MAGIC: PHÁT HIỆN CÁC MỐI ĐE DỌA DAI DẲNG NÂNG CAO THÔNG QUA HỌC BIỂU DIỄN ĐỒ THỊ BỊ CHE MẶT

MỤC LỤC

0. Tóm Tắt

Các mối đe dọa dai dẳng nâng cao (APTs), được các tác nhân tấn công tinh vi nhất sử dụng, đang trở nên ngày càng phổ biến và gây ra mối đe dọa lớn cho nhiều doanh nghiệp và tổ chức khác nhau. Phân tích dòng dõi dữ liệu trên các đồ thị nguồn gốc đã nổi lên như một phương pháp phổ biến trong phát hiện APT.

Tuy nhiên, các công trình trước đây đã bộc lộ một số nhược điểm: (1) yêu cầu dữ liệu chứa tấn công và kiến thức tiên nghiệm về APT, (2) không trích xuất được thông tin ngữ cảnh phong phú ẩn chứa trong các đồ thị nguồn gốc và (3) trở nên bất khả thi do chi phí tính toán và mức tiêu thụ bộ nhớ quá cao.

Trong bài báo này, nhóm nghiên cứu giới thiệu **MAGIC**, một phương pháp phát hiện APT tự giám sát mới lạ và linh hoạt, có khả năng thực hiện phát hiện đa mức độ chi tiết dưới các mức độ giám sát khác nhau. **MAGIC** tận dụng học biểu diễn đồ thị bị che mặt để mô hình hóa các thực thể và hành vi hệ thống lành tính, thực hiện trích xuất đặc trưng sâu hiệu quả và trừu tượng hóa cấu trúc trên các đồ thị nguồn gốc. Bằng cách phát hiện các hành vi hệ thống bất thường thông qua các phương pháp phát hiện ngoại lệ, **MAGIC** có thể thực hiện phát hiện APT ở cả cấp độ thực thể hệ thống và cấp độ nhật ký theo lô. **MAGIC** được thiết kế đặc biệt để xử lý sự trôi dạt khái niệm bằng cơ chế thích ứng mô hình và áp dụng thành công cho các điều kiện và kịch bản phát hiện phổ quát.

Nhóm nghiên cứu đánh giá **MAGIC** trên ba bộ dữ liệu được sử dụng rộng rãi, bao gồm cả các cuộc tấn công thực tế và mô phỏng. Kết quả đánh giá chỉ ra rằng **MAGIC** đạt được kết quả phát hiện đầy hứa hẹn trong tất cả các kịch bản và cho thấy ưu thế vượt trội so với các phương pháp phát hiện APT hiện đại về chi phí hiệu suất.

1. Giới thiệu

Các mối đe dọa dai dẳng nâng cao (APTs) là các cuộc tấn công mạng có chủ đích và tinh vi do các tác nhân tấn công lành nghề thực hiện và gây ra mối đe dọa lớn cho cả doanh nghiệp và tổ chức. Hầu hết các APT khai thác các lỗ hổng zero-day và đặc biệt khó phát hiện do tính chất lén lút và hay thay đổi của chúng. Các công trình gần đây về phát hiện APT tận dụng dòng dõi dữ liệu để thực hiện phát hiện APT. Dòng dõi dữ liệu chuyển đổi nhật ký kiểm toán thành các đồ thị nguồn gốc, trích xuất thông tin ngữ cảnh phong phú từ nhật ký kiểm toán và cung cấp một nền tảng hoàn hảo cho phân tích quan hệ nhân quả chi tiết và phát hiện APT. Các công trình ban đầu xây dựng các quy tắc dựa trên các mẫu APT điển hình hoặc cụ thể và đối chiếu nhật ký kiểm toán với các quy tắc đó để phát hiện các APT tiềm ẩn. Một số công trình gần đây áp dụng phương pháp phát hiện dị thường thống kê để phát hiện APT tập trung vào các yếu tố khác nhau của đồ thị nguồn gốc, ví dụ: các thực thể hệ thống, tương tác và cộng đồng. Tuy nhiên, hầu hết các công trình gần đây đều dựa trên các phương pháp học sâu (DL). Chúng sử dụng nhiều kỹ thuật học sâu khác nhau để mô hình hóa các mẫu APT hoặc hành vi hệ thống và thực hiện phát hiện APT theo kiểu phân loại hoặc phát hiện dị thường.

Trong khi các phương pháp hiện có này đã chứng minh khả năng phát hiện APT với độ chính xác hợp lý, chúng gặp phải nhiều tổ hợp các thách thức sau:

(1) Các phương pháp giám sát gặp phải vấn đề thiếu dữ liệu (LOD) vì chúng yêu cầu kiến thức tiên nghiệm về APT (tức là các mẫu tấn công hoặc nhật ký chứa tấn công). Ngoài ra, các phương pháp này đặc biệt dễ bị tổn thương khi đối mặt với các loại APT mới mà chúng chưa được huấn luyện để xử lý.

(2) Các phương pháp dựa trên thống kê chỉ yêu cầu dữ liệu lành tính để hoạt động, nhưng chúng không thể trích xuất được ngữ nghĩa sâu và mối tương quan của các hoạt động lành tính phức tạp ẩn chứa trong nhật ký kiểm toán, dẫn đến tỷ lệ dương tính giả cao.

(3) Các phương pháp dựa trên DL, đặc biệt là các phương pháp dựa trên chuỗi và dựa trên đồ thị, đã đạt được hiệu quả đầy hứa hẹn với chi phí tính toán nặng nề, khiến chúng không thực tế trong các kịch bản phát hiện thực tế.

Trong bài báo này, Nhóm tác giả giải quyết ba vấn đề trên bằng cách giới thiệu **MAGIC**, một phương pháp phát hiện APT tự giám sát mới lạ, tận dụng học biểu diễn đồ thị bị che mặt và các phương pháp phát hiện ngoại lệ đơn giản để xác định các thực thể hệ thống tấn công then chốt từ khối lượng lớn nhật ký kiểm toán. **MAGIC** đầu tiên xây dựng đồ thị nguồn gốc từ nhật ký kiểm toán theo các bước đơn giản nhưng phổ quát. Sau đó, **MAGIC** sử dụng một mô-đun biểu diễn đồ thị để thu được các nhúng bằng cách kết hợp các đặc trưng đồ thị và thông tin cấu trúc theo cách tự giám sát. Mô hình được xây dựng dựa trên các bộ mã hóa tự động đồ thị bị che mặt dưới sự giám sát chung của cả tái cấu trúc đặc trưng bị che mặt và tái cấu trúc cấu trúc dựa trên mẫu. Một phương pháp phát hiện ngoại lệ không giám sát được sử dụng để phân tích các nhúng đã tính toán và đạt được kết quả phát hiện cuối cùng. **MAGIC** được thiết kế để linh hoạt và có khả năng mở rộng. Tùy thuộc vào bối cảnh ứng dụng, **MAGIC** có khả năng thực hiện phát hiện đa mức độ chi tiết, tức là phát hiện sự tồn tại của APT trong các lô nhật ký hoặc định vị các đối tượng xấu ở cấp độ thực thể. Mặc dù **MAGIC** được thiết kế để thực hiện phát hiện APT mà không cần dữ liệu chứa tấn công, nhưng nó rất phù hợp cho các điều kiện bán giám sát và giám sát đầy đủ. Hơn nữa, **MAGIC** còn chứa một cơ chế thích ứng mô hình tùy chọn, cung cấp một kênh phản hồi cho người dùng của nó. Phản hồi này rất quan trọng để **MAGIC** tiếp tục cải thiện hiệu suất, chống lại sự trôi dạt khái niệm và giảm thiểu dương tính giả.

Nhóm tác giả triển khai **MAGIC** và đánh giá hiệu suất và chi phí của nó trên ba bộ dữ liệu tấn công APT khác nhau: bộ dữ liệu DARPA Transparent Computing E33, bộ dữ liệu StreamSpot4 và bộ dữ liệu Unicorn Wget5. Các bộ dữ liệu DARPA chứa các cuộc tấn công thực tế, trong khi bộ dữ liệu StreamSpot và Unicorn Wget được mô phỏng hoàn toàn trong môi trường có kiểm soát. Kết quả đánh giá cho thấy **MAGIC** có khả năng thực hiện phát hiện APT ở cấp độ thực thể với độ chính xác 97,26% và độ phủ 99,91% cũng như chi phí tối thiểu, ít tốn bộ nhớ hơn và nhanh hơn đáng kể so với các phương pháp phát hiện APT hiện đại (ví dụ, nhanh hơn ShadeWatcher6 51 lần).

Để mang lại lợi ích cho nghiên cứu trong tương lai và khuyến khích cải thiện hơn nữa **MAGIC**, Nhóm tác giả công khai mã nguồn của **MAGIC** và các bộ dữ liệu đã được tiền xử lý của Nhóm tác giả. Tóm lại, bài báo này có những đóng góp sau:

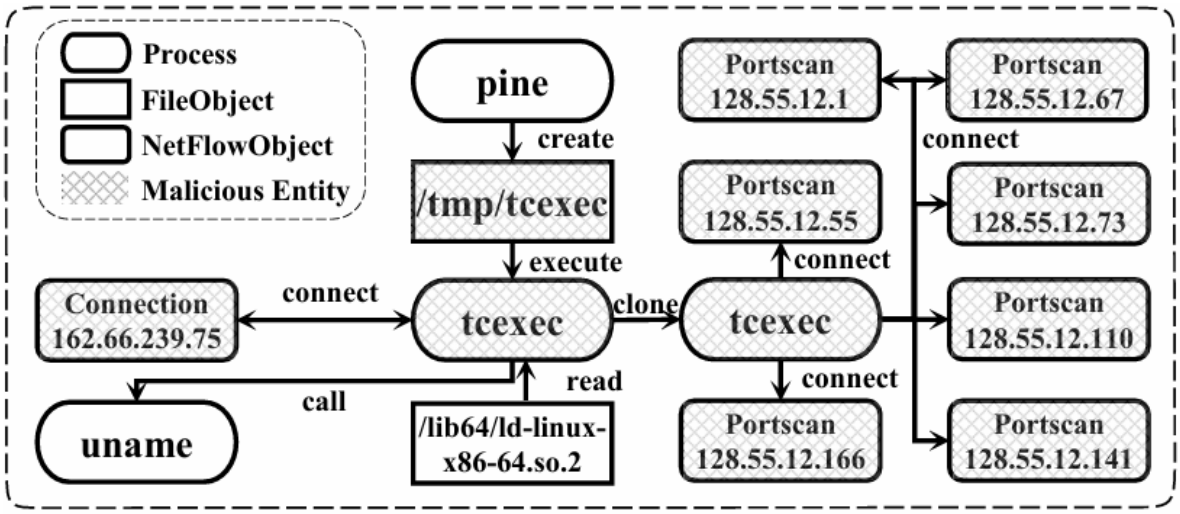
- Nhóm tác giả đề xuất **MAGIC**, một phương pháp phát hiện APT phổ quát dựa trên học biểu diễn đồ thị bị che mặt và các phương pháp phát hiện ngoại lệ, có khả năng thực hiện phát hiện đa mức độ chi tiết trên khối lượng lớn nhật ký kiểm toán.

- Nhóm tác giả đảm bảo tính thực tế của **MAGIC** bằng cách giảm thiểu chi phí tính toán với các bộ mã hóa tự động đồ thị bị che mặt mở rộng, cho phép **MAGIC** hoàn thành quá trình huấn luyện và phát hiện trong thời gian chấp nhận được ngay cả trong điều kiện hạn chế.

- Nhóm tác giả đảm bảo tính phổ quát của **MAGIC** bằng nhiều nỗ lực khác nhau. Nhóm tác giả tận dụng học biểu diễn đồ thị bị che mặt và các phương pháp phát hiện ngoại lệ, cho phép **MAGIC** thực hiện phát hiện chính xác dưới các mức độ giám sát khác nhau, ở các mức độ chi tiết phát hiện khác nhau và với nhật ký kiểm toán từ nhiều nguồn khác nhau.

- Nhóm tác giả đánh giá **MAGIC** trên ba bộ dữ liệu được sử dụng rộng rãi, bao gồm cả các cuộc tấn công APT thực tế và mô phỏng. Kết quả đánh giá cho thấy **MAGIC** phát hiện APT với kết quả đầy hứa hẹn và chi phí tính toán tối thiểu.

- Nhóm tác giả cung cấp một triển khai mã nguồn mở của **MAGIC** để mang lại lợi ích cho nghiên cứu trong tương lai của cộng đồng và khuyến khích cải thiện hơn nữa phương pháp của Nhóm tác giả.



Hình 1. Đồ thị nguồn gốc của một cuộc tấn công APT thực tế, khai thác lỗ hổng Pine Backdoor. Tất cả các thực thể và tương tác không liên quan đến tấn công đã được loại bỏ khỏi đồ thị nguồn gốc.

2. Bối cảnh

2.1. Ví dụ minh họa

Ở đây Nhóm tác giả cung cấp một minh họa chi tiết về một kịch bản APT mà Nhóm tác giả sử dụng trong toàn bộ bài báo. Pine backdoor với Drakon Dropper là một cuộc tấn công APT từ bộ dữ liệu DARPA Engagement 3 Trace. Trong cuộc tấn công, một kẻ tấn công

tạo một tệp thực thi độc hại (/tmp/tcexec) và gửi nó đến máy chủ mục tiêu qua một email lừa đảo. Sau đó, người dùng vô tình tải xuống và mở email. Bên trong email là một tệp thực thi được thiết kế để thực hiện quét cổng cho trinh sát nội bộ và thiết lập một kết nối im lặng giữa máy chủ mục tiêu và kẻ tấn công. **Hình 1** hiển thị đồ thị nguồn gốc của ví dụ minh họa của Nhóm tác giả. Các nút trong đồ thị đại diện cho các thực thể hệ thống và các mũi tên đại diện cho các tương tác có hướng giữa các thực thể. Đồ thị được hiển thị là một đồ thị con được trừu tượng hóa từ đồ thị nguồn gốc hoàn chỉnh bằng cách loại bỏ hầu hết các thực thể và tương tác không liên quan đến tấn công. Hình dạng nút khác nhau tương ứng với các loại thực thể khác nhau. Các thực thể được phủ sọc được coi là các thực thể độc hại.

2.2. Nghiên cứu trước đây và hạn chế

2.2.1. Các phương pháp giám sát

Đối với các công trình ban đầu [2–6], cần xây dựng các quy tắc heuristic đặc biệt để bao phủ tất cả các mẫu tấn công. Nhiều phương pháp phát hiện APT dựa trên DL [11, 14–16, 18] xây dựng các đồ thị nguồn gốc dựa trên cả dữ liệu lành tính và dữ liệu chứa tấn công, đồng thời phát hiện APT theo kiểu phân loại. Các phương pháp giám sát này có thể đạt được kết quả phát hiện gần như hoàn hảo trên các mẫu tấn công đã học, nhưng đặc biệt dễ bị tổn thương khi đối mặt với sự trôi dạt khái niệm hoặc các mẫu tấn công chưa từng thấy. Hơn nữa, đối với các phương pháp dựa trên quy tắc, việc xây dựng và duy trì các quy tắc heuristic có thể rất tốn kém và mất thời gian. Và đối với các phương pháp dựa trên DL, sự khan hiếm dữ liệu chứa tấn công đang cản trở khả năng triển khai thực tế của các phương pháp giám sát này. Để giải quyết vấn đề trên, **MAGIC** áp dụng kiểu phát hiện dị thường hoàn toàn tự giám sát, cho phép không cần dữ liệu chứa tấn công trong khi vẫn xử lý hiệu quả với các mẫu tấn công chưa từng thấy.

2.2.2. Các phương pháp dựa trên thống kê

Hầu hết các phương pháp dựa trên thống kê gần đây [7–9] phát hiện các tín hiệu APT bằng cách xác định các thực thể hệ thống, tương tác và cộng đồng dựa trên độ hiếm hoặc điểm dị thường của chúng. Tuy nhiên, độ hiếm của các thực thể hệ thống không nhất thiết chỉ ra sự bất thường của chúng và các điểm dị thường, thu được thông qua phân tích quan hệ nhân quả hoặc lan truyền nhãn, là việc trích xuất đặc trưng nông trên các đồ thị nguồn gốc. Để minh họa, quá trình tcexec thực hiện nhiều hoạt động quét cổng trên các địa chỉ IP khác nhau trong ví dụ minh họa của Nhóm tác giả (Xem Hình 1), điều này có thể được coi là một hành vi hệ thống bình thường. Tuy nhiên, xét đến việc quá trình tcexec, có nguồn gốc từ mạng bên ngoài, cũng đọc thông tin hệ thống nhạy cảm (uname) và kết nối với các địa chỉ IP công khai (162.66.239.75), chúng ta có thể dễ dàng xác định tcexec là một thực thể độc hại. Việc không trích xuất được ngữ nghĩa sâu và mối tương quan giữa các thực thể hệ thống thường dẫn đến hiệu suất phát hiện thấp và tỷ lệ dương tính giả cao của các phương pháp dựa trên thống kê. Tuy nhiên, **MAGIC** sử dụng một mô-đun biểu diễn đồ thị để thực hiện trích xuất đặc trưng đồ thị sâu trên các đồ thị nguồn gốc, dẫn đến các nhúng chất lượng cao.

2.2.3. Các phương pháp dựa trên DL

Gần đây, các phương pháp phát hiện APT dựa trên DL, bất kể là giám sát hay không giám sát, đều mang lại kết quả phát hiện rất hứa hẹn. Tuy nhiên, trên thực tế, hàng trăm GB nhật ký kiểm toán được tạo ra mỗi ngày trong một doanh nghiệp có quy mô trung bình. Do đó, các phương pháp dựa trên DL, đặc biệt là các phương pháp dựa trên chuỗi và dựa trên đồ thị [10, 12, 15–18], trở nên bất khả thi do chi phí tính toán của chúng. Ví dụ, ATLAS mất trung bình 1 giờ để huấn luyện trên 676MB nhật ký kiểm toán và ShadeWatcher mất 1 ngày để huấn luyện trên bộ dữ liệu DARPA E3 Trace với GPU. Bên cạnh đó, một số phương pháp dựa trên bộ mã hóa tự động đồ thị [25–27] gặp phải vấn đề chi phí bộ nhớ bùng nổ khi quy mô của các đồ thị nguồn gốc mở rộng. **MAGIC** tránh trở nên đòi hỏi về mặt tính toán bằng cách giới thiệu các bộ mã hóa tự động đồ thị bị che mặt và hoàn thành quá trình huấn luyện trên bộ dữ liệu DARPA E3 Trace chỉ trong vài phút. Đánh giá chi tiết về chi phí hiệu suất của **MAGIC** được trình bày trong Phần 6.4.

2.2.4. Các phương pháp đầu cuối

Ngoài ba hạn chế chính được thảo luận ở trên, cũng đáng nói rằng hầu hết các phương pháp phát hiện APT gần đây đều là các bộ phát hiện đầu cuối và tập trung vào một tác vụ phát hiện cụ thể. Ví dụ, ATLAS tập trung vào việc tái cấu trúc tấn công đầu cuối và Unicorn tạo ra các cảnh báo ở cấp độ hệ thống từ các luồng nhật ký. Thay vào đó, phương pháp của **MAGIC** là phổ quát và thực hiện phát hiện APT đa mức độ chi tiết trong nhiều kịch bản phát hiện khác nhau, đồng thời có thể được áp dụng cho nhật ký kiểm toán được thu thập từ các nguồn khác nhau.

2.3. Mô hình đe dọa và Định nghĩa

Đầu tiên, Nhóm tác giả trình bày mô hình đe dọa mà Nhóm tác giả sử dụng trong toàn bộ bài báo, sau đó định nghĩa chính thức các khái niệm quan trọng để hiểu cách **MAGIC** thực hiện phát hiện APT.

2.3.1. Mô hình mối đe dọa

Nhóm tác giả giả định rằng kẻ tấn công đến từ bên ngoài hệ thống và nhắm mục tiêu vào thông tin có giá trị bên trong hệ thống. Một kẻ tấn công có thể thực hiện các bước tinh vi để đạt được mục tiêu của mình nhưng để lại dấu vết có thể theo dõi trong nhật ký. Sự kết hợp giữa phần cứng hệ thống, hệ điều hành và phần mềm kiểm toán hệ thống là cơ sở tính toán đáng tin cậy của Nhóm tác giả. Các cuộc tấn công gây nhiễm độc và các cuộc tấn công trốn tránh không được xem xét trong mô hình đe dọa của Nhóm tác giả.

2.3.2. Đồ thị nguồn gốc

Một đồ thị nguồn gốc là một đồ thị có hướng và có chu trình được trích xuất từ nhật ký kiểm toán thô. Việc xây dựng một đồ thị nguồn gốc là thông lệ phổ biến trong nguồn gốc dữ liệu, vì nó kết nối các thực thể hệ thống và trình bày mối quan hệ tương tác giữa chúng. Một đồ thị nguồn gốc chứa các nút đại diện cho các thực thể hệ thống khác nhau (ví dụ: quy trình, tệp và socket) và các cạnh đại diện cho các tương tác giữa các thực thể hệ thống (ví dụ:

thực thi và kết nối), được gắn nhãn với các loại tương tác. Ví dụ, /tmp/tcexec là một thực thể hệ thống FileObject và cạnh giữa /tmp/tcexec và tcexec là một hoạt động thực thi từ một FileObject nhắm mục tiêu đến một Process (Xem Hình 1).

2.3.3. Phát hiện đa mức độ chi tiết

MAGIC có khả năng thực hiện phát hiện APT ở hai mức độ chi tiết: cấp độ nhật ký theo lô và cấp độ thực thể hệ thống. Khả năng phát hiện đa mức độ chi tiết của **MAGIC** dẫn đến một phương pháp phát hiện hai giai đoạn: đầu tiên thực hiện phát hiện cấp độ nhật ký theo lô trên các lô nhật ký liên tục và sau đó thực hiện phát hiện cấp độ thực thể hệ thống trên các lô dương tính để xác định kết quả phát hiện chi tiết. Việc áp dụng phương pháp này vào các thiết lập thực tế sẽ giảm thiểu hiệu quả khối lượng công việc, tiêu thụ tài nguyên và dương tính giả, đồng thời tạo ra các kết quả chi tiết.

- Phát hiện cấp độ nhật ký theo đợt. Ở mức độ chi tiết phát hiện APT này, nhiệm vụ chính là khi có nhật ký kiểm toán theo đợt từ một nguồn nhất quán, **MAGIC** cảnh báo nếu một APT tiềm ẩn được phát hiện trong một lô nhật ký. Tương tự như Unicorn, **MAGIC** không định vị chính xác các thực thể và tương tác hệ thống độc hại ở mức độ chi tiết phát hiện này.

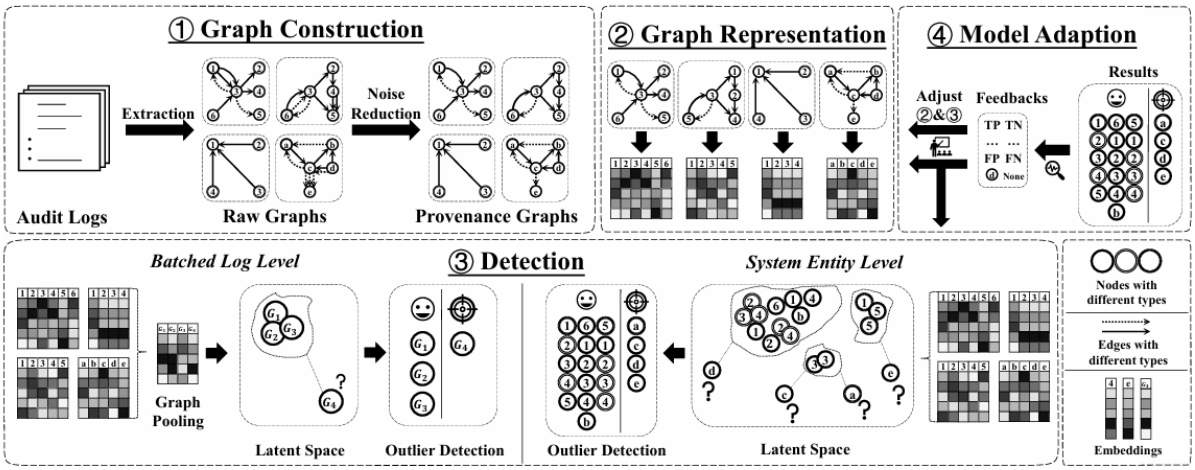
- Phát hiện cấp độ thực thể hệ thống. Nhiệm vụ phát hiện ở mức độ chi tiết phát hiện APT này là khi có nhật ký kiểm toán từ một nguồn nhất quán, **MAGIC** có khả năng định vị chính xác các thực thể hệ thống độc hại trong các nhật ký kiểm toán đó. Việc xác định các thực thể hệ thống quan trọng trong quá trình APT là rất quan trọng đối với các tác vụ tiếp theo như điều tra tấn công và khôi phục câu chuyện tấn công, vì nó cung cấp kết quả phát hiện dễ giải thích và giảm nhu cầu về các chuyên gia trong lĩnh vực cũng như các nỗ lực thủ công dư thừa.

3. Tổng quan về MAGIC

MAGIC là một phương pháp phát hiện APT tự giám sát mới lạ, tận dụng học biểu diễn đồ thị bị che mặt và các phương pháp phát hiện ngoại lệ, đồng thời có khả năng thực hiện phát hiện đa mức độ chi tiết hiệu quả trên khối lượng lớn nhật ký kiểm toán. Quy trình của **MAGIC** bao gồm ba thành phần chính: (1) xây dựng đồ thị nguồn gốc, (2) một mô-đun biểu diễn đồ thị và (3) một mô-đun phát hiện. Nó cũng cung cấp một tùy chọn (4) cơ chế thích ứng mô hình. Trong quá trình huấn luyện, **MAGIC** chuyển đổi dữ liệu huấn luyện bằng (1), học nhúng đồ thị bằng (2) và ghi nhớ các hành vi lành tính trong (3). Trong quá trình suy luận, **MAGIC** chuyển đổi dữ liệu mục tiêu bằng (1), thu được nhúng đồ thị bằng (2) đã được huấn luyện và phát hiện các ngoại lệ thông qua (3). Hình 2 đưa ra tổng quan về kiến trúc của **MAGIC**.

Các luồng nhật ký kiểm toán do phần mềm kiểm toán hệ thống thu thập thường được lưu trữ theo đợt. Trong quá trình xây dựng đồ thị nguồn gốc (1), **MAGIC** chuyển đổi các nhật ký này thành các đồ thị nguồn gốc tĩnh. Các thực thể hệ thống và tương tác giữa chúng được trích xuất và chuyển đổi thành các nút và cạnh tương ứng. Một số kỹ thuật giảm độ phức tạp được sử dụng để loại bỏ thông tin dư thừa.

Các đồ thị nguồn gốc đã được xây dựng sau đó được đưa qua mô-đun biểu diễn đồ thị (2) để thu được các nhúng đầu ra (tức là các biểu diễn vectơ toàn diện của các đối tượng). Được xây dựng dựa trên các bộ mã hóa tự động đồ thị bị che mặt và tích hợp tái cấu trúc cấu trúc dựa trên mẫu, mô-đun biểu diễn đồ thị nhúng, lan truyền và tổng hợp các thuộc tính của nút và cạnh vào các nhúng đầu ra, chứa cả nhúng nút và nhúng trạng thái hệ thống.



Hình 2. Tổng quan về quy trình phát hiện của MAGIC.

Mô-đun biểu diễn đồ thị chỉ được huấn luyện trên nhật ký kiểm toán lành tính để mô hình hóa các hành vi hệ thống lành tính. Khi thực hiện phát hiện APT trên nhật ký kiểm toán có khả năng chứa tấn công, **MAGIC** sử dụng các phương pháp phát hiện ngoại lệ dựa trên các nhúng đầu ra để phát hiện các ngoại lệ trong hành vi hệ thống (3). Tùy thuộc vào mức độ chi tiết của tác vụ, các nhúng khác nhau được sử dụng để hoàn thành việc phát hiện APT. Đối với các tác vụ ở cấp độ nhật ký theo đợt, các nhúng trạng thái hệ thống, phản ánh các hành vi chung của toàn bộ hệ thống, là mục tiêu phát hiện. Một ngoại lệ trong các nhúng như vậy có nghĩa là trạng thái hệ thống tương ứng của nó chưa từng thấy và có khả năng độc hại, điều này tiết lộ một tín hiệu APT trong lô đó. Đối với các tác vụ ở cấp độ thực thể hệ thống, mục tiêu phát hiện là các nhúng nút đó, đại diện cho hành vi của các thực thể hệ thống. Các ngoại lệ trong nhúng nút cho thấy các thực thể hệ thống đáng ngờ và phát hiện các mối đe dọa APT ở mức độ chi tiết cao hơn.

Trong các thiết lập phát hiện thực tế, **MAGIC** có hai ứng dụng được thiết kế trước. Đối với mỗi lô nhật ký được thu thập bởi phần mềm kiểm toán hệ thống, người dùng có thể trực tiếp sử dụng tính năng phát hiện ở cấp độ thực thể của **MAGIC** để xác định chính xác các thực thể độc hại trong lô, hoặc thực hiện phát hiện hai giai đoạn, như đã nêu trong Phần 2.3. Trong trường hợp này, **MAGIC** đầu tiên quét một lô và xem liệu có tín hiệu độc hại tồn tại trong lô hay không (phát hiện cấp độ nhật ký theo lô). Nếu nó cảnh báo dương tính, **MAGIC** sau đó thực hiện phát hiện ở cấp độ thực thể để xác định các thực thể hệ thống độc hại ở mức độ chi tiết cao hơn. Phát hiện ở cấp độ nhật ký theo lô ít đòi hỏi tính toán hơn đáng kể so với phát hiện ở cấp độ thực thể. Do đó, một quy trình hai giai đoạn như vậy có thể giúp người dùng của **MAGIC** tiết kiệm tài nguyên tính toán và tránh các báo động giả mà không ảnh hưởng đến độ chi tiết phát hiện của **MAGIC**. Tuy nhiên, nếu người dùng ưu tiên phát

hiện chi tiết trên tất cả các thực thể hệ thống, thì quy trình trước đó vẫn là một tùy chọn khả dụng.

Để đối phó với sự trôi dạt khái niệm và các cuộc tấn công chưa từng thấy, một cơ chế thích ứng mô hình tùy chọn được sử dụng để cung cấp các kênh phản hồi cho người dùng (4). Kết quả phát hiện được các nhà phân tích bảo mật kiểm tra và xác nhận được đưa trở lại **MAGIC**, giúp nó thích ứng với những thay đổi trong hành vi hệ thống lành tính theo cách bán giám sát. Trong những điều kiện như vậy, **MAGIC** đạt được kết quả phát hiện thậm chí còn hứa hẹn hơn, điều này được thảo luận trong Phần 6.3. Hơn nữa, **MAGIC** có thể dễ dàng được áp dụng cho việc phát hiện APT trực tuyến trong thế giới thực nhờ khả năng tự thích ứng với sự trôi dạt khái niệm và chi phí tính toán tối thiểu.

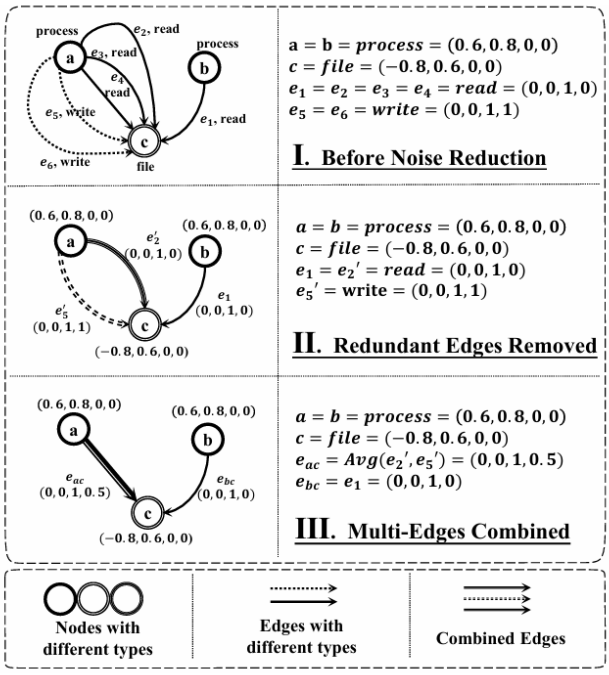
4. Chi tiết thiết kế

Trong phần này, Nhóm tác giả giải thích chi tiết cách **MAGIC** thực hiện phát hiện APT hiệu quả trên khối lượng lớn nhật ký kiểm toán. **MAGIC** chứa bốn thành phần chính: một giai đoạn xây dựng đồ thị để xây dựng các đồ thị dòng dõi được tối ưu hóa và nhất quán (Phần 4.1), một mô-đun biểu diễn đồ thị tạo ra các nhúng đầu ra với hiệu quả tối đa (Phần 4.2), một mô-đun phát hiện sử dụng các phương pháp phát hiện ngoại lệ để thực hiện phát hiện APT (Phần 4.3) và một cơ chế thích ứng mô hình để đối phó với sự trôi dạt khái niệm và các phản hồi chất lượng cao khác (Phần 4.4).

4.1. Xây dựng đồ thị nguồn gốc

MAGIC đầu tiên xây dựng một đồ thị dòng dõi từ nhật ký kiểm toán thô trước khi thực hiện biểu diễn đồ thị và phát hiện APT. Nhóm tác giả tuân theo ba bước để xây dựng một đồ thị dòng dõi nhất quán và được tối ưu hóa, sẵn sàng cho việc biểu diễn đồ thị. Phân tích cú pháp nhật ký. Bước đầu tiên là chỉ cần phân tích cú pháp từng mục nhật ký, trích xuất các thực thể hệ thống và các tương tác hệ thống giữa chúng. Sau đó, một đồ thị dòng dõi nguyên mẫu có thể được xây dựng với các thực thể hệ thống là các nút và các tương tác là các cạnh. Bây giờ Nhóm tác giả trích xuất thông tin phân loại liên quan đến các nút và cạnh. Đối với định dạng nhật ký đơn giản cung cấp nhãn thực thể và tương tác, Nhóm tác giả sử dụng trực tiếp các nhãn này. Đối với một số định dạng cung cấp các thuộc tính phức tạp của các thực thể và tương tác đó, Nhóm tác giả áp dụng băm đa nhãn (ví dụ: xxhash) để chuyển đổi các thuộc tính thành nhãn. Ở giai đoạn này, đồ thị dòng dõi là một đa đồ thị có hướng. Nhóm tác giả đã thiết kế một ví dụ để minh họa cách Nhóm tác giả xử lý đồ thị dòng dõi thô sau khi phân tích cú pháp nhật ký trong Hình 3. Nhúng ban đầu. Ở giai đoạn này, Nhóm tác giả chuyển đổi nhãn nút và cạnh thành vectơ đặc trưng có kích thước cố định (tức là nhúng ban đầu) có chiều d, trong đó d là chiều ẩn của mô-đun biểu diễn đồ thị của Nhóm tác giả. Nhóm tác giả áp dụng nhúng tra cứu, thiết lập mối quan hệ một-một giữa nhãn nút/cạnh với các vectơ đặc trưng d chiều. Như được minh họa trong Hình 3 (I và II), các quy trình a và b có cùng nhãn, do đó chúng được ánh xạ tới cùng một vectơ đặc trưng, trong khi a và c được nhúng vào các vectơ đặc trưng khác nhau vì chúng có nhãn khác nhau. Nhóm tác giả lưu ý rằng số lượng nhãn nút/cạnh duy nhất có thể có được xác định bởi nguồn dữ liệu (tức là định dạng nhật ký

kiểm toán). Do đó, nhúng tra cứu hoạt động trong một thiết lập quy nạp và không cần học các nhúng cho các nhãn chưa từng thấy. Giảm nhiễu. Đầu vào dự kiến của mô-đun biểu diễn đồ thị của Nhóm tác giả là các đồ thị đơn. Do đó, Nhóm tác giả cần kết hợp nhiều cạnh giữa các cặp nút. Nếu nhiều cạnh có cùng nhãn (cũng chia sẻ cùng một nhúng ban đầu) tồn tại giữa một cặp nút, Nhóm tác giả loại bỏ các cạnh dư thừa để chỉ một trong số chúng còn lại. Sau đó, Nhóm tác giả kết hợp các cạnh còn lại thành một cạnh cuối cùng. Nhóm tác giả lưu ý rằng giữa một cặp nút, các cạnh có nhiều nhãn khác nhau có thể vẫn còn. Sau khi kết hợp, nhúng ban đầu của cạnh duy nhất kết quả được thu được bằng cách lấy trung bình các nhúng ban đầu của các cạnh còn lại. Để minh họa, Nhóm tác giả cho thấy cách giảm nhiễu của Nhóm tác giả kết hợp đa cạnh và cách nó ảnh hưởng đến các nhúng ban đầu của cạnh trong Hình 3 (II và III). Đầu tiên, ba tương tác đọc và hai tương tác ghi giữa a và c được hợp nhất thành một cho mỗi nhãn. Sau đó, Nhóm tác giả kết hợp chúng lại với nhau, tạo thành một cạnh eac với nhúng ban đầu bằng trung bình nhúng ban đầu của các cạnh còn lại (e′2 và e′5). Nhóm tác giả cung cấp một so sánh giữa các bước giảm nhiễu của Nhóm tác giả và các công trình trước đây trong Phụ lục E.

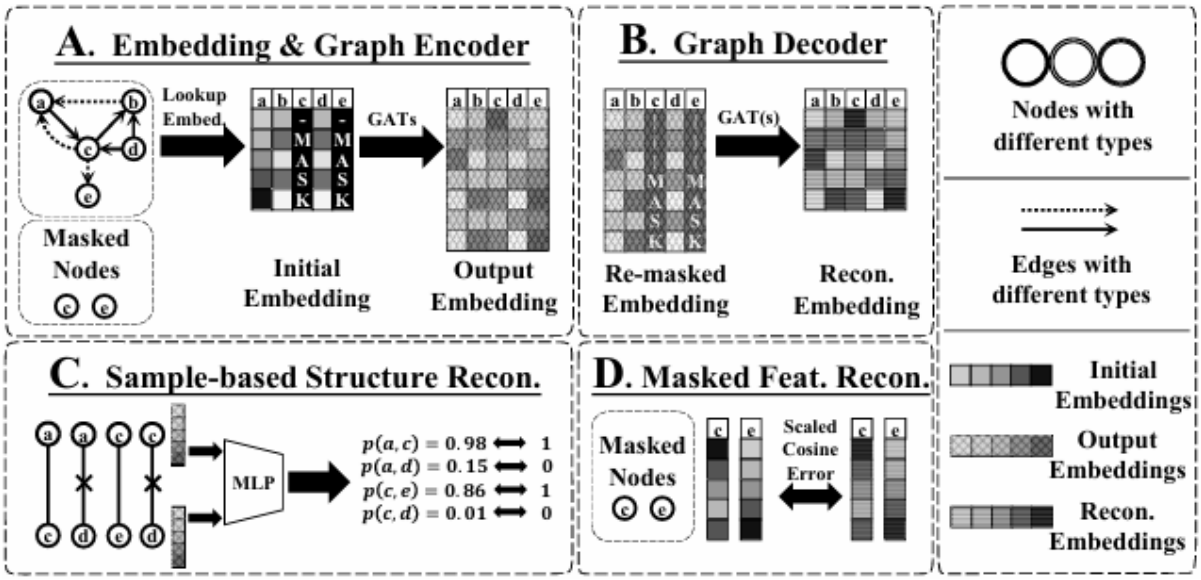


Hình 3. Ví dụ về các bước xây dựng đồ thị của MAGIC.

Sau khi thực hiện ba bước trên, **MAGIC** đã hoàn thành việc xây dựng một đồ thị dòng dõi nhất quán và bảo toàn thông tin, sẵn sàng cho các tác vụ tiếp theo. Trong quá trình xây dựng đồ thị dòng dõi, ít thông tin bị mất vì **MAGIC** chỉ làm tổn hại đến ngữ nghĩa ban đầu bằng cách khái quát hóa các mô tả chi tiết về các thực thể và tương tác hệ thống thành các nhãn. Tuy nhiên, trung bình 79,60% tổng số cạnh đã được giảm trên bộ dữ liệu DARPA E3 Trace, giúp tiết kiệm thời gian huấn luyện và mức tiêu thụ bộ nhớ của **MAGIC**.

4.2. Mô-đun biểu diễn đồ thị

MAGIC sử dụng một mô-đun biểu diễn đồ thị để thu được các nhúng chất lượng cao từ các đồ thị dòng dõi đã được đặc trưng hóa. Như được minh họa trong Hình 4, mô-đun biểu diễn đồ thị bao gồm ba giai đoạn: một quy trình che mặt để ẩn một phần các đặc trưng nút (tức là các nhúng ban đầu) cho mục đích tái cấu trúc (Phần 4.2.1), một bộ mã hóa đồ thị tạo ra các nhúng đầu ra của nút và trạng thái hệ thống bằng cách lan truyền và tổng hợp các đặc trưng đồ thị (Phần 4.2.2), một bộ giải mã đồ thị cung cấp các tín hiệu giám sát cho việc huấn luyện mô-đun biểu diễn đồ thị thông qua tái cấu trúc đặc trưng bị che mặt và tái cấu trúc cấu trúc dựa trên mẫu (Phần 4.2.3). Bộ mã hóa và giải mã tạo thành một bộ mã hóa tự động đồ thị bị che mặt, vượt trội trong việc tạo ra các nhúng nhanh và tiết kiệm tài nguyên.



Hình 4. Mô-đun biểu diễn đồ thị của MAGIC

4.2.1. Che đặc trưng

Trước khi huấn luyện mô-đun biểu diễn đồ thị của Nhóm tác giả, Nhóm tác giả thực hiện che mặt trên các nút, để bộ mã hóa tự động đồ thị bị che mặt có thể được huấn luyện dựa trên việc tái cấu trúc các nút này. Các nút bị che mặt được chọn ngẫu nhiên, bao phủ một tỷ lệ nhất định của tất cả các nút. Các nhúng ban đầu của các nút bị che mặt như vậy được thay thế bằng một mã thông báo mặt nạ đặc biệt xmask để che đi mọi thông tin ban đầu về các nút này. Tuy nhiên, các cạnh không bị che mặt vì các cạnh này cung cấp thông tin quý giá về mối quan hệ giữa các thực thể hệ thống. Tóm lại, với các nhúng ban đầu của nút xn, Nhóm tác giả che mặt các nút như sau:

$$emb_n = \begin{cases} x_n, & n \notin \widetilde{N} \\ x_{mask}, & n \in \widetilde{N} \end{cases}$$

trong đó $\tilde{N}$ là các nút bị che mặt được chọn ngẫu nhiên, emb_n là nhúng của nút n sẵn sàng cho việc huấn luyện mô-đun biểu diễn đồ thị. Quá trình che mặt này chỉ xảy ra trong quá trình huấn luyện. Trong quá trình phát hiện, Nhóm tác giả không che mặt bất kỳ nút nào.

4.2.2. Bộ mã hóa đồ thị

Các nhúng ban đầu thu được từ các bước xây dựng đồ thị chỉ xem xét các đặc trưng thô. Tuy nhiên, các đặc trưng thô không đủ để mô hình hóa các hành vi chi tiết của các thực thể hệ thống. Thông tin ngữ cảnh của một thực thể, chẳng hạn như vùng lân cận của nó, mối quan hệ nhiều bước và các mẫu tương tác của nó với các thực thể hệ thống khác đóng vai trò quan trọng để thu được các nhúng thực thể chất lượng cao. Ở đây, Nhóm tác giả sử dụng và mở rộng các bộ mã hóa tự động đồ thị bị che mặt để tạo ra các nhúng đầu ra theo cách tự giám sát. Bộ mã hóa tự động đồ thị bị che mặt bao gồm một bộ mã hóa và một bộ giải mã. Bộ mã hóa tạo ra các nhúng đầu ra bằng cách lan truyền và tổng hợp các đặc trưng đồ thị, và bộ giải mã tái cấu trúc các đặc trưng đồ thị để cung cấp các tín hiệu giám sát cho việc huấn luyện. Kiến trúc bộ mã hóa-giải mã như vậy duy trì thông tin ngữ cảnh và ngữ nghĩa trong các nhúng được tạo ra, trong khi chi phí tính toán của nó giảm đáng kể thông qua học bị che mặt.

Bộ mã hóa của mô-đun biểu diễn đồ thị của Nhóm tác giả chứa nhiều lớp mạng lưới chú ý đồ thị (GAT) xếp chồng lên nhau. Chức năng của một lớp GAT là tạo ra các nhúng nút đầu ra theo cả các đặc trưng (nhúng ban đầu) của chính nút đó và các nút lân cận của nó. Khác với các GNN thông thường, GAT giới thiệu một cơ chế chú ý để đo lường tầm quan trọng của các nút lân cận đó.

Để giải thích chi tiết, một lớp GAT nhận các nhúng nút được tạo ra bởi các lớp trước đó làm đầu vào và lan truyền các nhúng từ các nút nguồn đến các nút đích thành các thông điệp dọc theo các tương tác. Thông điệp chứa thông tin về nút nguồn và tương tác giữa nguồn và đích:

$$MSG(src, dst) = W_{msg}^{T}(h_{src}||emb_e).$$

Và cơ chế chú ý được sử dụng để tính toán các hệ số chú ý giữa nguồn thông điệp và đích của nó:

$$\alpha(src, dst) = LeakyReLU(W_{as}^{T} h_{src} + W_{am} MSG(src, dst)),$$
$$a(src, dst) = Softmax(\alpha(src, dst)).$$

Sau đó, đối với nút đích, GAT tổng hợp các thông điệp từ các cạnh đến để cập nhật nhúng nút của nó bằng cách tính tổng có trọng số của tất cả các thông điệp đến. Trọng số chính là các hệ số chú ý:

$$AGG(h_{dst}, h_{\mathcal{N}}) = W_{self} h_{dst} + \sum_{i \in \mathcal{N}} a(i, dst) MSG(i, dst),$$
$$h_n^l = AGG^l(h_n^{l-1}, h_{\mathcal{N}_n}^{l-1}).$$

trong đó $h^l{}_n$ là nhúng ẩn của nút n ở lớp thứ l của GAT, $h^{l-1}{}_n$ là của lớp $l-1$ và N_n là vùng lân cận một bước của n. Đầu vào của lớp GAT đầu tiên là các nhúng nút ban đầu. embe là

nhúng cạnh ban đầu và không đổi trong suốt mô-đun biểu diễn đồ thị. Was, Wam, Wsel f, Wmsg là các tham số có thể huấn luyện. Nhúng nút được cập nhật tạo thành một sự trừu tượng hóa chung về hành vi tương tác một bước của nút.

Nhiều lớp GAT như vậy được xếp chồng lên nhau để thu được nhúng nút cuối cùng *hn*, được nối với nhúng nút gốc và đầu ra của tất cả các lớp GAT:

$$h_n = emb_n||h_n^1||\cdots||h_n^l.$$

$\cdot||\cdot$ biểu thị phép nối. Càng nhiều lớp GAT được xếp chồng lên nhau, phạm vi lân cận càng rộng và mẫu tương tác nhiều bước của một nút càng được biểu diễn xa hơn trong nhúng của nó. Do đó, bộ mã hóa đồ thị kết hợp hiệu quả các đặc trưng ban đầu của nút và các hành vi tương tác nhiều bước để trừu tượng hóa các hành vi của thực thể hệ thống thành các nhúng nút. Bộ mã hóa đồ thị cũng áp dụng phép gộp trung bình cho tất cả các nhúng nút để tạo ra một nhúng toàn diện của chính đồ thị, tóm tắt toàn bộ trạng thái của hệ thống:

$$h_G = \frac{1}{|N|}\sum_{n_i \in N} h_{n_i}.$$

Các nhúng nút và các nhúng trạng thái hệ thống được tạo ra bởi bộ mã hóa đồ thị được coi là đầu ra của mô-đun biểu diễn đồ thị, được sử dụng trong các tác vụ tiếp theo trong các kịch bản khác nhau.

4.2.3. Bộ giải mã đồ thị

Bộ mã hóa đồ thị không cung cấp các tín hiệu giám sát hỗ trợ quá trình huấn luyện mô hình. Trong các bộ mã hóa tự động đồ thị điển hình, một bộ giải mã đồ thị được sử dụng để giải mã các nhúng nút và giám sát quá trình huấn luyện mô hình thông qua tái cấu trúc đặc trưng và tái cấu trúc cấu trúc. Tuy nhiên, các bộ mã hóa tự động đồ thị bị che mặt loại bỏ tái cấu trúc cấu trúc để giảm chi phí tính toán. Bộ giải mã đồ thị của Nhóm tác giả là sự kết hợp của cả hai, tích hợp tái cấu trúc đặc trưng bị che mặt và tái cấu trúc cấu trúc dựa trên mẫu để xây dựng một hàm mục tiêu tối ưu hóa mô-đun biểu diễn đồ thị.

Cho các nhúng nút *hn* thu được từ bộ mã hóa đồ thị, bộ giải mã đầu tiên che mặt lại các nút đã được che mặt và chuyển đổi chúng thành đầu vào của quá trình tái cấu trúc đặc trưng bị che mặt:

$$h_n^* = \begin{cases} W^* h_n, & n \notin \widetilde{N} \\ W^* v_{remask}, & n \in \widetilde{N} \end{cases},$$

Sau đó, bộ giải mã sử dụng một lớp GAT tương tự như đã mô tả ở trên để tái cấu trúc các nhúng ban đầu của các nút bị che mặt, cho phép tính toán một tổn thất tái cấu trúc đặc trưng:

$$x_n^* = AGG^*(h_n^*, h_{\mathcal{N}_v}^*),$$
$$L_{fr} = \frac{1}{|\widetilde{N}|}\sum_{n_i \in \widetilde{N}}(1 - \frac{x_{n_i}^T x_{n_i}^*}{||x_{n_i}|| \cdot ||x_{n_i}^*||})^\gamma.$$

trong đó L_{fr} là tổn thất tái cấu trúc đặc trưng bị che mặt thu được bằng cách tính toán tổn thất cosin được chia tỷ lệ giữa các nhúng ban đầu và được tái cấu trúc của các nút bị che mặt. Tổn

thất này thay đổi đáng kể giữa các mẫu dễ và khó, giúp tăng tốc quá trình học một cách hiệu quả. Mức độ chia tỷ lệ như vậy được kiểm soát bởi một siêu tham số γ.

Trong khi đó, tái cấu trúc cấu trúc dựa trên mẫu nhằm mục đích tái cấu trúc cấu trúc đồ thị (tức là dự đoán các cạnh giữa các nút). Thay vì tái cấu trúc toàn bộ ma trận kề, có độ phức tạp O(N2), tái cấu trúc cấu trúc dựa trên mẫu áp dụng lấy mẫu đối lập trên các cặp nút và dự đoán xác suất cạnh giữa các cặp đó. Chỉ các nút không bị che mặt tham gia vào tái cấu trúc cấu trúc. Các mẫu dương được xây dựng với tất cả các cạnh hiện có giữa các nút không bị che mặt, và các mẫu âm được lấy mẫu trong số các cặp nút không có cạnh hiện có giữa chúng.

Một MLP hai lớp đơn giản được sử dụng để tái cấu trúc các cạnh giữa các mẫu cặp nút, tạo ra một xác suất cho mỗi mẫu. Tổn thất tái cấu trúc có dạng một tổn thất entropy chéo nhị phân đơn giản trên các mẫu đó:

$$prob(n,n') = \sigma(MLP(h_n||h_{n'})),$$
$$L_{sr} = -\frac{1}{|\hat{N}|}\sum_{n\in\hat{N}}(log(1-prob(n,n^-)) + log(prob(n,n^+))).$$

trong đó (n,n^-) và (n,n^+) lần lượt là các mẫu âm và dương, và $\hat{N} = N - \tilde{N}$ là tập hợp các nút không bị che mặt. Tái cấu trúc cấu trúc dựa trên mẫu chỉ cung cấp sự giám sát cho các nhúng đầu ra. Thay vì sử dụng tích vô hướng, Nhóm tác giả sử dụng một MLP để tính toán xác suất cạnh vì các thực thể tương tác không nhất thiết phải tương tự nhau về hành vi. Ngoài ra, Nhóm tác giả không buộc mô hình phải học cách dự đoán xác suất cạnh. Chức năng của việc tái cấu trúc cấu trúc như vậy là tối đa hóa thông tin hành vi chứa trong các nhúng nút đã được trừu tượng hóa, để một MLP đơn giản là đủ để kết hợp và diễn giải thông tin đó thành xác suất cạnh.

Hàm mục tiêu cuối cùng $L = L_{fr} + L_{sr}$ kết hợp L_{fr} và L_{sr} và cung cấp các tín hiệu giám sát cho mô-đun biểu diễn đồ thị, cho phép nó học các tham số theo cách tự giám sát.

4.3. Mô-đun phát hiện

Dựa trên các nhúng đầu ra được tạo ra bởi mô-đun biểu diễn đồ thị, Nhóm tác giả sử dụng các phương pháp phát hiện ngoại lệ để thực hiện phát hiện APT theo cách không giám sát. Như đã giải thích chi tiết trong các phần trước, các nhúng này tóm tắt các hành vi hệ thống ở các mức độ chi tiết khác nhau. Mục tiêu của mô hình phát hiện của Nhóm tác giả là xác định các thực thể hoặc trạng thái hệ thống độc hại chỉ với kiến thức tiên nghiệm về các hành vi hệ thống lành tính. Các nhúng được tạo ra thông qua học biểu diễn đồ thị có xu hướng tạo thành các cụm nếu các thực thể tương ứng của chúng có các hành vi tương tác tương tự trong đồ thị [19, 25–27, 32]. Do đó, các ngoại lệ trong các nhúng trạng thái hệ thống cho thấy các hành vi hệ thống bất thường và đáng ngờ. Dựa trên hiểu biết sâu sắc này, Nhóm tác giả phát triển một phương pháp phát hiện ngoại lệ đặc biệt để thực hiện phát hiện APT.

Trong quá trình huấn luyện, các nhúng đầu ra lành tính đầu tiên được trừu tượng hóa từ các đồ thị dòng dõi huấn luyện. Những gì mô-đun phát hiện làm ở giai đoạn này chỉ đơn

giản là ghi nhớ các nhúng đó và sắp xếp chúng trong một Cây K-D. Sau khi huấn luyện, mô-đun phát hiện tiết lộ các ngoại lệ trong ba bước: tìm kiếm k-láng giềng gần nhất, tính toán độ tương tự và lọc. Với một nhúng mục tiêu, mô-đun phát hiện đầu tiên thu được k-láng giềng gần nhất của nó thông qua tìm kiếm Cây K-D. Quá trình tìm kiếm như vậy chỉ mất thời gian log(N), trong đó N là tổng số nhúng huấn luyện đã ghi nhớ. Sau đó, một tiêu chí tương tự được áp dụng để đánh giá độ gần gũi của nhúng mục tiêu với các láng giềng của nó và tính toán một điểm bất thường. Nếu điểm bất thường của nó cao hơn một siêu tham số θ, nhúng mục tiêu được coi là một ngoại lệ và thực thể hệ thống hoặc trạng thái hệ thống tương ứng của nó là độc hại. Một quy trình làm việc ví dụ của mô-đun phát hiện được chính thức hóa như sau, sử dụng khoảng cách Euclid làm tiêu chí tương tự:

$$\mathcal{N}_x = KNN(x)$$
$$dist_x = \frac{1}{|\mathcal{N}_x|} \sum_{x_i \in \mathcal{N}_x} ||x - x_i||$$
$$score_x = \frac{dist_x}{\overline{dist}}$$
$$result_x = \begin{cases} 1, & score_x \geq \theta \\ 0, & score_x < \theta \end{cases}$$

trong đó dist là khoảng cách trung bình giữa các nhúng huấn luyện và k-láng giềng gần nhất của chúng. Khi thực hiện phát hiện cấp độ nhật ký theo lô, mô-đun phát hiện ghi nhớ các nhúng trạng thái hệ thống lành tính phản ánh trạng thái hệ thống và phát hiện xem nhúng trạng thái hệ thống của một đồ thị dòng dõi mới đến có phải là một ngoại lệ hay không. Khi thực hiện phát hiện cấp độ thực thể hệ thống, mô-đun phát hiện thay vào đó ghi nhớ các nhúng nút lành tính chỉ ra hành vi của thực thể hệ thống và khi có một đồ thị dòng dõi mới đến, nó phát hiện các ngoại lệ trong các nhúng của tất cả các thực thể hệ thống.

4.4. Cơ chế thích ứng mô hình

Để một bộ phát hiện APT hoạt động hiệu quả trong các kịch bản phát hiện thực tế, cần phải xem xét đến sự trôi dạt khái niệm. Khi đối mặt với các hành vi hệ thống lành tính nhưng chưa từng thấy trước đây, **MAGIC** tạo ra kết quả phát hiện dương tính giả, điều này có thể gây hiểu lầm cho các ứng dụng tiếp theo (ví dụ: điều tra tấn công và khôi phục câu chuyện). Các công trình gần đây giải quyết vấn đề này bằng cách quên đi dữ liệu lỗi thời hoặc điều chỉnh mô hình của chúng cho phù hợp với những thay đổi hệ thống lành tính thông qua cơ chế thích ứng mô hình. **MAGIC** cũng tích hợp cơ chế thích ứng mô hình để chống lại sự trôi dạt khái niệm và học hỏi từ các dương tính giả được xác định bởi các nhà phân tích bảo mật. Hơi khác so với các công trình khác chỉ sử dụng dương tính giả để huấn luyện lại mô hình, **MAGIC** có thể được huấn luyện lại với tất cả các phản hồi. Như đã thảo luận trong các phần trước, mô-đun biểu diễn đồ thị trong **MAGIC** mã hóa các thực thể hệ thống thành các nhúng theo cách tự giám sát, mà không cần biết nhãn của nó. Bất kỳ dữ liệu chưa từng thấy nào, bao gồm cả các âm tính thật, đều là dữ liệu huấn luyện có giá trị cho mô-đun biểu diễn đồ thị để nâng cao khả năng biểu diễn của nó trên các hành vi hệ thống chưa từng thấy.

Mô-đun phát hiện chỉ có thể được huấn luyện lại với các phản hồi lành tính để theo kịp những thay đổi trong hành vi hệ thống. Và khi nó ghi nhớ ngày càng nhiều phản hồi lành tính, hiệu quả phát hiện của nó sẽ giảm xuống. Để giải quyết vấn đề này, Nhóm tác giả cũng triển khai một cơ chế giảm giá trên mô-đun phát hiện. Khi khối lượng các nhúng đã ghi nhớ vượt quá một lượng nhất định, các nhúng sớm nhất sẽ đơn giản bị loại bỏ khi các nhúng mới đến được học. Nhóm tác giả cung cấp cơ chế thích ứng mô hình như một giải pháp tùy chọn cho sự trôi dạt khái niệm và các hành vi hệ thống chưa từng thấy. Nên điều chỉnh **MAGIC** cho phù hợp với những thay đổi của hệ thống bằng cách cung cấp các mẫu dương tính giả đã được xác nhận cho cơ chế thích ứng mô hình của **MAGIC**.

5. Triển khai

Nhóm tác giả triển khai **MAGIC** với khoảng 3.500 dòng mã Python 3.8. Nhóm tác giả phát triển một số trình phân tích cú pháp nhật ký để đối phó với các định dạng nhật ký kiểm toán khác nhau, bao gồm StreamSpot, Camflow và CDM. Đồ thị dòng dõi được xây dựng bằng thư viện xử lý đồ thị Networkx và được lưu trữ ở định dạng JSON. Mô-đun biểu diễn đồ thị được triển khai thông qua PyTorch và DGL. Mô-đun phát hiện được phát triển bằng Scikit-learn. Đối với các siêu tham số của **MAGIC**, hệ số tỷ lệ γ trong tổn thất tái cấu trúc đặc trưng được đặt thành 3, số lượng láng giềng k được đặt thành 10, tốc độ học là 0,001 và hệ số suy giảm trọng số bằng $5 \times 10{-}4$. Nhóm tác giả sử dụng một bộ mã hóa đồ thị 3 lớp và tỷ lệ che mặt là 0,5 trong các thí nghiệm của mình. Chiều nhúng đầu ra d khác nhau trong hai kịch bản phát hiện, phát hiện ở cấp độ nhật ký theo lô và phát hiện ở cấp độ thực thể. Nhóm tác giả sử dụng d bằng 256 trong phát hiện ở cấp độ nhật ký theo lô và đặt d bằng 64 trong phát hiện ở cấp độ thực thể để giảm tiêu thụ tài nguyên. Ngưỡng phát hiện θ được chọn bằng cách tìm kiếm tuyến tính đơn giản riêng biệt trên mỗi bộ dữ liệu. Các siêu tham số có thể có các lựa chọn khác. Nhóm tác giả sẽ chứng minh tác động của các siêu tham số này lên **MAGIC** sau trong phần đánh giá. Trong phân tích siêu tham số của Nhóm tác giả, d được chọn từ {16, 32, 64, 128, 256}, l từ {1, 2, 3, 4} và r từ {0.3, 0.5, 0.7}. Đối với ngưỡng θ, nó được chọn giữa 1 và 10 trong phát hiện ở cấp độ nhật ký theo lô. Đối với phát hiện ở cấp độ thực thể, vui lòng tham khảo Phụ lục D.

6. Đánh giá

Nhóm tác giả sử dụng 131GB nhật ký kiểm toán thu được từ các phần mềm kiểm toán hệ thống khác nhau để đánh giá hiệu quả và hiệu suất của **MAGIC**. Đầu tiên, Nhóm tác giả mô tả cài đặt thử nghiệm của mình (Phần 6.1), sau đó trình bày chi tiết hiệu quả của **MAGIC** trong các kịch bản khác nhau (Phần 6.2), thực hiện phân tích dương tính giả và đánh giá tính hữu ích của cơ chế thích ứng mô hình (Phần 6.3) và phân tích chi phí hiệu suất thời gian chạy của **MAGIC** (Phần 6.4). Tác động của các thành phần và siêu tham số khác nhau của **MAGIC** được phân tích trong Phần 6.5. Ngoài ra, một nghiên cứu trường hợp chi tiết về ví dụ minh họa của Nhóm tác giả được thực hiện trong Phụ lục C để minh họa cách quy trình của **MAGIC** hoạt động để phát hiện APT. Các thí nghiệm này được thực hiện trên cùng một thiết bị.

6.1. Cài đặt thử nghiệm

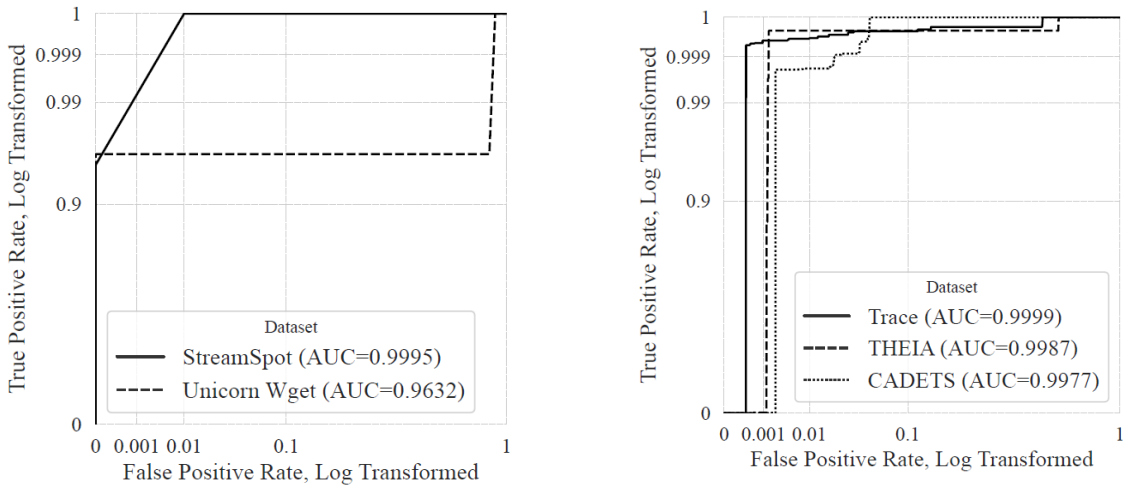
Nhóm tác giả đánh giá hiệu quả của **MAGIC** trên ba bộ dữ liệu công khai: bộ dữ liệu StreamSpot, bộ dữ liệu Unicorn Wget và bộ dữ liệu DARPA Engagement 3. Các bộ dữ liệu này khác nhau về dung lượng, nguồn gốc và độ chi tiết. Nhóm tác giả tin rằng bằng cách kiểm tra hiệu suất của **MAGIC** trên các bộ dữ liệu này, Nhóm tác giả có thể so sánh **MAGIC** với càng nhiều phương pháp phát hiện APT hiện đại càng tốt và khám phá tính phổ quát và khả năng ứng dụng của **MAGIC**. Nhóm tác giả cung cấp mô tả chi tiết về ba bộ dữ liệu như sau.

Dataset	Scenario	Malicious	#Log pieces	Avg. #Entity	Avg. #Interaction	Size(GB)
StreamSpot	CNN		100	8,989	294,903	0.9
	Download		100	8,830	310,814	1.0
	Gmail		100	6,826	37,382	0.1
	VGame		100	8,636	112,958	0.4
	YouTube		100	8,292	113,229	0.3
	Attack	✓	100	8,890	28,423	0.1
Unicorn Wget	Benign		125	265,424	975,226	64.0
	Attack	✓	25	257,156	949,887	12.6

Bảng 1. Bộ dữ liệu cho phát hiện cấp độ nhật ký theo lô.

Dataset	Scenario	Malicious	#Node	#Edge	Size (GB)
DARPA E3 Trace	Benign		3,220,594	4,080,457	15.40
	Extension Backdoor	✓	732		
	Pine Backdoor	✓	67,345		
	Phishing Executable	✓	5		
DARPA E3 THEIA	Benign		1,598,647	2,874,821	17.91
	Attack	✓	25,319		
DARPA E3 CADETS	Benign		1,614,189	3,303,264	18.38
	Attack	✓	12,846		

Bảng 2. Bộ dữ liệu cho phát hiện cấp độ thực thể hệ thống.



(a) Đường cong ROC trên phát hiện cấp độ nhật ký theo lô *(b) Đường cong ROC trên phát hiện cấp độ thực thể hệ thống*

Hình 5. Đường cong ROC trên tất cả các bộ dữ liệu.

6.1.1. Bộ dữ liệu StreamSpot

Bộ dữ liệu StreamSpot (Xem Bảng 1) là một bộ dữ liệu mô phỏng được thu thập và công khai bởi StreamSpot bằng hệ thống kiểm toán SystemTap. Bộ dữ liệu StreamSpot chứa 600 lô nhật ký kiểm toán giám sát các lệnh gọi hệ thống trong 6 kịch bản duy nhất. Năm trong số các kịch bản đó mô phỏng hành vi người dùng lành tính, trong khi kịch bản tấn công mô phỏng một cuộc tấn công drive-by-download. Bộ dữ liệu được coi là một bộ dữ liệu tương đối nhỏ và vì không có nhãn của các mục nhật ký và các thực thể hệ thống nào được cung cấp, Nhóm tác giả thực hiện phát hiện ở cấp độ nhật ký theo lô trên bộ dữ liệu StreamSpot tương tự như các công trình trước đây.

6.1.2. Bộ dữ liệu Unicorn Wget

Bộ dữ liệu Unicorn Wget (Xem Bảng 1) chứa các cuộc tấn công mô phỏng được thiết kế bởi Unicorn. Cụ thể, nó chứa 150 lô nhật ký được thu thập bằng Camflow, trong đó 125 lô lành tính và 25 lô chứa các cuộc tấn công chuỗi cung ứng. Các cuộc tấn công này, được phân loại là các cuộc tấn công lén lút, được thiết kế công phu để có hành vi tương tự như quy trình làm việc lành tính của hệ thống và được cho là khó xác định. Bộ dữ liệu này được coi là khó nhất trong số các bộ dữ liệu thử nghiệm của Nhóm tác giả do dung lượng lớn, định dạng nhật ký phức tạp và tính chất lén lút của các cuộc tấn công này. Tương tự như các phương pháp hiện đại, Nhóm tác giả thực hiện phát hiện ở cấp độ nhật ký theo lô trên bộ dữ liệu này.

6.1.3. Bộ dữ liệu DARPA E3

Bộ dữ liệu DARPA Engagement 3 (Xem Bảng 2), là một phần của chương trình DARPA Transparent Computing, được thu thập từ một mạng doanh nghiệp trong một cuộc đối đầu. Các cuộc tấn công APT khai thác các lỗ hổng khác nhau được thực hiện bởi đội đỏ để đánh cắp thông tin nhạy cảm. Các đội xanh cố gắng xác định các cuộc tấn công này bằng cách kiểm tra các máy chủ mạng và thực hiện phân tích nhân quả trên chúng. Các bộ dữ liệu con Trace, THEIA và CADETS được bao gồm trong đánh giá của Nhóm tác giả. Ba bộ dữ liệu con này bao gồm tổng cộng 51,69GB bản ghi kiểm toán, chứa tới 6.539.677 thực thể hệ thống và 68.127.444 tương tác. Do đó, Nhóm tác giả đánh giá khả năng phát hiện ở cấp độ thực thể hệ thống của **MAGIC** và giải quyết vấn đề chi phí trên các bộ dữ liệu này.

Đối với các bộ dữ liệu khác nhau, Nhóm tác giả sử dụng các cách chia bộ dữ liệu khác nhau để đánh giá mô hình và Nhóm tác giả chỉ sử dụng các mẫu lành tính để huấn luyện. Đối với bộ dữ liệu StreamSpot, Nhóm tác giả chọn ngẫu nhiên 400 lô trong số 500 lô lành tính để huấn luyện và phần còn lại để kiểm tra, dẫn đến một bộ kiểm tra cân bằng. Đối với bộ dữ liệu Unicorn Wget, 100 lô lành tính được chọn để huấn luyện. trong khi phần còn lại dành cho việc kiểm tra. Đối với bộ dữ liệu DARPA E3, Nhóm tác giả sử dụng cùng nhãn ground-truth như ThreaTrace và chia các mục nhật ký theo thứ tự xuất hiện của chúng. 80% các mục nhật ký sớm nhất được sử dụng để huấn luyện, trong khi phần còn lại được giữ lại để kiểm tra. Trong quá trình đánh giá, hiệu suất trung bình của **MAGIC** dưới 100 hạt giống ngẫu nhiên toàn cầu được báo cáo là kết quả cuối cùng, do đó các kết quả thực nghiệm có thể chứa các phần của thực thể hệ thống/lô nhật ký.

6.2. Hiệu quả

Hiệu quả của **MAGIC** trong việc phát hiện APT đa mức độ chi tiết được đánh giá trên ba bộ dữ liệu. Ở đây, Nhóm tác giả trình bày kết quả phát hiện của **MAGIC** trên mỗi bộ dữ liệu, sau đó so sánh nó với các phương pháp phát hiện APT hiện đại trên các bộ dữ liệu đó. Kết quả phát hiện. Các kết quả cho thấy **MAGIC** phát hiện APT thành công với độ chính xác cao trong các kịch bản khác nhau. Nhóm tác giả trình bày kết quả phát hiện của **MAGIC** trên mỗi bộ dữ liệu trong Bảng 3 và các đường cong ROC tương ứng của chúng trong Hình 5.

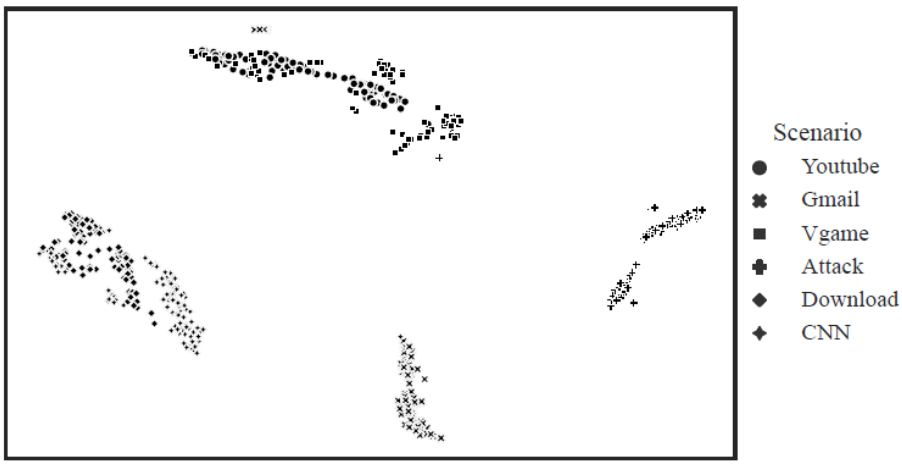
Trên các bộ dữ liệu dễ như bộ dữ liệu StreamSpot, **MAGIC** đạt được kết quả phát hiện gần như hoàn hảo. Điều này là do bộ dữ liệu StreamSpot chỉ thu thập hoạt động của một người dùng duy nhất trên mỗi lô nhật ký, dẫn đến hành vi hệ thống có thể dễ dàng tách biệt với nhau. Nhóm tác giả tiếp tục trình bày hiệu ứng này bằng cách trực quan hóa sự phân bố của các nhúng trạng thái hệ thống được trừu tượng hóa từ các lô nhật ký đó trong Hình 6. Các nhúng trạng thái hệ thống được phân thành 6 loại, khớp với 6 kịch bản liên quan đến bộ dữ liệu. Ngoài ra, điều này cho thấy mô-đun biểu diễn đồ thị của **MAGIC** vượt trội trong việc trừu tượng hóa hành vi hệ thống thành các nhúng như vậy.

Khi xử lý bộ dữ liệu Unicorn Wget, **MAGIC** đạt được trung bình độ chính xác 98,01% và độ phủ 96,00%, thấp hơn đáng kể so với bộ dữ liệu StreamSpot. Phong cách tự giám sát của **MAGIC** khiến nó khó phân biệt giữa các cuộc tấn công lén lút và các hành vi hệ thống lành tính. Tuy nhiên, **MAGIC** vẫn khôi phục thành công trung bình 24 trong số 25 lô nhật ký chỉ với 0,5 dương tính giả được tạo ra, tốt hơn bất kỳ bộ phát hiện hiện đại nào.

Trên các bộ dữ liệu DARPA, **MAGIC** đạt được độ phủ trung bình 99,91% và tỷ lệ dương tính giả 0,15% chỉ với các mục nhật ký lành tính để huấn luyện. Điều này cho thấy **MAGIC** nhanh chóng học cách mô hình hóa hành vi hệ thống. Tập kiểm tra trong kịch bản này không cân bằng, có nghĩa là tổng số thực thể lành tính ground-truth vượt xa số lượng thực thể độc hại. Trong số 1.386.046 thực thể kiểm tra, chỉ có 106.246 được gắn nhãn là độc hại. Tuy nhiên, có rất ít dương tính giả được tạo ra, vì **MAGIC** xác định các thực thể độc hại bằng cách phát hiện ngoại lệ và các thực thể bất thường tự nhiên được phát hiện là bất thường.

Granularity	Dataset		Train Ratio	Ground Truth		#TP	#FP	#TN	#FN	Precision	Recall	FPR	F1-Score	AUC
				#Benign	#Malicious									
Batched log level	StreamSpot		80%	100	100	100.0	0.59	99.41	0.0	99.41%	100.00%	0.59%	99.71%	99.95%
	Unicorn Wget		80%	25	25	24.0	0.5	24.5	1.0	98.02%	96.00%	2.00%	96.98%	96.32%
System entity level	DARPA E3 Trace	All	80%	616,025	68,082	68,072	569	615,456	10	99.17%	99.98%	0.09%	99.57%	99.99%
		Extension Backdoor			732	727			5					
		Pine Backdoor			67,345	67,342			3					
		Phishing Executable			5	3			2					
	DARPA E3 THEIA	All	80%	319,448	25,319	25,318	456	318,992	1	98.23%	99.99%	0.14%	99.11%	99.87%
	DARPA E3 CADETS	All	80%	344,327	12,846	12,816	759	343,568	30	94.40%	99.77%	0.22%	97.01%	99.77%

Bảng 3. Kết quả phát hiện của MAGIC trên các bộ dữ liệu khác nhau. Đối với phát hiện cấp độ nhật ký theo lô, mục tiêu phát hiện là các phần nhật ký. Và đối với phát hiện cấp độ thực thể hệ thống, các thực thể hệ thống là mục tiêu.



Hình 6. Không gian ẩn của các nhúng trạng thái hệ thống trong bộ dữ liệu StreamSpot. Mỗi điểm đại diện cho một đoạn nhật ký trong bộ dữ liệu, thuộc một trong sáu kịch bản: xem YouTube, kiểm tra Gmail, chơi vgame, trải qua một cuộc tấn công drive-by-download, tải xuống các tệp thông thường và xem CNN.

Dataset	Approach	Train Ratio	Supervision	Precision	F1-Score	Recall	FPR
StreamSpot	StreamSpot	80%	B	73%	81%	91%	6.6%
	Unicorn (baseline)	75%	B	95%	96%	93%	1.6%
	Prov-Gem	80%	B,A	100%	97%	94%	**0%**
	ThreaTrace	75%	B	98%	99%	99%	0.4%
	MAGIC (Ours)	80%	B	99%	**99%**	**100%**	0.6%
Unicorn Wget	Unicorn (baseline)	80%	B	86%	90%	95%	15.5%
	Prov-Gem	80%	B,A	100%	89%	80%	**0%**
	ThreaTrace	80%	B	93%	95%	**98%**	7.4%
	MAGIC (Ours)	80%	B	98%	**97%**	96%	2.0%
DARPA E3 Trace	DeepLog	N/A	B,A	41%	51%	68%	2.7%
	Log2vec (baseline)	N/A	B,A	54%	64%	78%	1.8%
	ThreaTrace	N/A	B	72%	83%	99%	1.1%
	ShadeWatcher	80%	B,SA	97%	99%	**99%**	0.3%
	MAGIC (Ours)	80%	B	**99%**	**99%**	99%	**0.1%**
DARPA E3 THEIA	DeepLog	N/A	B,A	16%	15%	14%	0.5%
	Log2vec (baseline)	N/A	B,A	62%	64%	66%	0.3%
	ThreaTrace	N/A	B	87%	93%	99%	**0.1%**
	MAGIC (Ours)	80%	B	**98%**	**99%**	**99%**	0.1%
DARPA E3 CADETS	DeepLog	N/A	B,A	23%	35%	74%	4.4%
	Log2vec (baseline)	N/A	B,A	49%	62%	85%	1.6%
	ThreaTrace	N/A	B	90%	95%	**99%**	0.2%
	MAGIC (Ours)	80%	B	**94%**	**97%**	99%	**0.2%**

Bảng 4. So sánh giữa MAGIC và các phương pháp phát hiện APT hiện đại trên các bộ dữ liệu khác nhau. Trong cột giám sát, B chỉ dữ liệu lành tính, A chỉ dữ liệu tấn công và SA chỉ dữ liệu tấn công trực tuyến.

Trong số các kết quả âm tính giả, Nhóm tác giả nhận thấy hầu hết chúng là các tệp và thư viện độc hại liên quan đến các cuộc tấn công. Điều này cho thấy **MAGIC** vượt trội trong việc phát hiện các quy trình và kết nối mạng độc hại có hành vi khác với các thực thể hệ thống lành tính. Tuy nhiên, **MAGIC** gặp khó khăn trong việc định vị các thực thể thụ động như các tệp và thư viện độc hại, có xu hướng có hành vi tương tự như các thực thể lành tính. May mắn thay, các tệp và thư viện trung gian này có thể dễ dàng được xác định trong quá trình khôi phục câu chuyện tấn công, với các quy trình và kết nối độc hại đã được phát hiện thành công. MAGIC so với các phương pháp hiện đại. Ba bộ dữ liệu được sử dụng để đánh giá MAGIC cũng được sử dụng bởi một số phương pháp phát hiện APT dựa trên học máy hiện đại. Ví dụ, Unicorn, Prov-Gem và ThreaTrace cho bộ dữ liệu Unicorn Wget và StreamSpot, ThreaTrace và ShadeWatcher cho bộ dữ liệu con E3-Trace. Các phương pháp yêu cầu thông tin tiên

nghiệm về APT, chẳng hạn như Holmes, Poirot và Morse, không được xem xét vì MAGIC không thể so sánh với chúng trong cùng một kịch bản phát hiện.

Kết quả so sánh giữa MAGIC và các phương pháp hiện đại khác trên mỗi bộ dữ liệu được trình bày trong Bảng 4. So sánh giữa MAGIC và các phương pháp không giám sát khác (tức là Unicorn và ThreaTrace) cho thấy chiến thắng hoàn toàn của phương pháp của Nhóm tác giả, tiết lộ hiệu quả của MAGIC trong việc mô hình hóa và phát hiện các ngoại lệ của hành vi hệ thống lành tính mà không cần sự giám sát từ các nhật ký chứa tấn công. Ngoài các phương pháp không giám sát, Prov-Gem là một bộ phát hiện APT giám sát dựa trên GAT. Tuy nhiên, nó không đạt được kết quả phát hiện tốt hơn ngay cả trên bộ dữ liệu StreamSpot dễ nhất. Điều này chủ yếu là do các lớp GAT đơn giản được giám sát trên các tác vụ phân loại không biểu cảm bằng các bộ mã hóa tự động đồ thị bị che mặt trong việc tạo ra các nhúng chất lượng cao. Một bộ phát hiện APT khác được đề cập, ShadeWatcher, áp dụng một phương pháp phát hiện bán giám sát. ShadeWatcher tận dụng TransR và các mạng nơ-ron đồ thị (GNN) để phát hiện APT dựa trên đề xuất và có thể đạt được tỷ lệ thu hồi tốt nhất trên bộ dữ liệu con E3-Trace. TransR, một phương pháp biểu diễn đồ thị tự giám sát trên Đồ thị tri thức, đóng góp lớn nhất vào độ chính xác phát hiện của ShadeWatcher. Thật không may, TransR cực kỳ tốn kém về chi phí tính toán. Ví dụ, ShadeWatcher mất tới 12 giờ để huấn luyện mô-đun TransR trên bộ dữ liệu con E3-Trace. Ngược lại, đánh giá về chi phí tính toán của MAGIC (Phần 6.4) cho thấy MAGIC có thể hoàn thành quá trình huấn luyện trên cùng lượng dữ liệu huấn luyện nhanh hơn 51 lần so với ShadeWatcher.

6.3. Phân tích dương tính giả

Đối với các ứng dụng thời gian thực, các bộ phát hiện APT phải cố gắng hết sức để ngăn chặn các báo động giả, vì các báo động giả này thường khiến các nhà phân tích bảo mật mệt mỏi và bối rối. Nhóm tác giả đánh giá tỷ lệ dương tính giả (FPR) của **MAGIC** trên nhật ký kiểm toán lành tính và điều tra cách cơ chế thích ứng mô hình của Nhóm tác giả giảm thiểu các báo động giả này. Bảng 3 cho thấy tỷ lệ dương tính giả của **MAGIC** trên mỗi bộ dữ liệu. Trong mỗi bộ dữ liệu, chỉ các nhật ký lành tính được sử dụng để huấn luyện và kiểm tra. **MAGIC** cho ra FPR thấp (trung bình 0,15%) với dữ liệu huấn luyện lớn. Điều này là do **MAGIC** mô hình hóa hành vi hệ thống lành tính bằng các nhúng tự giám sát, cho phép nó xử lý hiệu quả các thực thể hệ thống chưa từng thấy. FPR thấp như vậy cho phép ứng dụng **MAGIC** trong các thiết lập thực tế. Khi chỉ thực hiện phát hiện chi tiết ở cấp độ thực thể, **MAGIC** chỉ cho ra 569 báo động giả trên bộ dữ liệu Trace, với trung bình chỉ 40 báo động giả mỗi ngày. Các nhà phân tích bảo mật có thể dễ dàng xử lý số lượng báo động này và thực hiện các cuộc điều tra bảo mật trên chúng. Nếu áp dụng phát hiện hai giai đoạn được mô tả trong Phần 3, số lượng báo động giả trung bình mỗi ngày có thể giảm xuống còn 24.

Cơ chế thích ứng mô hình của Nhóm tác giả được thiết kế để giúp **MAGIC** học hỏi từ các hành vi chưa từng thấy mới quan sát được. Nhóm tác giả đánh giá cách cơ chế này giảm thiểu các dương tính giả trên nhật ký kiểm toán lành tính trong Bảng 5. Cụ thể, Nhóm tác giả kiểm tra **MAGIC** trên bộ dữ liệu Trace trong năm cài đặt khác nhau:

- Huấn luyện trên 80% mục nhật ký đầu tiên và kiểm tra trên 20% còn lại mà không có sự thích ứng, giống hệt cài đặt ban đầu của Nhóm tác giả.

- Huấn luyện trên 20% đầu tiên và kiểm tra trên 20% cuối cùng mà không có sự thích ứng, cho mục đích so sánh.

- Huấn luyện trên 20% đầu tiên, thích ứng trên các dương tính giả được tạo ra từ 20% tiếp theo và kiểm tra trên 20% cuối cùng.

- Huấn luyện trên 20% đầu tiên, thích ứng trên cả dương tính giả và âm tính thật được tạo ra từ 20% mục nhật ký tiếp theo và kiểm tra trên 20% cuối cùng.

- Huấn luyện trên 20% đầu tiên, thích ứng trên cả dương tính giả và âm tính thật được tạo ra từ 40% mục nhật ký tiếp theo và kiểm tra trên 20% cuối cùng.

Kết quả thực nghiệm chỉ ra rằng việc thích ứng mô hình với phản hồi lành tính liên tục làm giảm các dương tính giả. Có thể đạt được sự giảm thiểu hơn nữa bằng cách cung cấp cả dương tính giả và âm tính thật cho mô hình. Điều này là do mô-đun biểu diễn đồ thị có thể được huấn luyện lại với bất kỳ dữ liệu nào để nâng cao khả năng biểu diễn của nó, như đã mô tả trong Phần 4.4.

Dataset	Train Ratio	Adaption	Test Ratio	FPR
StreamSpot	80%	N/A	20%	0.59%
Unicorn Wget	80%	N/A	20%	2.00%
DARPA E3 Trace	80%	N/A	20%	0.089%
	20%	N/A	20%	0.426%
	20%	20% FP	20%	0.272%
	20%	20% FP & TN	20%	0.220%
	20%	40% FP & TN	20%	0.173%

Bảng 5. Tỷ lệ dương tính giả của MAGIC trên các bộ dữ liệu khác nhau. Hiệu quả của cơ chế thích ứng mô hình được kiểm tra trong các cài đặt khác nhau.

Phase	Component	Time consumption (s)		Peak Memory consumption (MB)
		with GPU	CPU only	
Graph Construction	N/A	642		2,610
Training	Graph Representation	151	685	1,564
	Detection	78		1,320
Inference	Graph Representation	5	10	2,108
	Detection	825		1,667

Bảng 6. Chi phí hiệu suất của MAGIC trên bộ dữ liệu con E3-Trace.

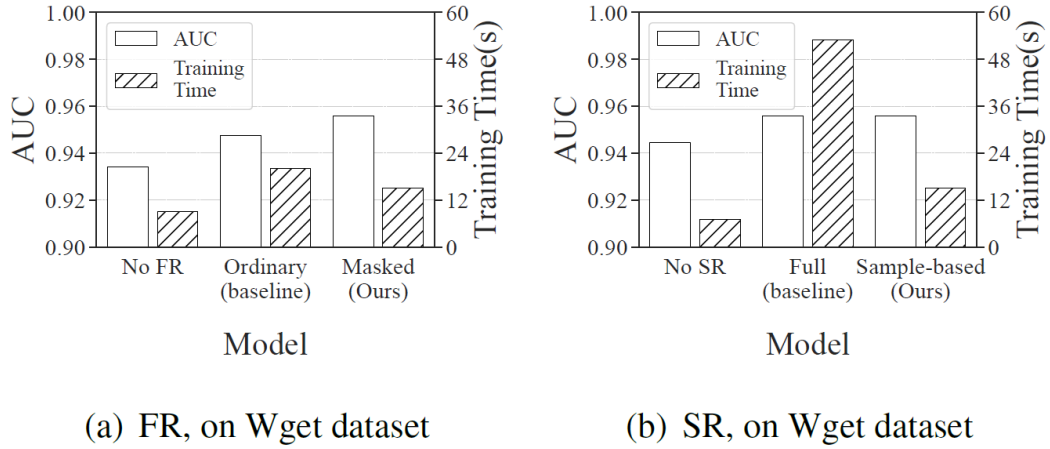
6.4. Chi phí hiệu suất

MAGIC được thiết kế để thực hiện phát hiện APT với chi phí tối thiểu, đảm bảo khả năng ứng dụng của nó trong nhiều điều kiện khác nhau. **MAGIC** hoàn thành quá trình huấn luyện và suy luận trong thời gian logarit và chiếm không gian tuyến tính. Nhóm tác giả cung cấp một phân tích chi tiết về độ phức tạp thời gian và không gian của nó trong Phụ lục B. Nhóm tác giả tiếp tục kiểm tra hiệu suất thời gian chạy của **MAGIC** trên bộ dữ liệu con E3-Trace và trình bày mức tiêu thụ thời gian và bộ nhớ của nó trong Bảng 6.

Trong các thiết lập thực tế, GPU có thể không phải lúc nào cũng có sẵn. Nhóm tác giả cũng kiểm tra hiệu quả của **MAGIC** mà không cần GPU. Chỉ với CPU có sẵn, giai đoạn huấn luyện trở nên chậm hơn rõ rệt. Hiệu quả của giai đoạn xây dựng đồ thị và phát hiện ngoại lệ không bị ảnh hưởng vì chúng được triển khai để chạy trên CPU. Nhóm tác giả cũng đo mức tiêu thụ bộ nhớ tối đa trong quá trình huấn luyện và suy luận. Mức tiêu thụ bộ nhớ thấp của **MAGIC** không chỉ ngăn chặn các vấn đề OOM trên các bộ dữ liệu lớn mà còn giúp **MAGIC** dễ tiếp cận hơn trong các điều kiện hạn chế.

Kết quả đánh giá về chi phí hiệu suất chứng minh tuyên bố rằng **MAGIC** có lợi thế hơn so với các bộ phát hiện APT hiện đại khác về hiệu quả. Ví dụ, ATLAS mất khoảng một giờ để huấn luyện mô hình của nó trên 676MB nhật ký kiểm toán và ShadeWatcher mất 1 ngày để huấn luyện trên bộ dữ liệu con E3-Trace. So với ShadeWatcher, **MAGIC** nhanh hơn 51 lần trong quá trình huấn luyện với cùng cài đặt tỷ lệ huấn luyện (tức là 80% mục nhập nhật ký để huấn luyện trên bộ dữ liệu con E3-Trace).

Các kết quả đánh giá này cũng minh họa rằng **MAGIC** hoàn toàn thực tế trong các điều kiện khác nhau. Xét đến việc bộ dữ liệu con E3-Trace được thu thập trong 2 tuần, mỗi ngày có 1,37GB nhật ký kiểm toán được tạo ra. Điều này có nghĩa là trong điều kiện chỉ có CPU, **MAGIC** chỉ mất 2 phút để phát hiện APT từ các nhật ký đó và hoàn thành việc thích ứng mô hình mỗi ngày. Hiệu quả đầy hứa hẹn như vậy khiến **MAGIC** trở thành một lựa chọn khả dụng cho cá nhân và doanh nghiệp nhỏ. Đối với các doanh nghiệp và tổ chức lớn hơn, họ tạo ra hàng trăm GB nhật ký kiểm toán mỗi ngày. Trong trường hợp này, hiệu quả của **MAGIC** có thể được đảm bảo bằng cách huấn luyện và thích ứng với GPU và song song hóa mô-đun phát hiện với các lõi CPU phân tán.



(a) FR, on Wget dataset (b) SR, on Wget dataset

Hình 7. Ảnh hưởng của các thành phần tái cấu trúc khác nhau đến hiệu suất và hiệu quả của MAGIC.

6.5. Nghiên cứu loại bỏ

Trong phần này, đầu tiên Nhóm tác giả giải quyết hiệu quả của các thành phần riêng lẻ quan trọng trong mô-đun biểu diễn đồ thị của **MAGIC**, sau đó thực hiện phân tích siêu tham số để đánh giá độ nhạy của **MAGIC**. Phân tích về các thành phần riêng lẻ và hầu hết các siêu tham số được thực hiện trên bộ dữ liệu Wget khó nhất, và phân tích siêu tham số về ngưỡng phát hiện θ được thực hiện trên tất cả các bộ dữ liệu. Phân tích thành phần riêng lẻ. Nhóm tác giả nghiên cứu cách tái cấu trúc đặc trưng (FR) cũng như tái cấu trúc cấu trúc (SR) ảnh hưởng đến hiệu suất của **MAGIC**, và Hình 7 trình bày tác động của các thành phần này đến cả kết quả phát hiện và chi phí hiệu suất. Cả FR và SR đều cung cấp sự giám sát cho mô-đun biểu diễn đồ thị của **MAGIC**. So với FR thông thường, FR được che mặt cải thiện nhẹ hiệu suất và giảm đáng kể thời gian huấn luyện. Tuy nhiên, SR dựa trên mẫu là một thành phần giảm độ phức tạp hiệu quả, giúp tăng tốc quá trình huấn luyện mà không làm mất hiệu suất so với SR đầy đủ. Phân tích siêu tham số. Chức năng của **MAGIC** được kiểm soát bởi một số siêu tham số, bao gồm chiều nhúng d, số lớp GAT l, tỷ lệ che mặt nút r và ngưỡng phát hiện ngoại lệ θ. Hình 8 minh họa cách các siêu tham số này ảnh hưởng đến hiệu suất mô hình trong các tình huống khác nhau. Hầu hết các trường hợp, các siêu tham số có ít tác động.

Nói chung, hiệu suất mô hình tương đối cao hơn đạt được với chiều nhúng lớn hơn và nhiều lớp GAT hơn bằng cách thu thập nhiều thông tin hơn từ các vùng lân cận xa hơn, như được hiển thị trong Hình phụ 8(a) và 8(b). Tuy nhiên, việc tăng d hoặc l làm tăng chi phí tính toán, dẫn đến thời gian huấn luyện và suy luận lâu hơn. Tỷ lệ che mặt mặc định là 0,5 cho kết quả tốt nhất. Điều này là do **MAGIC** không thể có đủ quá trình huấn luyện dưới tỷ lệ che mặt thấp. Và dưới tỷ lệ che mặt cao, các đặc trưng nút bị tổn thương nghiêm trọng, ngăn **MAGIC** học các nhúng nút thông qua tái cấu trúc đặc trưng. Việc tăng tỷ lệ che mặt làm tăng nhẹ gánh nặng tính toán.

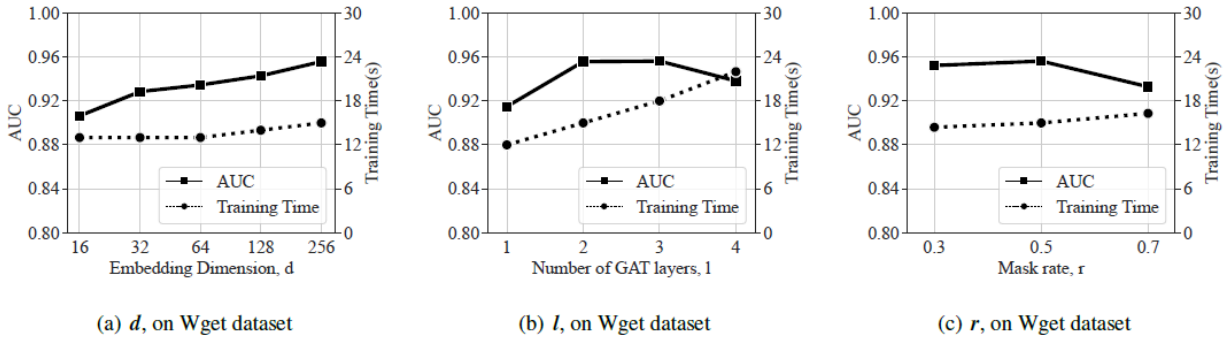
Nhóm tác giả tiếp tục xem xét các điểm bất thường của các thực thể để đánh giá độ nhạy của θ. Một θ thấp hơn tự nhiên dẫn đến hiệu suất thu hồi cao hơn với chi phí là nhiều dương tính giả hơn và ngược lại. Như được minh họa trong Hình 9, hầu hết các thực thể độc hại đều có điểm bất thường cao so với các thực thể lành tính và được tách biệt tốt với chúng

với rất ít sự trùng lặp. Các khoảng cách đáng kể giữa các điểm bất thường lành tính và độc hại hỗ trợ tuyên bố rằng **MAGIC** không phụ thuộc vào một ngưỡng θ chính xác để thực hiện phát hiện chính xác trong các tình huống thực tế. Nhóm tác giả định lượng các khoảng cách này trong Phụ lục D.

Đối với một bộ phát hiện không giám sát như **MAGIC**, các siêu tham số thường khó lựa chọn. Tuy nhiên, điều đó không đúng với **MAGIC**. Nhóm tác giả cũng cung cấp một hướng dẫn chung để lựa chọn siêu tham số trong Phụ lục D.

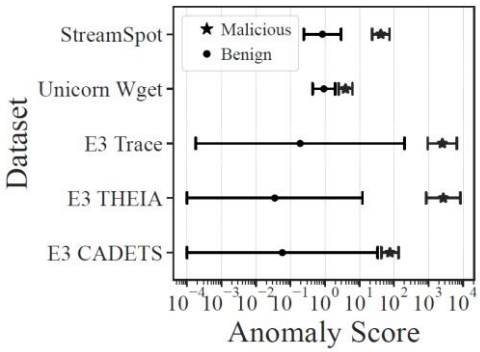
7. Thảo luận và Hạn chế

Chất lượng dữ liệu huấn luyện. **MAGIC** mô hình hóa hành vi hệ thống lành tính và phát hiện APT bằng cách phát hiện ngoại lệ. Tương tự như các phương pháp phát hiện APT dựa trên dị thường khác, Nhóm tác giả giả định rằng tất cả các hành vi hệ thống lành tính cập nhật đều được quan sát trong quá trình thu thập nhật ký huấn luyện. Tuy nhiên, nếu **MAGIC** được huấn luyện với dữ liệu chất lượng thấp không bao phủ đầy đủ hành vi hệ thống, có thể tạo ra nhiều dương tính giả. Phát hiện ngoại lệ. **MAGIC** triển khai một mô-đun phát hiện ngoại lệ dựa trên KNN để phát hiện APT. Mặc dù nó hoàn thành quá trình huấn luyện và suy luận trong thời gian logarit, nhưng hiệu quả của nó trên các bộ dữ liệu lớn vẫn chưa đạt yêu cầu. Để minh họa, mô-đun phát hiện của Nhóm tác giả mất 13,8 phút để kiểm tra 684.111 mục tiêu, chiếm 99% tổng thời gian suy luận. Các phương pháp phát hiện ngoại lệ khác, chẳng hạn như One-class SVM và Isolation Forest, không phù hợp với cài đặt phát hiện của Nhóm tác giả và không thể thích ứng với sự trôi dạt khái niệm. Các phương pháp dựa trên cụm và tìm kiếm KNN gần đúng có thể phù hợp hơn với các bộ dữ liệu khổng lồ. Trong khi đó, các phương pháp dựa trên KNN có thể được mở rộng sang GPU. Nhóm tác giả dành những cải tiến như vậy cho mô-đun phát hiện của mình cho các nghiên cứu trong tương lai. Các cuộc tấn công đối nghịch. Trong Phần 6.2, Nhóm tác giả chỉ ra rằng **MAGIC** xử lý tốt các cuộc tấn công lén lút, tránh bị phát hiện bằng cách hành động tương tự như hệ thống lành tính. Tuy nhiên, nếu kẻ tấn công biết chi tiết cách **MAGIC** hoạt động, họ có thể thực hiện các cuộc tấn công được thiết kế công phu để xâm nhập vào bộ phát hiện của Nhóm tác giả. Nhóm tác giả thực hiện một phân tích đơn giản về các cuộc tấn công đối nghịch và chứng minh tính mạnh mẽ của **MAGIC** chống lại chúng trong Phụ lục A. Khi các phương pháp dựa trên đồ thị ngày càng trở nên phổ biến và mạnh mẽ trong các ứng dụng phát hiện khác nhau, việc thiết kế và tránh các cuộc tấn công đối nghịch này trên cả đồ thị đầu vào và GNN là một chủ đề nghiên cứu thú vị.



(a) ***d***, on Wget dataset (b) ***l***, on Wget dataset (c) ***r***, on Wget dataset

Hình 8. Ảnh hưởng của các siêu tham số khác nhau đến hiệu suất và hiệu quả của MAGIC.



Hình 9. Điểm bất thường của các thực thể hệ thống. Nhóm tác giả loại trừ 5% điểm cao nhất và thấp nhất trên mỗi bộ dữ liệu.

8. Các công trình liên quan

MAGIC chủ yếu liên quan đến ba lĩnh vực nghiên cứu, bao gồm phát hiện APT, học biểu diễn đồ thị và các phương pháp phát hiện ngoại lệ. Phát hiện APT. Mục tiêu của phát hiện APT là phát hiện các tín hiệu APT, các thực thể độc hại và các tương tác không hợp lệ từ nhật ký kiểm toán. Các công trình gần đây chủ yếu dựa trên dòng dõi dữ liệu. Như đã đề xuất bởi, các bộ phát hiện dựa trên dòng dõi có thể được phân loại thành các phương pháp dựa trên quy tắc, dựa trên thống kê và dựa trên học máy. Các phương pháp dựa trên quy tắc [2–6] sử dụng kiến thức tiên nghiệm về các cuộc tấn công đã từng thấy và xây dựng các quy tắc heuristic độc đáo để phát hiện chúng. Các phương pháp dựa trên thống kê [7–9] xây dựng các thống kê để đo lường sự bất thường của các phần tử đồ thị dòng dõi và thực hiện phát hiện dị thường trên chúng. Các phương pháp dựa trên học máy [10–18, 45] tận dụng học sâu để mô hình hóa hành vi hệ thống lành tính hoặc các mẫu tấn công [11, 14–16, 18] và thực hiện phát hiện APT như là phân loại hoặc phát hiện dị thường. Trong số đó, các phương pháp dựa trên chuỗi phát hiện APT dựa trên các mẫu thực thi/quy trình làm việc của hệ thống và các phương pháp dựa trên đồ thị [10, 12, 15–18, 45] mô hình hóa các thực thể và tương tác thông qua GNN và phát hiện các hành vi bất thường như APT. Học biểu diễn đồ thị. Các kỹ thuật nhúng trên đồ thị bắt đầu từ mạng nơ-ron tích chập đồ thị (GCN) và được tăng cường hơn nữa bởi mạng nơ-ron chú ý đồ thị (GAT) và GraphSAGE. Các bộ mã hóa tự động đồ thị [19, 25–27] mang đến các giải pháp không giám sát cho học biểu diễn đồ thị. GAE, VGAE và GATE sử dụng tái cấu trúc đặc trưng và tái cấu trúc cấu trúc để học các nhúng đầu ra theo cách tự giám sát. Tuy nhiên, chúng tập trung vào các tác vụ dự đoán liên kết và phân cụm đồ thị, không

liên quan đến ứng dụng của Nhóm tác giả. Gần đây, các bộ mã hóa tự động đồ thị bị che mặt tận dụng tái cấu trúc đặc trưng bị che mặt và đã đạt được hiệu suất hiện đại trên nhiều ứng dụng khác nhau. Phát hiện ngoại lệ. Các phương pháp phát hiện ngoại lệ xem các ngoại lệ (tức là các đối tượng có thể không thuộc phân phối thông thường) là các dị thường và nhằm mục đích xác định chúng. Các phương pháp phát hiện ngoại lệ truyền thống bao gồm One-class SVM, Isolation Forest và Local Outlier Factor. Các phương pháp truyền thống này được sử dụng rộng rãi trong nhiều kịch bản phát hiện khác nhau, bao gồm phát hiện gian lận thẻ tín dụng và phát hiện giao dịch độc hại. Điều này chứng minh rằng các phương pháp phát hiện ngoại lệ hoạt động tốt trong việc phát hiện dị thường. Trong khi đó, bản thân các bộ mã hóa tự động là các công cụ hiệu quả để phát hiện ngoại lệ. Nhóm tác giả giải thích lý do tại sao Nhóm tác giả không sử dụng các bộ mã hóa tự động đồ thị làm bộ phát hiện dị thường trong Phụ lục F.

9. Kết luận

Nhóm tác giả đã giới thiệu **MAGIC**, một phương pháp phát hiện APT có khả năng ứng dụng phổ quát, hoạt động với hiệu quả tối đa và chi phí thấp. **MAGIC** tận dụng học biểu diễn đồ thị bị che mặt để mô hình hóa hành vi hệ thống lành tính từ nhật ký kiểm toán thô và thực hiện phát hiện APT đa mức độ chi tiết thông qua các phương pháp phát hiện ngoại lệ. Các đánh giá trên ba bộ dữ liệu được sử dụng rộng rãi trong nhiều kịch bản phát hiện khác nhau cho thấy **MAGIC** đạt được kết quả phát hiện đầy hứa hẹn với tỷ lệ dương tính giả thấp và chi phí tính toán tối thiểu.

10. Lời cảm ơn

Nhóm tác giả xin cảm ơn các nhà phản biện ẩn danh và người hướng dẫn của Nhóm tác giả vì những nhận xét chi tiết và có giá trị của họ. Công trình này được tài trợ bởi Quỹ Khoa học Tự nhiên Quốc gia Trung Quốc (số U1936213 và số 62032025), CNKLSTISS, Quỹ Nghiên cứu Cơ bản cho các Trường Đại học Trung ương, Đại học Tôn Dật Tiên (số 22lgqb26) và Chương trình Lãnh đạo Nghiên cứu Học thuật Thượng Hải (số 21XD1421500).