TOWARDS AUTOMATED SMART CONTRACTS VULNERABILITY DETECTION BASED ON DEEP LEARNING MODELS

Tran Anh Tu*, Dang Xuan Bao
Academy of Cryptography Techniques

ARTICLE INFO

Received: 04/7/2023
Revised: 30/8/2023
Published: 31/8/2023

KEYWORDS

Blockchain
Deep learning
Smart contracts
Vulnerability detection
Ethereum

ABSTRACT

Smart contracts are at the core of today's strong development of blockchain technology. However, due to the immutability and publicity of smart contracts, potential vulnerabilities in them become extremely dangerous. The issue of finding vulnerabilities in smart contracts thus attracts a lot of attention and is a hot issue in the development of blockchain applications. This paper proposes a model for applying deep learning technology in detecting and classifying vulnerabilities in smart contracts. The proposed model is developed based on 1D CNN architecture. This model gives better performance than traditional 2D CNN models in the problem of detecting vulnerabilities in smart contracts. We collect and label smart contract vulnerabilities using the Slither engine and perform proposed model evaluation with several traditional CNN models. The results show that the proposed model has a higher performance, reaching an accuracy of 98.18%. The results show the applicability of deep learning technology in software vulnerability detection in general and smart contracts in particular. This is the basis for developing more secure smart contracts in practice.

ỨNG DỤNG CÁC MÔ HÌNH HỌC SÂU TRONG TỰ ĐỘNG PHÁT HIỆN LỖ HỔNG TRONG CÁC MÃ HỢP ĐỒNG THÔNG MINH

Trần Anh Tú*, Đặng Xuân Bảo
Học viện Kỹ thuật Mật mã

THÔNG TIN BÀI BÁO

Ngày nhận bài: 04/7/2023
Ngày hoàn thiện: 30/8/2023
Ngày đăng: 31/8/2023

TỪ KHÓA

Chuỗi khối
Học sâu
Hợp đồng thông minh
Phát hiện lỗ hổng
Ethereum

TÓM TẮT

Hợp đồng thông minh là yếu tố cốt lõi tạo nên sự phát triển mạnh mẽ của công nghệ chuỗi khối ngày nay. Tuy nhiên, do tính bất biến và công khai của các hợp đồng thông minh mà các lỗ hổng tiềm ẩn trong nó trở lên hết sức nguy hiểm. Vấn đề tìm kiếm các lỗ hổng trong hợp đồng thông minh do đó thu hút nhiều sự quan tâm và là một vấn đề nóng trong quá trình phát triển của các ứng dụng chuỗi khối. Bài báo này đề xuất một mô hình ứng dụng công nghệ học sâu trong việc phát hiện và phân lớp lỗ hổng trong hợp đồng thông minh. Mô hình đề xuất được phát triển dựa trên kiến trúc mạng nơ ron tích chập CNN 1D. Mô hình này cho hiệu quả tốt hơn so với các mô hình CNN 2D truyền thống trong bài toán phát hiện lỗ hổng trong hợp đồng thông minh. Nghiên cứu này thực hiện việc thu thập và gán nhãn các lỗ hổng trên hợp đồng thông minh sử dụng công cụ Slither và thực hiện đánh giá mô hình đề xuất với một số mô hình CNN truyền thống. Kết quả cho thấy mô hình đề xuất có hiệu suất cao hơn, đạt tới độ chính xác 98,18%. Kết quả cho thấy khả năng ứng dụng công nghệ học sâu trong bài toán phát hiện lỗ hổng phần mềm nói chung và hợp đồng thông minh nói riêng. Đây là cơ sở để phát triển các hợp đồng thông minh an toàn hơn trên thực tế.

DOI: https://doi.org/10.34238/tnu-jst.8269

* Corresponding author. *Email: tutran@actvn.edu.vn*

1. Giới thiệu

Kể từ khi Satoshi Nakamoto xuất bản bài báo "*Bitcoin: A Peer-to-Peer Electronic Cash System*" vào năm 2008 [1] và sự ra đời chính thức của Bitcoin một năm sau đó, công nghệ chuỗi khối (blockchain) tử đã trở lên ngày càng thu hút nhiều sự quan tâm không chỉ trong lĩnh vực học thuật mà còn trong lĩnh vực ứng dụng. Hiện nay, công nghệ này đã được ứng dụng vào nhiều lĩnh vực như y tế, kinh tế, giáo dục, chính phủ điện tử [2]. Sự phổ biến của các ứng dụng chuỗi khối ngày nay bắt nguồn từ sự ra đời của khái niệm hợp đồng thông mình do Vitalik Buterin đề xuất vào năm 2015 trong nền tảng chuỗi khối thế hệ tiếp theo Ethereum [3]. Các khái niệm như hợp đồng thông minh, chuỗi khối và máy ảo Ethereum phi tập trung (EVM) của nó cung cấp cho các nhà phát triển khả năng xây dựng các ứng dụng phi tập trung (dApps) và do đó có khả năng triển khai được nhiều bài toán trên thực tế hơn thay vì chỉ dừng lại với vai trò là một hệ thống tiền tệ thanh toán.

Hợp đồng thông minh trên thực tế là các chương trình kiểm soát việc thực thi tự động cho nhiều giao dịch trong mạng ngang hàng [4]. Tuy nhiên, các hợp đồng thông minh nắm giữ tiền điện tử trị giá hàng tỷ đô la, khiến chúng trở lên hấp dẫn đối với những kẻ tấn công. Với số lượng hợp đồng thông minh ngày càng tăng, ngày càng có nhiều vấn đề về bảo mật được phơi bày. Những kẻ tấn công sử dụng ác ý các lỗ hổng hợp đồng thông minh để xâm nhập mạng chuỗi khối, gây ra tổn thất lớn cho cả hệ thống chuỗi khối và chủ sở hữu hợp đồng thông minh [5]. Theo thống kê của SlowMist Hacked [6], tính đến thời điểm hiện tại, các nền tảng chuỗi khối chứa ETH [7], EOS [8] và TRON [9] đã bị thiệt hại trị giá hơn 1,2 tỷ USD do cuộc tấn công bảo mật vào hợp đồng thông minh.

Các hợp đồng thông minh Ethereum được viết bằng Solidity, một ngôn ngữ lập trình thuộc dạng Turing hoàn chỉnh: điều này cho phép các nhà phát triển chuỗi khối dễ dàng triển khai các giải pháp logic kinh doanh phức tạp và là công cụ phát triển các ứng dụng phi tập trung (dApps). Ethereum [7], một trong những nền tảng chuỗi khối phổ biến nhất, đã triển khai hàng chục nghìn hợp đồng thông minh, kiểm soát một lượng lớn Ether (Tiền điện tử của Ethereum) trị giá hàng tỷ đô la. Điều này cũng tạo ra nhiều khả năng xảy ra lỗi và lỗ hổng, có thể được tìm thấy và sau đó bị khai thác bởi những người dùng có ác ý: đây là một vấn đề nghiêm trọng đối với hợp đồng thông minh, bởi vì những vấn đề này không thể được vá sau khi triển khai do tính chất bất biến của sổ cái.

Nhiều sự kiện bảo mật hợp đồng thông minh Ethereum do những kẻ tấn công gây ra cũng đã và đang xuất hiện, trong khi tổn thất đặc biệt nghiêm trọng do tính không thể đảo ngược và bất biến của hợp đồng thông minh. Chúng ta chỉ có thể nhìn Ether chảy vào túi của kẻ tấn công nhưng không thể ngăn chặn nó. Vào tháng 6 năm 2016, hơn 3,6 triệu Ether đã bị đánh cắp bởi tin tặc do khai thác lỗ hổng reentrancy của The DAO Attack [10], gây thiệt hại hơn 60 triệu USD. Ngoài ra, vào tháng 11 năm 2017, số Ether trị giá 300 triệu USD đã bị đóng băng do ví MultiSig của Parity [11]. Có thể coi rằng ngày càng có nhiều lỗ hổng được phát hiện và khai thác, làm nổi bật một yêu cầu cấp thiết đối với tính bảo mật của các hợp đồng thông minh. Do đó, các công cụ phát hiện lỗ hổng hiệu quả cho hợp đồng thông minh là rất cần thiết và cấp bách, đặc biệt là với sự phụ thuộc của các giao dịch mở rộng vào các hợp đồng thông minh trong hầu hết các nền tảng chuỗi khối.

Ngày càng có nhiều mối quan tâm và nghiên cứu liên quan đến tính an toàn của hợp đồng thông minh, với mục tiêu cố gắng tìm kiếm và xác định các lỗ hổng khác nhau của hợp đồng thông minh. Hướng tiếp cận truyền thống trong bài toán này là các phương pháp phân tích tĩnh mã nguồn và xây dựng tập luật phát hiện như phân tích thực thi hình thức và các quy tắc logic cứng do chuyên gia xác định. Dựa trên các phương pháp này, các công cụ phát hiện lỗ hổng tự động được phát triển để phát hiện các lỗ hổng trên hợp đồng thông minh trước khi triển khai hợp đồng thông minh trên mạng chuỗi khối. Các công cụ phát hiện lỗ hổng tự động dựa trên thực thi tượng trưng và thực thi phi tượng trưng thường được sử dụng do độ chính xác phát hiện cao đối

với các lỗ hổng đã biết. Tuy nhiên, các công cụ thực thi tượng trưng như Oyente [12], Osiris [13], Mythril [14] và Securify [15] tốn nhiều thời gian trong việc mô phỏng các đường dẫn biểu tượng và không phù hợp để phát hiện lỗ hổng hàng loạt vì khó khám phá tất cả đường dẫn thực thi trong một hợp đồng. Các công cụ thực thi phi tượng trưng như Slither [16] và Smartcheck [17] cũng tốn thời gian mô phỏng các đường dẫn biểu tượng và gây ra các phủ định sai vì chúng phụ thuộc nhiều vào các quy tắc phát hiện được xác định trước. Ngoài ra, các phương pháp này cần được thực hiện bởi nhân lực chuyên gia thông qua phân tích mã nguồn và thu thập các phần thực thi mã chính như một tính năng để kiểm tra xem chúng có thể dẫn đến hành vi giao dịch độc hại hay không. Điều này khá tốn kém và có thể thiếu chính xác.

Hướng tiếp cận hiện đại hơn xuất hiện gần đây là hướng tiếp cận phân tích động trong đó đưa vào các đầu vào và chạy thử các mã hợp đồng thông minh để xác định kết quả đầu ra và các tác động của nó. Với hướng tiếp cận này thì chi phí bỏ ra khá lớn do việc chạy thử các mã hợp đồng thông minh đòi hỏi rất nhiều tài nguyên, và các hợp đồng thông minh cũng có thể gọi ra các tài nguyên bên ngoài mạng đòi hỏi các chi phí bổ sung. Điều này làm cho phân tích động trở nên khá tốn kém trong việc xác định các lỗ hổng trong hợp đồng thông minh.

Để giải quyết các vấn đề trên, các phương pháp dựa trên học máy và học sâu đã được đề xuất để giúp nâng cao hiệu quả trong việc tự động phát hiện các lỗ hổng trong hợp đồng thông minh. Trong đó, phương pháp học sâu cho thấy độ chính xác phát hiện lỗ hổng tốt mà không cần định nghĩa rõ ràng về các quy tắc phát hiện từ tập dữ liệu huấn luyện [18]. Đặc biệt, các phương pháp phát hiện dựa trên mạng nơ-ron tích chập (CNN) nhìn chung cho thấy độ chính xác tốt khi phân lớp phần mềm độc hại và lỗ hổng trong phần mềm [19], [20]. Tuy nhiên, do các mô hình học sâu dựa trên CNN [21] được thiết kế để phân loại không phải mã phần mềm mà là hình ảnh nên chúng không thể phân tích chính xác ngữ nghĩa và ngữ cảnh của hợp đồng thông minh do đó gây ra nhiều kết quả dương tính giả và âm tính giả. Đã có nhiều nghiên cứu về các mạng CNN 1D đối với các bài toán như nhận dạng âm thanh, phân tích các dữ liệu dạng chuỗi, phân tích ngữ nghĩa,... Điều này cho thấy khả năng ứng dụng mạng CNN 1D trong bài toán với bài toán phát hiện lỗ hổng trong hợp đồng thông minh.

Để đảm bảo hiệu quả của phương pháp phát hiện lỗ hổng trong hợp đồng thông minh bằng mạng CNN, bài báo này đề xuất một mô hình mạng CNN 1D dựa trên kiến trúc ResNet phù hợp cho hoạt động trên các đoạn mã bytecode. Khác với các mô hình CNN 2D truyền thống mô hình được đề xuất sẽ có thể trích xuất được các thuộc tính ở mức thấp dễ dàng hơn. Nó cho phép duy trì sự tương quan trong ngữ nghĩa và ngữ cảnh của các mã hợp đồng thông minh khi thực hiện việc phân lớp. Để triển khai mô hình này, nghiên cứu này thực hiện việc tiền xử lý các mã bytecode trong hợp đồng thông minh để sinh ra các chuỗi RGB như việc tạo ảnh màu.

Đóng góp chính của bài báo này bao gồm: (1) Đề xuất một kiến trúc CNN mới để phân lớp các lỗ hổng trong hợp đồng thông minh trong khi vẫn giữ được ngữ nghĩa và ngữ cảnh của hợp đồng. Điều này khác với các kiến trúc CNN truyền thống vẫn được sử dụng trong bài toán này; (2) Thực hiện việc thu thập và tiền xử lý dữ liệu hợp đồng thông minh từ Etherscan và gắn nhãn tự động sử dụng công cụ Slither phục vụ cho việc đánh giá; (3) Thực hiện đánh giá kiến trúc đề xuất so với các kiến trúc CNN 2D truyền thống như AlexNet, GoogleNet, Interception,... để chỉ ra rằng mô hình được đề xuất cho kết quả tốt khi so sánh với các mô hình truyền thống.

2. Phương pháp nghiên cứu

Mục tiêu của bài báo này là thiết kế một phương pháp phát hiện lỗ hổng tự động có khả năng phân lớp và kết luận về một số dạng lỗ hổng phổ biến trong hợp đồng thông minh. Để đạt được mục tiêu này, bài báo này đã phát triển một phương pháp trong đó dữ liệu của các hợp đồng thông minh viết bằng solidity được tiền xử lý bằng cách thực hiện dịch các mã byte thành mã màu RGB và chuyển đổi chúng thành hình ảnh được mã hóa có kích thước cố định. Sau đó, hình ảnh được mã hóa được đưa vào mạng nơ-ron tích chập để trích xuất và học biểu diễn tự động. Ở đây, bài báo không thực hiện việc trích xuất thuộc tính từ mã nguồn solidity trước theo cách thủ

công. Tất cả quá trình học sẽ diễn ra tự động, giúp giảm chi phí và công sức của chuyên gia. Cách biểu diễn đặc trưng này có thể biểu diễn các thông tin phức tạp hơn trong mã nguồn solidity bằng các ảnh màu với khả năng biểu diễn 16777216 màu khác nhau (mỗi mẫu có 24 pixel) thay với hình ảnh thang độ xám chỉ có 256 màu (mỗi mẫu có 8 bit pixel). Do kích thước biểu diễn đầu vào phức tạp như vậy, do đó bài báo sử dụng các mô hình CNN để làm giảm đi kích thước so với các loại mạng kết nối đầy đủ. CNN cung cấp khả năng dùng chung tham số làm cho nó phù hợp hơn với cấu trúc phức tạp hơn. Nó không chỉ làm giảm số lượng tham số mà còn phản ánh sự phức tạp của hợp đồng thông minh, tiết kiệm thời gian tính toán khổng lồ với phương pháp hiện tại.

2.1. Thu thập và xử lý dữ liệu

Vì đây là một lĩnh vực nghiên cứu tương đối mới nên không có nhiều bộ dữ liệu nguồn mở về các hợp đồng thông minh được dán nhãn và hầu hết chúng đều khá nhỏ. Hai trong số các bộ dữ liệu phổ biến nhất hiện nay là SmartBugs Wild [22] và ScrawlID [23]. Đây là hai bộ dữ liệu được gắn nhãn bằng các công cụ khác nhau có tỷ lệ dương tính giả tương đối thấp. Tuy nhiên, số lượng mẫu thử trong các tập này tương đối nhỏ. SmartBugs Wild chỉ chứa hơn 6700 mẫu và ScrawIID là hơn 47000 mẫu. Điều này khiến chúng khá nhỏ để đào tạo một mô hình sâu từ đầu. Chính vì vậy, bài báo thực hiện việc tự thu thập dữ liệu bằng cách cào dữ liệu các hợp đồng thông minh từ Etherscan và cho chạy qua công cụ tự quét lỗ hổng để gán nhãn.

Công cụ pyspider được sử dụng để thực hiện cào dữ liệu của các hợp đồng thông minh đã được triển khai trên mạng ethereum mainnet trong khoảng thời gian từ tháng một năm 2023 đến tháng 3 năm 2023. Sau khi thực hiện thu thập, thu được 132564 mẫu hợp đồng thông minh. Các mẫu này sau đó được đưa qua công cụ phân tích tĩnh Slither để tiến hành việc gắn nhãn. Slither là một công cụ sử dụng kết hợp nhiều bộ phát hiện lỗ hổng dựa trên luật (rule-based) do đó các lỗ hổng được kết luận là các lỗ hổng đã từng tồn tại và được tìm ra và được xây dựng thành các luật phát hiện. Chính vì vậy, công cụ này có độ tin cậy cao và tránh được tỷ lệ dương tính giả. Các hợp đồng thông minh sau khi thu thập được đi qua công cụ này và nhận về các file JSON chứa các thông tin cụ thể về các bộ phát hiện và lỗ hổng tương ứng được kết luận. Một mẫu sẽ bao gồm các thuộc tính: address (địa chỉ), source_code (mã solidity) và bytecode (mã bytecode) và mẫu JSON mà công cụ Slither trả về.

Hình 1 dưới đây là một ví dụ về một mẫu thu thập được sau khi đi qua công cụ Slither.

{
'address': '0x006699d34AA3013605d468d2755A2Fe59A16B12B'
'source_code': 'pragma solidity 0.5.4; interface IERC20 { function balanceOf(address account) external ...'
'bytecode':
'0x6080604052348015610010576000080fd5b506004361061020257600035 7c0100000000000000000000000000000000000000000000000000000000
0000000000000900...'
'slither': '{"success": true, "error": null, "results": {"detectors": [{"check": "divide-before-multiply", "impact": "Medium",
"confidence": "Medium"}]}}'
}

Hình 1. *Mẫu thông tin hợp đồng thông minh sau khi đi qua công cụ Slither*

Thông qua 38 bộ phát hiện của công cụ Slither sử dụng, tác giả thu được bộ dữ liệu còn lại sau khi lọc bỏ một số mẫu lỗi không đi qua được công cụ bao gồm 120.608 mẫu. Ở đây, bộ dữ liệu thu thập được sẽ được chia thành 5 lớp:

- access-control: là lớp các lỗ hổng phổ biến trong mọi phần mềm không chỉ riêng với chuỗi khối. Thông thường, chức năng của hợp đồng được truy cập thông qua các hàm công khai (public) hoặc bên ngoài (external) của nó nhưng nếu khả năng hiển thị của một số trường/hàm không được đặt chính xác thành riêng tư (private), thì người dùng độc hại có thể có quyền truy cập vào chúng.
- arithmetic: đây là các lỗ hổng liên quan đến vấn đề xử lý số học như tràn số dưới hoặc tràn số trên. Đây là một dạng lỗ hổng nghiêm trọng trong các hợp đồng thông minh khi các số nguyên

bị tràn dẫn tới các kết quả tính toán không chính xác. Trong trường hợp tràn số trên, nhiều đoạn mã liên quan có thể là bàn đạp cho các cuộc tấn công DoS hoặc đánh cắp thông tin.

• reentrancy: đây có lẽ là loạt lỗ hổng nổi tiếng nhất trong Ethereum. Nó xuất hiện khi một lời gọi gọi tới một hợp đồng bên ngoài được cho phép tạo các lời gọi mới ngược trở lại hợp đồng gọi ban đầu trước khi việc thực thi khởi tạo được hoàn thành. Điều này có nghĩa là trạng thái của hợp đồng có thể bị thay đổi ở giữa quá trình thực thi của một hàm.

• others: Đây là các lỗ hổng khác không thuộc vào các lớp lỗ hổng ở trên. Ví dụ như các lỗ hổng dạng: uninitialized-state, một dạng lỗ hổng xuất hiện khi hợp đồng thông minh có một biến trạng thái không được khởi tạo; incorrect-equality, là dạng lỗ hổng kiểm tra điều kiện bằng của tài khoản xem nó có đủ Ether hoặc token để thực hiện giao dịch hay không, khi gặp lỗ hổng này kẻ tấn công có thể dễ dàng thực thi điều kiện kiểm tra; và backdoor, đơn giản là các hàm được gắn nhãn "backdoor".

• safe: là các mã hợp đồng thông minh được coi là sạch khi đi qua các bộ phát hiện của công cụ Slither mà không được kết luận là một lỗ hổng nghiêm trọng.

Để có thể sử dụng cho việc phân lớp, từ các mẫu JSON trả về từ công cụ Slither, bài báo thực hiện tiền xử lý để đưa nó về nhãn tương ứng với kết quả quét từ công cụ. Chẳng hạn quay lại ví dụ trên, với impact là Medium và không thuộc danh sách 5 nhãn được chỉ định. Do đó, nó được gán nhãn là 4 (tức là mã an toàn không chứa lỗ hổng) như biểu diễn trong Hình 2.

{
'address': '0x006699d34AA3013605d468d2755A2Fe59A16B12B'
'source_code': 'pragma solidity 0.5.4; interface IERC20 { function balanceOf(address account) external ...'
'bytecode':
'0x6080604052348015610010057600080fd5b506004361061020257600357c0100000000000000000000000000000000000000000000000000000000900
...'
'slither': [4]
}

Hình 2. *Mẫu thông tin hợp đồng thông minh sau khi gán nhãn*

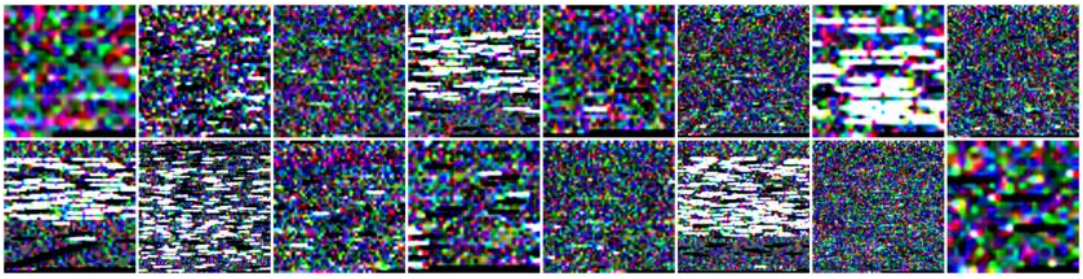
Từ dữ liệu thu thập được, phần bytecode sẽ được tách ra để thực hiện tiền xử lý nhằm thu được các hình ảnh màu RGB để đưa vào mô hình học sâu phục vụ cho việc phân lớp. Việc biến đổi từ bytecode về dạng ảnh có thể được mô tả như sau. Đầu tiên thực hiện lấy 6 ký tự hexa (tương ứng với 3 byte liên tiếp) để tạo thành một điểm ảnh RGB. Sở dĩ bài báo sử dụng 6 ký tự hexa là bởi mỗi mỗi điểm ảnh sẽ được biểu diễn bằng 3 byte tương ứng với 6 ký tự hexa.

Chẳng hạn, với nội dung của hợp đồng thông minh tại địa chỉ: 0xBB9bc244D798123fDe783fCc1C72d3Bb8C189413, chúng ta có thể thu được dạng bytecode của nó từ https://etherscan.io như trong Hình 3.

Hình 3. *Mã bytecode của hợp đồng thông minh trên Etherscan*

Bytecode của hợp đồng thông minh này sẽ được chuyển thành ảnh RGB bằng cách lấy 6 ký tự hexa tương ứng với 3 byte liên tiếp là một điểm ảnh. Chẳng hạn, với 6 ký tự đầu tiên của bytecode trong hợp đồng thông minh ở trên (606060) ta sẽ chuyển được thành một điểm ảnh RGB (R:96, G:96, B:96). Tương tự với 6 ký tự tiếp theo (405260) sẽ tương ứng với điểm ảnh RGB (R:64, G:82, B:96) và cứ tiếp tục như vậy. Sau quá trình này, chúng ta sẽ thu được một ảnh RGB để đưa vào mô hình học sâu CNN dùng cho việc phân lớp các hợp đồng thông minh.

Hình 4 là một ví dụ về một ảnh jpg được tạo thành từ chuyển đổi bytecode sang ảnh RGB.



Hình 4. *Chuyển đổi mã bytecode thành dạng ảnh JPG*

Mặc dù bài báo đề xuất mô hình CNN 1D, tuy nhiên để tiện cho việc đánh giá với các mô hình CNN 2D khác thì bài báo vẫn sử dụng chung cách thức biểu diễn dữ liệu dạng ảnh màu RGB. Khi đó với mô hình CNN 1D thì ảnh sẽ được xét theo các điểm ảnh từ trái sang phải từ trên xuống dưới để làm đầu vào cho mô hình. Để chuẩn hóa về kích thước, các hình ảnh sau khi thu được sẽ được chuẩn hóa về kích thước 128×128.

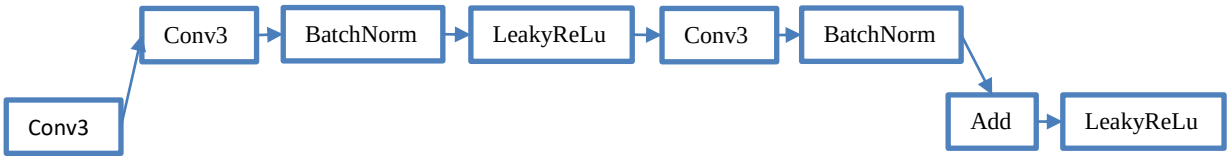
2.2. Mô hình huấn luyện

Bài báo này sử dụng mạng nơ-ron tích chập (CNN) để thực hiện việc phân loại các hình ảnh mầu RGB. Tuy nhiên, mô hình CNN truyền thống sẽ có thể gặp phải hai vấn đề sau:

- Kích thước bộ lọc truyền thống của mạng nơ-ron tích chập (CNN) là *3x3* hoặc *5x5*, do đó, các mã bytecode không tương quan có thể trở thành tương quan khi chúng được chuyển đổi thành hình ảnh. Các lỗi biên dịch của hợp đồng thông minh có thể tránh được phát hiện bằng cách tận dụng sự không khớp này.
- Hình ảnh màu thu được từ việc tổng hợp mã bytecode của hợp đồng thông minh không phải là hình ảnh tự nhiên; thay vào đó, chúng được hình thành từ mã nguồn solidity. Do đó, việc tổng hợp theo kiến trúc CNN 2D truyền thống chắc chắn sẽ phá hủy ngữ cảnh và ngữ nghĩa của phần mã độc hại, khiến việc phát hiện không chính xác.

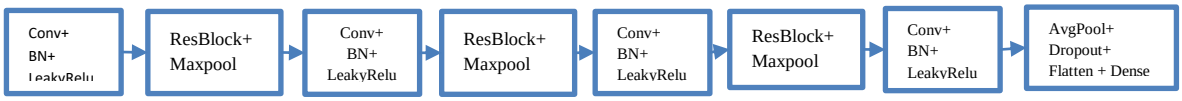
Để giải quyết hai vấn đề trên, bài báo này đề xuất sử dụng mô hình Conv1D. Kết quả đánh giá cho thấy mô hình này cho hiệu quả cao hơn với các mô hình 2D CNN truyền thống. Điều này có thể giải thích rằng các mô hình CNN 2 chiều làm mất đi sự tương quan của các mã bytecode trong hợp đồng thông minh trong khi mô hình CNN 1 chiều vẫn có khả năng giữ lại được các mối tương quan này. Các CNN 2D truyền thống được cấu trúc theo cách sao cho các lớp nông nắm bắt các thuộc tính cấp thấp, sau đó được tổng hợp thành các thuộc tính cấp cao trong các lớp tiếp theo. Tuy nhiên, các mẫu thú vị và hữu ích để phát hiện lỗ hổng trong mã bytecode rất có thể là các mẫu theo từng pixel ở mức độ thấp. Trong thực tế, khi mạng phát triển sâu hơn, chúng ta có xu hướng mất một số thông tin ở cấp độ pixel. Do đó, ngữ nghĩa và ngữ cảnh của hợp đồng thông minh có thể bị phá hủy. Ngược lại, việc áp dụng các tích chập 1D trên mã bytecode của hợp đồng được sử dụng làm tín hiệu (tức là: không được định hình lại dưới dạng hình ảnh RGB hình chữ nhật) có thể được trang bị tốt hơn để duy trì những thông tin này. Để khẳng định kết quả này, bài báo đã thực hiện nhiều thử nghiệm so sánh mô hình đề xuất Conv1D với các mô hình CNN 2D truyền thống khác nhau (bao gồm AlexNet, VGG19, ResNet-18, GoogleNet và Inception-v3) để đối chiếu kết quả thu được.

Đối với kiến trúc Conv1D, bài báo đã triển khai và thử nghiệm mạng nơ-ron covultional 1D lấy cảm hứng từ ResNet. Hình 5 dưới đây thể hiện kiến trúc của một khối ResBlock1D trong mô hình Conv1D được áp dụng.



Hình 5. *Kiến trúc của một khối ResBlock1D*

Hình 6 dưới đây thể hiện chi tiết các lớp của mạng Conv1D Resnet này:



Hình 6. *Kiến trúc chi tiết của mạng ResNet Conv1D được sử dụng*

3. Thử nghiệm và đánh giá

Dựa trên mô hình đề xuất và các phân tích ở trên, bài báo thử nghiệm và đánh giá mô hình đề xuất ResNet Conv1D trên bộ dữ liệu thu thập được so với các mô hình CNN 2D truyền thống khác. Phần thực nghiệm được thực hiện trên máy Windows 10, 64-bit với cấu hình phần cứng gồm: 64 GB DDR4 2400 RAM và CPU Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz 2.60 GHz, card đồ họa GPU NVIDIA GTX950M 2GB. Các thư viện được sử dụng là Pytorch, sklearn. Ngôn ngữ lập trình sử dụng là Python 3.10.

Chúng tôi đã thử các cấu hình các siêu tham số khác nhau cho từng loại mô hình khác nhau như: tốc độ học, thuật toán tối ưu (Adam hoặc SGD). Bộ dữ liệu thu thập được ở trên được chia ra thành 2 phần: 80% cho việc huấn luyện và 20% cho việc kiểm thử mô hình thu được. Thử nghiệm được thực hiện 10 lần. Mỗi lần chạy, dữ liệu kiểm thử và huấn luyện sẽ được chia lại ngẫu nhiên và đảm bảo tỷ lệ giữa các nhãn.

Bảng 1 dưới đây chỉ ra kết quả trung bình đạt được trên bộ kiểm thử cho mỗi lần chạy. Lưu ý rằng trong ngữ cảnh này khi mọi phần tử có thể thuộc nhiều hơn một lớp và các nhãn lớp không thực sự cân bằng thì độ chính xác không phải là thước đo lý tưởng để sử dụng. Do đó, chúng tôi cũng xem xét thêm chỉ số F1 score, tổng hợp các đóng góp của tất cả các lớp để tính toán số liệu trung bình và do đó xử lý các ví dụ của từng lớp với trọng số bằng nhau.

Bảng 1. *Độ chính xác của việc phân lớp đối với một số mô hình đánh giá*

Mô hình	Accuracy	F1 Score
ResNet Conv1D	0,9818	0,9748
ResNet-18	0,9524	0,9414
Inception-v3	0,9626	0,9612
AlexNet	0,8548	0,8486
VGG19	0,9232	0,8958
GoogleNet	0,8996	0,8966

Các CNN 2D có ưu điểm là không yêu cầu đầu vào bị cắt bớt theo bất kỳ cách nào. Trước tiên các hình ảnh được tạo ra bằng cách sử dụng tất cả các bytecode và sau đó chỉ cần thay đổi kích thước của chúng nếu cần. Tuy nhiên, chúng có những nhược điểm đã thảo luận ở trên, điều này có thể khiến chúng không phải là lựa chọn lý tưởng cho việc phân loại các lỗ hổng trên hợp đồng

thông minh. Trên thực tế, các mẫu lỗ hổng trong mã bytecode của Solidity có thể chỉ ở cấp độ của một chuỗi mã lệnh nhỏ và do đó có thể bị bỏ sót khi sử dụng tích chập có kích thước quá lớn.

Đối với mạng tích chập 1D do kích thước tính toán tối ưu, bài báo thực hiện việc cắt mẫu đầu vào với độ dài tối đa lớn hơn là 16384 (tương ứng với hình ảnh 128x128 được làm phẳng). Kết quả cho thấy như trong bảng trên, kiến trúc này là kiến trúc đạt được kết quả tốt nhất.

4. Kết luận

Bài báo này đã đề xuất một phương pháp phát hiện lỗ hổng hợp đồng thông minh mới bằng cách sử dụng kiến trúc CNN 1D. Kết quả đánh giá đã chỉ ra rằng mô hình được đề xuất có hiệu quả cao trong việc phát hiện các lỗ hổng trong mã hợp đồng thông minh. Phương pháp được đề xuất cho thấy hiệu quả cao hơn trong hiệu suất phát hiện với nhiều loại lỗ hổng khác nhau. Cụ thể, mô hình cho độ chính xác lên tới 98,18% so với các mô hình CNN 2D truyền thống. Từ những kết quả đánh giá này, chúng tôi tin rằng phương pháp được đề xuất có thể giúp cung cấp khả năng phát hiện lỗ hổng hợp đồng thông minh chính xác và hữu ích hơn để cải thiện môi trường phát triển hợp đồng thông minh an toàn.

TÀI LIỆU THAM KHẢO/ REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, no. 21260, pp.1-9, 2008.

[2] F. A. Sunny, P. Hajek, and M. Munk, "A Systematic Review of Blockchain Applications," *IEEE Access*, vol. 10, pp. 59155-59177, 2022.

[3] V. Buterin, "A next-generation smart contract and decentralized application platform," *White paper*, vol. 2, no. 37, pp. 1-36, 2014.

[4] C. Wu, "A review on recent progress of smart contract in blockchain," *IEEE Access*, vol. 10, pp. 50839-50863, 2022.

[5] S. S. Kushwaha, "Systematic review of security vulnerabilities in ethereum blockchain smart contract," *IEEE Access*, vol. 10, pp. 6605-6621, 2022.

[6] S. Hacked, "Slowmist Hacked," 2018. [Online]. Available: ttps://hacked.slowmist.io/en/. [Accessed June 26, 2023].

[7] Etherscan, "Etherscan," 2015. [Online]. Available: https://etherscan.io/. [Accessed June 22, 2023].

[8] EOS, "EOS," 2018. [Online]. Available: https://eos.io/. [Accessed June 26, 2023].

[9] TRON, "TRON," 2017. [Online]. Available: https://tron.network/. [Accessed June 26, 2023].

[10] Wikipedia, "The Dao Attack," 2016. [Online]. Available: https://en.wikipedia.org/wiki/. [Accessed June 26, 2023].

[11] L. Breidenbach, P. Daian, A. Juels, and E. G. Sirer, "An In-Depth Look at the Parity Multisig Bug," 2017. [Online]. Available: http://hackingdistributed.com/2017/07/22/deep-dive-parity-bug/. [Accessed June 26, 2023].

[12] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *2016 ACM SIGSAC Conf. Comput. Commun. Secure*, 2016, pp. 254-269.

[13] C. F. Torres, J. Schütte, and R. State, "Osiris: Hunting for integer bugs in Ethereum smart contracts," in *Proc. 34th Annu. Comput. Secur. Appl. Conf.*, 2018, pp. 664-676.

[14] Mythril, "Mythril," 2018. [Online]. Available: https://github.com/ConsenSys/mythril-classic. [Accessed June 26, 2023].

[15] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Bünzli, and M. Vechev, "Securify: Practical security analysis of smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 67-82.

[16] J. Feist, G. Grieco, and A. Groce, "Slither: A static analysis framework for smart contracts," in *Proc. IEEE/ACM 2nd Int. Workshop Emerg. Trends Softw. Eng. Blockchain (WETSEB)*, 2019, pp. 8-15.

[17] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov, "SmartCheck: Static analysis of Ethereum smart contracts," in *Proc. 1st Int. Workshop Emerg. Trends Softw. Eng. Blockchain*, 2018, pp. 9-16.

[18] W. Wang, J. Song, G. Xu, Y. Li, H. Wang, and C. Su, "ContractWard: Automated vulnerability detection models for Ethereum smart contracts," *IEEE Trans. Netw. Sci. Eng*, vol. 8, no. 2, pp. 1133-1144, 2021.

[19] H.-D. Huang, T. Ton, and H.-Y. Kao, "R2-d2: Color-inspired convolutional neural network (cnn)-based android malware detections," in *2018 IEEE international conference on big data (big data)*, 2018, pp. 2633-2642.

[20] R. Russell, L. Kim, L. Hamilton, T. Lazovich, J. Harer, O. Ozdemir, P. Ellingwood, and M. McConley, "Automated vulnerability detection in source code using deep representation learning," in *17th IEEE international conference on machine learning and applications (ICMLA)*, 2018, pp. 757-762.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1-12.

[22] T. Durieux, J. F. Ferreira, R. Abreu, and P. Cruz, "Empirical Review of Automated Analysis Tools on 47,587 Ethereum Smart Contracts," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 530-541.

[23] C. S. Yashavant, S. Kumar, and A. Karkare, "ScrawlD: A Dataset of Real World Ethereum Smart Contracts Labelled with Vulnerabilities," in *ArXiv:2202.11409*, 2022, pp. 1-5.

[24] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1-32, 2014.