Improving Cross-Platform Binary Analysis using Representation Learning via Graph Alignment.

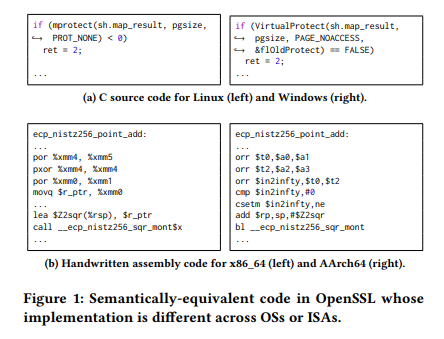
Cải thiện Phân tích Nhị phân Đa nền tảng bằng cách sử dụng Học biểu diễn thông qua Căn chỉnh đồ thị.

TÓM TẮT

* Phân tích Nhị phân Đa nền tảng yêu cầu một biểu diễn chung của các tập tin nhị phân trên các nền tảng khác nhau, dựa vào đó có thể thực hiện phân tích cụ thể. Công việc gần đây đã đề xuất học biểu diễn số học chiều thấp (tức là lời giải) của mã nhị phân sau khi tháo dỡ, và thực hiện phân tích nhị phân trong không gian biểu diễn này. Tuy nhiên, không may, các kỹ thuật hiện có đều không đạt được yêu cầu vì (i) chúng chỉ đối với một nền tảng cụ thể và tạo ra các biểu diễn không được căn chỉnh trên các nền tảng khác nhau, hoặc (ii) không được thiết kế để thu thập thông tin ngữ cảnh phong phú có sẵn trong mã nhị phân đã tháo dỡ.
* Chúng tôi giới thiệu một phương pháp mới dựa trên học sâu mang tên XBA, giải quyết các vấn đề nêu trên. Để làm được điều này, đầu tiên chúng tôi trừu tượng hoá các tập tin nhị phân dưới dạng đồ thị có kiểu, được gọi là đồ thị giải mã nhị phân (BDGs), mã hoá thông tin dòng điều khiển và thông tin ngữ cảnh phong phú khác của các thực thể khác nhau được tìm thấy trong tập tin nhị phân đã tháo dỡ, bao gồm các khối cơ bản, các hàm bên ngoài được gọi và các chuỗi chữ tham chiếu. Sau đó, chúng tôi xây dựng việc học biểu diễn mã nhị phân thành một bài toán căn chỉnh đồ thị, tức là tìm sự tương ứng giữa các nút của BGD được trích xuất từ hai tập tin nhị phân được biên dịch cho các nền tảng khác nhau. XBA sử dụng mạng tích chập đồ thị để học ý nghĩa của mỗi nút, (i) sử dụng thông tin ngữ cảnh phong phú của nó được mã hóa trong BGD và (ii) căn chỉnh các lời giải của nó trên các nền tảng. Bài toán mà chúng tôi đưa ra cho phép XBA học các căn chỉnh ý nghĩa giữa hai BGD một cách bán giám sát, chỉ đòi hỏi một số lượng hạn chế các cặp nút được căn chỉnh trên các nền tảng để huấn luyện. Đánh giá của chúng tôi cho thấy XBA có thể học các lời giải giàu ý nghĩa của các tập tin nhị phân được căn chỉnh trên các nền tảng mà không cần biết trước thông tin cụ thể của nền tảng. Bằng cách chỉ huấn luyện mô hình của chúng tôi với chỉ 50% số căn chỉnh chính xác, XBA đã có thể dự đoán trung bình 75% phần còn lại. Các nghiên cứu trường hợp của chúng tôi cũng cho thấy rằng các lời giải đã học mã hóa thông tin hữu ích cho việc phân tích nhị phân đa nền tảng.

1 GIỚI THIỆU

* Trong những năm gần đây, học sâu đã cho thấy thành công lớn trong nhiều nhiệm vụ phân tích nhị phân [11, 17, 18, 24, 32, 35, 44, 51, 59, 62, 65], vượt trội hơn so với các phương pháp truyền thống về độ chính xác và hiệu suất phân tích. Hầu hết các công trình trước đó nghiên cứu vấn đề học biểu diễn (embedding) của các thực thể khác nhau được tìm thấy trong mã nhị phân [17, 18, 35, 38, 62, 65], chẳng hạn như các chỉ thị [35, 61], các khối cơ bản [18, 65], và các hàm [17, 38]. Biểu diễn số học chiều thấp của các thực thể như vậy được tìm thấy trong mã nhị phân - bất kể chúng được thu thập như thế nào, chẳng hạn như bằng cách huấn luyện mạng neural sâu [17, 18], bằng cách sử dụng mã băm nhạy cảm địa phương [48], v.v. [21] - có thể nâng cao độ chính xác của các nhiệm vụ phân tích nhị phân hậu xử lý: nhận dạng sự tương đồng nhị phân [18], tìm kiếm mã nhân bản hoặc lỗ hổng [17], và các nhiệm vụ phân tích nhị phân ngữ nghĩa khác [11, 44].
* Bài báo này xem xét vấn đề học biểu diễn mã nhị phân có thể hỗ trợ phân tích nhị phân đa nền tảng [8, 20, 48, 59, 65]. Cụ thể, chúng tôi nhắm mục tiêu học mô hình biểu diễn có thể tạo ra các lời giải của tập tin nhị phân, có độ giàu ý nghĩa ngữ nghĩa và căn chỉnh trên các nền tảng khác nhau - trên các hệ điều hành (OS) hoặc kiến trúc bộ chỉ thị (ISAs) khác nhau. Biểu diễn mã nhị phân biên dịch cho các nền tảng khác nhau trong không gian vector thống nhất cho phép thực hiện các nhiệm vụ phân tích nhị phân hậu xử lý theo cách đa nền tảng [48, 59, 65]. Lợi ích chính của phân tích nhị phân đa nền tảng là những nỗ lực phân tích được thực hiện trên một nền tảng có thể dễ dàng được chuyển sang các nền tảng khác; ví dụ, một chữ ký lỗi, ví dụ, chỉ được xác định một lần và tìm kiếm lỗi có thể được thực hiện trên các tập tin nhị phân có sẵn trong các ISA khác nhau [48].



* Rất tiếc, các phương pháp hiện có tạo ra các biểu diễn mã nhị phân không giàu ý nghĩa và không được căn chỉnh trên các nền tảng khác nhau để hỗ trợ phân tích nhị phân đa nền tảng. Hãy xem xét hai cặp mã nguồn ví dụ được hiển thị trong Hình 1, trong đó mỗi cặp có cùng ngữ nghĩa, nhưng ở dạng cụ thể của một hệ điều hành (OS) hoặc kiến trúc bộ chỉ thị (ISA) khác nhau. Đối với hai cặp này, các phương pháp biểu diễn tiên tiến nhất [18, 65] không tạo ra các biểu diễn được căn chỉnh tốt trên các nền tảng khác nhau. Các phương pháp học không giám sát hiện có chỉ huấn luyện mô hình của họ cho một nền tảng duy nhất [17, 18], không thể tạo ra các biểu diễn được căn chỉnh trên các nền tảng. Trong khi đó, các phương pháp học có giám sát hiện có [59, 65] lại huấn luyện mô hình của họ trực tiếp tối ưu hóa cho sự căn chỉnh giữa các khối cơ bản [65] và các hàm [59] trên các nền tảng khác nhau. Là tập dữ liệu huấn luyện được sử dụng để giám sát quá trình huấn luyện mô hình của họ, họ sử dụng các biến thể mã nhị phân chạy trên nhiều nền tảng được biên dịch từ cùng mã nguồn. Tuy nhiên, họ chỉ sử dụng các đặc trưng nội bộ của từng khối cơ bản [65] hoặc từng hàm [59], và không sử dụng thông tin ngữ cảnh phong phú của chúng (ví dụ: chuyển dòng điều khiển nội hoặc ngoại hoặc tham chiếu đến chuỗi chữ) có sẵn trong mã nhị phân đã tháo dỡ.
* Bỏ qua thông tin ngữ cảnh này có thể vô tình dẫn đến việc tạo ra các mô hình có khả năng ý thức hạn chế trong cài đặt đa nền tảng, xem xét cách các phương pháp giám sát hiện có [65] xây dựng các tập dữ liệu huấn luyện được đánh nhãn của họ. Do đó, chúng tôi đề xuất một phương pháp tiếp cận bán giám sát dựa trên đồ thị để học mô hình biểu diễn, giúp tạo ra các biểu diễn mã nhị phân có ý nghĩa ngữ nghĩa và được căn chỉnh trên các nền tảng khác nhau - trên các ISA hoặc OSs. Kỹ thuật chính của chúng tôi là căn chỉnh cấp đồ thị được thực hiện trên mã nhị phân theo cấu trúc đồ thị trên các nền tảng: Một cặp tập tin nhị phân biên dịch cho các nền tảng khác nhau được trừu tượng thành các đồ thị có kiểu, gọi là đồ thị giải mã nhị phân (BDGs), mã hoá thông tin ngữ cảnh phong phú. Sau đó, hai mạng neural đồ thị chia sẻ tham số của chúng được huấn luyện, tối ưu trực tiếp cho sự căn chỉnh giữa hai đồ thị đó, sử dụng (i) thông tin căn chỉnh giữa các nền tảng một phần (do đó, hình thành một quá trình học bán giám sát), và (ii) thông tin ngữ cảnh phong phú được mã hóa trong mỗi BGD. Đặc biệt, chúng tôi sử dụng mạng neural đồ thị (GCNs) [6], lấy cảm hứng từ kiến trúc được chứng minh gần đây có hiệu quả trong việc học biểu diễn tri thức đa ngôn ngữ [56] trong lĩnh vực xử lý ngôn ngữ tự nhiên. Phương pháp căn chỉnh cấp đồ thị của chúng tôi cho phép XBA dự đoán các căn chỉnh tiềm năng giữa các thực thể không được đánh nhãn hoặc thậm chí là các thực thể thuộc các phần mềm hoàn toàn khác nhau. Một mô hình được huấn luyện bằng phương pháp của chúng tôi có thể tạo ra các biểu diễn giàu ý nghĩa hơn, được căn chỉnh tốt hơn trên các nền tảng hơn các phương pháp giám sát hiện có [59, 65], bởi vì họ không sử dụng thông tin ngữ cảnh phong phú được mã hóa trong BGD như phương pháp của chúng tôi.
* Để học tri thức ý nghĩa mã nhị phân đa nền tảng ngoài những gì có thể được trích xuất từ trình biên dịch, chúng tôi xây dựng các tập dữ liệu độc đáo bao gồm một bộ chương trình mã nguồn mở cấp thấp và các thư viện được viết để có khả năng di động trên nhiều ISA và OSs, chẳng hạn như GLIBC và OpenSSL. Phương pháp của chúng tôi hiệu quả trích xuất tri thức cụ thể cho mỗi nền tảng được nhúng trong các phần mềm mã nguồn mở đa nền tảng như vậy, chẳng hạn như mã assembly được viết bởi con người cho từng ISA và mã nguồn cụ thể của từng hệ điều hành, và học mô hình biểu diễn có thể hỗ trợ phân tích nhị phân đa nền tảng. Đánh giá của chúng tôi cho thấy các biểu diễn giàu ý nghĩa của mã nhị phân được căn chỉnh trên các nền tảng có thể được học mà không cần bất kỳ thông tin cụ thể của nền tảng nào hoặc nỗ lực đánh nhãn thủ công. Bằng cách chỉ huấn luyện mô hình của chúng tôi chỉ với 50% số căn chỉnh chính xác giữa các nền tảng, chúng tôi đã có thể dự đoán trung bình 75% (hoặc 74% tùy thuộc vào hướng dự đoán) số căn chỉnh còn lại. Các nghiên cứu trường hợp của chúng tôi cho thấy rằng các biểu diễn học thông qua căn chỉnh BGD đa nền tảng mã hóa tri thức hữu ích cho phân tích nhị phân đa nền tảng.

Các đóng góp của chúng tôi bao gồm:

* Chúng tôi đề xuất một phương pháp học biểu diễn mới có thể hỗ trợ phân tích nhị phân đa nền tảng. Chúng tôi sắp xếp lại nhiệm vụ này thành một bài toán căn chỉnh đồ thị; nghĩa là chúng tôi huấn luyện một mô hình nhúng mã nhị phân với mục tiêu căn chỉnh đồ thị.
* Chúng tôi giới thiệu một phương pháp mới dựa trên học sâu mang tên XBA, giúp học biểu diễn mã nhị phân bằng mục tiêu căn chỉnh đồ thị. Chúng tôi phát triển một định dạng đại diện cho mã nhị phân đã tháo dỡ dưới dạng đồ thị và sử dụng mạng tích chập đồ thị để thu thập thông tin ý nghĩa phong phú từ đồ thị.
* Để đánh giá XBA, chúng tôi giới thiệu các tập dữ liệu kiểm tra cho phân tích nhị phân đa nền tảng. Chúng tôi chỉ ra rằng XBA cải thiện các phương pháp cơ sở một cách đáng kể, và XBA có thể nắm bắt ngữ nghĩa của mã nhị phân từ một tập con nhỏ của các mẫu được đánh nhãn.

2 NỀN TẢNG VÀ ĐỘNG LỰC

2.1 Học biểu diễn mã nhị phân

* Công trình gần đây đã cho thấy rằng học biểu diễn vượt trội hơn so với các phương pháp truyền thống dựa trên heuristic trong nhiều nhiệm vụ phân tích nhị phân, chẳng hạn như phát hiện sự tương đồng nhị phân [18, 62, 65], dự đoán tên hàm [14] và xác định ranh giới hàm [4, 44, 51]. Chìa khóa thành công trong việc này là học các biểu diễn phân tán của mã nhị phân - ở cấp độ của các chỉ thị, khối cơ bản hoặc hàm - trong không gian vector, còn được gọi là nhúng, có thể được sử dụng cho các nhiệm vụ hậu xử lý khác nhau. Công trình trước đó tập trung vào xây dựng các mô hình được tiền huấn luyện có thể tạo ra các nhúng như vậy, và họ có thể điều chỉnh mô hình đã tiền huấn luyện đó hoặc sử dụng trực tiếp các nhúng thu được cho các nhiệm vụ hậu xử lý. Chúng tôi phân loại công trình trước đó thành hai phương pháp dựa trên cách học nhúng.
* Học Biểu Diễn Không Giám Sát: Dòng công việc về tạo nhúng cho mã nhị phân sử dụng một phương pháp học không giám sát. Các phương pháp không giám sát, theo định nghĩa, có lợi thế không đòi hỏi bất kỳ dữ liệu huấn luyện có nhãn nào; nó chỉ yêu cầu các mã nhị phân được tháo dỡ (tức là có thể xác định được các chỉ thị và ranh giới hàm, và có thể xây dựng đồ thị dòng điều khiển nội hoặc ngoại hoặc của cùng một chương trình). DeepBinDiff [18] và Asm2Vec [17] là các nghiên cứu tiêu biểu trong loại học không giám sát này. DeepBinDiff sử dụng Word2Vec [42] để nhúng các chỉ thị và các khối cơ bản, và tiếp tục điều chỉnh các nhúng của khối cơ bản với Text-Associated DeepWalk (TADW) [60] để mã hóa vị trí ngữ cảnh của khối cơ bản trong đồ thị dòng điều khiển nội chương trình trên cấp toàn bộ chương trình. Asm2Vec [17], trong khi đó, thực hiện các bước đi ngẫu nhiên trên đồ thị dòng điều khiển nội của mỗi hàm. Các chuỗi chỉ thị được tạo ra từ các bước đi ngẫu nhiên được sử dụng để huấn luyện mô hình nhúng mã nhị phân cấp hàm dựa trên mô hình PV-DM [33].
* Học Biểu Diễn Có Giám Sát: Một dòng công việc riêng biệt sử dụng học biểu diễn có giám sát. Ý tưởng chính ở đây là tạo ra các nhúng tốt cho một nhiệm vụ dự đoán cụ thể, chẳng hạn như phát hiện sự tương đồng nhị phân. Đầu tiên được đề xuất bởi Xu et al. [59], hầu hết các công trình trước đó huấn luyện một kiến trúc Siamese [10] trên các cặp được đánh nhãn của mã nhị phân. Xu et al. (Gemini) biểu diễn mỗi hàm dưới dạng đồ thị dòng điều khiển được mở rộng (ACFG) và sử dụng chúng làm đầu vào cho mô hình Structure2Vec [13]. Hai mô hình Structure2Vec giống nhau chia sẻ tham số xây dựng mạng Siamese, và mạng được huấn luyện trên các cặp ACFGs và nhãn tương đồng của chúng. Họ cuối cùng sử dụng các đầu ra của mô hình Structure2Vec đã được huấn luyện làm nhúng cho các hàm. Zuo et al. [65] (InnerEye), thay vào đó, biểu diễn mỗi khối cơ bản dưới dạng một chuỗi các thông số chỉ thị và sử dụng chúng làm đầu vào cho mô hình Neural Machine Translation (NMT) để học nhúng mã nhị phân ở cấp độ của các khối cơ bản. Một mạng Siamese được tạo thành từ hai mô hình NMT giống nhau và được huấn luyện trên các cặp được đánh nhãn của các khối cơ bản. Bằng cách sử dụng dữ liệu được đánh nhãn tương đồng theo cặp, các phương pháp có giám sát học nhúng trong đó các nhúng của các đoạn mã nhị phân tương tự sẽ gần nhau và các đoạn mã không tương tự sẽ cách xa nhau.

2.2 Phân Tích Nhị Phân Đa Nền Tảng

* Ngày nay, nhiều phần mềm được viết để hỗ trợ nhiều hệ điều hành (OSs) và kiến trúc tập lệnh (ISAs). Phần mềm có thể di động trên các hệ điều hành có thể chứa các thực hiện rời rạc cho cùng một chức năng cấp cao, ví dụ như tuân thủ một giao diện lập trình ứng dụng (API) cụ thể cho hệ điều hành. Tùy thuộc vào phần cứng được sử dụng để triển khai, các mã nhị phân cũng có thể được biên dịch cho nhiều kiến trúc tập lệnh; sự đa dạng này đặc biệt rõ ràng trong các firmware có sẵn trong nhiều kiến trúc tập lệnh [12], ví dụ như ARMv7, ARMv8, MIPS, v.v.
* Xử lý sự đa dạng này trong phân tích nhị phân là một nhiệm vụ khó khăn, do đó đã thu hút nhiều sự quan tâm trong việc tiến hành phân tích nhị phân theo cách độc lập với nền tảng. Các nghiên cứu trước đó đã đề xuất các kỹ thuật phân tích nhị phân đa nền tảng độc lập với nền tảng, được thiết kế cho việc tìm kiếm lỗ hổng [48, 49], phát triển khai thác mã code tái sử dụng [31], phát hiện sự tương đồng nhị phân [59, 65], hoặc tìm kiếm mã nhị phân nói chung [8]. Lợi ích chính của phân tích nhị phân đa nền tảng là những nỗ lực phân tích được thực hiện trên một nền tảng có thể dễ dàng được chuyển sang các nền tảng khác; ví dụ, chữ ký lỗi chỉ định được định nghĩa một lần duy nhất và việc tìm kiếm lỗi có thể được thực hiện cho các mã nhị phân có sẵn trong các ISA khác nhau [48]. Một dòng công trình trước đó [8, 19, 45, 48, 52] giải quyết vấn đề này bằng cách nâng các mã nhị phân lên thành các biểu diễn độc lập với hệ điều hành hoặc kiến trúc tập lệnh. Trái lại, Gemini [59] và InnerEye [65], các phương pháp học biểu diễn có giám sát được mô tả trong phần 2.1, sử dụng các mô hình Siamese để trực tiếp học sự tương đồng giữa mã nhị phân biên dịch cho các nền tảng khác nhau.

2.3 Nhúng cho Phân Tích Đa Nền Tảng.

* Một mô hình nhúng hiểu quan hệ ngữ nghĩa giữa các mã nhị phân biên dịch cho các nền tảng khác nhau sẽ tạo ra các vector nhúng phân biệt, trong đó (i) các cặp mã nhị phân có cùng ý nghĩa trên các nền tảng sẽ gần nhau, mặc dù chúng có vẻ khác nhau trên các nền tảng, và (ii) các cặp không có cùng ý nghĩa sẽ xa nhau.
* Phương pháp Học không giám sát so với Học có giám sát. Phương pháp học không giám sát có lợi thế không đòi hỏi bất kỳ dữ liệu đánh nhãn nào, nhưng có thể không tạo ra các nhúng nhúng bắt được mối quan hệ ngữ nghĩa giữa mã nhị phân trên các nền tảng. Các phương pháp nhúng không giám sát hiện có được mô tả trong phần 2.1 dựa hoàn toàn vào các đặc trưng có sẵn trong một nền tảng duy nhất [17, 18] trong quá trình huấn luyện, mà không tích hợp bất kỳ thông tin căn chỉnh ngữ nghĩa đa nền tảng nào. Điều này không phải lựa chọn lý tưởng, bởi vì, như chúng tôi sẽ giải thích bên dưới, một tập dữ liệu có nhãn, có thể cung cấp hướng dẫn hữu ích trong quá trình huấn luyện, có thể được xây dựng một cách hệ thống bằng cách sử dụng các trình biên dịch.
* Một cách tiếp cận học có giám sát được sử dụng trong các nghiên cứu trước đây về nhúng mã code qua các kiến trúc tập lệnh [59, 65] đã được chứng minh hiệu quả trong việc huấn luyện mô hình nhúng phân biệt. Phương pháp này yêu cầu một tập dữ liệu huấn luyện có nhãn, trong đó các cặp nhãn mức bộ tự tương đồng. Các nghiên cứu trước đây đã xây dựng một tập dữ liệu huấn luyện có nhãn tương tự như nhau thông qua nhiều lời gọi trình biên dịch [59, 65]. Ý tưởng là trước tiên biên dịch cùng một dòng mã nguồn viết bằng một ngôn ngữ lập trình độc lập với nền tảng như C/C++ hai lần cho hai nền tảng khác nhau, và ánh xạ mã nhị phân biên dịch trở lại vị trí của chúng trong mã nguồn gốc (hoặc trong biểu diễn trung gian độc lập với nền tảng được tạo ra trong quá trình biên dịch [65]). Các cặp mã nhị phân được ánh xạ vào cùng một vị trí tạo thành các cặp mã nhị phân có cùng ý nghĩa trên các nền tảng. Các cặp có ý nghĩa khác nhau, một cách khác, được tạo ra thông qua việc chọn ngẫu nhiên cặp mã nhị phân. Tổng hợp các cặp này tạo thành một tập dữ liệu huấn luyện có nhãn có thể được sử dụng để giám sát học với một kiến trúc Siamese.

A table with numbers and text

Description automatically generated

A table with numbers and letters

Description automatically generated

* Hạn chế của Các Phương pháp Giám sát Hiện có. Các phương pháp học có giám sát, chẳng hạn như việc huấn luyện một mô hình nhúng sử dụng kiến trúc Siamese, có thể căn chỉnh các cặp nhãn, nhưng các phương pháp hiện có chỉ xem xét việc ghép các khối cơ bản và không thể mã hóa thông tin ngữ cảnh [59, 65]. Việc xây dựng các mã nhị phân có ý nghĩa đồng nghĩa trên nhiều nền tảng mà có thể được thực hiện bằng nhiều lời gọi trình biên dịch trên cùng một mã nguồn cho các nền tảng khác nhau có giới hạn vì các lý do sau đây.
* Thứ nhất, như được minh họa trong Hình 1, các phiên bản của cùng một chức năng có thể không giống nhau trên các nền tảng. Mã nguồn cụ thể cho từng nền tảng không thể biên dịch cho các nền tảng khác nhau, do đó không thể đồng nghĩa về mặt ngữ nghĩa giữa các nền tảng mà không cần cố gắng thủ công.
* Thứ hai, trình biên dịch thông thường chỉ tạo ra một phần của tất cả các chỉ thị có sẵn trong một kiến trúc tập lệnh (ISA). Một ISA có thể hỗ trợ quá nhiều chỉ thị (ví dụ: hàng nghìn chỉ thị trong Intel x86\_64), bao gồm cả các chỉ thị không còn được sử dụng cho tương thích ngược (backward compatibility) và / hoặc các chỉ thị được xây dựng cho mục đích cụ thể (ví dụ: chỉ thị AES cho Intel x86\_64 [27] và ARM [3]) thường được sử dụng trong mã assembly viết tay cho mỗi ISA. Mã lắp ráp viết tay, hiển nhiên, không thể biên dịch cho các ISA khác nhau.
* Cuối cùng nhưng không kém phần quan trọng, thông tin ánh xạ từ mã nhị phân đến mã nguồn có thể không hoàn chỉnh. Không đảm bảo rằng trình biên dịch luôn giữ vị trí chính xác của một đoạn mã nhị phân trong mã nguồn gốc [16, 25, 36].

A screenshot of a computer

Description automatically generated

* Bảng 1 thể hiện số lượng các khối cơ bản không được căn chỉnh khi các mã nhị phân được biên dịch cho các nền tảng khác nhau. Tùy thuộc vào phần mềm, khoảng 3% đến 40% số khối cơ bản không được căn chỉnh, ngay cả khi chúng ta biên dịch cùng một phần mềm cho các hệ điều hành (OSs) / kiến trúc tập lệnh (ISAs) khác nhau. Một cái nhìn tỉ mỉ hơn vào các khối cơ bản không được căn chỉnh này làm cho vấn đề rõ ràng hơn. Như được thể hiện trong Bảng 2, không may, có nhiều mã ghi nhớ chỉ thị mà chỉ xuất hiện trong các khối cơ bản không được căn chỉnh này. Điều này phần là do mã cụ thể cho từng nền tảng, chẳng hạn như các hàm lắp ráp viết tay hoặc các hàm API cụ thể cho hệ điều hành, không thể biên dịch cho các nền tảng khác nhau. Các phương pháp học giám sát hiện có [65], vì không tích hợp thông tin ngữ cảnh phong phú xung quanh những khối cơ bản không được căn chỉnh này trong quá trình huấn luyện mô hình của họ, không thể bắt được ngữ nghĩa của chúng.

3 XBA: PHƯƠNG PHÁP ĐỀ XUẤT CỦA CHÚNG TÔI

* Chúng tôi giới thiệu XBA, một phương pháp mới dựa trên học sâu giải quyết những hạn chế của các nghiên cứu trước đây, từ đó tạo điều kiện cho phân tích nhị phân đa nền tảng. Chúng tôi đầu tiên thảo luận về những lựa chọn thiết kế của chúng tôi trong Bảng 3, so sánh với thiết kế của các phương pháp trước đó.
* Định dạng đầu vào: Chúng tôi phát triển một định dạng chứa thông tin ngữ cảnh phong phú của các khối cơ bản. Công việc trước đó xem xét một khối cơ bản chỉ là một chuỗi các chỉ thị mà không xem xét ngữ cảnh cục bộ của nó [65], hoặc là một nút đặt trong biểu đồ dòng điều khiển chương trình liên tục (iCFG) của một mã nhị phân cụ thể [18]. Để mã hóa ngữ cảnh phong phú của các khối cơ bản vượt xa dòng điều khiển chương trình, chúng tôi đề xuất một định dạng mới, đồ thị disassembly nhị phân (BDG), để biểu diễn các mã nhị phân (Phần 3.1).
* Phương pháp: Chúng tôi chọn một phương pháp có thể tận dụng thông tin ngữ cảnh phong phú được mã hóa trong định dạng đầu vào của chúng tôi. Duan et al. [18] sử dụng DeepWalk [47], nhưng phương pháp này chỉ cung cấp thông tin về dòng điều khiển được khám phá bằng cách lấy ngẫu nhiên các bước đi qua một iCFG. XBA, sử dụng các mạng gấp đồ thị (GCN), được biết đến để bắt lấy thông tin ngữ cảnh được mã hóa trong các đồ thị (BDGs).
* Huấn luyện: Chúng tôi sử dụng một mục tiêu căn chỉnh đồ thị, một phương pháp học bán giám sát, để huấn luyện các mạng GCN của chúng tôi. Các phương pháp không giám sát [18] chỉ dựa vào thông tin cấu trúc có sẵn trong một nền tảng duy nhất; chúng không thể căn chỉnh mã nhị phân qua các nền tảng. Trong khi học có giám sát [65] có thể căn chỉnh các mã nhị phân đó qua các nền tảng, phương pháp này không thể hoạt động mà không có các nhãn căn chỉnh. Do đó, chúng tôi áp dụng một phương pháp lai, căn chỉnh các mã nhị phân qua các nền tảng ở mức đồ thị, chỉ đòi hỏi một tập con nhỏ các căn chỉnh được nhãn.

A diagram of a complex network

Description automatically generated

* Tổng quan. Hình 2 mô tả tổng quan của XBA. Ở phía bên trái, chúng ta thể hiện một ví dụ về mã nguồn có cùng ý nghĩa được triển khai cho Linux và Windows trong cùng một phần mềm, OpenSSL, được viết để có thể di động trên các hệ điều hành. XBA nhận hai mã nhị phân được biên dịch cho các nền tảng khác nhau và học (hoặc suy ra) sự tương đồng giữa chúng. Chúng ta trước tiên disassemble các mã nhị phân đó và xây dựng các đồ thị disassembly nhị phân (BDG). Các đồ thị này chứa các khối cơ bản, các hàm ngoại vi, các chuỗi và các mối quan hệ giữa chúng. Sau đó, chúng ta xử lý các BDG này bằng các mạng gấp đồ thị (GCNs) và tạo ra các embedding cho các thực thể (ví dụ: các khối cơ bản, các hàm ngoại vi hoặc các chuỗi). Trong quá trình huấn luyện, chúng ta đảm bảo rằng các GCN tạo ra các embedding cho các thực thể được căn chỉnh gần nhau và các thực thể không được căn chỉnh (được tạo nhân tạo thông qua mẫu) được tách xa. Khi được huấn luyện, các GCN của chúng ta tạo ra các embedding cho các thực thể trong BDG (chứa thông tin ngữ cảnh phong phú), hữu ích cho các tác vụ phụ thuộc đa dạng.

3.1 Đồ thị phân tách nhị phân

* Phân tích nhị phân trên một mức trừu tượng là phân tích một chuỗi các mã thông báo nhận được từ một mã nhị phân đã được disassemble. Trong phân tích đa nền tảng, chúng ta gặp nhiều mã thông báo cụ thể cho từng nền tảng, và mối quan hệ ngữ nghĩa của chúng không dễ dàng xác định qua các nền tảng khác nhau. Ví dụ, không dễ dàng xác định rằng "read" trong một mã nhị phân Linux và "ReadFile" trong một mã nhị phân Windows có ý nghĩa tương tự nhau mà không có kiến thức cụ thể về các nền tảng này. Để giải quyết vấn đề này, chúng tôi trừu tượng một mã nhị phân đã disassemble thành một định dạng mới, đồ thị phân tách nhị phân (BDG), mã hóa thông tin ngữ cảnh phong phú từ đó chúng ta có thể rút ra ngữ nghĩa của các mã thông báo cụ thể cho từng nền tảng. Một BDG là một đồ thị có loại được gán cho các mối quan hệ của các mã thông báo cụ thể trong một mã nhị phân đã disassemble:
* Các nút. Một nút trong BDG có thể thuộc một trong các loại sau: 1 khối cơ bản, 2 hàm ngoại vi và 3 chuỗi chữ. Ở đây, chúng tôi chọn độ chi tiết của các khối cơ bản để trừu tượng hóa các chỉ thị được tìm thấy trong các phần mã của một mã nhị phân. Một lựa chọn khác là sử dụng độ chi tiết của các hàm [17, 21, 38, 59]. Tuy nhiên, công việc gần đây [18, 35, 48, 65] chứng minh rằng các biểu diễn học được ở độ chi tiết mịn hơn có thể hữu ích rộng rãi hơn so với các biểu diễn học được ở độ chi tiết thô hơn. Các biểu diễn học được ở độ chi tiết mịn hơn (tức là các embedding của khối cơ bản) có thể được sử dụng để huấn luyện các biểu diễn học được ở độ chi tiết thô hơn (tức là các embedding của hàm), theo một cách tương tự như học các embedding của câu và đoạn văn với các embedding của từ [33], hoặc học các embedding của siêu đồ thị và đồ thị với các embedding của nút đồ thị [2, 43].

A table with text and symbols

Description automatically generated

* Chúng tôi xác định các thuộc tính nút như sau: 1 Đối với mỗi khối cơ bản, chúng tôi sử dụng các chuỗi chỉ thị cho thuộc tính nút. Cụ thể hơn, mã ngữ và các toán hạng của mỗi chỉ thị được tìm thấy trong một khối cơ bản cụ thể được chuẩn hóa theo quy tắc được đề ra trong Bảng 4, và chuỗi các mã thông báo đã được chuẩn hóa được sử dụng làm thuộc tính nút của khối cơ bản. 2 Chúng tôi sử dụng tên ký hiệu của chúng làm thuộc tính nút cho các hàm ngoại vi, các chuỗi chỉ thị của chúng tồn tại bên ngoài một mã nhị phân đã disassemble. 3 Chúng tôi sử dụng các chuỗi chữ thông thường được tìm thấy trong một mã nhị phân đã disassemble làm thuộc tính nút.
* Mối quan hệ. Có bốn loại mối quan hệ có thể được xác định giữa các nút của BDG như sau: 1 Một truyền dẫn dòng điều khiển trong nội thư hướng, được xác định giữa hai khối cơ bản, chỉ ra rằng điều khiển chương trình chuyển từ một khối cơ bản này sang khối cơ bản khác. 2 Cuộc gọi trực tiếp (tức là các truyền dẫn dòng điều khiển giữa các hàm), được xác định giữa một khối cơ bản chứa một hoặc nhiều điểm gọi và các hàm được gọi. Chúng tôi cũng xác định 3 các tham chiếu mã-ngôn-ngữ lấy địa chỉ mã-ngôn-ngữ, khi một khối cơ bản chứa một hoặc nhiều chỉ thị lấy địa chỉ của một hàm khác. Cuối cùng, 4 các tham chiếu mã-chuỗi được xác định giữa một khối cơ bản và một nút chuỗi, khi khối cơ bản chứa một chỉ thị tham chiếu đến chuỗi chữ.

3.2 Phân tích mã nhị phân đa nền tảng và Vấn đề canh chỉnh đồ thị

* Bây giờ chúng tôi trình bày cách phân tích mã nhị phân đa nền tảng có thể được hình thành thành một vấn đề canh chỉnh đồ thị. Trong phân tích mã nhị phân đa nền tảng, thách thức chính là xác định các khối cơ bản có chức năng tương tự (hoặc không tương tự) với thông tin không đầy đủ. Ví dụ, chúng ta không thể giả định rằng hai hàm ngoại vi cụ thể của hệ điều hành, ví dụ, read trong Linux và ReadFile trong Windows thực hiện cùng một hoạt động. Để ước tính chức năng của chúng, chúng ta cần thông tin ngữ cảnh, chẳng hạn như cùng một khối cơ bản gọi các hàm đó hoặc cùng một đối số được truyền vào chúng. Trong ví dụ trên, nếu cả hai hàm đều nhận .sqliterc là đối số, chúng ta có thể ước tính rằng chúng có cùng mục đích, tức là tải tệp cấu hình SQLite.
* Vấn đề Canh chỉnh đồ thị. Ví dụ trên làm rõ rằng phân tích mã nhị phân đa nền tảng thành công đòi hỏi nắm bắt sự tương đồng (hoặc sự không tương đồng) giữa các thực thể (ví dụ, các khối cơ bản) được tìm thấy trong một mã nhị phân bằng thông tin hàng xóm của chúng. Chúng ta có thể, vì vậy, định hình nhiệm vụ này thành một vấn đề canh chỉnh đồ thị [9, 34, 54, 56]. Trên các BDG được xây dựng từ các mã nhị phân được biên dịch cho các nền tảng khác nhau, chúng ta nhằm tìm sự tương ứng giữa các nút trong BDG dựa trên một tập con nhỏ của các cặp nút đã được canh chỉnh trước đó. Cụ thể, chúng ta có thể viết lại nhiệm vụ canh chỉnh này như sau:
* Chúng tôi ký hiệu một BDG là 𝐺 = (𝐸, 𝑅, 𝐴, 𝑇 𝑅, 𝑇 𝐴), trong đó 𝐸, 𝑅, 𝐴 là các tập hợp của các thực thể, mối quan hệ và thuộc tính, tương ứng. 𝑇 𝑅 ⊂ 𝐸 × 𝑅 × 𝐸 là tập các ba từ liên quan,𝑇 𝐴 ⊂ 𝐸×𝐴×𝑉 là tập các ba từ thuộc tính và 𝑉 đại diện cho tập giá trị thuộc tính. Giả sử rằng hai mã nhị phân được biên dịch từ cùng một nguồn mã cho các nền tảng khác nhau 𝑃1 và 𝑃2. Chúng ta trước tiên xây dựng BDG của chúng 𝐺𝑃1 và 𝐺𝑃2, trong đó 𝐺𝑃𝑖 = (𝐸𝑖, 𝑅𝑖, 𝐴𝑖,𝑇 𝑅 𝑖,𝑇 𝐴 𝑖) cho 𝑖 ∈ {1, 2}. ***aaaaaaaaaaaaaaaaaaaa*** Giữa 𝐺𝑃1 và 𝐺𝑃2, có𝑚 nút được canh chỉnh trước đó, và chúng ta xác định chúng từ số dòng mã nguồn và thông tin tệp được giữ trong phần debug của mã nhị phân. Chúng tôi ký hiệu chúng là 𝑀+ =
* {(𝑒1𝑖, 𝑒2𝑖)|𝑒1𝑖 ∈ 𝐸
* ′
* 1
* ∧ 𝑒2𝑖 ∈ 𝐸
* ′
* 2
* }
* 𝑚
* 𝑖=1
* , trong đó 𝐸
* ′
* 𝑖
* là tập con của 𝐸𝑖. Cuối cùng, nhiệm vụ của chúng tôi là tìm sự tương ứng giữa tất cả các cặp thực thể không canh chỉnh trong các BDG.

3.3 Học biểu diễn dựa trên đồ thị

* XBA sử dụng học biểu diễn đồ thị dựa trên deep learning [7] để giải quyết vấn đề canh chỉnh này. Ở đây, chúng ta trước tiên huấn luyện một mô hình để chiếu các thực thể và thuộc tính trong các BDG vào không gian biểu diễn thống nhất. Sau khi được huấn luyện, XBA sử dụng mô hình này đã được huấn luyện trước để tạo ra các biểu diễn cho các thực thể và thuộc tính trong BDG. Như vậy, mô hình có thể tạo ra các biểu diễn chất lượng cho các thực thể không được nhìn thấy trong quá trình huấn luyện (ğ4.3) hoặc các biểu diễn có thể cải thiện hiệu suất của nhiều tác vụ phụ thuộc, ví dụ như phân tích tương tự đa nền tảng (ğ4.4).
* Chúng tôi sử dụng mạng convolutional đồ thị (GCN) [6, 29] để tạo các biểu diễn. GCN đã được nghiên cứu kỹ lưỡng và vượt trội hơn các phương pháp nhúng được sử dụng bởi các công trình trước đó [17, 18], chẳng hạn như random walks, trong các tác vụ canh chỉnh đồ thị đa dạng (xem công trình liên quan của chúng tôi ở phần 6).
* Kiến trúc của GCN. Chúng tôi sử dụng mô hình GCN có nhiều lớp được đề xuất bởi Kipf và Welling [29]. Nó chứa một ngăn xếp của nhiều lớp GCN. Mỗi lớp lấy ma trận nhúng nút 𝐻
* (𝑙) ∈ R
* 𝑛×𝑑
* (𝑙)
* , trong đó 𝑛 là số lượng nút và 𝑑
* (𝑙)
* là chiều đầu vào của lớp 𝑙. Đầu ra của lớp 𝑙 được tính toán như sau:
* A math formula with numbers and letters

  Description automatically generated
* Trong đó, 𝜎 là hàm kích hoạt mà chúng tôi sử dụng 𝑅𝑒𝐿𝑈(·); 𝐴 là ma trận liên kết kề 𝑛 × 𝑛 biểu diễn các mối quan hệ trong một BDG; 𝐴ˆ = 𝐴 + 𝐼, trong đó 𝐼 là ma trận đơn vị; 𝐷ˆ là ma trận đường chéo chứa bậc của 𝐴ˆ; 𝑊 (𝑙) ∈ R
* 𝑑
* (𝑙)×𝑑
* (𝑙+1)
* là ma trận trọng số và 𝑑
* (𝑙+1)
* là chiều đầu vào của lớp tiếp theo.
* Tính ma trận kề. Chúng tôi cần các mối quan hệ 𝑇
* 𝑅 để tính phương trình trên. Chúng tôi biểu diễn 𝑇
* 𝑅 dưới dạng ma trận liên kết kề 𝐴. Theo công trình trước đó [56], chúng tôi định nghĩa 𝐴 sao cho 𝑎𝑖𝑗 ∈ 𝐴 đo lường ảnh hưởng tiềm năng của nút 𝑒𝑖 đến nút 𝑒𝑗. Chúng tôi tính các giá trị này cho mỗi loại mối quan hệ 𝑟: chức năng, được ký hiệu bằng 𝑓 𝑢𝑛(𝑟), và chức năng nghịch đảo, được ký hiệu bằng 𝑖 𝑓 𝑢𝑛(𝑟). Sau đó, chúng tôi tính 𝑎𝑖𝑗, số lượng các mối quan hệ được định nghĩa giữa 𝑒𝑖 và 𝑒𝑗, được trọng số bởi 𝑓 𝑢𝑛(𝑟) và 𝑖 𝑓 𝑢𝑛(𝑟), như sau:
* A white background with black text

  Description automatically generated
* Tính toán các đặc trưng đầu vào. Các bộ ba thuộc tính 𝑇𝐴 được định nghĩa trong một BDG phải được mã hóa thành các vector số học có thể được đưa vào GCN như các vector đặc trưng đầu vào ban đầu 𝐻
* (0)
* . XBA xem xét tập hợp của tất cả các giá trị thuộc tính (ví dụ: các mã thông báo hướng dẫn được chuẩn hóa) liên kết với một thực thể 𝑒, bất kể loại của chúng, là các đặc trưng của 𝑒. Các biểu diễn vector số học của các đặc trưng này có thể được tính toán bằng cách sử dụng bất kỳ mã hóa hoặc mô hình nhúng nào. Ở đây, chúng tôi xem xét hai mô hình sau: (i) các vector đặc trưng thuộc tính được mã hóa trong mô hình túi từ (bag-of-words) [23], và (ii) các nhúng thu được từ DeepBinDiff và InnerEye [18, 65]. Trong đánh giá của chúng tôi, chúng tôi chỉ ra rằng XBA cải thiện hiệu quả của các nhúng này hơn nữa (xem phần 4.3).

3.4 Huấn luyện mô hình và tạo ra các nhúng

* Chúng ta nhớ rằng chúng ta chỉ có một tập con nhỏ các thực thể đã được căn chỉnh trước đó trong phân tích mã nhị phân trên nền tảng giao diện người dùng (cross-platform binary analysis). Trong tình huống này, việc sử dụng các phương pháp có giám sát được đề xuất trong các nghiên cứu trước đó [13, 59, 65] không thể học nhúng cho các thực thể không được căn chỉnh. Do đó, chúng ta tiếp cận một phương pháp bán giám sát: Các thực thể đã được căn chỉnh trước đó phục vụ làm tín hiệu giám sát cho phép tính toán các nhúng chất lượng cho các thực thể không được căn chỉnh.
* XBA sử dụng một kiến trúc Siamese để huấn luyện các GCNs của chúng tôi. Hai GCNs với cùng một kiến trúc và các tham số mô hình nhận các vector đặc trưng đầu vào và các ma trận kề, được xây dựng từ hai BDGs, làm đầu vào. Điều này cho phép chúng ta huấn luyện các GCNs của mình theo cách tương phản bằng cách sử dụng hàm mất mát hậu quả sau đây.
* Mục tiêu huấn luyện. Trong quá trình huấn luyện, chúng ta tối thiểu hóa khoảng cách 𝑓 giữa các nhúng từ các cặp thực thể được căn chỉnh tích cực trong khi tăng khoảng cách giữa các nhúng từ các cặp thực thể được căn chỉnh tiêu cực. Chúng tôi sử dụng một hàm mất mát bản lề dựa trên biên:
* A math equations on a white background

  Description automatically generated

A table with numbers and symbols

Description automatically generated

* Đây là phần tiếp theo về huấn luyện mô hình và tạo ra các nhúng.
* 𝑀+
* là tập các căn chỉnh tích cực từ các cặp thực thể đã được căn chỉnh trước đó, 𝑀−
* (𝑒1,𝑒2)
* là tập các căn chỉnh tiêu cực cho một cặp thực thể đã được căn chỉnh trước đó (𝑒1, 𝑒2), 𝑓 là khoảng cách L1 𝑓 (𝑥, 𝑦) = ||𝑥 − 𝑦||1, ℎ(𝑒) là nhúng của một thực thể 𝑒, và 𝛾 là tham số siêu hợp lệ, xác định khoảng cách tối thiểu giữa các căn chỉnh tích cực và tiêu cực. Để tính toán 𝑀+
* , chúng ta căn chỉnh trước các thực thể bằng cách sử dụng thông tin số dòng mã nguồn được lưu giữ trong phần debug của các mã nhị phân đã cho. 𝑀−
* (𝑒1,𝑒2)
* được thu được bằng cách thay thế 𝑒𝑖 bằng một nút được chọn ngẫu nhiên từ 𝐺𝑃1 hoặc 𝐺𝑃2.
* Sau khi được huấn luyện, mô hình GCN của chúng tôi có thể tạo ra các nhúng cho các thực thể 𝑒𝑖 được tìm thấy trong mã nhị phân đã cho. Chúng tôi đầu tiên xây dựng một BDG của mã nhị phân và lấy đồ thị con từ đó. Đồ thị con phải bao gồm các thực thể 𝑒𝑖 và các thực thể hàng xóm 𝑛 bước, cũng như các mối quan hệ giữa chúng, trong đó 𝑛 là số lớp của GCN. Chúng tôi đưa đồ thị con vào GCN của chúng tôi và nó sẽ tạo ra các nhúng ℎ(𝑒𝑖) cho các 𝑒𝑖. ℎ(𝑒𝑖) có thể được sử dụng cho nhiều tác vụ phân tích mã nhị phân sau này, chẳng hạn như tính điểm tương đồng giữa hai đoạn mã nhị phân, ngay cả khi chúng được lấy từ mã nhị phân được biên dịch cho các nền tảng khác nhau.

4 ĐÁNH GIÁ

Trong phần này, chúng tôi trả lời các câu hỏi nghiên cứu sau:

* RQ1. XBA có tạo ra nhúng tương tự cho các khối cơ bản

không được căn chỉnh trước, tức là không bao gồm trong các căn chỉnh hạt giống không?

* RQ2. Các vector nhúng mã nhị phân học được với XBA

có mã hóa thông tin hữu ích cho phân tích nhị phân đa nền tảng không?

* RQ3. Một mô hình XBA có thể tổng quát hóa trên các nhị phân không xuất hiện trong quá trình huấn luyện?
* RQ4. Bao nhiêu sự giám sát, tức là lượng căn chỉnh hạt giống, cần thiết trong quá trình huấn luyện một mô hình XBA?
* RQ5. Mỗi loại quan hệ đóng góp bao nhiêu vào hiệu suất của XBA?

4.1 Bộ dữ liệu đánh giá

* Chúng tôi xây dựng các bộ dữ liệu đánh giá riêng cho XBA trong phân tích nhị phân đa nền tảng. Các bộ dữ liệu đánh giá được sử dụng bởi hầu hết các nghiên cứu trước đó về nhúng mã nhị phân không luôn chứa các thành phần cụ thể cho từng hệ điều hành hoặc kiến trúc hệ điều hành. Để học nhúng phong phú hơn (do đó cho phép phân tích đa nền tảng hoặc đa kiến trúc phong phú hơn), chúng tôi cẩn thận xây dựng các bộ dữ liệu riêng của chúng tôi, bao gồm nhiều thành phần cụ thể của từng nền tảng như sau:
* Bộ dữ liệu đa nền tảng: Chúng tôi chọn năm phần mềm khác nhau hỗ trợ Linux và Windows theo cách tự nhiên: (i) CURL, (ii) SQLITE3, (iii) chương trình giao diện dòng lệnh của OpenSSL (OPENSSL), (iv) thư viện mật mã của OpenSSL (LIBCRYPTO), và (v) Apache HTTPD. Chúng tôi biên dịch chúng thành các nhị phân (hoặc thư viện chia sẻ) của Linux ELF và Windows PE32+ với bao gồm thông tin gỡ lỗi.
* Bộ dữ liệu đa kiến trúc: Do kiến trúc lớp của các hệ điều hành hiện đại, phần lớn mã kiến trúc hệ thống cụ thể của từng ISA đều nằm trong các thư viện hệ thống cấp thấp. Chúng tôi do đó chọn (i) GLIBC, thư viện C mặc định trong nhiều bản phân phối Linux, và (ii) LIBCRYPTO trong OpenSSL, một thư viện mật mã hỗ trợ nhiều ISA. Cả hai thư viện đều chứa mã cụ thể của từng ISA, chẳng hạn như mã lắp tay được tùy chỉnh cho mỗi ISA được hỗ trợ. Chúng tôi biên dịch chúng trên Linux cho x86\_64 và AArch64 thành các thư viện chia sẻ định dạng ELF với thông tin gỡ lỗi.
* Chúng tôi phân tích mã nhị phân này bằng IDA Pro [26]. XBA xây dựng các đồ thị khối cơ bản (BDGs) tương ứng từ mã đã phân tích. Chúng tôi căn chỉnh trước các thực thể trong BDGs dựa trên thông tin dòng mã nguồn (tức là thông tin gỡ lỗi) được giữ bởi các trình biên dịch. Thống kê của bộ dữ liệu của chúng tôi được tổng kết trong Bảng 5. Chúng ta có thể thấy, vì những lý do được giải thích ở mục 2.3, có nhiều thực thể chúng tôi không thể căn chỉnh dựa trên thông tin gỡ lỗi. Tuy nhiên, các BDGs được xây dựng từ bộ dữ liệu của chúng tôi ngầm chứa thông tin ngữ nghĩa phong phú giữa các thực thể, được định nghĩa bởi các mối quan hệ. XBA bắt giữ ngữ nghĩa của các thực thể không được căn chỉnh thông qua việc học bán giám sát và có thể tìm ra các căn chỉnh giữa chúng.

4.2 Cài đặt và Thiết lập

* Chúng tôi đã triển khai XBA bằng Python v3.8 và TensorFlow v2.7.0 [1]. Chúng tôi dựa trên việc triển khai GCNs của Wang và cộng sự [56]. Trừ khi có ghi chú khác, chúng tôi sử dụng GCNs 5 lớp và thiết lập kích thước đơn vị ẩn của mỗi lớp 𝑑 là 200, tức là nhúng đầu ra sẽ có kích thước là 200. Trong hàm mất mát của chúng tôi, chúng tôi đặt khoảng cách giữa các cặp tương tự và không tương tự là 𝛾 là 0.1, và đối với mỗi cặp đã được căn chỉnh trước, chúng tôi tạo ra từ 50 đến 100 mẫu căn chỉnh tiêu cực, được chọn khác nhau cho mỗi nhị phân. Chúng tôi đã thực hiện tất cả các thí nghiệm trên một máy được trang bị CPU AMD EPYC 7282 với 256GB RAM và hai GPU NVIDIA RTX 3090.
* Các mô hình so sánh. Chúng tôi chọn các phương pháp mã nhúng mã nhị phân (và mã hóa) sau đây làm các mô hình so sánh: InnerEye [65], DeepBinDiff [18], và mô hình bag-of-words (BoW) [23]. Trong BoW, chúng tôi sử dụng từ vựng được xây dựng bằng cách sử dụng phương pháp mã hóa từ của InnerEye. InnerEye sử dụng một phương pháp học có giám sát với kiến trúc Siamese, trong khi DeepBinDiff sử dụng một phương pháp học không giám sát. Chúng tôi đánh giá cẩn thận hai phương pháp so sánh cơ bản này: chúng tôi bao gồm tất cả các mã thông báo đã được chuẩn hóa bỏ qua trong các nghiên cứu gốc và cẩn thận hoàn thiện việc triển khai được mã nguồn mở bởi các tác giả [1].
* Cả hai mô hình InnerEye và DeepBinDiff đã được huấn luyện trên bộ dữ liệu của chúng tôi bằng các tham số mặc định của chúng, và sử dụng các nhúng được tạo ra bởi các mô hình này để đánh giá XBA.
* Tiêu chí đánh giá. Chúng tôi sử dụng độ chính xác Hit@𝐾 là tiêu chí đánh giá của chúng tôi, đây là tiêu chí tiêu chuẩn trong nhiều nhiệm vụ phát hiện độ tương đồng [17, 34, 38, 56]. Điểm số cao có nghĩa là các cặp mã nhị phân có ý nghĩa tương đồng qua các nền tảng có các biểu diễn tương tự nhau hơn so với các cặp khác trong không gian nhúng. Chúng tôi chọn 𝐾 từ {1, 10, 50, 100}.
* Huấn luyện mô hình. Chi tiết huấn luyện thay đổi tùy thuộc vào các thí nghiệm, trừ khi có ghi chú khác, các mô hình cơ bản và XBA đều được huấn luyện như sau. Chúng tôi đã huấn luyện hai mô hình cho InnerEye: một trên tất cả các bộ kiểm tra đa nền tảng và một trên tất cả các bộ kiểm tra đa kiến trúc. Đối với DeepBinDiff, vì nó là một phương pháp tiếp cận trên toàn chương trình, chúng tôi đã huấn luyện các nhúng DeepBinDiff riêng lẻ trên mỗi bộ dữ liệu kiểm tra. XBA cũng được huấn luyện riêng biệt cho mỗi nhị phân. Đối với mỗi mô hình XBA, chúng tôi đã sử dụng các đặc trưng đầu vào (i) được mã hóa bằng BoW (được hiển thị như BoW Encoding + XBA) hoặc (ii) các nhúng được tạo ra bởi DeepBinDiff (DeepBinDiff + XBA). Chúng tôi đã dừng huấn luyện sớm trước khi mô hình trở nên quá khớp, và số epoch tương ứng thay đổi tùy thuộc vào từng nhị phân.

4.3 RQ1. Độ chính xác căn chỉnh

* Trước tiên, chúng tôi xem xét mức độ chính xác mà XBA có thể dự đoán các căn chỉnh qua các nền tảng. Ở đây, chúng tôi ngẫu nhiên chia các cặp thực thể đã được căn chỉnh trước (tức là các cặp có nhãn căn chỉnh) thành hai phần bằng nhau. Chúng tôi sử dụng nửa đầu tiên là bộ dữ liệu để huấn luyện mô hình XBA và InnerEye có 3 lớp, và nửa còn lại để đánh giá tính hữu ích của các nhúng. Chúng tôi báo cáo độ chính xác dự đoán dưới dạng Hit@𝐾. Bảng 6 cho thấy các kết quả.
* XBA vượt trội hơn các phương pháp cơ sở. Trong tất cả các đánh giá đa nền tảng và đa kiến trúc, XBA vượt trội hơn các phương pháp cơ sở. So với các phương pháp cơ sở, các cải tiến về độ chính xác tăng khi sử dụng các số liệu đo chính xác hơn (tức là Hit@100→Hit@1). Trong các căn chỉnh đa nền tảng, chúng tôi quan sát rằng XBA vượt trội hơn các phương pháp cơ sở từ 1% đến 25% trong Hit@1 và Hit@10. Đáng ngạc nhiên, XBA cải thiện độ chính xác từ 27% đến 75% trong tất cả các căn chỉnh đa kiến trúc trong Hit@1 và Hit@10; trong khi các phương pháp cơ sở chỉ có độ chính xác 2% trong các tình huống tương tự. Đặc biệt ngạc nhiên hơn, XBA cải thiện các mã hóa BoW lên tới 7 lần trong Hit@10 (11.34 → 83.41). Kết quả của chúng tôi cho thấy việc sử dụng bài toán căn chỉnh đồ thị trong phân tích nhị phân đa nền tảng cho phép chúng tôi tạo ra các nhúng có ý nghĩa và được căn chỉnh ngữ nghĩa qua các nền tảng.
* So sánh với các phương pháp có giám sát hiện có. InnerEye là một công trình đại diện sử dụng học có giám sát để huấn luyện mô hình nhúng, tuy nhiên, không may, nó cho thấy hiệu suất tệ nhất trong các tình huống đa nền tảng. Chúng tôi đầu tiên tìm lý do từ định dạng dữ liệu mà InnerEye sử dụng. InnerEye chỉ biểu diễn các khối cơ bản bằng các đặc trưng nội bộ của chúng, tức là các chuỗi hướng dẫn, không mã hóa bất kỳ thông tin láng giềng nào. Như chúng tôi đã thảo luận trong mục 3.3, phân tích đa nền tảng yêu cầu thông tin ngữ cảnh nằm ngoài phạm vi của các khối cơ bản. Trong khi đó, XBA, khi chú ý đến cả các đặc trưng phân bổ và các đặc trưng quan hệ phong phú được mã hóa trong các đồ thị khối cơ bản (BDGs), thực hiện đáng kể tốt hơn trong việc căn chỉnh các thực thể không được bao gồm trong dữ liệu huấn luyện.
* So sánh với các phương pháp không có giám sát hiện có. Chúng tôi quan sát rằng DeepBinDiff [18], một phương pháp học không giám sát đại diện, thực hiện tốt hơn so với InnerEye trong các tình huống đa nền tảng. Điều này có thể được quy cho phương pháp nhúng được sử dụng bởi DeepBinDiff, có thể mã hóa ngữ cảnh của một khối cơ bản trong một đồ thị dòng điều khiển liên thủ công bằng cách sử dụng Text-Associated Deep-Walk [60]. Tuy nhiên, DeepBinDiff thực hiện tệ hơn đáng kể (2% đến 12%) trong các tình huống đa kiến trúc, nơi các từ vựng (ví dụ: các từ viết mã lệnh) được sử dụng như các đặc trưng của các nút có nhiều khác biệt trên các nền tảng. Sự tương đồng ngữ nghĩa của chúng do các sự khác biệt cú pháp không được lấy mẫu bởi DeepBinDiff, vì nó sử dụng học không giám sát; nghĩa là, nó không cung cấp giám sát cho việc căn chỉnh đa nền tảng trong quá trình huấn luyện.
* Kết hợp XBA với các phương pháp hiện có. Chúng tôi nhấn mạnh rằng XBA tương thích với các phương pháp cơ sở hiện có. Trong các thí nghiệm với DeepBinDiff, chúng tôi sử dụng các nhúng DeepBinDiff làm vectơ đặc trưng đầu vào của XBA (được hiển thị như DeepBinDiff + XBA). Kết quả của chúng tôi cho thấy XBA cải thiện các nhúng DeepBinDiff sao cho chúng có căn chỉnh tốt hơn qua các nền tảng; do đó, chúng có hiệu quả tốt hơn cho phân tích nhị phân đa nền tảng. Sự cải thiện này nhất quán trong tất cả các thí nghiệm của chúng tôi với DeepBinDiff. Thú vị là trong OPENSSL, các nhúng DeepBinDiff được cải thiện bởi XBA vượt trội hơn các nhúng XBA được huấn luyện bằng các nhúng được mã hóa bằng BoW. Kết quả cho thấy có thể có sự tương hợp giữa XBA và các phương pháp nhúng khác.