ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC
PHƯƠNG PHÁP HỌC MÁY TRONG AN TOÀN THÔNG TIN

TÊN ĐỀ TÀI: HỌC MÁY TRONG PHÁT HIỆN SỰ TƯƠNG ĐỒNG CỦA TỆP NHỊ PHÂN

TÊN TIẾNG ANH: MACHINE LEARNING IN BINARY FILE SIMILARITY DETECTION

Giảng viên hướng dẫn
ThS.Phan Thế Duy

Sinh viên thực hiện
Võ Hoàng Khanh - 20521458
Phan Quang Khải 20521429
Nguyễn Thành Công - 20521143

Mục lục

Mục lục **2**
Lời cảm ơn 3
1 Tổng quan 4
1.1 Giới thiệu vấn đề 4
1.2 Tính ứng dụng 5
1.3 Mục tiêu, đối tượng và phạm vi nghiên cứu 5
2 Cơ sở lý thuyết 7
2.1 Sự tương đồng mã nhị phân 7
2.2 Học sâu 8
2.3 Một số kiến trúc mạng nơ-ron 9
3 Mô hình phát hiện sự tương đồng mã nhị phân 11
4 Thí nghiệm và đánh giá 12
4.1 Thiết lập thí nghiệm 12
4.2 Quá trình thực nghiệm 12
4.3 Tiền xử lý dữ liệu 14
4.4 Kết quả thí nghiệm 14
5 Kết luận 17
5.1 Kết quả đạt được 17
5.2 Một số thách thức 17
5.3 Hướng phát triển 17

Tài liệu tham khảo **18**

Lời cảm ơn

Trong suốt quá trình nghiên cứu và hoàn thành đồ án này, nhóm em đã may mắn nhận được sự định hướng, giúp đỡ, các ý kiến đóng góp quý báu từ giáo viên bộ môn.

Nhóm xin gửi lời cảm ơn sâu sắc tới thầy Phan Thế Duy vì sự tận tâm, nhiệt tình hướng dẫn và hỗ trợ em trong suốt quá trình nghiên cứu. Sự chỉ bảo và động viên của thầy đã giúp nhóm vượt qua nhiều khó khăn và thử thách.

Dù cho đồ án này có thể chưa đạt được sự hoàn hảo như mong đợi, nhưng nhóm đã nỗ lực hết mình và luôn hy vọng rằng sẽ có cơ hội để hoàn thiện và cải thiện nó trong tương lai.

Nhóm 2 xin chân thành cảm ơn thầy.

1 Tổng quan

1.1 Giới thiệu vấn đề

Phát hiện sự tương đồng mã nhị phân là quá trình phân tích và so sánh các đoạn mã nhị phân để xác định mối quan hệ tương đồng giữa chúng. Điều này có thể áp dụng cho nhiều mục đích, bao gồm phân loại mã độc hại, phát hiện lỗ hổng bảo mật, phân tích bản vá, và phân tích dòng dõi phần mềm.

Tầm quan trọng của vấn đề này được thể hiện qua lượng lớn tài liệu nghiên cứu: các nhà nghiên cứu trong nhiều lĩnh vực như bảo mật hệ thống, ngôn ngữ lập trình và học máy đã xuất bản rất nhiều bài báo (thường ở các hội nghị hàng đầu) đề xuất các phương pháp mới cho tương đồng nhị phân. Cuộc cạnh tranh này đã thúc đẩy sự phát triển nhanh chóng và hoàn thiện nhiều giải pháp khác nhau.[4]

Tuy nhiên, dù có khối lượng nghiên cứu đáng kể, vẫn còn nhiều câu hỏi quan trọng chưa được trả lời:

- Các phương pháp khác nhau so sánh như thế nào khi được đánh giá trên cùng một tập dữ liệu và bằng các tiêu chí giống nhau?
- Vai trò của các tập hợp đặc trưng khác nhau là gì?
- So sánh giữa các kiến trúc khác nhau khó khăn hơn so với làm việc trên một kiến trúc duy nhất không?
- Có hướng nghiên cứu cụ thể nào hứa hẹn hơn cho việc thiết kế các kỹ thuật mới trong tương lai không?

Một số thách thức chính khiến các nhà nghiên cứu vẫn chưa thể có câu trả lời thỏa đáng cho những câu hỏi trên.

- **Thiếu đánh giá tiêu chuẩn**
- **Đa dạng tập hợp đặc trưng**
- **Phức tạp khi so sánh giữa các kiến trúc**

Sự tương đồng hàm nhị phân đóng vai trò quan trọng trong nhiều lĩnh vực nghiên cứu bảo mật hệ thống:

- **Kỹ thuật đảo ngược**: Các phương pháp tương đồng mã nhị phân có thể giúp so khớp một hàm chưa biết với các hàm đã biết trong cơ sở dữ liệu trước đó, tiết kiệm thời gian đáng kể cho việc phân tích đảo ngược.
- **Phát hiện và vá lỗ hổng**: Khi đã biết một hàm dễ bị tấn công, các kỹ thuật tương đồng có thể tìm ra các trường hợp xuất hiện của hàm đó trong nhiều tệp nhị phân, giúp xác định và vá lỗi nhanh chóng.

- **So sánh nhị phân và phân tích bản vá**: Việc so sánh hai tệp nhị phân để nhận biết sự khác biệt và phân tích các bản vá đòi hỏi đo lường độ tương đồng hàm hiệu quả.

- **Phân tích nguồn gốc phần mềm và phân cụm phần mềm độc hại**: Sự tương đồng hàm nhị phân giúp theo dõi sự phát triển của phần mềm và phân cụm các mẫu phần mềm độc hại dựa trên các chức năng chung.

1.2 Tính ứng dụng

Nghiên cứu này được lấy cảm hứng từ BinDeep [1], nhóm dựa trên ý tưởng cơ bản của phương pháp là sử dụng mạng neural siamese để đo lường sự tương đồng của các hàm nhị phân.

Cùng với sự phát triển của học sâu, nhóm muốn khai thác tiềm năng của học sâu trong việc mô hình hóa và phân tích sư tương đồng mã nhị phân.

1.3 Mục tiêu, đối tượng và phạm vi nghiên cứu

1.3.1 Mục tiêu

Trong đề tài của mình , nhóm đề xuất xây dựng một khung mô hình phát hiện sự tương đồng mã nhị phân dựa vào học máy và học sâu.

1.3.2 Đối tượng nghiên cứu

Đối tượng nghiên cứu của nhóm sẽ là xây dựng mô hình học sâu để phân tích sự tương đồng của các hàm trong mã nhị phân có cùng kiến trúc nhưng khác trình biên dịch và khác cấp độ tối ưu trong trình biên dịch (optimization level).

Trong thực nghiệm đồ án môn học này chúng em sử dụng tập datastet lấy từ bài báo [4] Trong tập dataset của tác giả sử dụng 7 mã nguồn khác nhau bao gồm nmap, clamav, openssl, curl, unrar, zlib, z3. Bảng 1.1 thể hiện số lượng các loại tệp tin con cũng như tổng số lượng tên các hàm duy nhất trong mỗi mã nguồn.

Mỗi mã nguồn sẽ được biên dịch thành các file thực thi khác nhau bào gồm

- Kiến trúc CPU: x86, x64, arm32, arm64, mip32, mip64

- Optimization level khác nhau: O0, O1, O2, O3, O4

- Trình biên dịch: gcc, clang

- Phiên bản trình biên dịch: 5, 7, 9, 3.5, 5.5

Bảng 1.1: Bảng thống kê số lượng các hàm trong mỗi tệp tin của từng loại mẫu nhị phân

Loại tệp tin	nmap	zlib	clamav	openssl	unrar	curl
Số lượng tệp tin con	3	4	6	10	1	1
Số lượng các hàm duy nhất trong mỗi tệp	4191	328	2347	12547	525	771

1.3.3 Phạm vi nghiên cứu

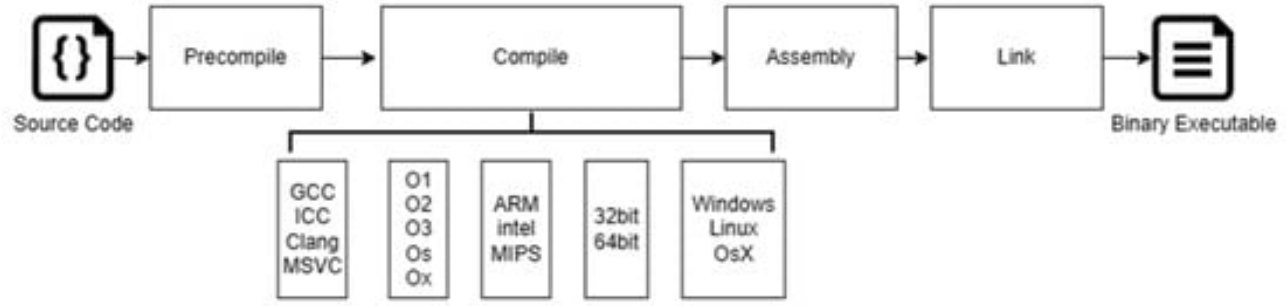
Trong khung mô hình đề xuất, nhóm tập trung phát triển thuật toán học máy và học sâu để phát hiện sự tương đồng mã nhị phân với kết quả cao nhất có thể.

2 Cơ sở lý thuyết

2.1 Sự tương đồng mã nhị phân

2.1.1 Mã nhị phân

Mã nhị phân là mã máy được tạo ra từ quá trình biên dịch mã nguồn và có thể được CPU thực thi trực tiếp. Mã này bao gồm các chữ số nhị phân (0 và 1), khiến nó khó đọc đối với con người. Để phân tích mã nhị phân dễ dàng hơn, các kỹ thuật dịch ngược được sử dụng để chuyển đổi mã máy sang ngôn ngữ assembly, cùng với các công cụ như trình gỡ lỗi giúp đơn giản hóa quá trình kiểm tra thủ công.



Hình 2.1: Quy trình biên dịch từ mã nguồn sang file có thể thực thi

Như minh họa trong Hình 2.1, quy trình chuyển đổi mã nguồn thành mã nhị phân điển hình thường bao gồm bốn giai đoạn: tiền biên dịch, biên dịch, lắp ráp và liên kết. Giai đoạn tiền biên dịch chủ yếu thực hiện các công việc như mở rộng tệp tiêu đề, thay thế macro và loại bỏ nhận xét. Giai đoạn biên dịch tiến hành phân tích từ vựng, cú pháp và ngữ nghĩa trên mã nguồn, tối ưu hóa và chuyển đổi nó thành mã hợp ngữ. Giai đoạn lắp ráp tiếp tục chuyển đổi mã hợp ngữ thành mã máy. Cuối cùng, giai đoạn liên kết tích hợp các tệp đối tượng đã biên dịch thành dạng nhị phân để tạo ra tệp thực thi cuối cùng.

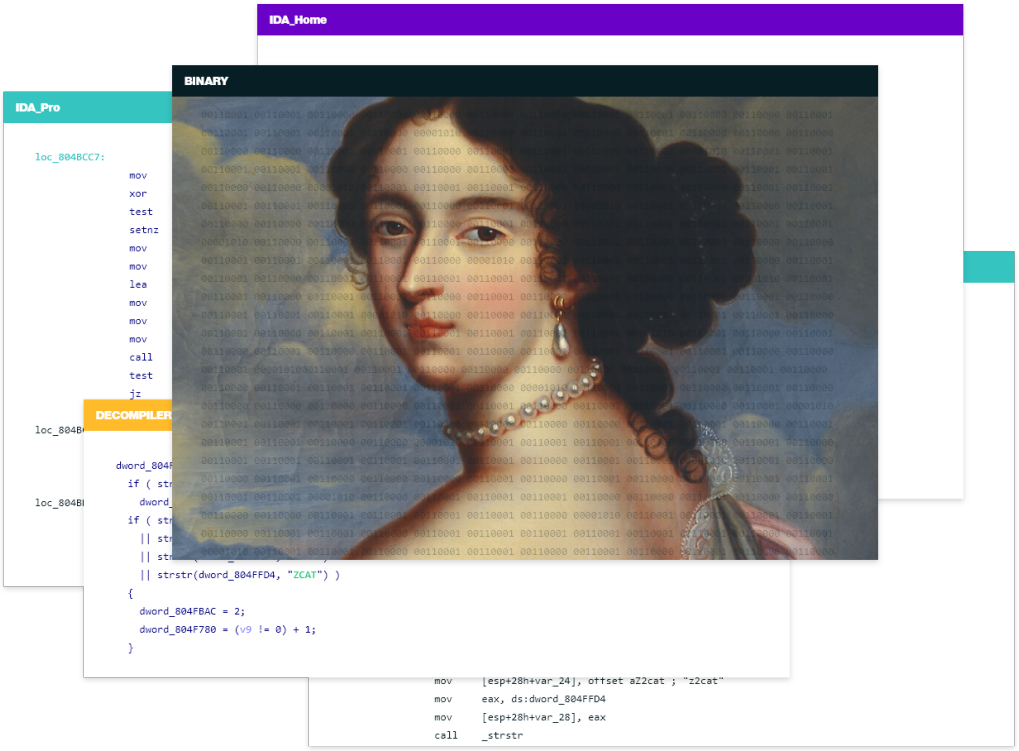
2.1.2 Trình dịch ngược

Một trình dịch ngược - decompiler là một công cụ phần mềm có khả năng chuyển đổi mã máy (bytes code) thành mã nguồn có cấu trúc cao hơn, chẳng hạn như mã assembly. Việc sử dụng decompiler có thể giúp phân tích và nắm bắt rõ hơn về chức năng và logic của một chương trình đã được biên dịch. Hiện nay có rất nhiều decompiler khác nhau như: IDA, Ghidra, Radare2, RetDec,...

Trong đề tài này, nhóm sử dụng IDA Pro để giải mã mã nhị phân và sau đó thu được chuỗi lệnh cho mỗi hàm.

Để giới thiệu IDA Pro, ta được biết đến với vai trò là một trình dịch ngược, có khả năng tạo ra các bản đồ của quá trình thực thi để hiển thị các lệnh nhị phân thực sự được bộ xử lý thực thi dưới dạng ký hiệu (ngôn ngữ assembly). Các kỹ thuật tiên

tiến đã được triển khai vào IDA Pro để nó có thể tạo ra mã nguồn ngôn ngữ assembly từ mã máy có thể thực thi và làm cho mã phức tạp này dễ đọc hơn đối với con người.



Hình 2.2: Trình dịch ngược IDA Pro

2.1.3 Sự tương đồng mã nhị phân

Sự tương đồng mã nhị phân (binary code similarity) là quá trình so sánh hai hoặc nhiều đoạn mã nhị phân, chẳng hạn như các khối cơ bản, hàm hoặc toàn bộ chương trình, để xác định sự giống nhau và khác nhau giữa chúng. Việc so sánh này đặc biệt quan trọng trong các tình huống mà mã nguồn của chương trình không có sẵn, như với các chương trình thương mại sẵn có (COTS), chương trình cũ và phần mềm độc hại.

Sự tương đồng mã nhị phân có thể dựa trên nhiều kỹ thuật phân tích khác nhau, bao gồm phân tích tĩnh (static analysis), phân tích động (dynamic analysis) và phân tích tượng trưng (symbolic analysis), với mục tiêu cuối cùng là giúp các nhà phân tích và nhà phát triển hiểu rõ hơn về các tính năng và hành vi của mã nhị phân.

2.2 Học sâu

Học sâu là một lĩnh vực của trí tuệ nhân tạo (Artificial Intelligence) tập trung vào việc xây dựng và huấn luyện các mô hình học máy sâu để tự động học từ dữ liệu. Phương pháp này lấy cảm hứng từ cách hoạt động của não người, nơi mà các mạng nơ-ron sinh học xử lý thông tin thông qua việc kết nối các nút nơ-ron.

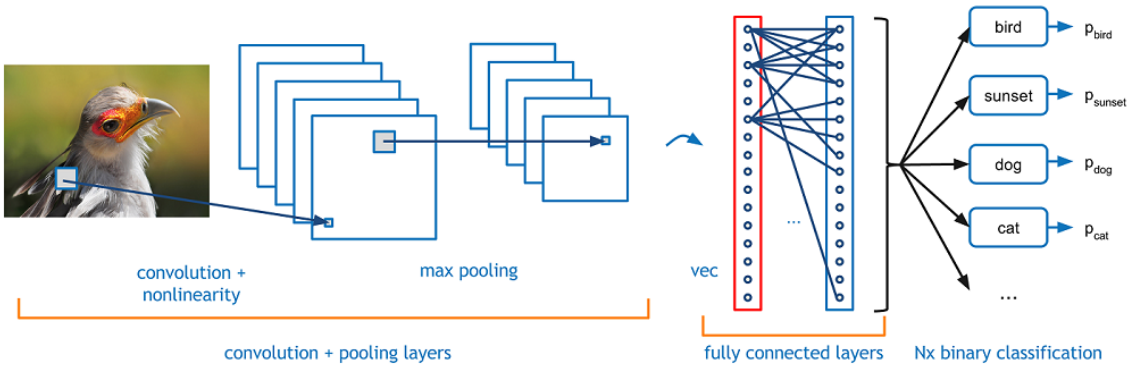
Mục tiêu chính của học sâu là xây dựng các mô hình máy tính có khả năng học và hiểu dữ liệu phức tạp để đưa ra dự đoán, phân loại, và giải quyết các bài toán phức tạp. Học sâu có ứng dụng rộng rãi trong nhiều lĩnh vực, bao gồm nhận dạng hình ảnh và video, xử lý ngôn ngữ tự nhiên, nhận dạng giọng nói, phân tích dữ liệu y tế, xe tự hành và nhiều lĩnh vực khác. Những mô hình này không chỉ giúp cải thiện độ chính xác và hiệu quả trong các tác vụ cụ thể mà còn mở ra nhiều khả năng mới trong việc giải quyết các vấn đề mà trước đây được coi là khó khăn hoặc không thể thực hiện được.

2.3 Một số kiến trúc mạng nơ-ron

Ở phần này, chúng tôi sẽ tập trung vào cấu trúc mạng Convolutional Neural Network (CNN), Long short term memory (LSTM) và Siamese.

2.3.1 Convolutional Neural Network (CNN)

- Kiến trúc CNN được sử dụng chủ yếu trong xử lý ảnh.
- Bao gồm các lớp tích chập (convolutional) để trích xuất đặc trưng từ ảnh và các lớp gộp (pooling layers) để giảm kích thước của đặc trưng.
- Cuối cùng, các lớp kết nối đầy đủ (fully connected layers) được sử dụng để phân loại và dự đoán.

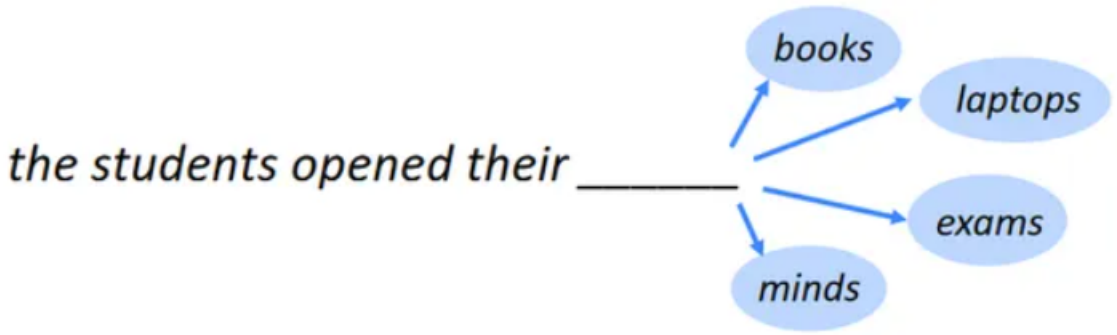


Hình 2.3: Ví dụ về một mô hình CNN để phân loại hình ảnh

2.3.2 Long short term memory

- Là một mạng thần kinh hồi quy (RNN) nhân tạo
- Kiến trúc LTSM được sử dụng chủ yếu trong xử lý văn bản, ngôn ngữ tự nhiên.
- Bao gồm các tế bào LSTM (LSTM cells) giúp lưu trữ và duy trì thông tin qua nhiều bước thời gian.

- Các tế bào LSTM có khả năng học và ghi nhớ các phụ thuộc dài hạn trong dữ liệu chuỗi, giúp chúng giải quyết vấn đề vanishing gradient thường gặp trong mạng nơ-ron truyền thống.

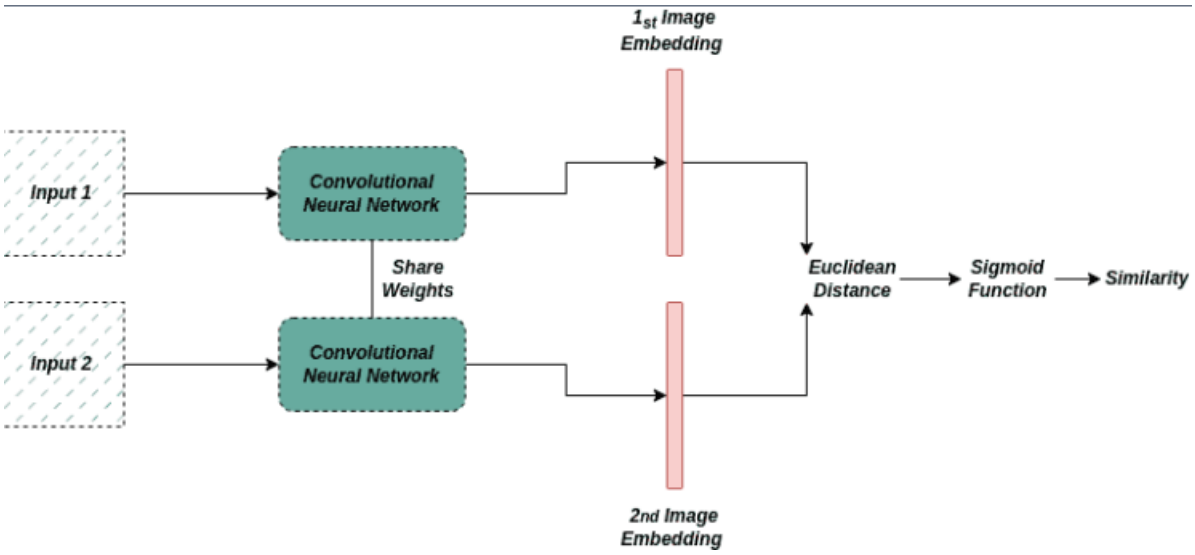


Hình 2.4: Ví dụ về ý tưởng của LSTM cho việc xử lý ngôn ngữ.

2.3.3 Siamese Neural Network

Siamese Neural Network (SNN) là một kiến trúc mạng nơ-ron chứa hai hoặc nhiều mạng con giống hệt nhau. "Giống hệt nhau" ở đây có nghĩa là, chúng có cùng cấu hình với cùng thông số và trọng số. Việc cập nhật các thông số được phản ánh đồng thời trên cả hai mạng con của nó.

SNN được sử dụng để tìm sự giống nhau của các dữ liệu đầu (Input Data) vào bằng cách so sánh các vectơ đặc trưng của chúng.



Hình 2.5: Mô hình SNN

3 Mô hình phát hiện sự tương đồng mã nhị phân

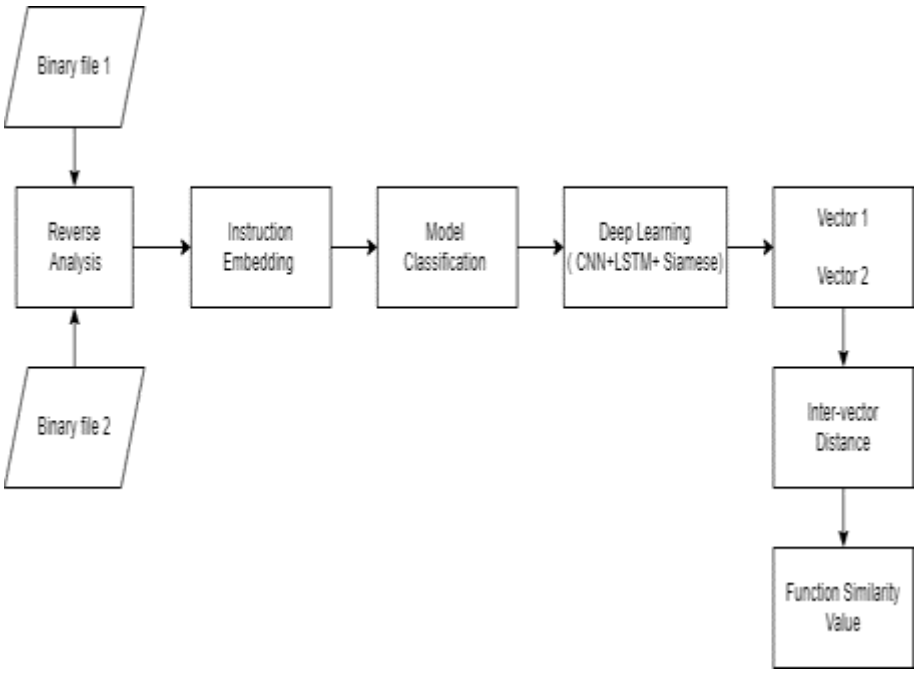
Một khung công cụ phát hiện sự tương đồng mã nhị phân mới, BinDeep, bằng cách sử dụng các kỹ thuật học sâu được phát triển để giải quyết những vấn đề liên quan đến việc các phần mềm có thể bị chèn mã độc hại hay những lỗ hổng bảo mật tồn tại do sai sót lập trình.

Ý tưởng cơ bản của phương pháp là sử dụng mạng nơ-ron siamese (SNN) để đo lường sự tương đồng của các hàm nhị phân. Để có được đầu vào cho mạng nơ-ron, đầu tiên phân rã mã nhị phân và trích xuất chuỗi lệnh như là các đặc điểm. Tiếp theo, sử dụng mô hình xử lý ngôn ngữ tự nhiên cổ điển để chuyển đổi các chuỗi lệnh thành các vector.Phương pháp này sử dụng cấu trúc mạng hỗn hợp, kết hợp mạng CNN và LSTM để đo lường sự tương đồng của các hàm nhị phân. [1]

Nhóm sẽ thực hiện dựa trên ý tưởng và thuật toán của bài báo về BinDeep

Khung đề xuất sẽ gồm 3 phần:

- Sử dụng IDA để trích xuất mã nhị phân, sau đó sử dụng NLP models để chuẩn hóa các chuôi lệnh thành vectors.
- Sử dụng mô hình học sâu để nhận diện kiến trúc CPU và mức độ optimization của hàm nhị phân
- Sử dụng mô hình mạng nơ-ron Siamese để phát hiện sự tương đồng mã nhị phân.



Hình 3.1: Khung mô hình đề xuất dựa trên ý tưởng BinDeep

4 Thí nghiệm và đánh giá

4.1 Thiết lập thí nghiệm

Môi trường thí nghiệm của nhóm được thực hiện trên goolge colab pro [2]. Môi trường này được sử dụng chuyên sâu cho học máy và không cần setup và được sử dụng TPU và GPU miễn phí. Google Colab Note của nhóm có sẵn tại địa chỉ [3]. Ngoài ra toàn bộ dataset cũng như các tệp tin liên quan khác cũng có tại địa chỉ [1]

4.2 Quá trình thực nghiệm

4.2.1 Xây dựng tập dữ liệu dạng bảng và gán nhãn

Từ tập dataset [4] nhóm thực hiện chọn lọc các phạm vi thực nghiệm. Các mã nguồn có cùng kiến trúc là x86, x64. Các optimization level là O0, O1, O2, O3, O4 và trình biên dịch là gcc, arm với các phiên bản trình biên dịch khác nhau. Trong số bảy mã nguồn trong tập dataset [4] nhóm chỉ sử dụng sáu trên bảy mẫu mã nguồn, ngoại trừ z3 lý do các tệp tin trong mã nguồn này có kích thước lớn và thời gian chiết xuất lâu.

Từ những tệp tin nhị phân ban đầu, nhóm thực hiện dùng IDA plugin python trong IDA pro để trích xuất ra các hàm và mã biên dịch hợp ngữ của từng tệp tin nhị phân. Nhóm chọn lọc các hàm có số lượng instruction line lớn hơn 30.

Với các hàm và mã hợp ngữ triết xuất ra nhóm sẽ lưu vào cơ sở dữ liệu. Để chiết xuất các hàm tương đồng trong từng loại tệp tin PE, nhóm sẽ chiết xuất các tên hàm chung trong từng loại tệp tin, ví dụ loại tệp tin *curl* trong tất cả các mẫu có 900 hàm và không có trong các loại tệp tin khác. Sau đó nhóm sẽ

1. Với các hàm tương đồng nhóm sẽ lấy các hàm trong cùng mã nguồn như nmap, curl v.v có cùng kiến trúc x86, hoặc x64 nhưng khác trình biên dịch và khác optimization level.

2. Với các hàm khác nhau, nhóm sẽ chọn các hàm khác nhau trong các mã nguồn khác nhau và khác tên hàm, nhưng có cùng kiến trúc là x86 hoặc x64.

Số lượng mẫu và tiêu chí của nhóm được thể hiện trong bảng 4.3. Số lượng mẫu của nhóm gồm 54 406 các cặp hàm tương đồng và 54 405 các cặp hàm không tương đồng. Các hàm tương đồng sẽ gán nhãn là 1 và các hàm không tương đồng gán nhãn là 0.

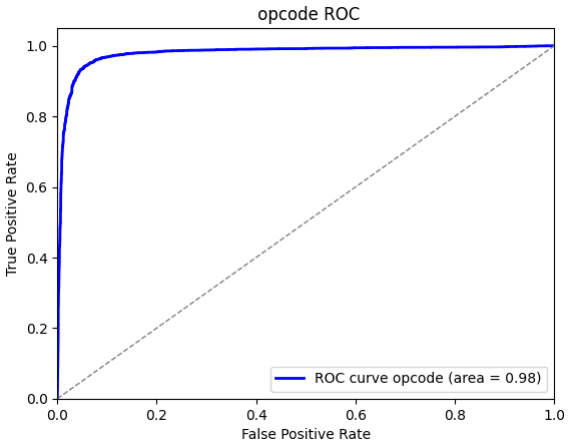
[1] https://drive.google.com/drive/folders/1uiGpMrPXVmzovGnmzBz8wCrI5PUG6F-K?usp=sharing

4.2.2 Chiết xuất thuộc tính

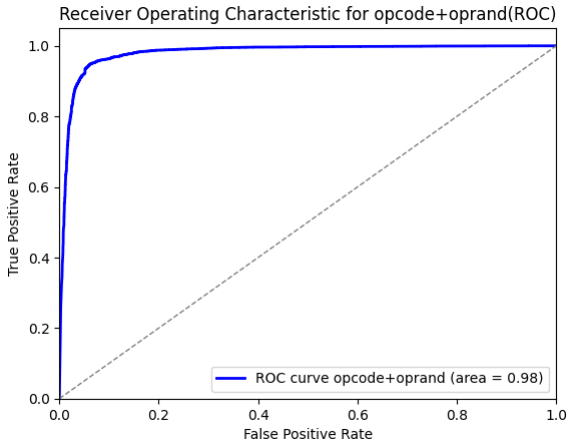
Theo hương dẫn của tác giả nhóm sẽ chọn ra hai thuộc tính để huấn luyện học sâu bao gồm:

- Chuỗi tuần tự các Opcode (lệnh máy) + Operand (Số nguyên, con chỏ, địa chỉ): ví dụ push ebp, add eax, 12, v.v. Loại dữ liệu là chuỗi.
- Chuỗi tuần tự Opcode: ví dụ add, mov, sub, jmp, call, push v.v. Loại dữ liệu là chuỗi.

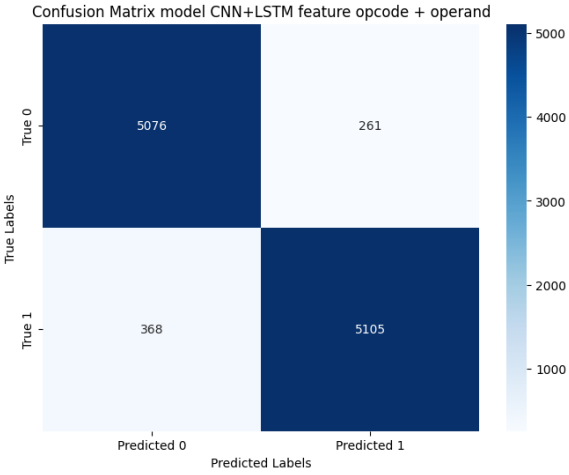
Hình 4.1: Biểu đồ ROC-AUC cho đặc tính Opcode



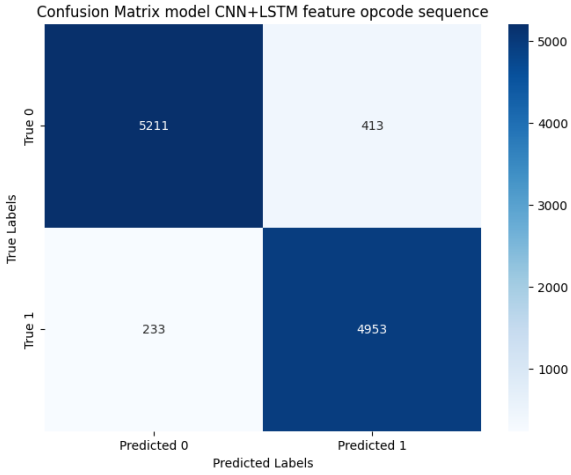
Hình 4.2: Biểu đồ ROC-AUC cho đặc tính Opcode và Operand



Hình 4.3: Biểu đồ confusion matrix cho đặc tính opcode và operand



Hình 4.4: Biểu đồ confusion matrix cho đặc tính opcode



Lý do chọn Opcode và Operand là vì những ngữ nghĩa trong thanh ghi sẽ thể hiện nhiều sự tương đồng [1].

Với đặc tính kết hợp Opcode và Operand, theo hướng dẫn của tác giả [1], nhóm sẽ tiến hành chuẩn hóa về tám dạng và được thể hiện trong bảng 4.1

Bảng 4.1: chuẩn hóa các dòng lệnh trong hợp ngữ

Lệnh hợp ngữ ban đầu	**Lệnh hợp ngữ sau khi chuẩn hóa**
push ebp	push TypeOne
mov ebp, esp	mov TypeOne, TypeOne
sub esp, 18h	sub TypeOne, TypeFive
mov dword ptr [esp+74h], 0	mov TypeThree, TypeFive
lea eax, [ebp+esi*2+0]	lea TypeOne, TypeFour
cmp eax, 210h	cmp TypeOne, TypeFive
jl loc 804B271	jl TypeSeven
mov eax, 1	mov TypeOne, TypeFive
call exit	call TypeSix

4.3 Tiền xử lý dữ liệu

Với dữ liệu dạng bảng thu thập được, nhóm tiến hành sử dụng Word2Vec, một kỹ thuật trong xử lý ngôn ngữ tự nhiên (NLP) có khả năng hiểu được ngữ nghĩa của từ. Trong Word2Vec có hai loại là CBOW và Skip-Gram. Theo đề xuất của tác giả nhóm sử dụng Skip-gram vì với dữ liệu lớn thì Skip-gram học tốt hơn.

4.3.1 Xây dựng mô hình học sâu

Nhóm sử dụng mô hình học sâu kết hợp CNN, LSTM và mạng Siamese. Việc kết hợp ba mô hình nãy sẽ cải thiện đáng kể hiệu quả trong việc so sánh tương đồng trong mã nhị phân [1].

4.3.2 Quá trình huấn luyện

Nhóm huấn luyện trên 10 epoch và sử dụng 10 cross-validation để tăng tính công bằng cho quá trình đánh giá.

4.3.3 Phương pháp đánh giá

Nhóm sử dụng các tiêu chí trong bảng 4.2 để đánh giá kết quả của mô hình học sâu.

4.4 Kết quả thí nghiệm

Bảng 4.4 thể hiện kết quả đánh giá sử dụng mô hình kết hợp CNN+LSTM+Siamese trên hai đặc tính Opcode và Operand. So sánh với đặc tính Opcode thì khi kết hợp hai đặc tính Opcode và Operand thì chỉ số Accuracy đạt cao hơn và chỉ số FPR đạt thấp hơn điều này cho thấy kết hợp hai đặc tính sẽ giúp cải thiện đáng kể khi xây dựng mô hình so sánh sự tương đồng trong mã nhị phân.

Bảng 4.2: Các phép đo đánh giá được sử dụng trong việc đánh giá kết quả mô hình.

Đánh giá	**Mô tả**	**Giải thích**
Accuracy	Đánh giá tần suất mô hình học máy dự đoán đúng kết quả	Độ chính xác càng cao thì hiệu suất càng tốt
Tỷ lệ False Negative (FNR)	Tỷ lệ các mẫu dương tính nhưng được dự đoán là âm tính trong tập test	FNR càng thấp thì hiệu suất càng tốt
Tỷ lệ False Positive (FPR)	Tỷ lệ của tất cả các mẫu âm tính nhưng vẫn được dự đoán là dương tính trong tập test	FPR càng thấp thì hiệu suất càng tốt
Độ chính xác (Precision)	Đánh giá tần suất mô hình học máy dự đoán đúng lớp dương tính.	Độ chính xác càng cao thì hiệu suất càng tốt
Recall	Đánh giá tần suất mô hình học máy nhận diện đúng các trường hợp dương tính (true positives) từ tất cả các mẫu dương tính thực sự trong tập dữ liệu	Recall càng cao thì hiệu suất càng tốt
Điểm F1 (F1 Score)	Giá trị trung bình điều hòa của độ chính xác và thu hồi của một mô hình phân loại	Điểm F1 càng cao thì hiệu suất càng tốt
Đường cong (ROC)	Biểu đồ thể hiện hiệu suất của một mô hình phân loại tại tất cả các ngưỡng phân loại.	
(AUC)	Đánh giá khả năng phân loại của mô hình	AUC càng cao thì khả năng phân loại của mô hình càng cao

Bảng 4.3: Mô tả tập dữ liệu dạng bảng

Loại mẫu	Số lượng mẫu	Kiến trúc CPU	Trình biên dịch	Phiên bản	Otimization level
Tương đồng	54 046	x86, x64	gcc, clang	5, 7, 9, 3.5	O0,O1,O2,O3
Không tương đồng	54 046	x86, x64	gcc, clang	5, 7, 9, 3.5	O0,O1,O2,O3

Bảng 4.4: Kết quả thí nghiệm

Đặc tính	Accuracy	Precision	Recall	F1-score	FPR	FNR
Opcode	0.940	0.955	0.923	0.939	0.042	0.076
Opcode + Operand	0.942	0.932	0.951	0.942	0.068	0.049

Ngoài ra các kết quả đạt được trong bảng 4.4 cũng thể hiện sự hiệu quả khi kết hợp ba mô hình CNN, LSTM và Siamese.

Hai biểu đồ 4.1 và 4.2 thể hiện lần lượt ROC curve của đặc tính Opcode và kết hợp hai đặc tính (Operand + Opcode). Nhận thấy cả hai chỉ số AUC đều là 0.98 điều này cho thấy mô hình có khả năng phân loại chính xác các hàm tương đồng lên tới 98%. Ngoài ra cả hai biểu đồ này tương đồng nhau điều này cho thấy không có sư khác biệt đáng kể khi sử dụng đặc tính Opcode hay kết hợp đặc tính Opcode và Operand.

Để mô tả chi tiết kết quả so sánh độ hiệu quả của mô hình học sâu kết hợp giữa LSTM và CNN trên các đặc tính kết hợp (Opcode + Operand) và đặc tính Opcode. Nhóm sử dụng confusion matrix, ý nghĩa kết quả của confusion matrix đươc thể hiện trong bảng 4.5. Confusion matrix giúp xác định số lượng True Positive (TP), False Positive (FP), True Negative (TN), và False Negative (FN). Các biểu đồ confusion matrix sẽ minh họa sự phân phối của các dữ liệu được phân loại.

Hai biểu đồ 4.4 và 4.4 lần lượt thể hiện kết quả confusion matrix trên hai đặc tính chuỗi opcode và đặc tính kết hợp (opcode + operand). Hai biểu đồ này cho thấy không có sự khác biệt đáng kể khi chỉ sử dụng đặc tính kết hợp hay kết hợp hai đặc tính opcode và operand. Mặc dù cả số lượng dương tính đúng và âm tính đúng của hai biểu đồ này đều cao nhưng có một số khác biệt. Với biểu đồ đặc tính kết hợp 4.3 thì số lượng dương tính đúng(xác định đúng hai hàm tương đồng) thấp hơn nhưng số lượng âm tính đúng (xác định đúng hai hàm không tương đồng) cao hơn so với mô hình chỉ sử dụng đặc tính opcode 4.4.

Bảng 4.5: Ý nghĩa của của kết quả confusion matrix

	Dự đoán tương đồng	**Dự đoán không tương đồng**
Thực tế tương đồng	Dương tính đúng (TP)	Âm tính sai (FN)
Thực tế không tương đồng	Dương tính sai (FP)	Âm tính đúng (TN)

5 Kết luận

5.1 Kết quả đạt được

Nhóm nghiên cứu tập trung về phương pháp phát hiện sự tương đồng trong mã nhị phân bằng học máy, học sâu.

Qua việc xây dựng và tìm hiểu mô hình này, nhóm chúng tôi đã hiểu sâu hơn về các hướng nghiên cứu liên quan, các kỹ thuật về trích xuất mã nhị phân,các kiến trúc CPUs, hiểu được về học máy, học sâu, các mạng đồ thị thần kinh, hiểu được các hạn chế để góp phần cải thiện dần.

- Tìm hiểu về kiến trúc CPUs như :x86,ARM,...
- Tìm hiểu về cách trích xuất các hàm đặc trưng của mã nhị phân
- Tìm hiểu về công cụ IDA Pro.
- Tìm hiểu và xây dựng các mô hình mạng CNN, LSTM,Siamese Neural Network

5.2 Một số thách thức

Trong quá trình thực nghiệm, việc xử lý dữ liệu vẫn còn nhiều vấn đề. Nguyên nhân do sự đa dạng trong biên dịch, cùng một mã nguồn tạo ra nhiều mã nhị phân khác nhau, đi kèm là phải có sự hiểu biết về cấu trúc máy tính cấp thấp.

Bên cạnh đó, hiệu suất tính toán cũng là một chướng ngại lớn, với thời gian xử lý khá lớn, khiến nhóm không thể triển khai với một lượng dữ liệu lớn.

5.3 Hướng phát triển

Cùng với một số kết quả đạt được, nhóm đề xuất một số cải thiện như sau:

- Tuy việc triển khai thành công, nhưng khả năng thực tế lại không được đánh giá cao. Nhóm muốn phát triển thêm để mô hình có thể được áp dụng trong một lĩnh vực cụ thể, ví dụ : phát hiện mã độc.

Tài liệu tham khảo

[1] Tian Donghai. “BinDeep: A deep learning approach to binary code similarity detection”. In: *Expert Systems with Applications* 168 (2021).

[2] Google. *Google Colaboratory*. 2024. URL: https://colab.google/.

[3] group2. *BinDeep project*. https://colab.research.google.com/drive/13v7SOeruoFrfz_SjsUq4cOdRvO044y9U?usp=sharing. 2024.

[4] Andrea Marcelli. “How machine learning is solving the binary function similarity problem”. In: *31st USENIX Security Symposium (USENIX Security 22)* (2022), pp. 2099–2116.