xác suất của cạnh thuộc trong chín loại có thể.

Mô hình đồng thời huấn luyện encoder và decoder chỉ trên đồ thị nguồn gốc của hành vi lành tính. Mục tiêu của việc đào tạo là giảm thiểu sự khác biệt giữa loại cạnh thực tế (khi một cạnh mới xuất hiện trong đồ thị) và loại được dự đoán bởi decoder từ biểu diễn của nó. Sự khác biệt này là lỗi tái tạo. Ở thời điểm kiểm tra, decoder gán một lỗi tái tạo nhỏ cho một cạnh nếu biểu diễn của nó mã hóa cấu trúc đồ thị giống với ngữ cảnh cấu trúc đã được học từ đồ thị hành vi bình thường. Ngược lại, một lỗi tái tạo lớn được gán, độ lớn của nó phụ thuộc vào mức độ sai lệch trong cả hai ngữ cảnh.

3.3.1 Bộ mã hóa

Mô hình sử dụng một kiến trúc mạng đồ thị thời gian (temporal graph network - TGN) để mã hóa các đặc trưng của đồ thị xuất xứ thành các biểu diễn cạnh. Tại thời điểm t, mô hình tạo ra một biểu diễn cạnh cho cạnh mới bằng một mô hình dựa trên GNN gọi là GAE (Graph Attention Embedding) hoặc GraphSAGE.

3.3.2 Bộ nhúng đồ thị

Trong mô hình, trạng thái của cấu trúc được đại diện bởi trạng thái của tất cả các nút trong cấu trúc đó. Mỗi trạng thái nút là một vector đặc trưng mô tả lịch sử của các thay đổi đồ thị liên quan đến nút đó. Khi một nút mới xuất hiện trong đồ thị, trạng thái của nó được khởi tạo thành vector đặc trưng với tất cả các giá trị bằng không, vì không có thông tin lịch sử nào về nút đó.

Mỗi cạnh trong đồ thị được mã hóa dưới dạng một chuỗi nối của vector đặc trưng nhúng của nút nguồn và nút đích, và sử dụng mã hóa one-hot cho mỗi loại cạnh. Khi các cạnh mới thay đổi vùng lân cận của nút, mô hình cập nhật trạng thái của nút để phản ánh sự thay đổi này.

Việc phát hiện các nút bất thường trong đồ thị nguồn gốc là một thách thức lớn, đặc biệt khi không có sẵn các mẫu tấn công đã biết trước (prior knowledge).

Phương pháp truyền thống của việc phát hiện sự bất thường thường sử dụng dữ liệu thiện và bất thường để huấn luyện một mô hình phân loại nhị phân trong chế độ giám sát. Tuy nhiên, vì chúng ta giả định không có kiến thức trước về các mẫu tấn công trong giai đoạn huấn luyện, nên không thể huấn luyện mô hình với dữ liệu đã được gán nhãn nhị phân.

Ở đây chúng tôi sử dụng 2 phương pháp nhúng là GraphSAGE và GAE.

3.3.2.1 Tổng quan về GraphSAGE

GraphSAGE (Graph Sample and Aggregate) là một kỹ thuật học sâu trên đồ thị, được thiết kế để trích xuất các đặc trưng của nút bằng cách tổng hợp thông tin từ các nút láng giềng của nó. Thay vì xử lý toàn bộ đồ thị, GraphSAGE sử dụng một

phương pháp lấy mẫu để chỉ xem xét một tập con của các nút láng giềng, giúp giảm chi phí tính toán và cải thiện hiệu quả. GraphSAGE sử dụng một phương pháp trích xuất đặc trưng và gán nhãn cho các nút, cho phép huấn luyện mô hình trong chế độ giám sát mà không cần dữ liệu bất thường

Các thành phần chính của GraphSAGE:

- **Lấy mẫu nút láng giềng:** GraphSAGE lấy mẫu một số nút láng giềng để giảm bớt sự phức tạp tính toán.
- **Tổng hợp thông tin:** Sử dụng các hàm tổng hợp để kết hợp các vector đặc trưng của các nút láng giềng. Các hàm tổng hợp phổ biến bao gồm Mean, LSTM, và Pooling.
- **Kết hợp đặc trưng:** Vector đặc trưng của mỗi nút được kết hợp với vector đặc trưng tổng hợp từ các nút láng giềng để tạo ra vector đặc trưng mới.

3.3.2.2 Chi tiết về GraphSAGE

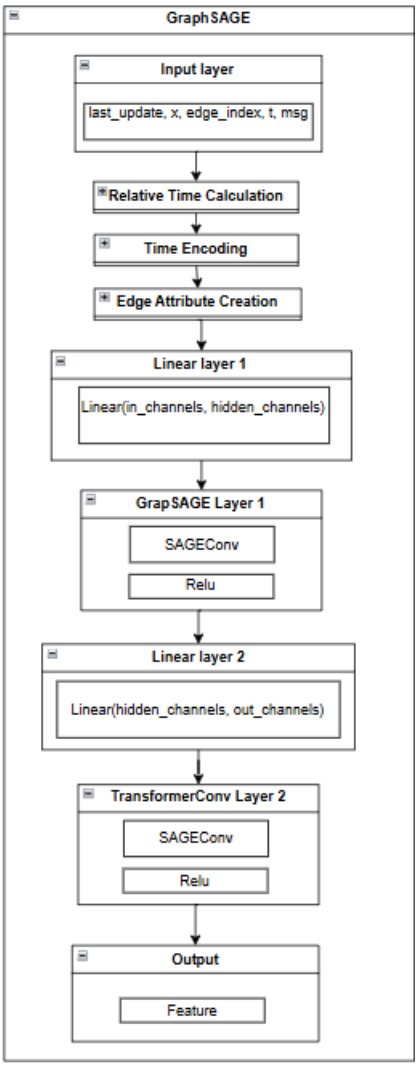
Khởi tạo vector đặc trưng: Mỗi nút trong đồ thị được khởi tạo với một vector đặc trưng ban đầu. Đối với các nút mới xuất hiện, vector đặc trưng của chúng sẽ được khởi tạo với tất cả các giá trị bằng không vì không có thông tin lịch sử nào về các nút này.

Mã hóa thời gian Mã hóa thời gian là một bước quan trọng để phản ánh sự thay đổi trong thời gian của các nút. Chênh lệch thời gian giữa các nút láng giềng được mã hóa và kết hợp với các thông điệp truyền qua các cạnh để tạo ra các vector đặc trưng cho các cạnh.

Trích Xuất Đặc Trưng Với GraphSAGE: Mô hình GraphSAGE bao gồm các lớp Linear và SAGEConv để biến đổi và trích xuất đặc trưng của nút. Cụ thể, mô hình bao gồm các bước sau:

- **Linear Transformation:** Biến đổi vector đặc trưng ban đầu của nút thông qua các lớp Linear trước khi đưa vào các lớp SAGEConv.
- **SAGEConv Layers:** Sử dụng các lớp SAGEConv để tổng hợp và kết hợp thông tin từ các nút láng giềng, giúp cải thiện vector đặc trưng của nút.

Kiến trúc của mô hình GraphSAGE với các lớp Linear có thể được mô tả chi tiết trong hình 3.2.



HÌNH 3.2: Hình ảnh mô tả kiến trúc của mô hình GraphSAGE

Đầu vào:

- X: Vector đặc trưng của các nút.
- last_update: Thời gian cập nhật cuối cùng của các nút.
- edge_index: Danh sách các cạnh trong đồ thị.

- t: Thời gian hiện tại.
- msg: Thông điệp liên quan đến các cạnh.

Mã Hóa Thời Gian (time_enc): Mã hóa sự chênh lệch thời gian giữa các nút bằng cách sử dụng lớp mã hóa thời gian (time_enc).

Kết Hợp Thông Tin: Kết hợp các vector mã hóa thời gian (rel_t_enc) với thông điệp (msg) để tạo ra edge_attr.

Biến Đổi Đặc Trưng Với SAGEConv:

- Linear1: Sử dụng lớp Linear đầu tiên (lin1) để biến đổi đặc trưng của các nút.
- SAGEConv1: Sử dụng lớp SAGEConv đầu tiên (sage_conv1) để tổng hợp thông tin từ các nút láng giềng.
- Linear2: Sử dụng lớp Linear thứ hai (lin2) để tiếp tục biến đổi đặc trưng của các nút.
- SAGEConv2: Sử dụng lớp SAGEConv thứ hai (sage_conv2) để tiếp tục tổng hợp và kết hợp thông tin từ các nút láng giềng.

GraphSAGE là một phương pháp hiệu quả để trích xuất và học các đặc trưng từ đồ thị. Bằng cách kết hợp thông tin từ các nút láng giềng và sử dụng các lớp Linear và SAGEConv, mô hình GraphSAGE có thể phát hiện các bất thường trong đồ thị một cách hiệu quả. Điều này giúp tăng cường bảo mật cho hệ thống và cung cấp các thông tin quan trọng trong phân tích dữ liệu đồ thị.

3.3.2.3 Tổng quan về GAE

Graph Attention Embedding (GAE) là một kỹ thuật học sâu trên đồ thị, sử dụng các cơ chế chú ý (attention mechanisms) để trích xuất các đặc trưng từ các nút và cạnh trong đồ thị. GAE có khả năng xử lý các đồ thị không đồng nhất, nơi mà các nút và cạnh có thể có các loại và đặc trưng khác nhau. Cơ chế chú ý trong GAE cho phép mô hình tập trung vào các nút và cạnh quan trọng, cải thiện khả năng trích xuất đặc trưng và phát hiện bất thường.

Các thành phần chính của GAE:

- **TransformerConv:** Lớp convolution dựa trên cơ chế chú ý từ mô hình Transformer. Nó cho phép mô hình học được trọng số của các nút láng giềng dựa trên tầm quan trọng của chúng.
- **Time Encoding:** Mã hóa thời gian để kết hợp thông tin thời gian vào vector đặc trưng của cạnh. Điều này giúp mô hình nắm bắt được sự thay đổi theo thời gian của các quan hệ trong đồ thị.

Chi tiết về GAE

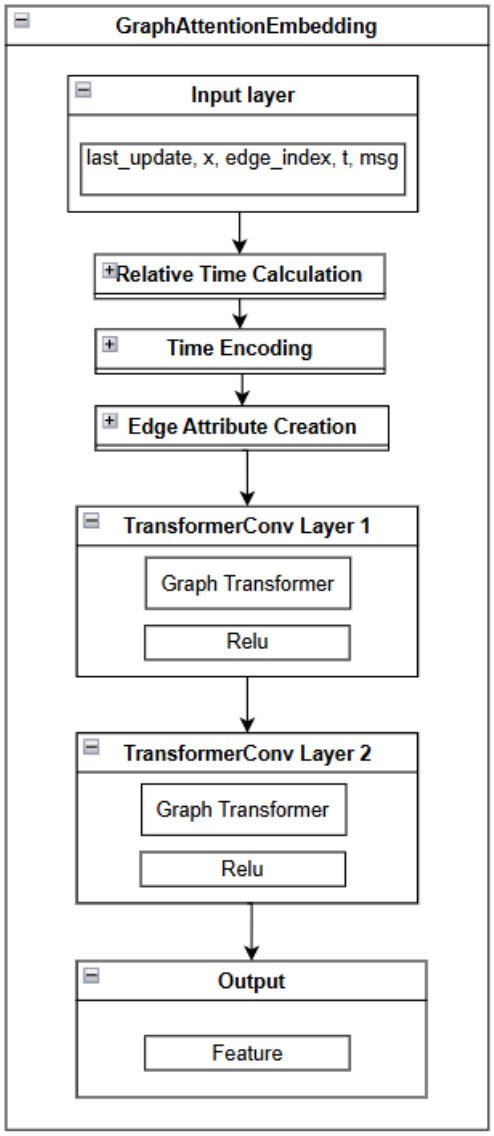
Khởi Tạo Vector Đặc Trưng: Mỗi nút trong đồ thị được khởi tạo với một vector đặc trưng ban đầu. Đối với các nút mới xuất hiện, vector đặc trưng của chúng sẽ được khởi tạo với tất cả các giá trị bằng không vì không có thông tin lịch sử nào về các nút này. Việc khởi tạo này giúp mô hình có thể học được từ dữ liệu mới và cập nhật các đặc trưng của nút theo thời gian.

Mã Hóa Thời Gian: Mã hóa thời gian là một bước quan trọng để phản ánh sự thay đổi trong thời gian của các nút. Chênh lệch thời gian giữa các nút láng giềng được mã hóa và kết hợp với các thông điệp truyền qua các cạnh để tạo ra các vector đặc trưng cho các cạnh. Mã hóa thời gian giúp mô hình hiểu được sự thay đổi theo thời gian và từ đó xác định được các bất thường liên quan đến thời gian.

Trích Xuất Đặc Trưng Với GAE: Mô hình Graph Attention Embedding bao gồm các lớp TransformerConv để biến đổi và trích xuất đặc trưng của nút. Cụ thể, mô hình bao gồm các bước sau:

- **Mã Hóa Thời Gian:** Tính toán sự chênh lệch thời gian giữa các nút và mã hóa sự chênh lệch này bằng lớp mã hóa thời gian (time_enc).
- **Kết Hợp Thông Tin:** Kết hợp các vector mã hóa thời gian (rel_t_enc) với thông điệp (msg) để tạo ra edge_attr.
- **Biến Đổi Đặc Trưng Với TransformerConv:** Sử dụng các lớp TransformerConv để tổng hợp và kết hợp thông tin từ các nút láng giềng dựa trên cơ chế chú ý.

Kiến trúc của mô hình Graph Attention Embedding được mô tả chi tiết trong hình 3.3.



HÌNH 3.3: Hình ảnh mô tả kiến trúc của mô hình GAE

Đầu vào:

- X: Vector đặc trưng của các nút.
- last_update: Thời gian cập nhật cuối cùng của các nút.
- edge_index: Danh sách các cạnh trong đồ thị.

- t: Thời gian hiện tại.
- msg: Thông điệp liên quan đến các cạnh.

Mã Hóa Thời Gian (time_enc): Mã hóa sự chênh lệch thời gian giữa các nút bằng cách sử dụng lớp mã hóa thời gian (time_enc).

Kết Hợp Thông Tin: Kết hợp các vector mã hóa thời gian (rel_t_enc) với thông điệp (msg) để tạo ra edge_attr.

Biến Đổi Đặc Trưng Với TransformerConv:

- TransformerConv1: Sử dụng lớp TransformerConv đầu tiên để biến đổi và trích xuất đặc trưng của nút.
- TransformerConv2: Sử dụng lớp TransformerConv thứ hai để tiếp tục trích xuất và kết hợp thông tin từ các nút láng giềng.

Graph Attention Embedding là một phương pháp hiệu quả để trích xuất và học các đặc trưng từ đồ thị, đặc biệt là trong việc phát hiện các cạnh bất thường. Bằng cách sử dụng cơ chế chú ý từ mô hình Transformer, GAE có thể học được trọng số của các nút láng giềng dựa trên tầm quan trọng của chúng, từ đó cải thiện hiệu quả của việc trích xuất đặc trưng. Điều này giúp tăng cường bảo mật cho hệ thống và cung cấp các thông tin quan trọng trong phân tích dữ liệu đồ thị.

3.3.3 Cập nhật trạng thái

3.3.3.1 Nhu cầu cập nhật trạng thái

Khi một cạnh mới e_t xuất hiện trong đồ thị tại thời điểm t, mô hình cần cập nhật trạng thái của các nút nguồn (v_{src}) và nút đích (v_{dst}) của cạnh đó vì vùng lân cận của các nút này đã thay đổi. Việc cập nhật này đảm bảo rằng thông tin về cạnh mới e_t được phản ánh trong các trạng thái của v_{src} và v_{dst}.

3.3.3.2 Mô hình GRU (Gated Recurrent Unit)

Để cập nhật trạng thái của các nút, chúng tôi sử dụng một mô hình GRU (Gated Recurrent Unit). GRU là một loại mạng nơ-ron tái diễn (Recurrent Neural Network - RNN) được thiết kế để xử lý các chuỗi dữ liệu và có khả năng ghi nhớ thông tin từ các bước thời gian trước đó. Biểu thức cập nhật trạng thái của các nút nguồn và đích được định nghĩa như sau:

$$s_t(v_{\text{src}}) = \text{GRU}(s_{t-}(v_{\text{src}}), e_t) \tag{3.1}$$

$$s_t(v_{\text{dst}}) = \text{GRU}(s_{t-}(v_{\text{dst}}), e_t) \tag{3.2}$$

Trong đó:

- $s_t(v_{\text{src}})$ và $s_t(v_{\text{dst}})$ là trạng thái cập nhật của các nút v_{src} và v_{dst} tại thời điểm t.
- $s_{t-}(v_{\text{src}})$ và $s_{t-}(v_{\text{dst}})$ là trạng thái của các nút v_{src} và v_{dst} trước thời điểm t.
- e_t là thông tin về cạnh mới tại thời điểm t.

3.3.3.3 Quá trình lan truyền thông tin

Thông tin về cạnh mới e_t được lan truyền tới các trạng thái cập nhật của các nút v_{src} và v_{dst}. Điều này có nghĩa là:

- **Thông tin về e_t**: Cạnh mới e_t ảnh hưởng trực tiếp đến các trạng thái của các nút nguồn và đích của nó, tức là thông tin về e_t được lưu giữ trong các trạng thái cập nhật $s_t(v_{\text{src}})$ và $s_t(v_{\text{dst}})$.
- **Các Cạnh Mới Xuất Hiện Sau t**: Các cạnh mới xuất hiện sau t có thể tích hợp thông tin từ e_t nếu e_t nằm trong vùng lân cận của chúng. Điều này giúp các embedding của các cạnh mới phản ánh thông tin lịch sử một cách đầy đủ và chính xác.

3.3.3.4 Không rò rỉ thông tin

Một điểm quan trọng là thông tin về cạnh mới e_t không được lan truyền tới embedding hiện tại z tại thời điểm t. Điều này nhằm đảm bảo rằng thông tin về e_t không bị rò rỉ khi sử dụng z để dự đoán e_t. Điều này giúp tránh việc mô hình học được thông tin về cạnh mới từ embedding hiện tại, từ đó tăng tính chính xác và trung thực của mô hình khi dự đoán các cạnh mới.

3.3.4 Bộ giải mã

Bộ giải mã trong hệ thống của chúng tôi kết hợp giữa mạng nơ-ron quan hệ (Relation Network) và multi-layer perceptron (MLP) để dự đoán loại của cạnh nối giữa nút nguồn và nút đích. Dưới đây là mô tả chi tiết về các thành phần này và cách chúng hoạt động cùng nhau để đạt được mục tiêu dự đoán.

3.3.4.1 Mạng Nơ-ron Quan hệ (Relation Network)

Mạng nơ-ron quan hệ là một kiến trúc học sâu được thiết kế để học các mối quan hệ giữa các đối tượng. Trong bài toán dự đoán cạnh, mạng nơ-ron quan hệ nhận các vector nhúng của nút nguồn và nút đích làm đầu vào và học cách biểu diễn mối quan hệ giữa chúng.

- **Lớp Nhập liệu**: Mạng nơ-ron quan hệ nhận các vector nhúng của nút nguồn (z_src) và nút đích (z_dst).
- **Kết hợp Vector**: Các vector nhúng này được kết hợp lại thành một vector duy nhất bằng cách nối (concatenate) chúng lại với nhau.
- **Lớp Ẩn**: Vector kết hợp sau đó được đưa qua một hoặc nhiều lớp ẩn (hidden layers) để học các đặc trưng mối quan hệ phức tạp hơn.
- **Lớp Đầu ra**: Đầu ra của mạng nơ-ron quan hệ là một vector nhúng mới biểu diễn mối quan hệ giữa nút nguồn và nút đích.

Mạng nơ-ron quan hệ giúp hệ thống học được các đặc trưng mối quan hệ tiềm ẩn giữa các nút, điều này rất quan trọng trong việc phát hiện các tấn công phức tạp và không rõ ràng.

3.3.4.2 Multi-Layer Perceptron (MLP)

Multi-layer perceptron là một loại mạng nơ-ron nhân tạo nhiều lớp, mỗi lớp bao gồm nhiều neuron. MLP được sử dụng rộng rãi trong các bài toán phân loại và hồi quy. Trong bài toán này, MLP được sử dụng để dự đoán loại của cạnh dựa trên đầu ra từ mạng nơ-ron quan hệ.

- **Lớp Nhập liệu**: Đầu vào của MLP là vector nhúng từ mạng nơ-ron quan hệ.
- **Lớp Ẩn**: Vector này được đưa qua nhiều lớp ẩn với các hàm kích hoạt (activation functions) như ReLU hoặc Tanh để học các đặc trưng phức tạp.
- **Lớp Đầu ra**: Lớp cuối cùng của MLP có kích thước bằng số loại cạnh có thể có (trong trường hợp này là chín), và sử dụng hàm Softmax để đưa ra xác suất của mỗi loại cạnh.

MLP cho phép hệ thống xử lý và phân loại các loại cạnh khác nhau một cách hiệu quả, nhờ vào khả năng học và trích xuất các đặc trưng phức tạp từ dữ liệu.

3.3.4.3 Kết hợp Mạng Nơ-ron Quan hệ và MLP

Trong mô hình của chúng tôi, đầu ra từ mạng nơ-ron quan hệ (biểu diễn mối quan hệ giữa nút nguồn và nút đích) được đưa vào MLP để dự đoán loại của cạnh. Quá trình hoạt động cụ thể như sau:

- **Lớp Nhập liệu**: Vector nhúng của nút nguồn và nút đích được đưa vào mạng nơ-ron quan hệ.
- **Biểu diễn Mối quan hệ**: Mạng nơ-ron quan hệ học và xuất ra một vector nhúng biểu diễn mối quan hệ giữa hai nút.
- **Dự đoán loại Cạnh**: Vector nhúng này sau đó được đưa vào MLP để dự đoán xác suất của từng loại cạnh.

Việc kết hợp này cho phép mô hình của chúng tôi tận dụng được cả khả năng học mối quan hệ giữa các nút từ mạng nơ-ron quan hệ và khả năng phân loại chính xác từ MLP, từ đó cải thiện độ chính xác và hiệu quả trong việc phát hiện các tấn công.

3.3.4.4 Quá trình Huấn luyện và Đánh giá

Trong quá trình huấn luyện, mô hình giảm thiểu sai số tái tạo (RE) giữa dự đoán của bộ giải mã và loại cạnh thực tế từ đồ thị không có độc hại. Công thức tính sai số tái tạo là:

$$\text{RE} = \text{CrossEntropy}(P, L) \tag{3.3}$$

Trong đó, P là xác suất dự đoán từ bộ giải mã, và L là vector one-hot biểu diễn loại cạnh thực tế, với xác suất của loại cạnh thực tế là 1 và phần còn lại là 0. Trong quá trình kiểm tra, mô hình gán giá trị RE thấp cho những cạnh mà cấu trúc và ngữ cảnh thời gian giống nhau với những điều được học từ đồ thị không có độc hại, nhưng giá trị RE cao nếu chúng chệch lệch đáng kể so với hành vi hệ thống bình thường đã biết.

Như vậy, bằng cách kết hợp mạng nơ-ron quan hệ và MLP, mô hình của chúng tôi có khả năng học được các mối quan hệ phức tạp giữa các nút và sử dụng thông tin này để dự đoán chính xác loại cạnh, góp phần vào việc phát hiện các tấn công phức tạp trong hệ thống.

3.4 Phát hiện bất thường

Mô hình xây dựng các hàng đợi cửa sổ thời gian để phát hiện các sự cố bất thường trong quá trình triển khai. Để làm điều này, mô hình xác định một tập hợp các nút đáng ngờ trong mỗi cửa sổ thời gian dựa trên lỗi tái tạo của các cạnh. Hai cửa sổ thời gian có các nút đáng ngờ trùng nhau được đặt vào hàng đợi cùng lúc. Khi một cửa sổ thời gian mới được thêm vào hàng đợi, mô hình cập nhật điểm bất thường của hàng đợi, dựa cũng vào lỗi tái tạo. Nếu điểm bất thường vượt quá một ngưỡng, mô hình coi hàng đợi đó là bất thường và kích hoạt một cảnh báo. Do đó, mô hình thực hiện phát hiện bất thường định kỳ tại các khoảng thời gian cố định.

Mô hình coi một nút trong cửa sổ thời gian là đáng ngờ nếu nút đó đáp ứng hai thuộc tính sau:

- **Tính bất thường:** nút nguồn hoặc nút đích của một cạnh có lỗi tái tạo (RE) lớn hơn một ngưỡng tái thiết.

- **Độ hiếm:** thực thể hệ thống tương ứng của nó không xuất hiện thường xuyên trong quá trình thực thi lành tính. Sử dụng tần suất nghịch đảo của văn bản (inverse document frequency - IDF) [5]. Cụ thể, đối với một nút v, chúng tôi tính:

$$\mathrm{IDF}(v) = \ln\left(\frac{N}{N_v + 1}\right) \tag{3.4}$$

  Trong đó N là tổng số khoảng thời gian và N_v là số khoảng thời gian chứa nút v. Một nút v nhận được giá trị IDF tối đa nếu nó không tồn tại trong quá khứ, tức là $N_v = 0$. Điều này có nghĩa rằng giá trị IDF của một nút càng cao thì nút đó càng hiếm. Mô hình xem một nút là hiếm nếu giá trị IDF của nó vượt qua ngưỡng hiếm α. Sau đó, mô hình xác định một tập hợp các nút đáng ngờ S_T thỏa mãn cả tính bất thường và tính hiếm cho mỗi khoảng thời gian T.

3.5 Điều tra bất thường

Để hỗ trợ quản trị viên hệ thống giải quyết cảnh báo, mô hình tự động tạo ra các đồ thị tóm tắt cuộc tấn công nhỏ gọn từ các hàng đợi thời gian bất thường. Điều này bao gồm việc xác định cộng đồng các cạnh có lỗi tái tạo cao để cải thiện khả năng đọc. Việc giảm đồ thị là cần thiết, vì khác với hình ảnh và văn bản, đồ thị khó nhìn và hiểu ngay cả đối với các chuyên gia. Mô hình làm điều này mà không phụ thuộc vào bất kỳ kiến thức tấn công trước nào; do đó, khả năng của nó trong việc tái tạo dấu vết tấn công chính xác hơn không bị giới hạn bởi các cuộc tấn công đã biết trước đây.

Khả năng tái dựng lại các câu chuyện tấn công một cách đầy đủ nhưng ngắn gọn là một tiêu chí thiết kế quan trọng trong mô hình. Điều này đặc biệt quan trọng đối với các hệ thống phát hiện bất thường, đặc biệt là những hệ thống sử dụng học sâu. Lý do là vì tái dựng cuộc tấn công giúp (1) thiết lập sự tin tưởng vào các quyết định, (2) tạo điều kiện cho yếu tố con người trong quá trình hiểu các bất

thường của hệ thống, và (3) đẩy nhanh quá trình xác định và giảm thiểu các kết quả dương tính giả (FP).

Mô hình áp dụng các kỹ thuật giảm kích thước đồ thị chứng minh tiêu biểu để giảm kích thước của đồ thị [21] mà không làm thay đổi ngữ nghĩa của nó. Ví dụ: gộp các cạnh từ các nút nguồn và đích giống nhau nếu chúng thuộc cùng loại. Việc giảm đồ thị không ảnh hưởng đến việc phát hiện sự bất thường, vì mô hình chỉ thực hiện giảm kích thước sau khi phát hiện một hàng đợi bất thường.

Sau khi giảm đồ thị, mô hình sử dụng thuật toán phát hiện cộng đồng Louvain [9] để xác định những cộng đồng. Các hoạt động tấn công thường tạo thành một cộng đồng dày đặc của các nút được kết nối qua các cạnh có lỗi tái tạo cao, tách chúng khỏi các nút khác không liên quan đến cuộc tấn công. Louvain bắt đầu với mỗi nút đại diện cho một cộng đồng riêng biệt. Đối với mỗi nút, nó di chuyển từ cộng đồng hiện tại của mình sang một trong các cộng đồng lân cận của nó mà dẫn đến cải thiện lớn nhất (nếu có) về các Modularity, đo lường mức độ mật độ kết nối trong cộng đồng so với kết nối giữa các cộng đồng. Louvain chạy quá trình này lặp đi lặp lại cho đến khi các Modularity không còn tăng nữa.

$$\text{Modularity} = \sum_{c}\left(\frac{\Sigma_{\text{in}}}{2m} - \left(\frac{\Sigma_{\text{tot}}}{2m}\right)^2\right) \tag{3.5}$$

Trong đó, Σ_{in} là tổng của RE của các cạnh trong cộng đồng c, và Σ_{tot} là tổng của RE của các cạnh kề với cộng đồng c (tức là các cạnh có một trong những đỉnh liên quan nằm trong cộng đồng và đỉnh còn lại nằm bên ngoài). m là tổng của RE của tất cả các cạnh trong hàng đợi. Đỉnh sẽ tiếp tục ở trong cộng đồng của nó nếu việc di chuyển nó không đạt được sự gia tăng về Modularity.

Các cộng đồng kết quả sau đó được đơn giản hóa để tạo ra các đồ thị tóm tắt ứng cử. Những đồ thị này mô tả một cách ngắn gọn hành vi độc hại kéo dài qua các khoảng thời gian dài và liên quan đến nhiều giai đoạn của chuỗi tấn công, mặc dù đôi khi chúng cũng có thể đại diện cho hoạt động hệ thống bất thường nhưng vẫn không độc hại (do tính chất của việc phát hiện sự bất thường).

Chương 4

HIỆN THỰC VÀ ĐÁNH GIÁ, THẢO LUẬN

Tóm tắt chương

Trong chương này, chúng tôi sẽ trình bày chi tiết cách hiện thực hóa phương pháp phát hiện tấn công APT đã trình bày trong Chương 3. Đồng thời, chúng tôi sẽ mô tả cách xây dựng môi trường thực nghiệm, các kịch bản thử nghiệm và các kết quả đạt được khi triển khai và kiểm thử hệ thống phát hiện tấn công APT dựa trên đồ thị nguồn gốc. Các bước thực nghiệm được thiết kế để đánh giá hiệu suất và tính hiệu quả của mô hình trong việc phát hiện các hoạt động xâm nhập và tái hiện lại dấu vết của các cuộc tấn công.

4.1 Các câu hỏi nghiên cứu

Trong quá trình hiện thực và đánh giá, chúng tôi tập trung trả lời các câu hỏi nghiên cứu sau:

- **Q1:** Mô hình có thể phát hiện chính xác các bất thường trong một hệ thống đang bị tấn công, đặc biệt là khi các cuộc tấn công diễn ra chậm và ngầm (ví dụ như APT trong 2.1) và do đó khó phát hiện hay không?
- **Q2:** Mô hình so sánh như thế nào với các hệ thống hiện đại?

- **Q3:** Mô hình có thể tái tạo chính xác hành vi tấn công từ đồ thị nguồn gốc ban đầu không?
- **Q4:** Hiệu suất tổng thể của mô hình là gì?

4.2 Giả định phạm vi

Chúng tôi xem xét các kẻ tấn công cố gắng kiểm soát hệ thống và duy trì sự hiện diện dai dẳng bằng cách khai thác các lỗ hổng phần mềm hay triển khai các backdoor. Tuy nhiên, chúng tôi không xem xét các cuộc tấn công ở cấp độ phần cứng, kênh phụ hoặc kênh bí mật, vì hành vi của chúng thường không được hệ thống kiểm tra cấp nhân cụ thể ghi lại. Hệ thống của chúng tôi là một hệ thống phát hiện dựa trên sự bất thường; do đó chúng tôi giả định thêm rằng các hệ thống máy chủ không bị ảnh hưởng của kẻ tấn công khi mô hình học từ các đồ thị nguồn gốc của hệ thống thực thi lành mạnh và quan sát kỹ lưỡng hoạt động hệ thống trong giai đoạn học ban đầu này.

Cơ sở tính toán đáng tin cậy (TCB) bao gồm hệ điều hành cơ bản, khung kiểm tra và mã phân tích, điều này cũng là tiêu chuẩn trong các PIDS hiện có. Do đó, chúng tôi không xem xét các cuộc tấn công cấp nhân và giả định việc sử dụng các kỹ thuật tăng cường hệ thống hiện có để giảm thiểu bất kỳ sự xâm phạm khung kiểm tra tiềm ẩn nào. Cuối cùng, chúng tôi giả định tính toàn vẹn của dữ liệu đầu ra (tức là các đồ thị nguồn gốc) từ khung kiểm tra. Các hệ thống nguồn gốc an toàn hiện có và các kỹ thuật ghi nhật ký không thể giả mạo, có thể đảm bảo tính toàn vẹn của nhật ký và phát hiện bất kỳ sự can thiệp độc hại nào với các nhật ký nguồn gốc.

4.3 Hiện thực mô hình

4.3.1 Các thư viện

Các thư viện liên quan bao gồm:

- **tqdm**: Thư viện để hiển thị thanh tiến trình trong terminal.

- **scikit-learn==1.2.0**: Thư viện học máy và khai phá dữ liệu.
- **networkx==2.8.7**: Thư viện để làm việc với đồ thị.
- **xxhash==3.2.0**: Thư viện xxHash cho việc băm dữ liệu.
- **graphviz==0.20.1**: Thư viện để vẽ đồ thị.

GPU sử dụng PyTorch:

- **pytorch==1.13.1**: Framework học sâu dựa trên tensor với hỗ trợ GPU.
- **torchvision==0.14.1**: Thư viện hỗ trợ thị giác máy tính cho PyTorch.
- **torchaudio==0.13.1**: Thư viện hỗ trợ âm thanh cho PyTorch.
- **torch_geometric==2.0.0**: Thư viện học đồ thị cho PyTorch.
- **pyg_lib**, **torch_scatter**, **torch_sparse**, **torch_cluster**, **torch_spline_conv**: Các phụ thuộc của thư viện PyTorch Geometric.

4.3.2 Tập dữ liệu DARPA

Chúng tôi sử dụng bộ dữ liệu DARPA trong khuôn khổ chương trình Transparent Computing (TC) [6]. Chương trình Transparent Computing của DARPA nhằm phát triển các công nghệ và một hệ thống nguyên mẫu thí nghiệm để cung cấp cả khả năng phát hiện thời gian thực và pháp chứng của APT cũng như thực thi các chính sách mong muốn một cách chủ động. TC đã tổ chức một số hoạt động đối kháng giả lập các APT thực tế trên các mạng doanh nghiệp. Trong các hoạt động này, một nhóm đỏ (red team) đã tiến hành một loạt các cuộc tấn công vào các dịch vụ quan trọng về an ninh của doanh nghiệp (ví dụ: máy chủ web, email và SSH) đồng thời thực hiện các hoạt động hợp pháp như duyệt web, kiểm tra email và đăng nhập SSH. Một nhóm riêng biệt đã triển khai các hệ thống thu thập nguồn gốc khác nhau (ví dụ: CADETS, ClearScope, THEIA,...) trên các nền tảng khác nhau để ghi lại hoạt động của toàn bộ hệ thống máy chủ. Dữ liệu nguồn gốc từ buổi tập thứ ba (E3) và thứ năm (E5) là công khai.

Bộ dữ liệu	Nút	Cạnh (triệu)	Cạnh tấn công	Tỉ lệ cạnh tấn công (%)
CADETS_E3	178,965	10.1	1,248	0.012
THEIA_E3	690,105	32.4	3,119	0.01

BẢNG 4.1: Bảng tóm tắt bộ dữ liệu trong thí nghiệm

Chúng tôi sử dụng các bộ dữ liệu DARPA để chỉ ra rằng mô hình có thể phát hiện chính xác các bất thường mặc dù chúng ẩn mình trong một lượng lớn hoạt động hợp pháp trong khoảng thời gian dài; trích xuất chính xác đồ thị nguồn gốc ban đầu (mô tả cả hoạt động hợp pháp và tấn công) thành một đồ thị tóm tắt tấn công gọn gàng mà không cần kiến thức về cuộc tấn công trước đó, mặc dù hoạt động tấn công hiếm hơn rất nhiều lần. Hơn nữa, chúng tôi sử dụng bộ dữ liệu để so sánh với PIDS khác như Flash.

Trong thí nghiệm của mình, chúng tôi thực hiện với 2 bộ dữ liệu là CADETS_E3 và THEIA_E3 (sau đây gọi là CADETS và THEIA). Bảng 4.1 mô tả bộ dữ liệu được sử dụng trong thí nghiệm.

4.3.3 Xử lý dữ liệu

4.3.3.1 Tạo cơ sở dữ liệu

Bước đầu tiên, chúng tôi tạo cơ sở dữ liệu để lưu trữ. Bộ dữ liệu được lưu vào 3 bảng là Netflow, Object và File.

- **Xử lý dữ liệu Netflow:** xử lý dữ liệu NetFlow nhằm trích xuất thông tin về các kết nối mạng và lưu trữ vào cơ sở dữ liệu. Quá trình bao gồm việc xử lý thông tin về địa chỉ IP, cổng, và tạo bảng dữ liệu chứa các thông tin này.
- **Xử lý dữ liệu Object:** xử lý dữ liệu về đối tượng từ tập tin log, bao gồm các sự kiện liên quan và lưu trữ vào cơ sở dữ liệu. Đối tượng này có thể là bất kỳ thực thể nào được xác định trong hệ thống.
- **Xử lý dữ liệu File:** xử lý và lưu trữ dữ liệu về các đối tượng file từ tập tin log. Thông tin về đường dẫn được trích xuất và lưu vào cơ sở dữ liệu.

BẢNG 4.2: Bảng tóm tắt bộ dữ liệu dựa trên loại nút

Bộ dữ liệu	Netflow nodes	Subject nodes	File nodes	Events
CADETS_E3	155,322	224,146	234,245	29,727,441
THEIA_E3	186,100	279,369	793,899	44,400,000

Sau đó, chúng tôi tiến hành tạo danh sách đối tượng từ các bảng dữ liệu Netflow, Object và File. Danh sách này sau đó được lưu trữ trong bảng node2id trong cơ sở dữ liệu để thực hiện việc ánh xạ giữa các đối tượng và các chỉ số.

Xử lý dữ liệu sự kiện: xử lý và lưu trữ dữ liệu về sự kiện từ tập tin log. Thông tin về mối quan hệ giữa các đối tượng được trích xuất và lưu trữ vào cơ sở dữ liệu.

BẢNG 4.3: Bảng tóm tắt dữ liệu theo ngày của CADETS_E3

Ngày	Số lượng sự kiện	Độ dài danh sách cạnh
2	600,488	445,378
3	2,453,951	1,758,192
4	2,395,521	1,677,700
5	2,746,323	1,936,142
6	2,892,308	2,022,481
7	3,265,428	2,284,034

BẢNG 4.4: Bảng tóm tắt dữ liệu theo ngày của THEIA_E3

Ngày	Số lượng sự kiện	Độ dài danh sách cạnh
3	9,799,359	8,230,837
4	6,672,049	4,930,304
5	2,127,824	1,489,011
9	712,059	685,635
10	6,389,285	6,274,151
11	7,511,085	7,285,220
12	7,293,214	7,024,937
13	3,895,125	3,759,064

4.3.3.2 Embedding

Tạo biểu diễn vector cho đối tượng:

- Đầu tiên, chúng tôi thu thập tất cả các nhãn đối tượng từ đồ thị cơ sở dữ liệu, bao gồm netflow, file và subject.
- Đối với mỗi nhãn đối tượng, chúng tôi xây dựng biểu diễn thang cấp cao bằng cách sử dụng các hàm chuyển đổi đường dẫn và địa chỉ IP.
- Sử dụng FeatureHasher từ thư viện scikit-learn, chúng tôi ánh xạ mỗi biểu diễn thang cấp cao sang một biểu diễn số học với số chiều được xác định bởi **node_embedding_dim**.
- Các biểu diễn số học này sau đó được lưu vào tệp **node2higvec** để sử dụng trong các bước sau.

Tạo Vector One Hot cho mối quan hệ:

- Chúng tôi tạo vector một hot cho mỗi loại mối quan hệ có thể xuất hiện trong đồ thị.
- Vector một hot được tạo bằng **torch.nn.functional.one_hot**, và chúng tôi lưu trữ chúng trong tệp **rel2vec**.

Tạo đồ thị vector hóa cho các sự kiện:

- Đối với mỗi khoảng thời gian trong đồ thị cơ sở dữ liệu (ví dụ: mỗi ngày), chúng tôi trích xuất tất cả các sự kiện.
- Chúng tôi xây dựng danh sách các cạnh (edges) từ các sự kiện, chỉ giữ lại những cạnh có loại mối quan hệ được xác định trước (được xác định bởi **include_edge_type**).
- Dùng các biểu diễn số học của đối tượng và vector một hot của mối quan hệ, chúng tôi xây dựng đồ thị vector hóa thời gian **(TemporalData)** cho mỗi khoảng thời gian.
- Đồ thị vector hóa sau đó được lưu trữ vào tệp **graph_4_ [day].TemporalData.simple**.

4.3.4 Xây dựng mô hình

4.3.4.1 Mô hình GraphSAGE

Khởi tạo vector đặc trưng: Mỗi nút trong đồ thị được khởi tạo ban đầu với một vector đặc trưng. Đối với các nút mới, vector đặc trưng này thường được khởi tạo với giá trị mặc định như vector không.

Mã hóa thời gian: Mã hóa sự chênh lệch thời gian giữa các nút láng giềng và kết hợp thông tin thời gian này với thông điệp truyền qua các cạnh để tạo ra các vector đặc trưng cho các cạnh.

Biến đổi đặc trưng với GraphSAGE:

- Sử dụng các lớp Linear để biến đổi đặc trưng ban đầu của các nút.
- Sử dụng các lớp SAGEConv để tổng hợp và kết hợp thông tin từ các nút láng giềng, từ đó cải thiện vector đặc trưng của nút.

4.3.4.2 Mô hình Graph Attention Embedding (GAE)

Khởi tạo vector đặc trưng: Mỗi nút trong đồ thị được khởi tạo ban đầu với một vector đặc trưng. Đối với các nút mới, vector đặc trưng này thường được khởi tạo với giá trị mặc định, ví dụ như vector không.

Mã hóa thời gian: GAE sử dụng mã hóa thời gian để tích hợp thông tin thời gian vào quá trình học tập. Mã hóa sự chênh lệch thời gian giữa các nút láng giềng và kết hợp thông tin thời gian này với thông điệp truyền qua các cạnh để tạo ra các vector đặc trưng cho các cạnh.

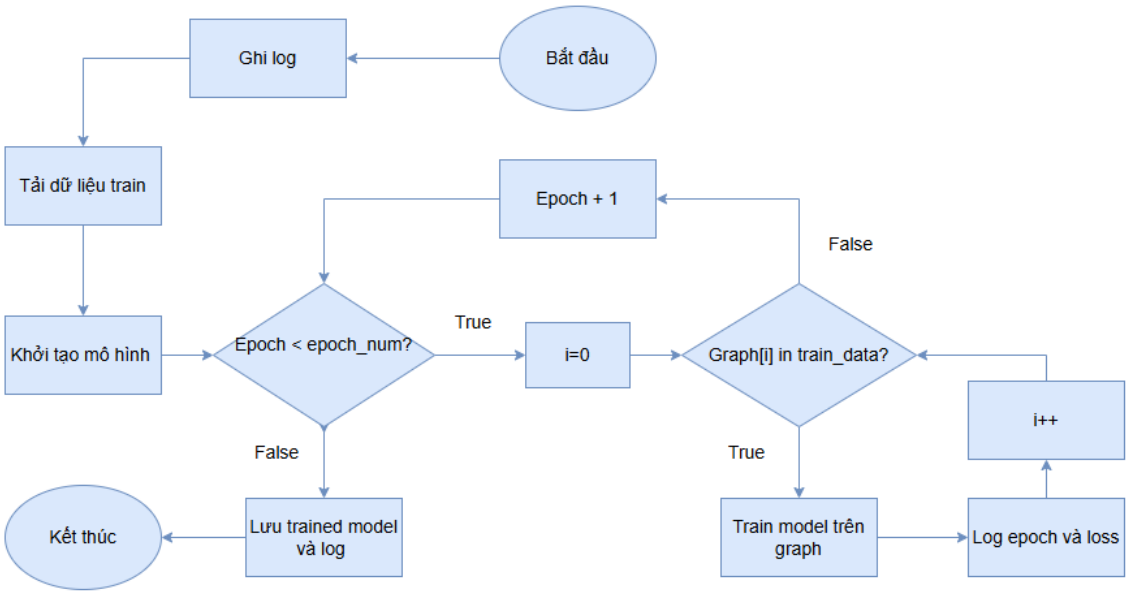
Biến đổi đặc trưng với GraphSAGE:

- GAE sử dụng các lớp TransformerConv để biến đổi và trích xuất đặc trưng của nút. Các lớp này áp dụng cơ chế chú ý để học các trọng số quan trọng của các nút láng giềng dựa trên mối quan hệ và thông tin trong đồ thị.
- Trước khi đưa vào 'TransformerConv', các đặc trưng được kết hợp với thông tin thời gian đã được mã hóa. Điều này giúp mô hình GAE hiểu được sự thay đổi thời gian và các mối quan hệ trong đồ thị.

Mô hình dự đoán liên kết: Mô hình dự đoán liên kết trong hệ thống của chúng tôi kết hợp giữa mạng nơ-ron quan hệ (Relation Network) và mạng nơ-ron lan truyền ngược nhiều lớp (Multi-Layer Perceptron - MLP). Mô hình này được sử dụng để dự đoán loại của các cạnh trong đồ thị dựa trên các đặc trưng mối quan hệ giữa các nút.

Trong quá trình dự đoán, mô hình nhận các vector đặc trưng từ mạng nơ-ron quan hệ, biến đổi và học các đặc trưng phức tạp hơn thông qua MLP. Kết quả cuối cùng là một vector biểu diễn xác suất của từng loại cạnh có thể có.

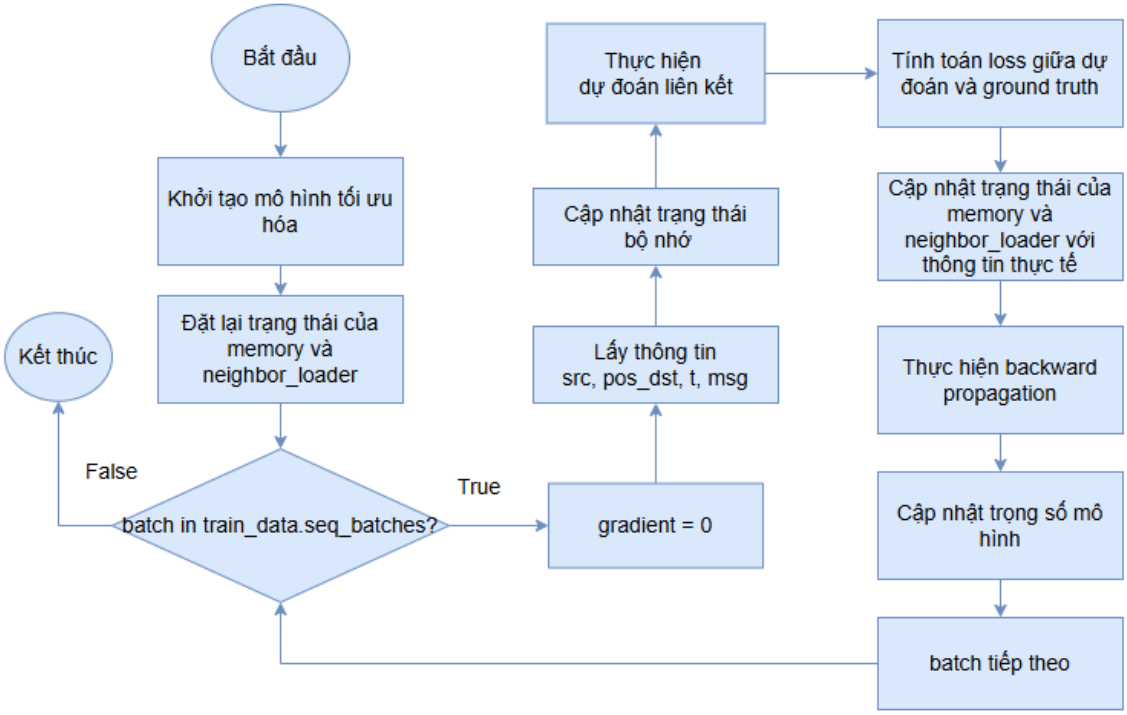
4.3.4.3 Train



HÌNH 4.1: Luồng hoạt động của chế độ huấn luyện

Quá trình đào tạo mô hình được mô tả ở hình 4.1 và 4.2, gồm các bước như sau:

Đầu tiên, khởi tạo mô hình và bộ tối ưu hóa. Mô hình được khởi tạo bằng cách sử dụng ba thành phần chính: **memory**, **gnn** (Graph Neural Network), và **link_pred** (MLP + Realtion Network). Một tối ưu hóa Adam được sử dụng với các tham số như tốc độ học (lr), epsilon (eps), và giảm trọng số (weight_decay).

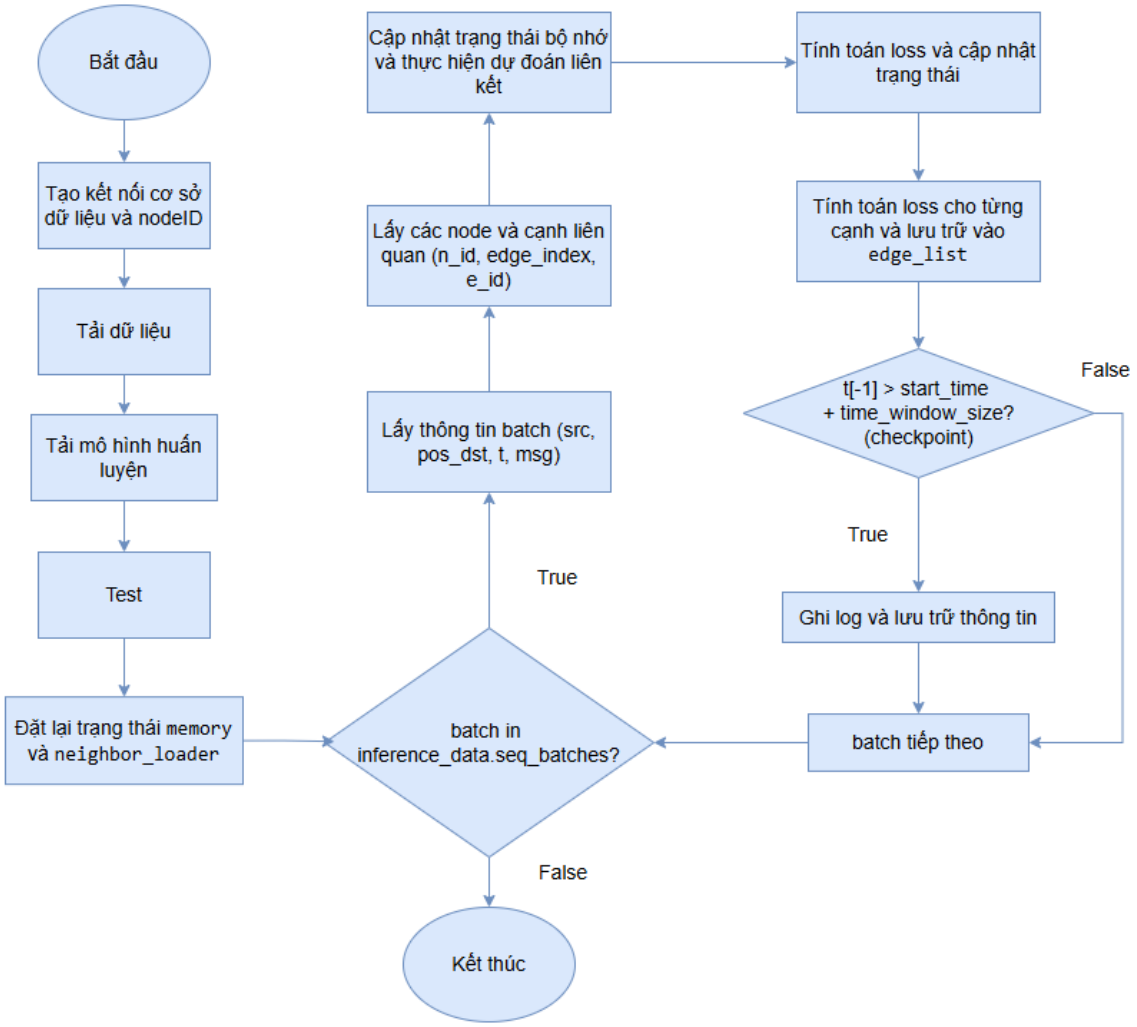


HÌNH 4.2: Chi tiết hàm train

Sau đó, đặt lại trạng thái của memory và neighbor_loader ở đầu mỗi epoch.

Cuối cùng, thực hiện quá trình duyệt qua các batch của dữ liệu đào tạo:

- Đặt gradient bằng 0.
- Lấy thông tin về nút nguồn, nút đích, thời điểm và thông điệp.
- Xác định các nút và cạnh liên quan thông qua neighbor_loader.
- Cập nhật trạng thái của bộ nhớ và thực hiện dự đoán liên kết.
- Tính toán loss giữa dự đoán và ground truth.
- Cập nhật trạng thái của memory và neighbor_loader với thông tin thực tế.
- Thực hiện backward propagation và cập nhật trọng số mô hình.
- Tổng hợp loss để theo dõi sự cụ thể của quá trình huấn luyện.



HÌNH 4.3: Luồng hoạt động của chế độ kiểm thử

4.3.4.4 Test

Quá trình kiểm tra mô hình được mô tả ở hình 4.3, gồm các bước:

Đầu tiên, nhận dữ liệu thử nghiệm **(inference_data)**, các thành phần của mô hình gồm: **(memory, gnn, link_pred, neighbor_loader)**; bản đồ từ nodeID sang node labels **(nodeid2msg)**, và đường dẫn để lưu trữ kết quả **(path)**.

Tiếp theo, xác định **time_with_loss** là một từ điển để lưu trữ giá trị mất mát và thời gian cho mỗi khoảng thời gian.

Sau đó, chia dữ liệu thử nghiệm thành các batch, thực hiện tiến trình kiểm thử với mỗi batch.

Tiến trình kiểm thử

- Chuyển sang chế độ đánh giá (eval) để tắt dropout và các hoạt động có thể gây lệch khi kiểm thử.
- Lấy thông tin về các nút nguồn (src), nút đích (pos_dst), thời gian (t), và thông điệp (msg).
- Cập nhật tập hợp các nút duy nhất (unique_nodes) và tổng số cạnh (total_edges).
- Xác định các nút và cạnh liên quan (n_id, edge_index, e_id) thông qua neighbor_loader.
- Lấy các biểu diễn của các nút từ bộ nhớ và thực hiện dự đoán liên kết (pos_out) thông qua mô hình.
- Ghi nhận kết quả và loss cho mỗi cạnh trong batch.
- Cập nhật trạng thái của bộ nhớ và loader với thông tin thực tế.
- Kiểm soát thời gian và lưu trữ loss cho mỗi khoảng thời gian.

BẢNG 4.5: Bảng tóm tắt dữ liệu được gán nhãn tấn công của CADETS_E3

Danh sách tấn công	Số lượng sự kiện
2018-04-06 11:18:26 2018-04-06 11:33:35	22,528
2018-04-06 11:33:35 2018-04-06 11:48:42	15,360
2018-04-06 11:48:42 2018-04-06 12:03:50	18,432
2018-04-06 12:03:50 2018-04-06 14:01:32	8,192

BẢNG 4.6: Bảng tóm tắt dữ liệu được gán nhãn tấn công của THEIA_E3

Danh sách tấn công	Số lượng sự kiện
2018-04-10 13:31:14 2018-04-10 13:46:36	228,352
2018-04-10 14:02:17 2018-04-10 14:17:34	29,696
2018-04-10 14:17:34 2018-04-10 14:33:18	2,529,280
2018-04-10 14:33:18 2018-04-10 14:48:47	84,992
2018-04-10 14:48:47 2018-04-10 15:03:54	134,144
2018-04-12 12:39:06 2018-04-12 12:54:44	65,536
2018-04-12 12:54:44 2018-04-12 13:09:55	88,064
2018-04-12 13:09:55 2018-04-12 13:25:06	98,304
2018-04-12 13:25:06 2018-04-12 13:40:07	135,168

4.3.4.5 Xây dựng hàng đợi bất thường

Sau khi thực hiện quá trình test, mã nguồn tiếp tục xây dựng hàng đợi có vấn đề dựa trên thông tin mất mát và thời gian.

- **cal_ anomaly_loss(loss_list, edge_list):** Hàm này tính toán số lượng, tổng loss trung bình, và tập hợp các nút và cạnh bất thường dựa trên một ngưỡng được xác định từ loss trung bình và độ lệch chuẩn.
- **compute_IDF():** Hàm này tính toán trọng số IDF cho các nút trong đồ thị, dựa trên thông tin mất mát từ các file log.
- **al_set_rel(s1, s2, node_IDF, tw_list):** Hàm này đo lường mức độ liên kết giữa hai tập hợp nút, dựa trên trọng số IDF và thông tin mất mát.
- **anomalous_queue_construction (node_IDF, tw_list, graph_dir_path):**Hàm chính xây dựng hàng chờ có vấn đề dựa trên thông tin mất mát và thời gian cho các khoảng thời gian trong đồ thị.

4.3.4.6 Điều tra bất thường

Mặc dù hàng đợi cửa sổ thời gian bất thường làm giảm đáng kể kích thước của đồ thị, nhưng chúng vẫn có thể chứa hàng nghìn nút và cạnh. Để giảm bớt gánh nặng cho các nhà phân tích, mô hình tự động hóa quy trình điều tra cuộc tấn công bằng cách xây dựng các đồ thị con biểu diễn tóm tắt bất thường từ hàng đợi cửa sổ thời gian bất thường. Điều này cũng có nghĩa là mô hình không dựa vào bất kỳ kiến thức tấn công nào trước đó. Hình 4.4 là một đồ thị con biểu diễn tóm tắt bất thường được mô hình xây dựng lại. Những đồ thị này mô tả ngắn gọn hành vi độc hại kéo dài trong thời gian dài và liên quan đến nhiều giai đoạn của một chuỗi tấn công, mặc dù đôi khi chúng cũng có thể đại diện cho hoạt động hệ thống bất thường nhưng không gây hại (do tính chất của việc phát hiện dựa trên bất thường).

Để thực hiện công việc này, trước tiên chúng tôi áp dụng kỹ thuật giảm kích thước đồ thị nguồn gốc [21] mà không thay đổi ngữ nghĩa của nó bằng cách gộp các cạnh từ các nút nguồn và đích giống nhau nếu chúng cùng loại. Việc giảm

thiểu đồ thị không ảnh hưởng đến việc phát hiện bất thường, vì mô hình chỉ thực hiện giảm thiểu sau khi nó phát hiện ra một hàng đợi bất thường.

Sau đó, chúng tôi sử dụng thuật toán Louvain [9] để phân cộng đồng trên đồ thị toàn cục. Trong đó, tạo đồ thị con cho mỗi cộng đồng và lưu vào từ điển **communities**. Tiếp tục duyệt qua từng cộng đồng để vẽ và lưu trữ đồ thị dưới dạng hình ảnh. Cuối cùng, sử dụng GraphViz [7] để vẽ và lưu trữ đồ thị cộng đồng nhỏ. Đồ thị sẽ được lưu dưới dạng file PDF.

HÌNH 4.4: Đồ thị con biểu diễn tóm tắt bất thường

4.4 Tiến hành thực nghiệm

Hệ thống của chúng tôi được thực nghiệm trên hệ điều hành Ubuntu 22.04 với cấu hình RAM là 46GB và dung lượng là 422GB. Chi tiết cấu hình được mô tả tại bảng 4.7. Ngoài ra, ngôn ngữ chính được sử dụng để xây dựng hệ thống là Python 3.9.

BẢNG 4.7: Bảng thông tin môi trường thực nghiệm của hệ thống

Thông tin	**Chi tiết**
Hệ điều hành	22.04.4 LTS (Jammy)
CPU	Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz
RAM	46 GB
GPU	NVIDIA GeForce GTX 1050 Ti
Dung lượng ổ cứng	422 GB

4.4.1 Thiết lập các tham số cho mô hình

Bảng 4.8 thống kê các tham số được sử dụng trong mô hình của chúng tôi.

4.4.2 Triển khai thực nghiệm

4.4.2.1 Chia dữ liệu train, test

Chúng tôi chia dữ liệu cho tập train và test theo ngày, chi tiết được mô tả trong bảng 4.9 và 4.10.

4.4.2.2 Các kịch bản triển khai

Để trả lời các câu hỏi nghiên cứu được đề ra, chúng tôi đã liên kết các kịch bản thử nghiệm với từng câu hỏi nghiên cứu cụ thể như sau:

- **Kịch bản 1: Dự đoán trên tập dữ liệu CADETS**
  - **Liên kết với câu hỏi Q1 4.1:** Kịch bản này được thiết kế để đánh giá khả năng của mô hình trong việc phát hiện các bất thường trong một hệ thống đang bị tấn công. Dữ liệu CADETS có thể chứa các cuộc tấn công chậm và ngầm, do đó việc dự đoán trên tập dữ liệu này giúp chúng tôi xác định liệu mô hình có thể phát hiện chính xác các cuộc tấn công APT hay không.
  - **Liên kết với câu hỏi Q3 4.1:** Bằng cách sử dụng mô hình để dự đoán trên dữ liệu kiểm thử, chúng tôi có thể kiểm tra xem mô hình có thể tái tạo chính xác hành vi tấn công từ đồ thị nguồn gốc ban đầu hay không.

BẢNG 4.8: Bảng các tham số được sử dụng trong mô hình

THAM SỐ	GIÁ TRỊ	MÔ TẢ
node_embedding_dim	16	Kích thước nhúng nút
node_state_dim	100	Kích thước của trạng thái nút
neighbor_size	20	Số lượng nút lân cận được lấy mẫu
edge_dim	100	Kích thước nhúng cạnh
time_dim	100	Kích thước của mã hóa thời gian
BATCH	1024	Kích thước batch cho quá trình huấn luyện và kiểm thử
lr	0.00005	Tốc độ học của tối ưu hóa Adam
eps	1e-08	Epsilon cho tối ưu hóa Adam
weight_decay	0.01	Giảm trọng số cho tối ưu hóa Adam
epoch_num	10	Epoch là một chu kỳ hoàn chỉnh của việc đưa toàn bộ dữ liệu huấn luyện qua mô hình một lần
time_window_size	60000000000×15	Kích thước cửa sổ thời gian (15 phút)
beta_day6	100	Ngưỡng cho ngày 6
beta_day7	100	Ngưỡng cho ngày 7

BẢNG 4.9: Bảng chia dữ liệu train, test của bộ dữ liệu CADETS_E3

Loại dữ liệu	Ngày	Tổng số sự kiện
Train	2-3-4	3,881,270
Test	6-7	4,306,515

BẢNG 4.10: Bảng chia dữ liệu train, test của bộ dữ liệu THEIA_E3

Loại dữ liệu	Ngày	Tổng số sự kiện
Train	3-4-5	14,650,152
Test	10-11-12	20,584,308

Điều này được đánh giá thông qua các chỉ số như Precision, Recall và AUC.

- **Kịch bản 2: Dự đoán trên tập dữ liệu THEIA**
  - **Liên kết với câu hỏi Q2 4.1:** Kịch bản này cho phép chúng tôi so sánh mô hình với các hệ thống hiện đại bằng cách sử dụng tập dữ liệu lớn hơn và phức tạp hơn. Kết quả dự đoán trên tập dữ liệu THEIA sẽ được so sánh với các kết quả từ các hệ thống khác để đánh giá hiệu suất của mô hình.
  - **Liên kết với câu hỏi Q4 4.1:** Kịch bản này cũng giúp chúng tôi đo lường hiệu suất tổng thể của mô hình. Các chỉ số đánh giá như Accuracy, Precision, Recall và AUC sẽ được sử dụng để đưa ra cái nhìn tổng quan về hiệu suất của mô hình.

4.4.2.3 Phân tích

Kết quả thử nghiệm cho từng kịch bản được tổng hợp và phân tích để trả lời các câu hỏi nghiên cứu:

- **Q1: Mô hình có thể phát hiện chính xác các bất thường trong một hệ thống đang bị tấn công không?**
  - **Kịch bản 1** cho thấy mô hình có khả năng phát hiện chính xác các cuộc tấn công APT thông qua các chỉ số cao về Precision và Recall. Điều này chứng tỏ mô hình có thể nhận diện các bất thường, ngay cả khi chúng diễn ra chậm và ngầm.
- **Q2: Mô hình so sánh như thế nào với các hệ thống hiện đại?**
  - **Kịch bản 2** cho thấy mô hình đạt được kết quả cao trên tập dữ liệu THEIA, so sánh tích cực với các hệ thống hiện đại khác. Đặc biệt, các chỉ số như AUC và Accuracy cho thấy hiệu suất vượt trội của mô hình.
- **Q3: Mô hình có thể tái tạo chính xác hành vi tấn công từ đồ thị nguồn gốc ban đầu không?**

  - Các kết quả từ **Kịch bản 1** cho thấy mô hình có khả năng tái tạo chính xác hành vi tấn công từ đồ thị nguồn gốc ban đầu, minh chứng qua các chỉ số Precision và Recall cao.

- **Q4: Hiệu suất tổng thể của mô hình là gì?**

  - Kết quả từ **Kịch bản 2** cho thấy hiệu suất tổng thể của mô hình là rất tốt, với các chỉ số đánh giá cao trên tập dữ liệu lớn. Điều này chứng tỏ mô hình có khả năng tổng quát hóa tốt và hoạt động hiệu quả trong nhiều tình huống thực tế khác nhau.

4.4.3 Kết quả thực nghiệm

4.4.3.1 Thông số

Kết quả thực nghiệm của chúng tôi được trình bày chi tiết trong bảng 4.11. Mô hình cho kết quả dựa trên các thông số:

- TP (True Positives): Số lượng các dự đoán dương tích cực đúng, tức là số lượng các trường hợp dự đoán là positive và thực tế cũng là positive.
- TN (True Negatives): Số lượng các dự đoán âm tích cực đúng, tức là số lượng các trường hợp dự đoán là negative và thực tế cũng là negative.
- FP (False Positives): Số lượng các dự đoán dương tích cực sai, tức là số lượng các trường hợp dự đoán là positive nhưng thực tế là negative.
- FN (False Negatives): Số lượng các dự đoán âm tích cực sai, tức là số lượng các trường hợp dự đoán là negative nhưng thực tế là positive.
- Precision (Positive Predictive Value): Tỉ lệ giữa số lượng True Positives và tổng số dự đoán positive (TP / (TP + FP)). Precision đo lường khả năng của mô hình không đưa ra dự đoán positive khi nó không chắc chắn.
- Recall (Sensitivity, True Positive Rate): Tỉ lệ giữa số lượng True Positives và tổng số thực tế positive (TP / (TP + FN)). Recall đo lường khả năng của mô hình phát hiện toàn bộ số trường hợp positive.

BẢNG 4.11: Bảng kết quả mô hình

Bộ dữ liệu	Mô hình	TP	TN	FP	FN	Pre.	Rec.	Acc.	AUC
CADETS	GAE	4	171	4	0	0.5	1.0	0.97	0.98
	GraphSAGE	4	174	1	0	0.8	1.0	0.99	0.99
THEIA	GAE	7	215	3	2	0.7	0.77	0.97	0.88
	GraphSAGE	9	216	2	0	0.81	1.0	0.99	0.99

BẢNG 4.12: Bảng thống kê biểu đồ tóm tắt tấn công của bộ dữ liệu CADETS_E3

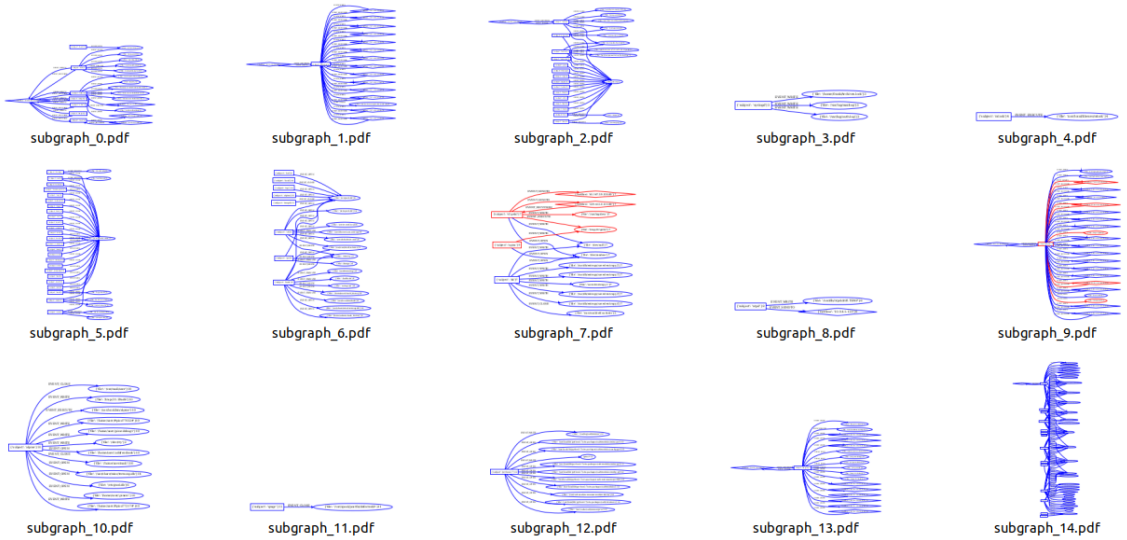
Nút	Cạnh	Cạnh trong cửa sổ thời gian	Tỉ lệ giảm (X lần)
18	26	115,712	4,450X

- Accuracy: Tỉ lệ giữa tổng số dự đoán đúng (TP + TN) và tổng số mẫu (TP + TN + FP + FN). Accuracy đo lường khả năng tổng thể của mô hình đưa ra dự đoán đúng.
- AUC (Area Under the ROC Curve): Diện tích dưới đường cong ROC (Receiver Operating Characteristic). AUC đo lường khả năng phân biệt giữa các lớp positive và negative; giá trị 1.0 đại diện cho một mô hình hoàn hảo, trong khi giá trị 0.5 chỉ ra sự phân loại ngẫu nhiên.

4.4.3.2 Biểu đồ tái tạo tóm tắt tấn công

Trong các bộ dữ liệu DARPA, mô hình có thể tái dựng lại hoạt động tấn công thực sự mô tả APT, trong khi chỉ báo cáo một vài đồ thị ứng viên vô hại. Bảng 4.12 cho thấy kích thước của đồ thị tóm tắt cuộc tấn công mà mô hình tạo ra từ hàng đợi thời gian bất thường trong mỗi bộ dữ liệu DARPA. Chúng tôi thấy rằng các đồ thị tóm tắt ứng viên là nhỏ (nhờ vào sự giảm thiểu đồ thị trong 3.5). Ví dụ, đối với bộ dữ liệu CADETS_E3, mô hình đạt được giảm thiểu cạnh 4,450 lần, thu hẹp tổng số cạnh cần kiểm tra thủ công từ 115,712 triệu trong các hàng đợi thời gian bất thường xuống chỉ còn 26. Điều này có nghĩa là các quản trị viên hệ thống có thể nhanh chóng và dễ dàng suy luận về các đồ thị tóm tắt ứng viên, loại bỏ những đồ thị vô hại và xác định hoạt động tấn công thực sự.

Hình 4.5 mô tả các biểu đồ tóm tắt tấn công được mô hình xây dựng lại.



HÌNH 4.5: Hình ảnh mô tả các biểu đồ tóm tắt tấn công

4.5 Đánh giá

Dựa trên bảng kết quả mô hình được trình bày trong Bảng 4.11, chúng tôi đưa ra một số nhận xét chi tiết về hiệu suất của hai mô hình Graph Attention Embedding (GAE) và GraphSAGE trên hai bộ dữ liệu khác nhau: CADETS và THEIA.

4.5.1 Đánh giá trên bộ dữ liệu CADETS

Đối với bộ dữ liệu CADETS, mô hình GAE và GraphSAGE đều đạt được kết quả ấn tượng với độ chính xác cao. Cụ thể, mô hình GAE đạt được độ chính xác (Accuracy) là 0.97 và AUC là 0.98, trong khi mô hình GraphSAGE có độ chính xác cao hơn với 0.99 và AUC là 0.99.

Mô hình GAE:

- Số lượng TP (True Positives) là 4, cho thấy khả năng phát hiện chính xác các trường hợp dương tính.
- TN (True Negatives) là 171, FP (False Positives) là 4 và không có FN (False Negatives).

- Precision của mô hình GAE là 0.5, cho thấy rằng một nửa các dự đoán dương tính là chính xác, nhưng Recall đạt giá trị tuyệt đối 1.0, tức là mô hình không bỏ sót bất kỳ trường hợp dương tính nào.

Mô hình GraphSAGE:

- Số lượng TP cũng là 4, cho thấy tương đương với mô hình GAE về khả năng phát hiện các trường hợp dương tính.
- TN là 174, FP chỉ là 1, và cũng không có FN.
- Precision của mô hình GraphSAGE là 0.8, cao hơn so với mô hình GAE, và Recall vẫn duy trì ở mức 1.0, cho thấy khả năng phát hiện chính xác các trường hợp dương tính mà không bỏ sót.

Nhìn chung, trên bộ dữ liệu CADETS, mô hình GraphSAGE tỏ ra vượt trội hơn mô hình GAE, đặc biệt ở các chỉ số Precision và FP, làm tăng độ tin cậy của các dự đoán dương tính.

4.5.2 Đánh giá trên bộ dữ liệu THEIA

Trên bộ dữ liệu THEIA, cả hai mô hình đều tiếp tục thể hiện khả năng dự đoán tốt, nhưng mô hình GraphSAGE vẫn cho thấy sự vượt trội hơn so với mô hình GAE.

Mô hình GAE:

- TP là 7, TN là 215, FP là 3 và FN là 2.
- Precision là 0.7 và Recall là 0.77, cho thấy rằng mô hình GAE vẫn phát hiện được nhiều trường hợp dương tính, nhưng có một số trường hợp bị bỏ sót (FN=2).
- Độ chính xác (Accuracy) là 0.97 và AUC là 0.88, cho thấy hiệu suất tổng thể tốt nhưng chưa đạt mức xuất sắc.

Mô hình GraphSAGE:

- TP là 9, TN là 216, FP là 2 và không có FN.

BẢNG 4.13: Bảng so sánh với mô hình Kairos. Pre. = Precision; Rec. = Recall; Acc. = Accuracy

Bộ dữ liệu	Mô hình	TP	TN	FP	FN	Prec.	Rec.	Acc.
CADETS	GAE	4	171	4	0	**0.5**	**1.0**	0.97
	GraphSAGE	4	174	1	0	**0.8**	**1.0**	**0.99**
	Kairos	3	171	4	1	0.42	0.75	0.97

- Precision là 0.81 và Recall là 1.0, cho thấy mô hình này không bỏ sót bất kỳ trường hợp dương tính nào và các dự đoán dương tính hầu hết đều chính xác.
- Độ chính xác (Accuracy) là 0.99 và AUC là 0.99, cho thấy hiệu suất tổng thể xuất sắc trên bộ dữ liệu THEIA.

4.5.3 Thảo luận

Dựa trên kết quả trên cả hai bộ dữ liệu, chúng ta có thể kết luận rằng mô hình GraphSAGE thể hiện hiệu suất cao hơn so với mô hình GAE. Điều này được thể hiện qua các chỉ số Precision, Recall, Accuracy và AUC. Đặc biệt, trên cả hai bộ dữ liệu CADETS và THEIA, mô hình GraphSAGE không bỏ sót bất kỳ trường hợp dương tính nào (Recall=1.0) và duy trì mức độ chính xác rất cao (Accuracy gần như 0.99 trên cả hai bộ dữ liệu). Bảng 4.13 cho thấy kết quả mô hình của chúng tôi khi kết hợp GAE và GraphSAGE cho kết quả tốt hơn so với mô hình Kairos với bộ dữ liệu CADETS trên cùng môi trường thực nghiệm.

Mô hình GAE mặc dù có hiệu suất khá tốt, nhưng vẫn có một số điểm yếu, đặc biệt là ở chỉ số Precision và số lượng False Positives (FP) cao hơn so với GraphSAGE. Điều này cho thấy rằng trong các ứng dụng yêu cầu độ chính xác cao và độ tin cậy trong dự đoán, mô hình GraphSAGE có thể là lựa chọn ưu tiên.

Ngoài ra việc xuất hiện một số FPs trong mô hình là do một số nguyên nhân:

- **Mô hình tiếp tục gán lỗi tái tạo cao:** Mô hình tiếp tục gán lỗi tái tạo cao cho các cạnh mà nút của chúng đã bị ảnh hưởng bởi kẻ tấn công, ngay cả sau khi kẻ tấn công ngừng hoạt động. Điều này có thể dẫn đến việc xem xét các thực thể là đã bị chiếm đóng khi chúng thực tế không còn tham gia vào cuộc tấn công.

- **Quyết định về FPs trong ground truth:** Trong ground truth, các thực thể giữa sau khi cuộc tấn công thường bị bỏ qua, vì chúng không còn là phần của cuộc tấn công nữa. Sự khác biệt trong quyết định về việc xác định thực thể nào là một phần của cuộc tấn công và thực thể nào không có thể dẫn đến sự khác biệt giữa các FP "giả mạo" và kết quả thực nghiệm thực tế.
- **Thực thể không tham gia vào cuộc tấn công sau khi bị chiếm đóng:** Trong một số trường hợp, các thực thể mà mô hình vẫn xem xét là bị chiếm đóng có thể không còn tham gia vào hoạt động tấn công. Điều này làm tăng số lượng FPs.

4.5.4 So sánh với một số mô hình khác

Việc so sánh công bằng giữa mô hình của chúng tôi và các hệ thống phát hiện xâm nhập dựa trên đồ thị nguồn gốc khác hiện nay là rất khó khăn vì một số lý do. *Thứ nhất*, phần lớn các hệ thống PIDS dựa trên chữ ký, trong khi mô hình của chúng tôi phát hiện các bất thường. Hiệu suất của các hệ thống PIDS dựa trên chữ ký phụ thuộc vào chất lượng của các chữ ký, mà thường là kiến thức độc quyền không được công khai. Việc so sánh giữa các hệ thống PIDS dựa trên chữ ký và các hệ thống PIDS dựa trên phát hiện bất thường có thể dễ dàng bị thiên lệch do việc thao túng các chữ ký để có thể khớp với cuộc tấn công. Do đó, chúng tôi loại trừ các hệ thống PIDS dựa trên chữ ký khỏi việc so sánh. *Thứ hai*, hầu hết các hệ thống PIDS dựa trên phát hiện bất thường là mã nguồn đóng và được đánh giá bằng cách sử dụng các bộ dữ liệu riêng tư. Vì vậy rất khó để xác minh tính chính xác khi không có quyền truy cập vào các bộ dữ liệu để tái tạo các kết quả ban đầu. *Thứ ba*, các hệ thống PIDS khác nhau có thể sử dụng các chỉ số khác nhau để báo cáo hiệu suất phát hiện của họ, càng làm phức tạp việc so sánh và tạo ra so sánh sai lệch về hiệu suất.

Do những khó khăn này, chúng tôi chọn Flash [19] làm hệ thống PIDS chính để so sánh, vì chúng là hệ thống phát hiện bất thường mã nguồn mở và được các tác giả đánh giá bằng một phần của bộ dữ liệu DARPA, trong đó có CADETS_E3 và THEIA_E3. Bảng 4.14 so sánh mô hình của chúng tôi khi sử dụng GAE và GraphSAGE với Flash. Lưu ý rằng Flash phát hiện bất thường dựa trên thông số của các nút.

BẢNG 4.14: Bảng so sánh với mô hình Flash. Pre. = Precision; Rec. = Recall; Acc. = Accuracy

Bộ dữ liệu	Mô hình	TP	TN	FP	FN	Prec.	Rec.	Acc.
CADETS	GAE	4	171	4	0	0.5	**1.0**	0.97
	GraphSAGE	4	174	1	0	0.8	**1.0**	0.99
	Flash	12,851	706,148	818	1	0.94	0.99	0.99
THEIA	GAE	7	215	3	2	0.7	0.77	0.97
	GraphSAGE	9	216	2	0	0.81	**1.0**	0.99
	Flash	25,318	3,503,044	2,282	44	0.92	0.99	0.99

Có thể thấy mô hình của chúng tôi luôn đảm bảo Recall ở mức 1.0 (trừ trường hợp mô hình GAE đối với bộ dữ liệu THEIA), điều này quan trọng trong việc phát hiện tấn công do không bỏ sót tấn công. Bên cạnh đó, các mô hình đều có tỉ lệ phát hiện (Accuracy) cao. Tuy nhiên, Flash sử dụng các thông số TP, TN, FP, FN dựa trên số nút, do đó bảng so sánh là tương đối và không đánh giá được mô hình nào tốt hơn dựa trên thông số Accuracy. Vì vậy, chúng tôi mong muốn rằng trong tương lai mô hình của chúng tôi cũng như các mô hình khác có thể khắc phục những điểm này để có được so sánh công bằng hơn.

Chương 5

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Tóm tắt chương

Trong chương cuối này, chúng tôi sẽ tổng hợp những công việc đã thực hiện, đưa ra kết luận về nghiên cứu, nêu quan điểm về ưu và nhược điểm của các phương pháp đã sử dụng, đồng thời đề xuất cải thiện và hướng phát triển trong tương lai.

5.1 Kết luận

Trong khóa luận này, chúng tôi đã nghiên cứu, đề xuất và đánh giá một giải pháp phát hiện xâm nhập dựa trên đồ thị nguồn gốc để phát hiện các cuộc tấn công APT. Hệ thống sử dụng kiến trúc mã hóa-giải mã dựa trên mạng nơ-ron đồ thị, học sự phát triển theo thời gian của các thay đổi cấu trúc của đồ thị nguồn gốc để định lượng mức độ bất thường của từng sự kiện hệ thống. Sau đó, dựa trên thông tin chi tiết này, tái cấu trúc dấu vết tấn công, tạo ra các đồ thị tóm tắt gọn nhẹ mô tả chính xác hoạt động độc hại mà không cần kiến thức tấn công trước đó. Một số kết quả đạt được từ việc xây dựng mô hình của chúng tôi như sau:

- Tìm hiểu sự phát triển theo thời gian của những thay đổi cấu trúc trong đồ thị nguồn gốc.
- Xây dựng mô hình với bộ dữ liệu CADETS_E3 và THEIA_E3 của DARPA.

- Xây dựng lại dấu vết tấn công, tạo ra các biểu đồ tóm tắt nhỏ gọn mô tả chính xác hoạt động bất thường.
- Thực hiện đánh giá mô hình.

Qua việc xây dựng hệ thống này, chúng tôi đã hiểu hơn về các hướng nghiên cứu liên quan, hiểu được các hạn chế để góp phần cải thiện dần. Sau toàn bộ quá trình, chúng tôi nhận thấy một số ưu và nhược điểm của mô hình như sau:

Ưu điểm

- Giải quyết được bài toán đặt ra, đó là 4 khía cạnh phổ biến của nghiên cứu PIDS, bao gồm: Phạm vi (Liệu PIDS có thể phát hiện các cuộc tấn công hiện đại mà xâm nhập qua các ranh giới ứng dụng không?); Tính bất khả tri tấn công (Liệu PIDS có thể phát hiện các cuộc tấn công mới mà không cần kiến thức trước về các đặc điểm tấn công không?); Tính kịp thời (Liệu PIDS có thể giám sát hiệu quả các hệ thống chủ khi chúng hoạt động không?) và Khả năng tái cấu trúc tấn công (Liệu PIDS có thể chắt lọc hoạt động tấn công từ các đồ thị nguồn gốc lớn để các quản trị viên hệ thống có thể dễ dàng hiểu và nhanh chóng phản ứng với sự xâm nhập hệ thống không?)
- Đạt hiệu quả đáng kể trong quá trình thực nghiệm, mô hình có khả năng giám sát hệ thống hiệu quả, hoạt động lâu dài trong thời gian chạy, hoạt động tốt các hệ thống hiện đại và chịu chi phí với hiệu suất tối thiểu.

Nhược điểm

- Mô hình học đòi hỏi một lượng lớn dữ liệu huấn luyện và tài nguyên tính toán cao hơn so với các mô hình học máy truyền thống. Do đó, việc tìm và xử lý bộ dữ liệu là một công việc cũng tốn rất nhiều thời gian và tài nguyên.
- Bên cạnh đó, dữ liệu về các cuộc tấn công APT chiếm rất ít so với dữ liệu lành tính trong các hoạt động bình thường. Chúng tôi cũng đã nghiên cứu việc xử lý học với ít dữ liệu bằng Few-shot Learning, tuy nhiên các công trình chỉ dừng lại ở mức xử lý với dữ liệu ảnh. Điều này cũng mở ra hướng phát triển cho tương lai về công việc áp dụng Few-shot Learning đối với dữ liệu đồ thị nguồn gốc.

5.2 Hướng phát triển

Chúng tôi xin đề xuất một số hướng nghiên cứu và phát triển liên quan đến khóa luận như sau:

- **Xem xét các công nghệ, kỹ thuật sẽ xuất hiện trong tương lai**: Trong tương lai có thể có các mô hình, thuật toán mới cho hiệu suất và kết quả tốt hơn, chúng tôi sẽ cố gắng áp dụng để cải thiện kết quả tính toán, nâng cao hiệu suất, phát triển cải thiện mô hình.
- **Thử nghiệm với bộ dữ liệu khác nhau**: Chúng tôi mong muốn mô hình của mình có thể được thử nghiệm với bộ dữ liệu lớn hơn hoặc các bộ dữ liệu khác.
- **Đánh giá và tăng cường tính bền vững của mô hình**: Thực hiện các cuộc tấn công cũng như xây dựng các cơ chế phòng thủ cho mô hình.
- **Xử lý học với ít dữ liệu bằng Few-shot Learning:** Như đã trình bày, hầu hết các công trình nghiên cứu Few-shot Learning hiện nay đang mức xử lý với dữ liệu ảnh. Chúng tôi cho rằng công việc áp dụng Few-shot Learning đối với dữ liệu đồ thị nguồn gốc là một hướng đi đầy hứa hẹn. Theo đó, chúng tôi đề xuất mô hình với 7Way-nShot, với 7Way tương ứng với với số lượng nhãn ở lớp cuối cùng, tương đương 7 loại cạnh, và nShot tương ứng với số lượng mẫu cho mỗi loại cạnh này.

Tài liệu tham khảo

[1] A. Alsaheel et al. "ATLAS: A sequence-based learning approach for attack investigation". In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 3005–3022.

[2] T. Bilot et al. "Graph neural networks for intrusion detection: A survey". In: *IEEE Access* (2023).

[3] Ping Chen, Lieven Desmet, and Christophe Huygens. "A Study on Advanced Persistent Threats". In: *15th IFIP International Conference on Communications and Multimedia Security (CMS)*. Aveiro, Portugal, 2014, pp. 63–72. DOI: 10.1007/978-3-662-44885-4_5. URL: https://inria.hal.science/hal-01404186/document.

[4] Zijun Cheng et al. "KAIROS: Practical Intrusion Detection and Investigation using Whole-system Provenance". In: *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2024.

[5] Kenneth Church and William Gale. "Inverse document frequency (idf): A measure of deviations from poisson". In: *Natural language processing using very large corpora*. Springer, 1999, pp. 283–295.

[6] DARPA Transparent Computing. *Transparent Computing Engagement 3 Data Release*. 2020. URL: https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md.

[7] John Ellson et al. "Graphviz and dynagraph—static and dynamic graph drawing tools". In: *Graph drawing software* (2004), pp. 127–148.

[8] FireEye Labs. *FireEye Advanced Threat Report 2013*. 2014. URL: https://www.ismsforum.es/ficheros/descargas/fireeye-advanced-threat-report-20131395657540.pdf.

[9] L Guillaume. "Fast unfolding of communities in large networks". In: *Journal Statistical Mechanics: Theory and Experiment* 10 (2008), P1008.

[10] X. Han et al. "SIGL: Securing Software Installations Through Deep Graph Learning". In: *30th USENIX Security Symposium (USENIX Security 21)*. 2021, pp. 2345–2362.

[11] X. Han et al. "Unicorn: Runtime provenance-based detector for advanced persistent threats". In: *arXiv preprint arXiv:2001.01525* (2020).

[12] W. U. Hassan, A. Bates, and D. Marino. "Tactical Provenance Analysis for Endpoint Detection and Response Systems". In: *Symposium on Security and Privacy (S&P'20)*. IEEE. 2020.

[13] M. Kapoor et al. "PROV-GEM: Automated Provenance Analysis Framework using Graph Embeddings". In: *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Dec. 2021, pp. 1720–1727.

[14] Kaspersky. *APT Q1 2023 playbook: advanced techniques, broader horizons, and new targets*. 2023. URL: `https://www.kaspersky.com/about/press-releases/2023_apt-q1-2023-playbook-advanced-techniques-broader-horizons-and-new-targets`.

[15] M. Li et al. "The study of APT attack stage model". In: *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*. IEEE, June 2016, pp. 1–5.

[16] Mandiant. *APT1: Exposing One of China's Cyber Espionage Units*. 2013. URL: `https://www.mandiant.com/resources/apt1-exposing-one-of-chinas-cyber-espionage-units`.

[17] M. S. Milajerdi et al. "Holmes: real-time apt detection through correlation of suspicious information flows". In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2019, pp. 1137–1152.

[18] S. M. Milajerdi et al. "POIROT: Aligning Attack Behavior with Kernel Audit Records for Cyber Threat Hunting". In: *Conference on Computer and Communications Security (CCS'19)*. ACM. 2019.

[19] Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. "FLASH: A Comprehensive Approach to Intrusion Detection via Provenance Graph Representation Learning". In: *IEEE Symposium on Security and Privacy (S&P)*. 2024.

[20] S. Wang et al. "Threatrace: Detecting and tracing host-based threats in node level through provenance graph learning". In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 3972–3987.

[21] Zhang Xu et al. "High fidelity data reduction for big data security dependency analyses". In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 504–516.

[22] F. Yang et al. "PROGRAPHER: An Anomaly Detection System based on Provenance Graph Embedding". In: *32nd USENIX Security Symposium (USENIX Security 23)*. 2023, pp. 4355–4372.

[23] J. Zengy et al. "Shadewatcher: Recommendation-guided cyber threat analysis using system audit records". In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2022, pp. 489–506.

[24] Jie Zhou et al. "Graph neural networks: A review of methods and applications". In: *AI open* 1 (2020), pp. 57–81.