

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

VŨ ANH KIỆT

KHÓA LUẬN TỐT NGHIỆP
PHÁT HIỆN LỖ HỔNG TRONG HỢP ĐỒNG THÔNG
MINH TRÊN MẠNG LIÊN CHUỖI KHỐI BẰNG
PHƯƠNG PHÁP HỌC MÁY
**SMART CONTRACT VULNERABILITIES AUTOMATIC
DETECTION ON THE CROSS-CHAIN NETWORK USING
MACHINE LEARNING**

CỬ NHÂN NGÀNH AN TOÀN THÔNG TIN

TP. Hồ Chí Minh - 2023

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

VŨ ANH KIỆT - 20520605

KHÓA LUẬN TỐT NGHIỆP
PHÁT HIỆN LỖ HỔNG TRONG HỢP ĐỒNG THÔNG
MINH TRÊN MẠNG LIÊN CHUỖI KHỐI BẰNG
PHƯƠNG PHÁP HỌC MÁY
**SMART CONTRACT VULNERABILITIES AUTOMATIC
DETECTION ON THE CROSS-CHAIN NETWORK USING
MACHINE LEARNING**

CỦ NHÂN NGÀNH AN TOÀN THÔNG TIN

GIẢNG VIÊN HƯỚNG DẪN:

TS. Phạm Văn Hậu
ThS. Trần Tuấn Dũng

TP. Hồ Chí Minh - 2023

LỜI CẢM ƠN

Đầu tiên, nhóm xin gửi lời cảm ơn chân thành đến Ban giám hiệu Trường Đại học Công nghệ Thông tin - Đại học Quốc Gia Thành Phố Hồ Chí Minh vì đã tạo điều kiện về cơ sở vật chất và điều kiện thuận lợi cho quá trình học tập và nghiên cứu cũng như việc thực hiện khoá luận. Nhóm xin trân trọng cảm ơn quý thầy cô giảng viên dạy tại trường vì đã truyền đạt những kiến thức bổ ích và những kinh nghiệm thực tế quý báu thông qua những tiết học. Nhóm xin chân thành cảm ơn thầy Phạm Văn Hậu và thầy Trần Tuấn Dũng đã tận tình hướng dẫn, góp ý và động viên nhóm trong suốt quá trình nghiên cứu. Các thầy không chỉ truyền đạt cho nhóm kiến thức mà còn dạy cách học, cách suy nghĩ và phương pháp nghiên cứu khoa học để hoàn thành khoá luận. Nhóm cũng xin gửi lời cảm ơn đến gia đình và bạn bè đã động viên, đóng góp ý kiến trong quá trình thực hiện công việc nghiên cứu. Cuối cùng, do kiến thức còn nhiều hạn chế nên đồ án chắc chắn không tránh khỏi những thiếu sót. Chúng tôi rất mong nhận được những nhận xét, ý kiến đóng góp, phê bình từ quý thầy cô để đồ án được hoàn thiện hơn.

Võ Anh Kiệt

MỤC LỤC

LỜI CẢM ƠN	i
MỤC LỤC	ii
DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT	iv
DANH MỤC CÁC HÌNH VẼ	v
DANH MỤC CÁC BẢNG BIỂU	vi
TÓM TẮT KHOÁ LUẬN	1
 CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI	2
1.1 Tổng quan tình hình nghiên cứu	2
1.2 Các công trình nghiên cứu liên quan	7
1.3 Mục tiêu, đối tượng, và phạm vi nghiên cứu	12
1.3.1 Mục tiêu nghiên cứu	12
1.3.2 Đối tượng nghiên cứu	12
1.3.3 Phạm vi nghiên cứu	12
1.3.4 Cấu trúc khoá luận tốt nghiệp	13
 CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	14
2.1 Tổng quan về các hệ thống phân tán	14
2.1.1 Tổng quan về hệ thống mạng Blockchain - Chuỗi khối . .	14
2.1.2 Tổng quan về hệ thống mạng liên chuỗi khối	17
2.2 Tổng quan về hợp đồng thông minh	19
2.2.1 Hợp đồng thông minh	19
2.2.2 Ngôn ngữ solidity	21
2.2.3 Các loại lỗ hổng trong hợp đồng	27
2.3 Các phương pháp phát hiện lỗ hổng tự động	34
2.3.1 Phương pháp học máy	34

2.3.2 Phương pháp học sâu	41
CHƯƠNG 3. MÔ HÌNH ĐỀ XUẤT	47
3.1 Tổng quan mô hình	47
3.1.1 Các thành phần chính của mô hình	47
3.1.2 Mô hình cầu nối	49
3.2 Phát hiện các lỗ hỏng bằng phương pháp học máy và học sâu . .	51
3.2.1 Xây dựng tập dữ liệu	51
3.2.2 Dán nhãn các mẫu và xử lý dữ liệu	53
3.2.3 Ứng dụng các mô hình học máy tổng việc phát hiện các lỗ hỏng	54
3.2.4 Ứng dụng các mô hình học sâu trong việc phát hiện các lỗ hỏng	55
CHƯƠNG 4. THÍ NGHIỆM VÀ ĐÁNH GIÁ	57
4.1 Thiết lập thực nghiệm	57
4.1.1 Môi trường thực nghiệm	57
4.1.2 Chỉ số đánh giá	58
4.1.3 Kịch bản thực nghiệm	60
4.2 Kết quả thực nghiệm	62
4.2.1 Kết quả về mặt hiệu suất	62
4.2.2 Kết quả về mặt thời gian	63
4.3 Thảo luận	64
CHƯƠNG 5. KẾT LUẬN	66
5.1 Kết luận	66
5.2 Hướng phát triển	67
TÀI LIỆU THAM KHẢO	68

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

BERT	Bidirectional Encoder Representations from Transformers
RoBERTa	Robustly Optimized BERT Pre-Training Approach
XGBoost	Extreme Gradient Boosting
SVM	Support Vector Machine
CNN	Convolutional Neural Network
LSTM	Long short term memory
FNN	Feedforward Neural Network
ML	Machine learning
DL	Deep learning
SC	Smart Contract
Smart Contract	Hợp đồng thông minh
ETH	Đồng tiền kỹ thuật số Ethereum

DANH MỤC CÁC HÌNH VẼ

Hình 1.1	Mô hình tổng quan của Xu và nhóm nghiên cứu	7
Hình 1.2	Mô hình tổng quan của Deng và nhóm nghiên cứu	8
Hình 1.3	Mô hình tổng quan của Huang và nhóm nghiên cứu	9
Hình 1.4	Mô hình tổng quan của Jiang và nhóm nghiên cứu	10
Hình 1.5	Mô hình tổng quan của Parizi và nhóm nghiên cứu	11
Hình 2.1	Một số lĩnh vực ứng dụng của công nghệ Blockchain	15
Hình 2.2	Các ứng dụng khi sử dụng nền tảng blockchain	17
Hình 2.3	Giao tiếp liên chuỗi giữa hai chuỗi khối thông qua sidechain	18
Hình 2.4	Sự khác nhau giữa hợp đồng truyền thống và hợp đồng thông minh	21
Hình 2.5	Quá trình biên dịch và thực thi hợp đồng thông minh	23
Hình 2.6	Sự tương quan của source code, bytecode và opcode của hợp đồng thông minh	23
Hình 2.7	Cấu trúc của bytecode trong hợp đồng thông minh	24
Hình 2.8	Cấu trúc của opcode trong hợp đồng thông minh	25
Hình 2.9	Mô phỏng lỗi hỏng Reentrancy	28
Hình 2.10	Mô phỏng lỗi hỏng Integer Overflow	29
Hình 2.11	Mô phỏng lỗi hỏng Integer Underflow	30
Hình 2.12	Mô phỏng lỗi hỏng Unprotected Ether Withdrawal	31
Hình 2.13	Mô phỏng lỗi hỏng Timestamp Dependence	32
Hình 2.14	Mô phỏng lỗi hỏng Front Running	33
Hình 2.15	Mô hình học máy Decision Tree	34
Hình 2.16	Mô hình học máy Random Forest	36
Hình 2.17	Mô hình học máy XGBoost	37

Hình 2.18 Mô hình học máy Support Vector Machine	39
Hình 2.19 Mô hình học máy Logistic Regression	40
Hình 2.20 Mô hình học sâu Convolutional Neural Network	41
Hình 2.21 Mô hình học sâu Long Short-Term Memory	43
Hình 2.22 Mô hình học sâu Feedforward neuron network	44
Hình 2.23 Mô hình học sâu RoBERTa	46
Hình 3.1 Mô hình tổng quan	48
Hình 3.2 Các bước chuyển đổi dữ liệu qua cầu nối Sidechain	50
Hình 3.3 Phân bố mẫu lành tính và độc hại	51
Hình 3.4 Phân bố mẫu lành tính và độc hại tương ứng với các lỗ hổng	52
Hình 3.5 Quá trình xử lý dữ liệu	54
Hình 4.1 Thời gian thực thi của các mô hình học máy trong việc phát hiện lỗ hổng bảo mật trong các hợp đồng thông minh liên chuỗi .	64

DANH MỤC CÁC BẢNG BIỂU

Bảng 4.1	Môi trường thực nghiệm mạng liên chuỗi	57
Bảng 4.2	Môi trường thực nghiệm các phương pháp học máy	58
Bảng 4.3	Hiệu suất của các mô hình học máy trong việc phát hiện lỗ hởng bảo mật trong các hợp đồng thông minh liên chuỗi	62
Bảng 4.4	Thời gian thực thi của các mô hình học máy trong việc phát hiện lỗ hởng bảo mật trong các hợp đồng thông minh liên chuỗi . .	63

TÓM TẮT KHOÁ LUẬN

Các hợp đồng thông minh là những ứng dụng phi tập trung hoạt động trên nền tảng công nghệ blockchain, tự thực thi các điều khoản đã thỏa thuận một cách minh bạch và chính xác. Trong vài năm gần đây, rất nhiều hợp đồng thông minh đã được phát triển và ứng dụng trên mạng chuỗi chéo tạo điều kiện cho tương tác và trao đổi dữ liệu giữa các blockchain khác nhau. Tuy nhiên, bất kỳ ứng dụng nào được triển khai đều tiềm ẩn những rủi ro về bảo mật đặc biệt là sự xuất hiện các lỗ hổng trong các hợp đồng thông minh khiến chúng dễ bị tấn công, đe dọa tới an ninh mạng.

Một số các nghiên cứu đã tập trung vào việc xác định và phát hiện các lỗ hổng trong hợp đồng thông minh bằng các phương pháp kiểm tra ký tự và thực thi nhưng các phương pháp hiện tại vẫn chưa đạt được khả năng phân tích toàn diện. Vì vậy ở nghiên cứu này, nhóm đề xuất các phương pháp học máy và phương pháp học sâu trong việc phân tích các lỗ hổng này.

Ở nghiên cứu này, chúng tôi giới thiệu các phương pháp học máy và học sâu dựa trên ChainSniper - một khung phân tích tích hợp học máy dựa trên sidechain để tự động đánh giá lỗ hổng hợp đồng thông minh chéo chuỗi. Phương pháp mà nhóm đề xuất là xây dựng một tập dữ liệu quy mô lớn gồm 300 đoạn mã có gán nhãn thủ công, được gọi là CrossChainSentinel, đã được biên soạn để huấn luyện các mô hình phân biệt mã dễ bị tấn công và mã an toàn. Các đoạn mã này bao gồm các lỗ hổng: Reentrancy, Interger Overflow/Underflow và Unprotected Ether Withdrawl. Kết quả thử nghiệm chứng minh khả năng ứng dụng học máy và học sâu giúp nâng cao hiệu quả kiểm tra hợp đồng thông minh cho các ứng dụng phi tập trung phân tán trên nhiều blockchain. Độ chính xác phát hiện đạt mức đáng kể, khẳng định tiềm năng của ChainSniper trong việc tăng cường an ninh thông qua đánh giá tự động và toàn diện mã hợp đồng.

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

Phần này đưa ra tổng quan về bối cảnh vấn đề và các nghiên cứu trước đó. Đồng thời, chúng tôi cũng mô tả phạm vi và kết cấu của luận văn.

1.1. Tổng quan tình hình nghiên cứu

Sự ra đời của công nghệ blockchain đã thúc đẩy việc xây dựng các mạng phi tập trung, cung cấp khả năng lưu trữ, trao đổi và xác minh dữ liệu một cách đáng tin cậy và có tính xác thực cao [1, 2]. Điều này nhờ vào cơ chế đồng thuận phân tán giữa các nút tham gia mạng. Tuy nhiên, để đạt được những tính năng vượt trội đó, các hệ thống blockchain thường được xây dựng độc lập với các quy tắc và giao thức riêng biệt. Điều này dẫn đến tình trạng các blockchain không thể tương tác và giao tiếp với nhau một cách trực tiếp.

Chính vì thế, công nghệ liên chuỗi (cross-chain) đã ra đời như một giải pháp cho phép kết nối và tương tác giữa nhiều mạng blockchain với nhau [3]. Công nghệ này hỗ trợ việc chuyển giao tài sản số giữa các chuỗi khối khác nhau một cách an toàn và hiệu quả [4]. Điều này thúc đẩy khả năng tích hợp, mở rộng và phát triển hệ sinh thái blockchain [5]. Các giải pháp liên chuỗi như Polkadot đã cung cấp một framework cho phép nhiều blockchain độc lập, có cấu trúc và chức năng khác nhau có thể giao tiếp, tương tác với nhau. Đây là một bước tiến quan trọng trong sự phát triển của ngành công nghệ blockchain và các ứng dụng phi tập trung trên thế giới.

Công nghệ liên chuỗi nhằm kết nối các hệ sinh thái blockchain bị cô lập, cho phép tài sản và dữ liệu được chuyển giao và chia sẻ một cách thuận tiện giữa các blockchain khác nhau [6]. Đóng góp này rất quan trọng trong việc giải quyết các hạn chế về khả năng mở rộng, tốc độ và chức năng mà các blockchain riêng lẻ thường gặp phải. Hiện tại tồn tại ba cách tiếp cận chính để kết nối và cho

phép chuyển tài sản giữa các chuỗi khối khác nhau: các giải pháp công chứng, khóa băm, và các relays/sidechains [7–9]. Cơ chế công chứng là phương pháp đồng thuận trong đó các bên thứ ba tin cậy (công chứng viên) xác minh giao dịch bằng chữ ký số trước khi chúng được thêm vào blockchain, nhằm ngăn giao dịch kép. Giải pháp khóa băm có thể triển khai thông qua cổng để truy cập các hợp đồng khóa thời gian khóa băm (Hash Time Lock Contract - HTLC) trên blockchain từ xa, đảm bảo việc nhận thanh toán trước khi phục dụng tài sản trên blockchain đích. Sidechains là các blockchain phụ được gắn nối với blockchain chính thông qua thanh gài 2 chiều, cho phép chuyển tài sản giữa các chuỗi, thêm tính năng mới cho blockchain chính mà không cần sửa đổi giao thức, giải quyết các hạn chế về khả năng mở rộng và quyền riêng tư. Tuy nhiên, giao thức tương tác chuỗi khối này vẫn tồn tại các nguy cơ về bảo mật và quyền riêng tư cần giải quyết.

Hợp đồng thông minh được coi như các chương trình máy tính có khả năng tự động thực thi các điều khoản và điều kiện của một thỏa thuận mà không cần sự can thiệp của bên thứ ba [10]. Chúng cho phép tự động hóa quá trình xác minh và bảo đảm các giao dịch tuân thủ theo logic đã được lập trình sẵn. Tuy nhiên, hợp đồng thông minh cũng chứa đựng những rủi ro tiềm ẩn, đặc biệt là các cuộc tấn công lợi dụng lỗ hổng bảo mật trong mã nguồn của chúng.

Các cuộc tấn công chéo chuỗi cụ thể nhắm vào việc khai thác những yếu điểm trong các hợp đồng thông minh hoạt động trên nhiều blockchain khác nhau. Chúng có thể dẫn đến thất thoát tài chính nghiêm trọng. Ví dụ, một vụ tấn công gần đây đã khai thác lỗ hổng trong một hợp đồng thông minh chéo chuỗi quản lý tài sản trị giá hơn 600 triệu đô la [11]. Năm 2016, dự án gây quỹ The DAO trên nền tảng Ethereum cũng bị tấn công dẫn đến việc đánh cắp hơn 50 triệu đô la do lỗ hổng trong hợp đồng thông minh. Các sự cố tấn công hợp đồng thông minh tiếp tục xảy ra, ví dụ như vụ tấn công vào ví Parity tháng 7/2017 khiến mất mát 30 triệu đô la tiền điện tử Ether. Hay vụ đánh cắp gần 300.000 đô la từ nền tảng KingDice tháng 8/2017 cũng do lợi dụng lỗ hổng trong

mã hợp đồng. Gần đây nhất là loạt vụ tấn công vào các hợp đồng thông minh trên Binance Smart Chain năm 2021, trong đó có vụ đánh cắp hơn 200 triệu đô la thông qua hợp đồng của Venus Protocol [12, 13]. Như vậy, việc phân tích và bảo mật hợp đồng thông minh trước khi triển khai là vô cùng quan trọng để hạn chế rủi ro mất mát tài sản.

Hợp đồng thông minh là các giao thức kỹ thuật số nhằm mục đích đơn giản hóa, xác minh hoặc thực thi việc thương lượng hay thực hiện hợp đồng. Chúng có nhiều ứng dụng, bao gồm các dịch vụ tài chính, thị trường dự đoán và Internet vạn vật [14]. Những hợp đồng này hoạt động trơn tru trên các nền tảng blockchain, tự động kích hoạt các hành động dựa trên các điều kiện định sẵn, loại bỏ nhu cầu về trung gian. Do đó, chúng tạo điều kiện cho các giao dịch không cần tin tưởng và tự động hóa các quy trình trong mạng lưới blockchain. Do sự phổ biến ngày càng tăng của công nghệ blockchain và áp dụng rộng rãi trên nhiều ngành nghề, lĩnh vực, việc phân tích và kiểm tra bảo mật cho các hợp đồng thông minh là vô cùng cấp bách và quan trọng trước khi chúng được triển khai thực tế [15]. Bởi lẽ, các hợp đồng thông minh hoạt động dựa trên các đoạn mã tự thực thi, bất kỳ lỗi hỏng hoặc sai sót nào cũng có thể dẫn đến những hậu quả nghiêm trọng. Mặc dù công tác kiểm tra và phân tích mã sử dụng các phương pháp thủ công là rất cần thiết, tuy nhiên quá trình này lại tiêu tốn rất nhiều thời gian, công sức và dễ phát sinh sai sót của con người. Đặc biệt, trong các hệ thống blockchain liên kết chéo như Polkadot, nơi hỗ trợ sự giao tiếp giữa nhiều chuỗi khối khác nhau, việc bảo mật và kiểm thử càng trở nên phức tạp và khó khăn hơn rất nhiều. Hiện tại, hệ sinh thái Ethereum đang phải đối mặt với rất nhiều lỗi hỏng bảo mật nghiêm trọng như các cuộc tấn công lặp lại (Reentrancy) [16], tràn số (Overflow/Underflow) [17] cũng như việc rút token mà không được bảo vệ trong các hợp đồng thông minh [18]. Những vấn đề này tiềm ẩn rủi ro lớn đến các ứng dụng phi tập trung (dApps) đang hoạt động trên Ethereum. Cần có những giải pháp, công cụ hữu hiệu hơn để kiểm thử bảo mật cho hợp đồng thông minh, theo kịp với tốc độ phát triển nhanh chóng của

các ứng dụng blockchain hiện nay.

Các mạng chuỗi liên kết chéo (cross-chain) cho phép xây dựng các ứng dụng phi tập trung phức tạp hơn thông qua kết nối giữa nhiều blockchain, tuy nhiên đồng thời cũng làm gia tăng số lượng các lỗ hổng tiềm ẩn mà tin tặc có thể lợi dụng để tấn công [19]. Một trong những mối đe dọa lớn nhất đó là cuộc tấn công lặp lại (reentrancy), cho phép đối tượng tấn công lặp đi lặp lại lời gọi hàm của hợp đồng thông minh trước khi hoàn thành yêu cầu trước đó, dẫn đến các hậu quả khó lường [20]. Bên cạnh đó, các lỗ hổng tràn số và underflow cũng rất nguy hiểm, xảy ra khi giá trị vượt quá giới hạn cho phép, gây ra hậu quả khôn lường [21]. Đặc biệt, lỗ hổng rút tiền điện tử mà không được xác thực (unprotected ether withdrawal) cũng đang nhận được sự quan tâm, khi một hợp đồng thông minh không minh chính xác yêu cầu rút tiền, cho phép hacker rút Ether một cách bất hợp pháp [22]. Những hậu quả từ các lỗ hổng bảo mật nói trên đã và đang lan rộng trong hệ sinh thái Ethereum, gây thiệt hại tài chính lớn cho nhiều bên liên quan [12]. Thực trạng đáng báo động này cho thấy tầm quan trọng của việc xây dựng và tuân thủ nghiêm ngặt các chính sách, quy trình bảo mật trong quá trình phát triển các ứng dụng blockchain. Các công ty, tổ chức cần chú trọng đầu tư nguồn lực cho hoạt động kiểm tra, phân tích lỗ hổng và cải thiện mã nguồn của sản phẩm trước khi đưa vào sử dụng. Điều này góp phần hạn chế rủi ro và thiệt hại có thể xảy ra do lỗi bảo mật của các ứng dụng blockchain.

Học máy là một nhánh của trí tuệ nhân tạo cho phép máy tính tự động học hỏi, cải thiện dựa trên kinh nghiệm mà không cần lập trình rõ ràng từng bước thực hiện. Các tiến bộ gần đây trong học máy, đặc biệt là sự ra đời của kiến trúc học sâu, đã cho phép xử lý các khối lượng dữ liệu lớn, đưa ra dự đoán và quyết định vượt xa khả năng con người trong một số tác vụ [23, 24]. Đối với các hợp đồng thông minh, các mô hình máy học có thể phân tích mã để xác định các vấn đề như lỗ hổng vòng lặp, các cuộc tấn công tràn số nguyên, và tấn công từ chối dịch vụ (DoS) [25]. Bằng cách học từ các ví dụ về cả hợp đồng thông minh dễ bị

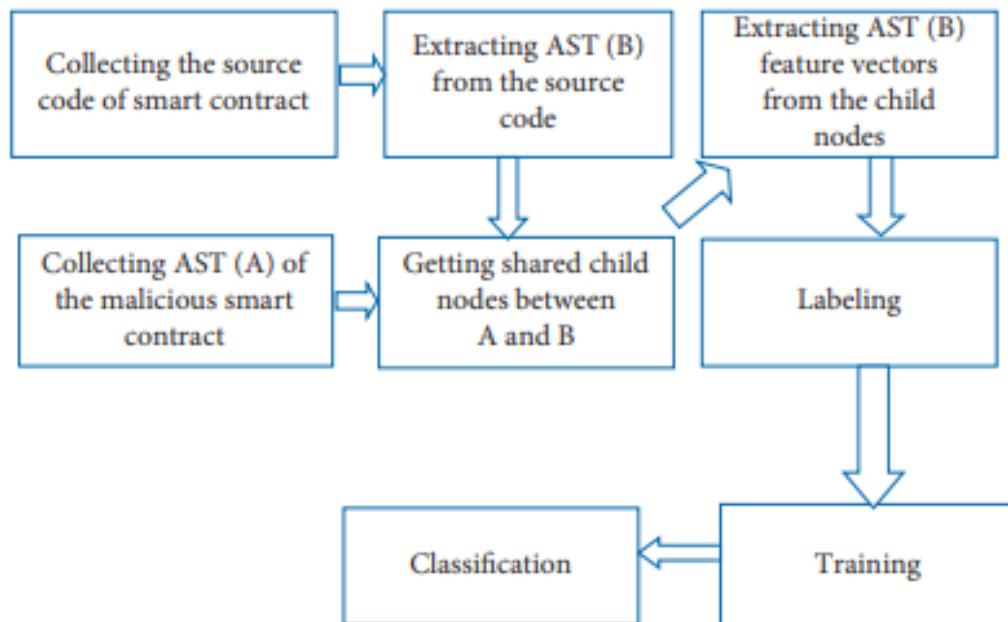
tổn thương và an toàn, các mô hình máy học có thể xác thực các hợp đồng một cách hiệu quả ở quy mô lớn. Việc ứng dụng học máy trong phân tích và phân loại mã nguồn đang cho thấy những hứa hẹn đáng khích lệ. Các nghiên cứu dựa trên ý tưởng này để xây dựng các mô hình có khả năng phát hiện tự động các đoạn code chứa lỗi hoặc lỗ hổng bảo mật. Chúng tôi cũng phát triển một bộ dữ liệu mã nguồn Solidity chú trọng đến phát hiện lỗ hổng tiềm ẩn. Sử dụng bộ dữ liệu này, chúng tôi đào tạo nhiều mô hình phân loại khác nhau như cây quyết định, rừng ngẫu nhiên, SVM, LSTM, CNN,... nhằm nâng cao khả năng phát hiện các đoạn code dễ bị tấn công. Học sâu là một nhánh của học máy sử dụng các mạng nơ-ron nhân tạo có kiến trúc sâu với nhiều tầng. Chúng có thể tự động trích xuất các đặc trưng và mẫu từ dữ liệu, giúp xử lý hiệu quả đầu vào phức tạp như hình ảnh, âm thanh, ngôn ngữ tự nhiên. Ứng dụng học sâu trong phân tích mã nguồn cũng cho thấy kết quả tích cực, ví dụ mô hình Roberta được chứng minh có độ chính xác cao trong nhiệm vụ phân loại và dự đoán lỗ hổng. Đây là hướng nghiên cứu hứa hẹn nhiều tiềm năng để tự động hóa quá trình rà soát và bảo mật source code.

Nhằm tăng cường tính bảo mật và độ tin cậy cho các giao dịch chuyển tiền giữa các blockchain thông qua cầu nối sidechain, chúng tôi đã nghiên cứu và phát triển giải pháp ChainSniper. Giải pháp này tích hợp các tính năng ghi lại nhật ký sự kiện một cách chặt chẽ, quét tự động các hợp đồng thông minh để phát hiện lỗ hổng tiềm ẩn và giám sát thời gian thực khi chúng được triển khai. Để đánh giá độ chính xác và hiệu quả của ChainSniper cũng như các mô hình học máy trong việc dự đoán lỗ hổng hợp đồng, chúng tôi đã biên soạn một tập dữ liệu mới có tên CrossChainSentinel. Tập dữ liệu này bao gồm 300 mẫu hợp đồng thông minh với chủ thích và nhãn thủ công. Cụ thể, 158 hợp đồng trong số đó là lành tính, 142 hợp đồng còn lại đã được tiêm chèn một số lỗ hổng phổ biến như lỗi reentrancy, tràn số, underflow và không được bảo vệ khi rút tiền điện tử. Sau đó, chúng tôi đã huấn luyện và so sánh nhiều mô hình học máy khác nhau trên tập dữ liệu CrossChainSentinel để đánh giá khả năng dự đoán chính xác

của chúng, nhận diện được các hợp đồng độc hại trong môi trường chéo chuỗi. Kết quả thử nghiệm ban đầu cho thấy các kỹ thuật học máy có thể được ứng dụng hiệu quả trong việc nâng cao độ an toàn cho quá trình kiểm toán và phân tích lỗ hổng của các hợp đồng thông minh trong môi trường liên kết chéo giữa nhiều blockchain với nhau.

1.2. Các công trình nghiên cứu liên quan

Hiện tại, đã có một số nghiên cứu đáng kể đề cập đến vấn đề nhóm đang nghiên cứu như Xu và nhóm nghiên cứu [26] trình bày một cái nhìn tổng quan toàn diện về ứng dụng các kỹ thuật học máy để phát hiện lỗ hổng trong hợp đồng thông minh (**Hình 1.1**). Họ sử dụng chung các nút con để phân tích và kết hợp mô hình k láng giềng gần nhất nhằm nâng cao hiệu quả phát hiện lỗ hổng. Qua các thử nghiệm so sánh, phương pháp của Xu vượt trội hơn công cụ Oyente và SmartCheck về độ chính xác. Tuy nhiên, mô hình chủ yếu tập trung vào ngôn ngữ Solidity nên cần tinh chỉnh thêm để xác định chính xác các dòng code dễ bị lỗi.

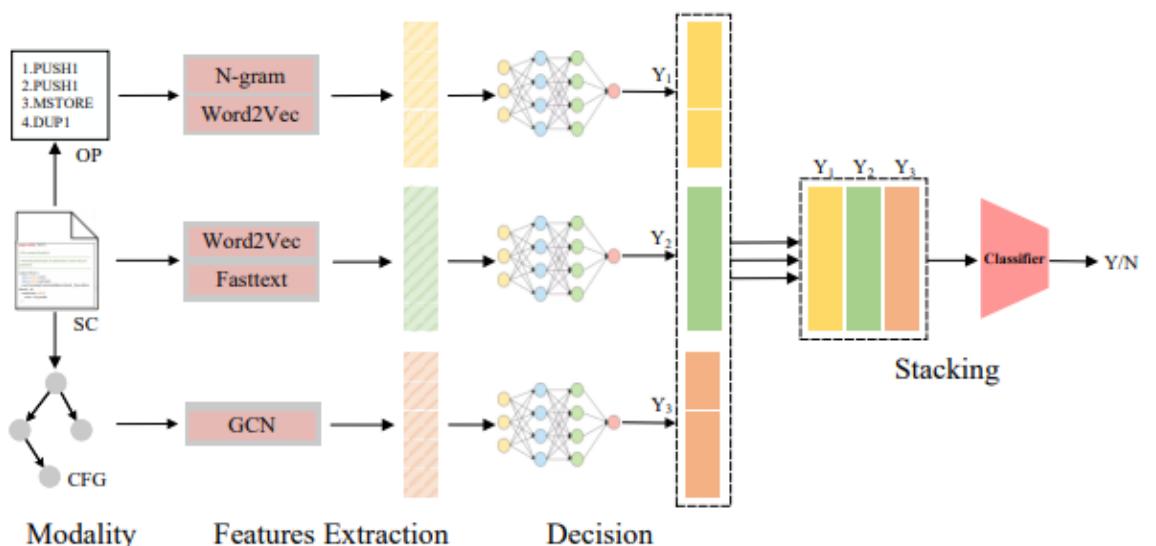


Hình 1.1: Mô hình tổng quan của Xu và nhóm nghiên cứu

Các nỗ lực sơ khai đã được ghi nhận trong bài nghiên cứu [27] đã thực hiện một đánh giá tổng quan lấy ý kiến đa chiều (multivocal literature review - MLR) về những thách thức bảo mật và riêng tư trong tương tác giữa các blockchain. Các tác giả đã xác định một số lỗ hổng then chốt bao gồm: các cuộc tấn công loại wormhole nhắm vào các lỗ hổng bảo mật để đánh cắp tài sản, tấn công từ chối dịch vụ làm tê liệt hệ thống, tấn công dựa trên thời gian thực hiện các giao dịch, sử dụng mật mã không tương thích giữa các blockchain dẫn đến mất tài sản trong quá trình chuyển đổi, và rò rỉ thông tin định danh trong các hợp đồng khóa thời gian băm mà các bên sử dụng để bảo mật giao dịch.

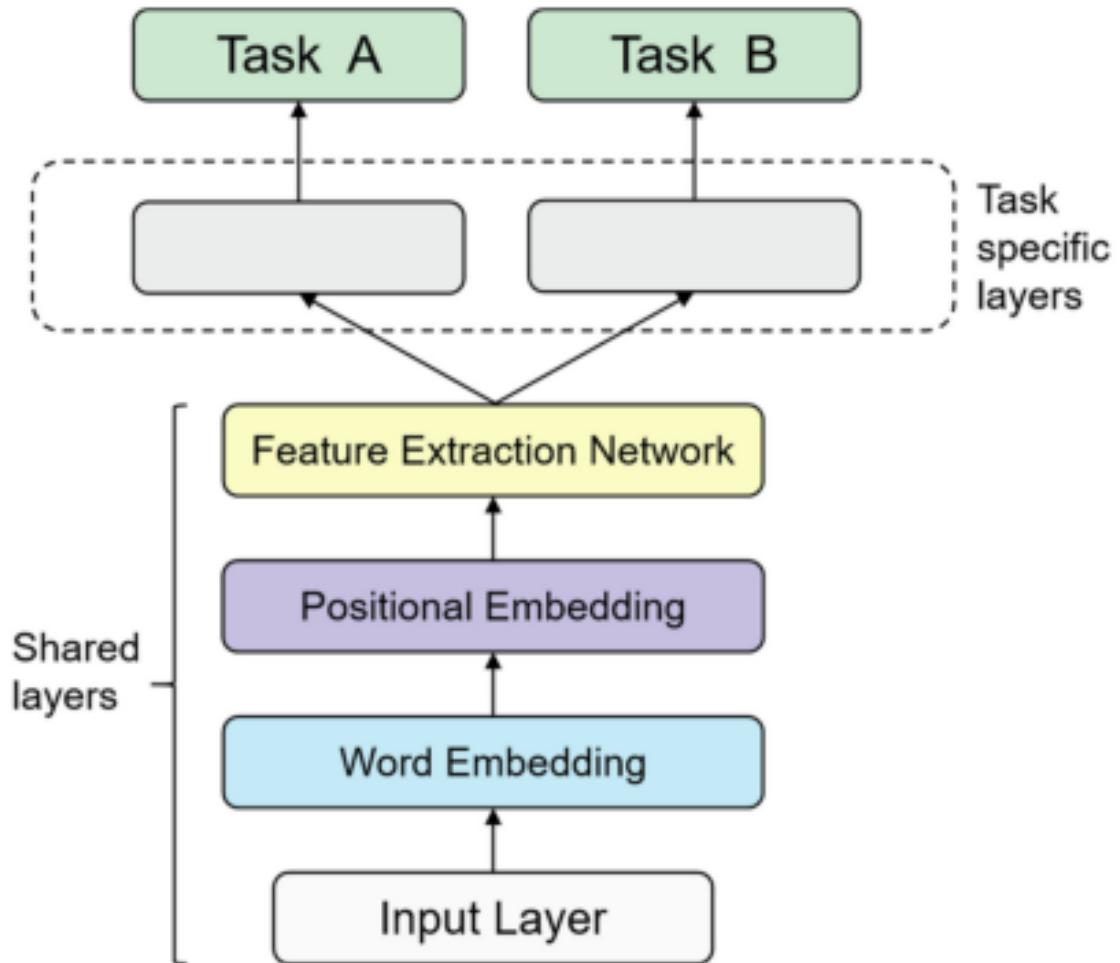
Trong một nghiên cứu tiên phong [28], He và cộng sự đã làm sáng tỏ phổ các lỗ hổng trong hợp đồng thông minh và đề xuất các biện pháp khắc phục nhằm giảm thiểu những rủi ro này. Họ đã tóm tắt ngắn gọn và so sánh giữa các công cụ hiện có được tinh chỉnh để giải quyết các lỗ hổng đó.

Mặt khác, Deng và nhóm nghiên cứu [29] đề xuất phương pháp mới phát hiện lỗ hổng hợp đồng thông minh dựa trên học sâu và kết hợp nhiều chế độ quyết định (**Hình 1.2**). Họ trích xuất 5 đặc trưng khác nhau đại diện cho các hợp đồng và đạt độ chính xác cao nhờ quy trình tinh vi. Tuy nhiên, nghiên cứu bỏ qua học không giám sát dù nó có tiềm năng cho vấn đề phát hiện lỗ hổng.



Hình 1.2: Mô hình tổng quan của Deng và nhóm nghiên cứu

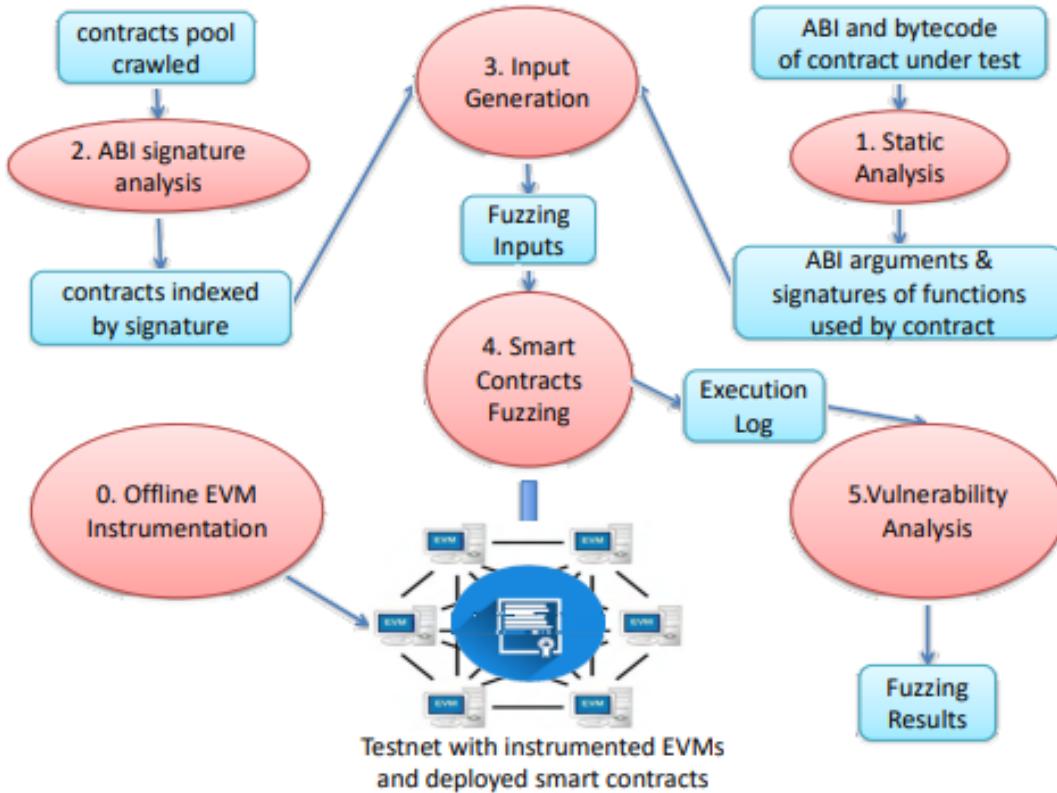
Huang và nhóm nghiên cứu [30] đề xuất mô hình phát hiện lỗ hổng hợp đồng thông minh dựa trên học máy đa nhiệm (**Hình 1.3**). Cụ thể, mô hình bao gồm lớp dưới chia sẻ để trích xuất đặc trưng từ các lệnh, và các nhánh nhiệm vụ phụ trợ để phát hiện và nhận dạng lỗ hổng. Thông qua thiết lập nhiệm vụ nhận dạng bổ trợ, mô hình học các đặc trưng có hướng để nâng cao khả năng dự đoán. Kết quả thử nghiệm cho thấy phương pháp xác định hiệu quả các lỗi như toán học, lặp lại và các cuộc gọi không xác định. Nhìn chung, học máy đa nhiệm rất có tiềm năng trong việc tự động hóa phân tích bảo mật của các hợp đồng.



Hình 1.3: Mô hình tổng quan của Huang và nhóm nghiên cứu

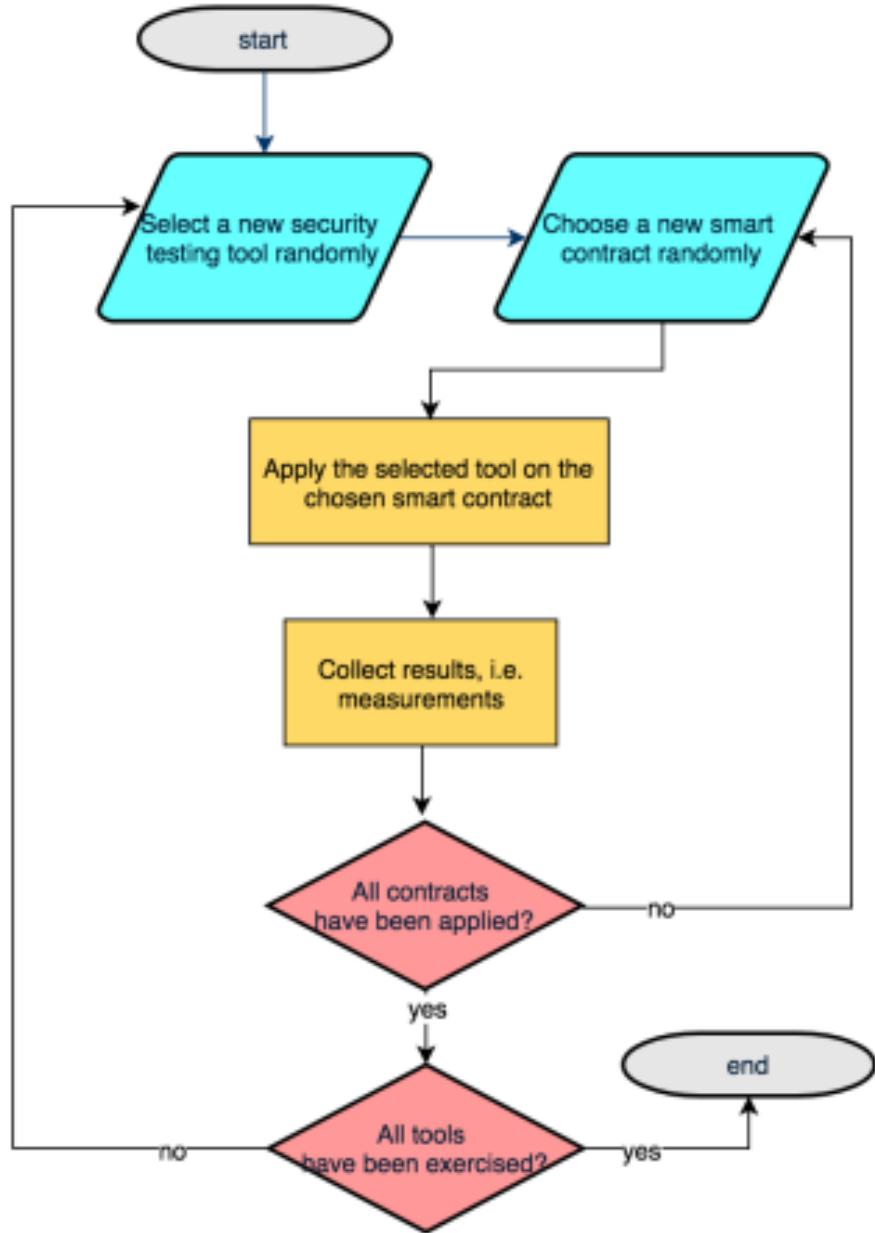
Bên cạnh đó, Jiang và cộng sự [31] chứng minh độ hiệu quả của kỹ thuật fuzzing và giám sát thời gian chạy đối với việc phát hiện lỗ hổng hợp đồng

(**Hình 1.4**). Công cụ ContractFuzzer của họ phân tích đặc tả ABI để sinh test case, xác định các biến kiểm soát để kiểm tra lỗ hổng, ghi nhật ký chi tiết EVM và giám sát thực thi để đánh dấu các lỗi tiềm ẩn. Thử nghiệm cho thấy công cụ phát hiện chính xác hơn 450 lỗ hổng, bao gồm cả lỗi DAO.



Hình 1.4: Mô hình tổng quan của Jiang và nhóm nghiên cứu

Ngoài ra, Parizi và cộng sự [32] đã đóng góp đáng kể vào lĩnh vực này bằng cách giới thiệu một công cụ kiểm thử toàn diện được thiết kế để phát hiện lỗ hổng, minh họa kết quả của công cụ của họ, và cung cấp các so sánh sâu sắc với các đối tác hiện có (**Hình 1.5**). Tuy nhiên, các nghiên cứu này không áp dụng cho khái niệm chuỗi khối liên kết ngang (cross-chain) với các lỗ hổng trong hợp đồng thông minh.



Hình 1.5: Mô hình tổng quan của Parizi và nhóm nghiên cứu

1.3. Mục tiêu, đối tượng, và phạm vi nghiên cứu

1.3.1. Mục tiêu nghiên cứu

Mục tiêu của đề tài này là nghiên cứu và phát triển một phương pháp sử dụng học máy để phát hiện lỗ hổng trên các hợp đồng thông minh trong môi trường mạng liên chuỗi khối. Điều này nhằm tăng cường tính bảo mật và độ tin cậy của các hợp đồng thông minh, từ đó giúp ngăn chặn và giảm thiểu các rủi ro tiềm ẩn, bảo vệ quyền lợi của các bên tham gia.

1.3.2. Đối tượng nghiên cứu

Các đối tượng của nghiên cứu bao gồm:

- Công nghệ chuỗi khối
- Công nghệ liên chuỗi
- Bảo mật smart contract
- Công nghệ học máy

1.3.3. Phạm vi nghiên cứu

Phạm vi của đề tài bao gồm:

- Tìm hiểu về công nghệ mạng liên chuỗi khối và triển khai tấn công trên mạng liên chuỗi: Nắm vững kiến thức về cấu trúc, hoạt động và ứng dụng của mạng liên chuỗi khối, đặc biệt về hợp đồng thông minh trên nền tảng mạng liên chuỗi và thực hiện tấn công trên mạng liên chuỗi khối.
- Xác định lỗ hổng và sự cần thiết phát hiện lỗ hổng: Định rõ các lỗ hổng phổ biến trên hợp đồng thông minh trên mạng liên chuỗi và lý do cần phải phát hiện chúng.

- Thiết kế mô hình học máy cho việc phát hiện lỗ hổng: Xây dựng một mô hình học máy phù hợp để phát hiện lỗ hổng trên hợp đồng thông minh trên mạng liên chuỗi, sử dụng các kỹ thuật học máy hiện đại và tối ưu hóa để đảm bảo hiệu suất và độ chính xác cao.
- Đánh giá mô hình: Áp dụng mô hình phát hiện lỗ hổng đã thiết kế vào một số hợp đồng thông minh thực tế trên mạng liên chuỗi khôi và đánh giá hiệu suất của mô hình.

1.3.4. Cấu trúc khoá luận tốt nghiệp

Cấu trúc khoá luận tốt nghiệp bao gồm:

- Chương 1: Tổng quan đề tài Giới thiệu tổng quan về đề tài và các công trình nghiên cứu liên quan
- Chương 2: Trình bày các cơ sở lý thuyết và nền tảng liên quan đến đề tài
- Chương 3: Trình bày mô hình ChainSniper được sử trong việc phát hiện lỗ hổng trong hợp đồng thông minh trong mạng liên chuỗi
- Chương 4: Trình bày kết quả thực nghiệm và đánh giá kết quả
- Chương 5: Kết luận và hướng phát triển

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Trong phần này, chúng ta sẽ đào sâu vào lĩnh vực nghiên cứu trước đó liên quan đến các hệ thống phân tán, hợp đồng thông minh và các lỗ hổng, và các phương pháp được sử dụng để phát hiện những lỗ hổng này.

2.1. Tổng quan về các hệ thống phân tán

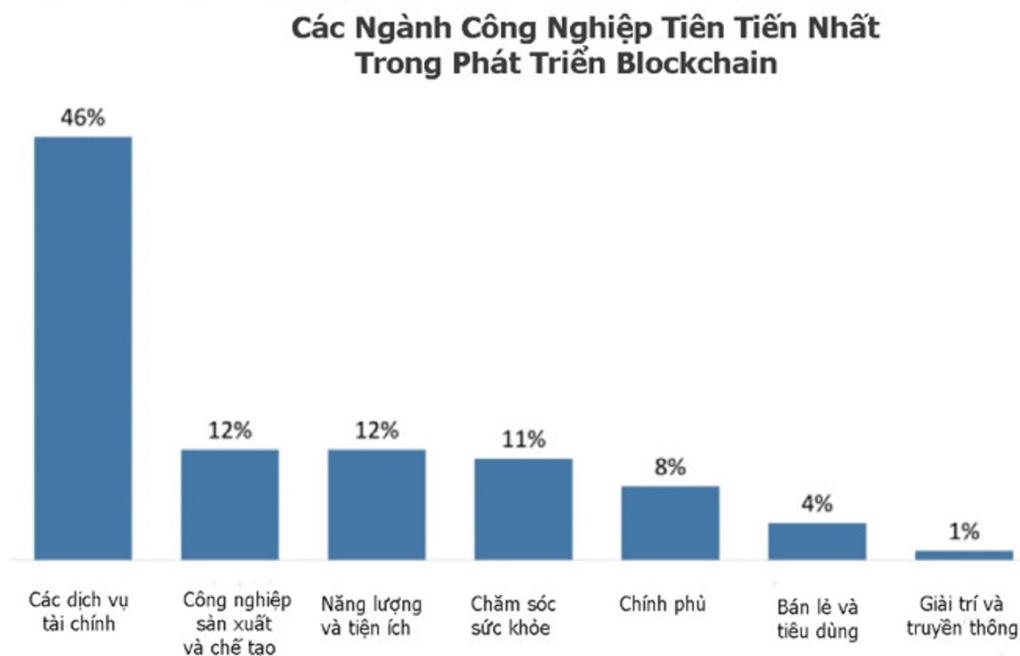
2.1.1. *Tổng quan về hệ thống mạng Blockchain - Chuỗi khối*

Blockchain là một công nghệ cách mạng đang thay đổi cách thức chúng ta tương tác và trao đổi thông tin trong thế giới số. Xuất phát từ việc hỗ trợ các loại tiền điện tử như Bitcoin, blockchain hiện nay đã vượt thoát khỏi giới hạn ban đầu để trở thành một nền tảng then chốt cho nhiều ứng dụng khác. Về bản chất, blockchain là một chuỗi các khối dữ liệu được liên kết với nhau một cách an toàn và phi tập trung. Nó tạo nên một hệ thống phân tán, minh bạch, bất biến và đồng thuận giữa các thành viên trong mạng.

So với các cơ sở dữ liệu truyền thống, blockchain có một số đặc điểm nổi bật:

- **An toàn, bảo mật:** Dữ liệu được mã hóa và xác thực bằng công nghệ mật mã, ngăn chặn nguy cơ giả mạo.
- **Minh bạch:** Tất cả các thành viên đều có quyền truy cập và xác minh các giao dịch trên blockchain.
- **Phi tập trung:** Dữ liệu được lưu trữ và quản lý trên một mạng lưới máy tính phân tán, không có điểm trung tâm. Điều này giúp hệ thống hoạt động liên tục và không bị lỗi đơn điểm.
- **Bất biến:** Dữ liệu sau khi ghi vào blockchain sẽ không thể thay đổi được nữa.

Với những ưu điểm vượt trội đó, blockchain đang được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau thúc đẩy mạnh mẽ sự phát triển của công nghệ mới (**Hình 2.1**). Các chuyên gia công nghệ dự đoán trong tương lai blockchain sẽ chuyển đổi mạnh mẽ mọi khía cạnh của cuộc sống con người.



Hình 2.1: Một số lĩnh vực ứng dụng của công nghệ Blockchain

Cơ sở lý thuyết của Blockchain dựa trên một vài nguyên lý cốt lõi:

- Thứ nhất, các khối dữ liệu (block) được liên kết với nhau bằng cách sử dụng mã băm (hash). Mã băm này đảm bảo tính toàn vẹn của dữ liệu và kết nối logic giữa các khối với nhau. Các khối mới được thêm vào cuối chuỗi theo đúng trình tự thời gian, tạo nên một lịch sử giao dịch không thể bị thay đổi hay làm giả mạo.
- Thứ hai, cơ chế đồng thuận giữa các nút trong mạng blockchain. Cơ chế phổ biến nhất là Proof of Work (PoW) và Proof of Stake (PoS). Điều này đảm bảo rằng mọi nút đều chấp nhận và thống nhất về trạng thái, dữ liệu hiện tại của chuỗi khối. Nó giúp blockchain duy trì tính an toàn, minh bạch và phi tập trung.

- Thứ ba, khả năng lập trình và thực thi các hợp đồng thông minh (smart contracts) trên nền tảng blockchain. Điều này cho phép tự động hóa quá trình giao dịch, thực hiện các điều khoản hợp đồng mà không cần sự can thiệp của bên thứ ba.

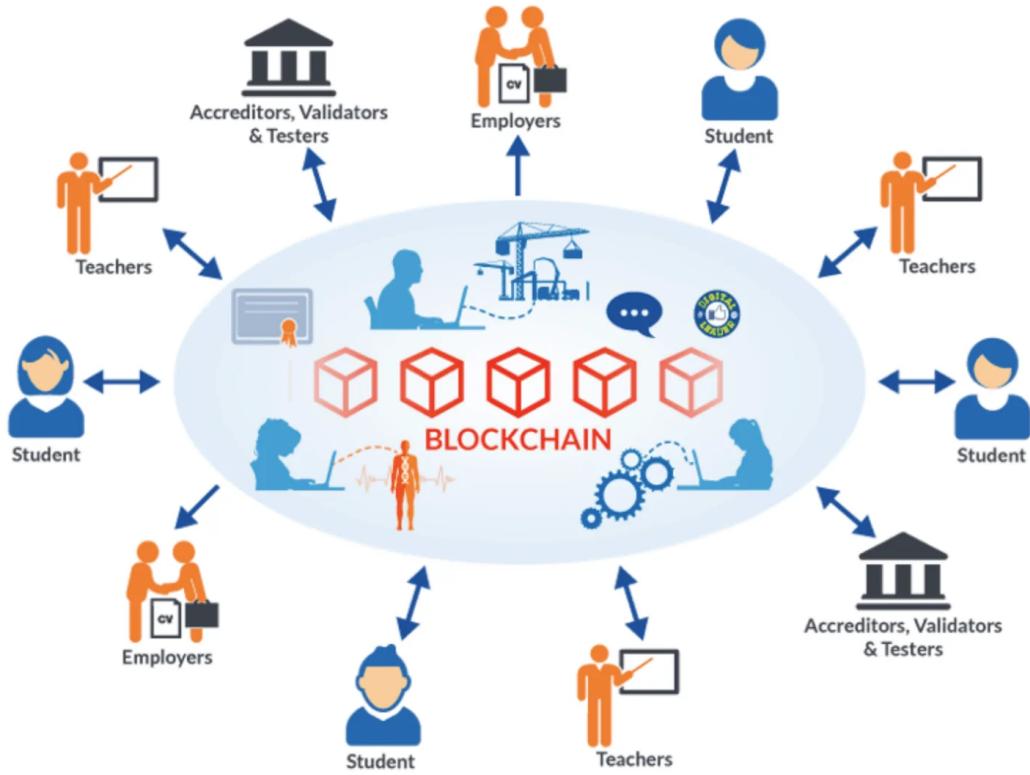
Nhờ vào những nguyên lý và công nghệ then chốt đó mà blockchain có thể hoạt động độc lập như một hệ thống phân tán, an toàn, minh bạch và hiệu quả mà không cần sự kiểm soát của bất kỳ tổ chức trung ương nào. Đó chính là những đặc tính cốt lõi làm nên sự cách mạng của công nghệ blockchain ngày nay.

Blockchain không chỉ dừng lại ở vai trò là công nghệ đỡ đầu cho các loại tiền kỹ thuật số, mà còn được ứng dụng rộng rãi vào nhiều lĩnh vực quan trọng khác của đời sống kinh tế - xã hội.

Trong lĩnh vực tài chính, blockchain tăng tốc độ giao dịch, giảm chi phí, đồng thời cung cấp tính an toàn và minh bạch cao trong các giao dịch tài chính. Nó có thể ứng dụng để xác thực danh tính, chuyển tài sản, thực hiện các hợp đồng tài chính mà không cần đến bên thứ ba. Điều này tiết kiệm thời gian, chi phí đồng thời đảm bảo tính đồng bộ và chính xác cao.

Trong lĩnh vực chuỗi cung ứng và logistics, việc ứng dụng blockchain giúp các doanh nghiệp có thể theo dõi chính xác nguồn gốc, xuất xứ của sản phẩm từ khi mới sản xuất cho đến khi vận chuyển đến tay người tiêu dùng. Điều này nâng cao tính minh bạch, giúp phát hiện nhanh các gian lận thương mại cũng như đảm bảo an toàn thực phẩm. Tính bất biến của blockchain cũng ngăn chặn nguy cơ làm giả hàng hóa và đánh cắp dữ liệu trong quá trình vận chuyển.

Trong lĩnh vực y tế, blockchain cho phép lưu trữ và chia sẻ hồ sơ, dữ liệu bệnh án một cách an toàn giữa các bệnh viện, phòng khám và bác sĩ, tránh được tình trạng bệnh nhân phải kê khai lại nhiều lần. Các nghiên cứu y sinh cũng có thể được phổ biến nhanh chóng và an toàn hơn thông qua blockchain.



Hình 2.2: Các ứng dụng khi sử dụng nền tảng blockchain

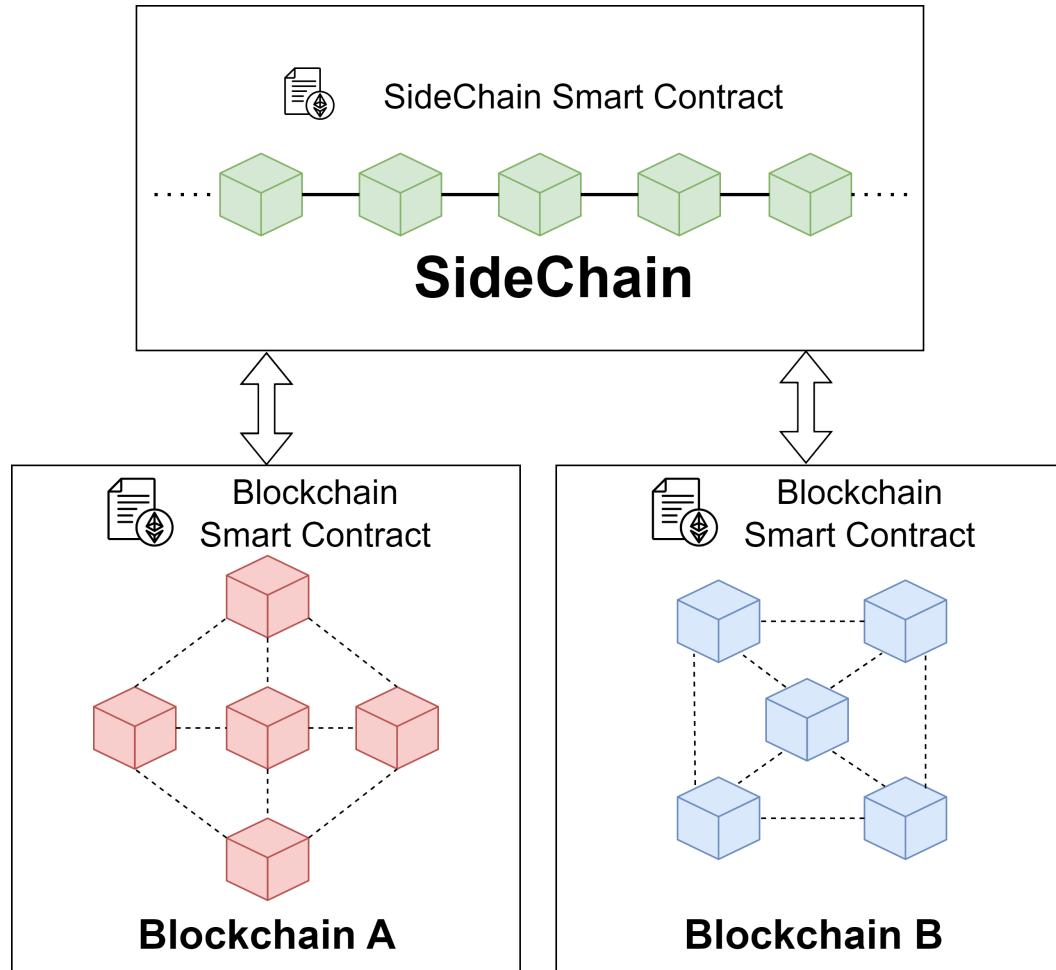
Như vậy, với khả năng phân quyền, xác thực phi tập trung, blockchain đang mở ra triển vọng tươi sáng cho tương lai mà ở đó thông tin có thể được lưu trữ, chia sẻ và truy xuất một cách an toàn, công bằng và không cần bên trung gian. Rõ ràng vai trò và tác động của blockchain không chỉ dừng lại ở tiền điện tử mà còn sâu rộng và đầy tiềm năng hơn thế rất nhiều.

2.1.2. Tổng quan về hệ thống mạng liên chuỗi khồi

Mạng liên chuỗi (cross-chain) là một công nghệ blockchain nâng cao, cho phép xây dựng các giao thức để kết nối và tương tác giữa nhiều chuỗi khối (blockchain) khác nhau (**Hình 2.3**). Các chuỗi khối này có thể dựa trên những công nghệ khác nhau, do nhiều tổ chức phát triển độc lập.

Mạng liên chuỗi ra đời để cải thiện hiện trạng cô lập giữa các blockchain/chuỗi

khối, và xây dựng một hệ sinh thái mở, tự do trao đổi dữ liệu, tài sản và ứng dụng. Điều này sẽ thúc đẩy đổi mới và mở rộng tiềm năng của blockchain.



Hình 2.3: Giao tiếp liên chuỗi giữa hai chuỗi khối thông qua sidechain

Các đặc điểm chính của mạng liên chuỗi:

- **Tính tương thích:** Các blockchain có thể giao tiếp, trao đổi dữ liệu hiệu quả với nhau qua mạng liên chuỗi. Yêu cầu cách biểu diễn dữ liệu nhất quán giữa các blockchain.
- **Tính an toàn và bảo mật:** Sử dụng các cơ chế bảo mật mạnh mẽ để đảm bảo an ninh trong quá trình trao đổi dữ liệu giữa các chuỗi khối. Phổ biến là các thuật toán mã hóa, ký số điện tử hiện đại.
- **Tính sở hữu đa chuỗi:** Người dùng có thể dễ dàng di chuyển và kiểm soát

tài sản giữa các chuỗi khối khác nhau. Nâng cao tính linh hoạt và hiệu quả trong sử dụng tài sản trên nhiều nền tảng blockchain.

2.2. Tổng quan về hợp đồng thông minh

2.2.1. Hợp đồng thông minh

Hợp đồng thông minh, còn gọi là hợp đồng điện tử thông minh, là một hình thức thỏa thuận điện tử trên nền tảng công nghệ blockchain. Loại hợp đồng này có khả năng tự động thực thi các điều khoản đã được lập trình sẵn mà không cần sự can thiệp của bên thứ ba.

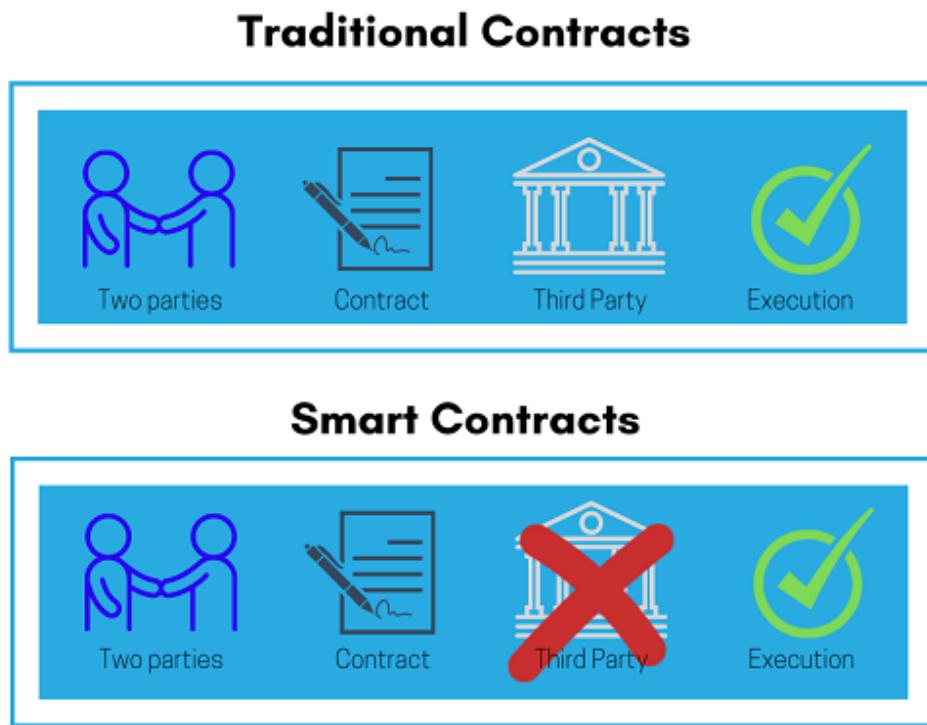
Ưu điểm của hợp đồng thông minh là tính minh bạch và khả năng kiểm soát chi phí, thời gian thực hiện hợp đồng tốt hơn so với hợp đồng truyền thống (**Hình 2.4**). Các điều khoản của hợp đồng được biểu diễn thành mã hoá không thể thay đổi và lưu trữ công khai trên một mạng lưới phân tán. Điều này hạn chế tối đa rủi ro tranh chấp, giả mạo thông tin khi thực thi hợp đồng.

Các tính chất đặc trưng của hợp đồng thông minh gồm:

- Hợp đồng thông minh sở hữu khả năng tự động thực thi các điều khoản mà không cần sự tham gia của bên thứ ba. Các quy tắc và điều kiện của hợp đồng đã được định nghĩa cụ thể bằng ngôn ngữ lập trình. Khi các điều kiện xảy ra, các mã lệnh này sẽ tự động chạy để kích hoạt hợp đồng.
- Hợp đồng thông minh cũng có đặc điểm là được sao chép và phân phối trên nhiều máy tính khác nhau trong hệ thống blockchain. Sự phân tán này nhằm mục đích bảo mật và đảm bảo tính toàn vẹn, tránh hiện tượng xóa bỏ hay sửa đổi trái phép nội dung hợp đồng.
- Bên cạnh đó, hợp đồng thông minh cũng có tính bất biến, nghĩa là nội dung của nó không thể bị thay đổi sau khi được đưa lên hệ thống blockchain. Đây là yếu tố rất quan trọng để đảm bảo tính an toàn và đáng tin cậy của các giao dịch.

- Đặc trưng tiếp theo là sự tin cậy. Hợp đồng hoạt động theo đúng cách nó được xây dựng mà không hề bị can thiệp bởi con người, đảm bảo mọi thứ diễn ra chính xác và tránh sai sót. Bên cạnh đó, cơ chế đồng thuận trong hệ thống blockchain cũng góp phần đảm bảo tính tin cậy của thông tin.
- Ngoài ra, hợp đồng thông minh còn có ưu điểm là mang tính công khai, minh bạch. Mọi người đều có thể truy cập và kiểm tra mã nguồn của nó kể cả khi không tham gia vào thỏa thuận đó. Đây là một trong những ưu điểm vượt trội của hợp đồng thông minh.
- Cuối cùng, nó còn tiết kiệm đáng kể về thời gian và chi phí cho các bên tham gia. Bởi lẽ hợp đồng tự vận hành mà không cần đối tác hay công chứng viên hỗ trợ.

Mặc dù vậy, hợp đồng thông minh chưa thể thay thế hoàn toàn cho hợp đồng truyền thống trong mọi trường hợp (**Hình 2.4**). Chúng không phù hợp với mọi loại hợp đồng có sẵn. Việc áp dụng hợp đồng thông minh còn tùy thuộc vào bản chất, phạm vi của từng loại thỏa thuận cụ thể. Ngoài ra, các quy định pháp lý hiện hành cùng môi trường pháp lý tại từng quốc gia, khu vực riêng biệt cũng là những yếu tố quyết định đến khả năng ứng dụng của công nghệ hợp đồng thông minh. Như vậy, hợp đồng thông minh vẫn còn những hạn chế nhất định và không thể áp dụng rộng rãi cho mọi trường hợp. Sự phù hợp cần được cân nhắc dựa trên đặc điểm, phạm vi của từng loại hợp đồng cùng các quy định pháp lý tại mỗi quốc gia.



Hình 2.4: Sự khác nhau giữa hợp đồng truyền thống và hợp đồng thông minh

2.2.2. Ngôn ngữ solidity

2.2.2.1. Source code

Solidity là ngôn ngữ lập trình cấp cao được thiết kế riêng cho việc phát triển các hợp đồng thông minh trên nền tảng Ethereum. Đây là ngôn ngữ typ (kiểu tĩnh) có cú pháp tương tự JavaScript, C++ và Python. Nhờ đó, các lập trình viên dễ dàng nắm bắt và làm quen với Solidity.

Trong Solidity, mọi thứ đều được mã hóa dưới dạng hợp đồng và các hàm (function). Các hàm này chứa các điều kiện, logic và các bước thực thi mà hợp đồng cần thực hiện. Lập trình viên sẽ sử dụng solidity để khai báo các biến, hàm và xây dựng tương tác giữa chúng để mô phỏng bất kỳ quy tắc nghiệp vụ nào. Dựa trên đó, hợp đồng thông minh sẽ tự động thực thi.

Một trong những ưu điểm nổi trội của Solidity là khả năng tạo ra các hợp

đồng thông minh tuân thủ mô hình OOP (lập trình hướng đối tượng). Điều này giúp chương trình dễ cấu hình, bảo trì và nâng cấp. Hơn nữa, Solidity còn hỗ trợ thư viện và giao thức tích hợp đa dạng các tính năng tiện ích cho phát triển blockchain như quản lý danh tính, gửi/nhận token, giao dịch...

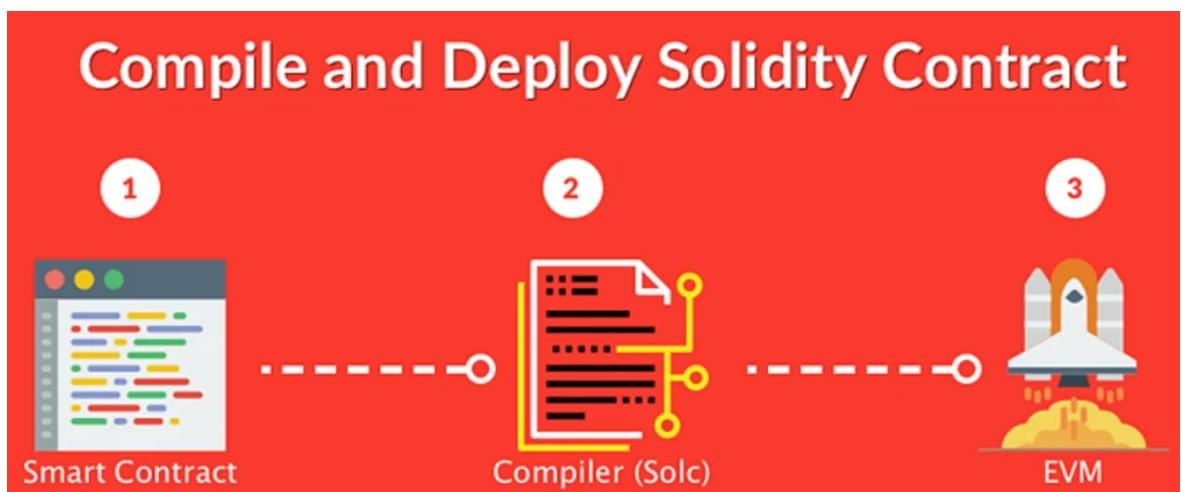
Như vậy, Solidity là công cụ lý tưởng nhất hiện nay để xây dựng các ứng dụng blockchain và hợp đồng thông minh trên Ethereum. Ngôn ngữ này mang đến sự đơn giản, dễ sử dụng nhưng vẫn đảm bảo được tính linh hoạt và mạnh mẽ trong phát triển.

Cấu trúc của một hợp đồng thông minh:

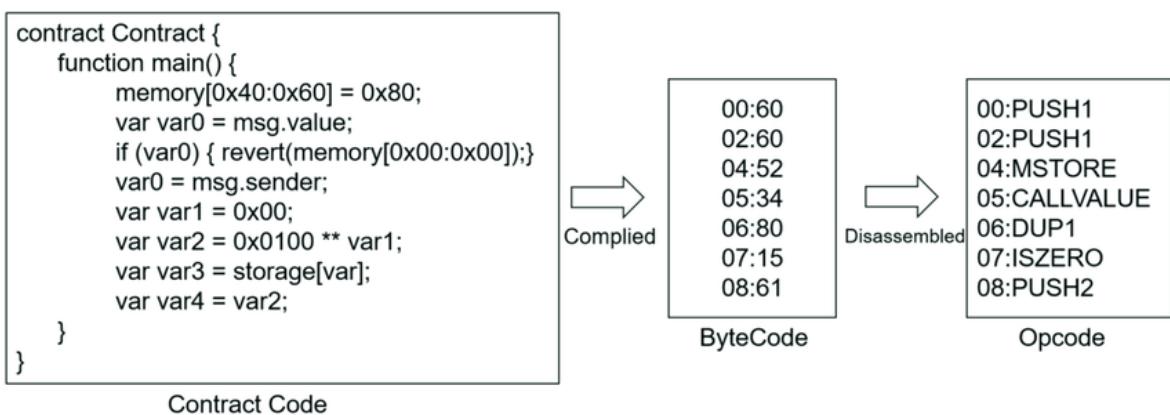
- **Pragma directive:** Đây là dòng khai báo phiên bản trình biên dịch Solidity mà source code sẽ được viết trên đó. Việc khai báo này giúp đồng bộ phiên bản giữa code và compiler, tránh rủi ro mất tương thích.
- **Contract:** Đây là khái niệm cơ bản nhất trong Solidity, tương đương với lớp (class) trong OOP. Contract sẽ chứa toàn bộ dữ liệu và hàm thành viên cần thiết để cấu thành nên hợp đồng thông minh.
- **State Variables:** Đây là các biến được khai báo bên trong contract, lưu trữ trạng thái và dữ liệu của hợp đồng. Chúng được lưu trữ vĩnh viễn trên blockchain.
- **Functions:** Các hàm là nơi định nghĩa chức năng và nghiệp vụ của hợp đồng. Chúng gồm các câu lệnh, điều kiện và logic cần thiết để thực thi.
- **Events:** Sự kiện được kích hoạt khi có hành động nào đó xảy ra trên hợp đồng. Chúng giúp ghi lại log và xác nhận các hoạt động diễn ra.

Hơn thế nữa, các công cụ biên dịch hợp đồng thông minh ra đời nhằm mục đích đơn giản hóa việc phát triển hợp đồng thông minh. Chúng cho phép người dùng viết hợp đồng bằng các ngôn ngữ phổ biến hơn như Python, sau đó tự động biên dịch ra các ngôn ngữ chuyên dụng như Solidity. Nhờ đó, quá trình phát triển hợp đồng thông minh trở nên dễ dàng và tiếp cận được nhiều lập trình

viên hơn. Điều này giúp mở rộng khả năng ứng dụng của công nghệ blockchain và hợp đồng thông minh. Quá trình biên dịch smart contract từ ngôn ngữ lập trình sang bytecode được thực hiện bởi trình biên dịch như solc cho Solidity. Bytecode sau đó được triển khai lên blockchain thông qua các giao dịch. Khi smart contract được thực thi, bytecode sẽ được biên dịch tiếp thành opcode - đây là ngôn ngữ máy ở mức thấp hơn mà EVM có thể đọc và thực thi trực tiếp. Quá trình biên dịch này giúp tối ưu hóa hiệu suất thực thi smart contract, vì opcode có kích thước nhỏ hơn và EVM chỉ cần đọc và thực thi các lệnh đơn giản (**Hình 2.5** và **Hình 2.6**).



Hình 2.5: Quá trình biên dịch và thực thi hợp đồng thông minh



Hình 2.6: Sự tương quan của source code, bytecode và opcode của hợp đồng thông minh

2.2.2.2. Bytecode

Bytecode là mã máy ảo Ethereum (EVM) được sinh ra khi biên dịch các hợp đồng thông minh viết bằng ngôn ngữ lập trình Solidity (**Hình 2.7**). Bytecode cho phép chuyển đổi mã nguồn Solidity thành các câu lệnh máy có thể đọc và thực thi được bởi Ethereum Virtual Machine. Đây là bước trung gian cần thiết để triển khai hợp đồng lên blockchain Ethereum. Các biến, cấu trúc dữ liệu, câu lệnh điều khiển và các hàm được khai báo bằng ngôn ngữ Solidity cấp cao sẽ được biên dịch xuống các kiểu dữ liệu và câu lệnh cơ bản hơn trong bytecode. Ví dụ biến kiểu string trong Solidity sẽ trở thành một mảng các ký tự byte. Biến bool được biểu diễn bằng giá trị 0 hoặc 1.

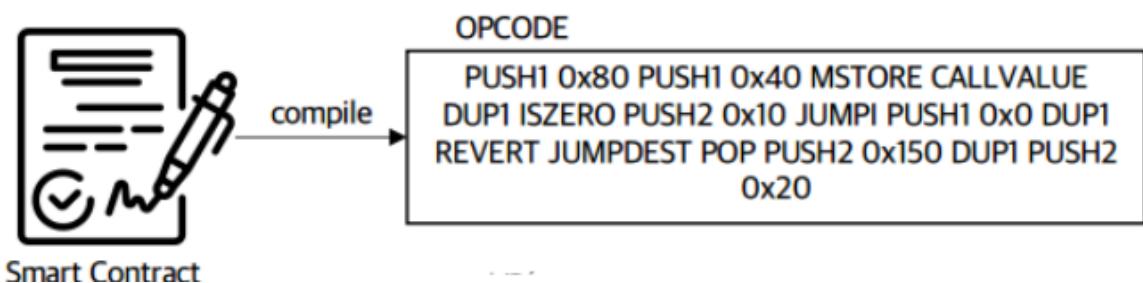
Hình 2.7: Cấu trúc của bytecode trong hợp đồng thông minh

Trong giai đoạn biên dịch mã nguồn Solidity, trình biên dịch sẽ chuyển đổi toàn bộ các hợp đồng thông minh thành một chuỗi các lệnh máy gọi là opcode. Mỗi opcode tương ứng với một hành động, thao tác hoặc lệnh cụ thể mà EVM có thể hiểu và thực thi được. Một số opcode phổ biến bao gồm các lệnh toán học

ADD, MUL, dùng cho phép cộng, nhân; lệnh SSTORE ghi dữ liệu vào bộ nhớ, SLOAD đọc dữ liệu; JUMP dùng để nhảy tới vị trí khác trong chương trình. Tập hợp các opcode này tạo thành bytecode cho hợp đồng. Sau khi được biên dịch, các hợp đồng thông minh Solidity sẽ được triển khai trên blockchain Ethereum dưới dạng bytecode. Đây chính là các câu lệnh mà EVM có thể đọc và thực thi khi có giao dịch gọi tới hợp đồng. Bytecode cho phép chuyển mã Solidity cấp cao về ngôn ngữ máy có thể thực thi được trên nền tảng Ethereum. Nó là cầu nối để code do người viết có thể vận hành trên môi trường blockchain. Khi triển khai, bytecode sẽ được lưu lại bên trong các khối của Ethereum.

2.2.2.3. Opcode

Opcode là các lệnh máy ảo được sử dụng trong smart contract để thực thi logic của contract. Các opcode này tương tự như các lệnh assembly trong lập trình máy tính. Mỗi opcode đại diện cho một hành động cụ thể mà smart contract có thể thực hiện, chẳng hạn như ghi dữ liệu, đọc dữ liệu, thực hiện phép toán, nhảy đến địa chỉ opcode tiếp theo. Các opcode cho phép smart contract thao tác với bộ nhớ, thực thi logic và giao tiếp với blockchain. Chúng được thiết kế để thực thi một cách an toàn và đáng tin cậy. Các lập trình viên smart contract sẽ sử dụng các opcode thông qua ngôn ngữ lập trình smart contract để viết logic cho contract. Sau đó mã nguồn sẽ được biên dịch thành các opcode tương ứng để máy ảo Ethereum thực thi.



Hình 2.8: Cấu trúc của opcode trong hợp đồng thông minh

Các loại opcode thông dụng:

- Các opcode thao tác dữ liệu: SLOAD, SSTORE, MLOAD, MSTORE
- Các opcode tính toán: ADD, MUL, SUB, DIV, EXP, MOD
- Các opcode logic: NOT, AND, OR, XOR, LT, GT
- Các opcode nhảy: JUMP, JUMPI

Chức năng của opcode thực hiện các thao tác dữ liệu:

- SLOAD: Đọc giá trị từ một vị trí bộ nhớ lưu trữ (storage) của hợp đồng thông minh. SLOAD sẽ truy xuất giá trị tại một địa chỉ bộ nhớ lưu trữ cụ thể và đẩy giá trị đó lên stack.
- SSTORE: Lưu trữ một giá trị vào một vị trí bộ nhớ lưu trữ của hợp đồng thông minh. SSTORE nhận 2 tham số, tham số đầu tiên là địa chỉ bộ nhớ lưu trữ và tham số thứ 2 là giá trị cần lưu.
- MLOAD: Tải một word (32 byte) từ bộ nhớ (memory) của EVM (Ethereum Virtual Machine) và đẩy lên stack.
- MSTORE: Lưu một word từ stack vào bộ nhớ. Nó nhận vào 2 tham số, tham số đầu tiên là vị trí bộ nhớ và tham số thứ 2 là giá trị cần lưu.

Như vậy, SLOAD và SSTORE làm việc với bộ nhớ lưu trữ lâu dài, trong khi MLOAD và MSTORE làm việc với bộ nhớ tạm thời của EVM. Chúng cho phép đọc, ghi dữ liệu giữa các vùng nhớ khác nhau.

Chức năng của opcode thực hiện các phép tính toán số học:

- ADD: Phép cộng hai số. Lấy hai giá trị trên đầu stack, cộng lại và đẩy kết quả lên stack.
- MUL: Phép nhân hai số. Tương tự ADD, lấy hai giá trị đầu stack nhân với nhau và đẩy kết quả lên.
- SUB: Phép trừ hai số. Lấy giá trị thứ 2 trên stack trừ đi giá trị đầu stack.

- DIV: Chia hai số nguyên. Tương tự SUB, chia số thứ 2 cho số đầu tiên.
- EXP: Lũy thừa - Giá trị thứ 2 mũ giá trị đầu tiên.
- MOD: Chia lấy dư - Tính số dư của phép chia giá trị đầu cho giá trị thứ 2.

Các opcode này cho phép thực hiện các phép toán cơ bản, hỗ trợ việc xử lý số liệu và tính toán trong smart contract. Chúng lấy các giá trị input từ stack, thực hiện phép toán và đẩy kết quả lên stack.

JUMP và JUMPI cho phép điều khiển luồng code một cách linh hoạt hơn, tạo ra các cấu trúc rẽ nhánh và lặp trong smart contract. Chức năng của opcode nhảy:

- JUMP: Nhảy đến một địa chỉ tuyệt đối trong code. Nó nhận vào một tham số duy nhất là địa chỉ cần nhảy tới. Sau khi thực thi JUMP, con trỏ lệnh sẽ chuyển đến địa chỉ được chỉ định và tiếp tục thực thi.
- JUMPI: Nhảy có điều kiện dựa trên kết quả kiểm tra logic. JUMPI nhận vào 2 tham số: Tham số 1: Địa chỉ cần nhảy tới nếu điều kiện khớp (tương tự như trong JUMP). Tham số 2: Giá trị điều kiện (khác 0 là True). Nếu giá trị điều kiện là True (khác 0), JUMPI sẽ chuyển con trỏ đến địa chỉ nhảy. Ngược lại con trỏ sẽ tiếp tục xuống dòng tiếp theo.

2.2.3. Các loại lỗ hổng trong hợp đồng

2.2.3.1. Reentrancy

Lỗ hổng reentrancy xảy ra do cách thức hoạt động của Ethereum Virtual Machine (EVM). Cụ thể, mỗi khi một hàm smart contract được gọi, EVM sẽ tạo ra một stack frame mới. Tại đây, các biến local sẽ được sao chép giá trị từ state của blockchain. Sau khi hàm kết thúc, giá trị cập nhật của các biến local mới được commit vào blockchain state.

Chính vì thế, nếu trong quá trình thực thi, hàm smart contract gọi tới một hàm khác trước khi kết thúc (ví dụ gọi đến một hàm bên ngoài hay hàm fall-back()), điều này cho phép mã độc hại được thực thi lại trên cùng một stack frame. Hacker có thể lợi dụng điều này để gọi đệ quy hàm bị lỗi, rút lui tiền nhiều lần trước khi trạng thái thực sự được cập nhật. **Hình 2.9** đã mô phỏng lại cuộc tấn công trong smart contract.

```
pragma solidity ^0.8.0;

contract ReentrancyExample {
    mapping(address => uint256) public balances;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw(uint256 amount) public {
        require(balances[msg.sender] >= amount, "I_balance");

        // Malicious reentrancy attack
        (bool success, ) = msg.sender.call{value: amount}("");
        require(success, "Reentrancy attack failed");

        balances[msg.sender] -= amount;
    }
}
```

Hình 2.9: Mô phỏng lỗ hổng Reentrancy

Điển hình như trường hợp hàm rút tiền ETH trong ví smart contract bị lỗi. Hacker có thể liên tục gọi lại hàm này để "đánh lừa" contract rằng số dư vẫn chưa thay đổi, cho phép rút hết số ETH trong ví. Đây chính là lỗ hổng reentrancy - một trong những lỗ hổng nghiêm trọng nhất có thể dẫn tới mất mát lớn.

Các contract thông minh cần cẩn trọng trong việc thiết kế các hàm dễ bị lạm dụng, đặc biệt là những hàm liên quan tới tiền hay quyền sở hữu tài sản.

2.2.3.2. Integer Overflow/Underflow

Integer Overflow/Underflow xảy ra khi một biến số nguyên vượt quá giới hạn lưu trữ của kiểu dữ liệu. Cụ thể, đối với kiểu uint256 trong Solidity chỉ có thể lưu trữ các số nguyên từ 0 đến $2^{256}-1$.

Khi thực hiện các phép tính số học như cộng, trừ, nhân, chia, nếu kết quả vượt quá giới hạn trên thì sẽ xảy ra overflow. Ngược lại, nếu kết quả nhỏ hơn 0 thì sẽ là underflow.

Trong smart contract, hacker có thể cố tình đưa vào các giá trị input khiến overflow/underflow xảy ra, dẫn đến kết quả sai và làm lợi cho chúng.

Chẳng hạn trong hàm rút tiền, hacker có thể gửi vào một số âm rất lớn, khiến biến cộng với số âm này bị underflow và đảo dấu thành số dương rất lớn. Điều này cho phép chúng rút nhiều tiền hơn số dư thực tế. **Hình 2.10** và **Hình 2.11** đã mô phỏng lại cuộc tấn công trong smart contract.

```
pragma solidity ^0.8.0;

contract IntegerOverflowUnderflowExample {
    mapping(address => uint256) public balances;

    function deposit(uint256 amount) public {
        balances[msg.sender] += amount;
    }

    function withdraw(uint256 amount) public {
        require(balances[msg.sender] >= amount, "Insufficient balance");

        // Integer underflow vulnerability
        balances[msg.sender] -= amount;

        // Malicious integer overflow attack
        uint256 maliciousAmount = type(uint256).max - balances[msg.sender] + 1;
        balances[msg.sender] += maliciousAmount;
    }
}
```

Hình 2.10: Mô phỏng lỗ hổng Integer Overflow

```

pragma solidity ^0.8.0;

contract IntegerUnderflowExample {
    mapping(address => uint256) public balances;

    function deposit(uint256 amount) public {
        balances[msg.sender] += amount;
    }

    function withdraw(uint256 amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance");

        // Integer underflow vulnerability
        balances[msg.sender] -= amount;

        // Transfer the amount to the user
        payable(msg.sender).transfer(amount);
    }
}

```

Hình 2.11: Mô phỏng lỗ hổng Integer Underflow

2.2.3.3. Unprotected Ether Withdrawal

Unprotected Ether Withdrawal xảy ra khi smart contract không giới hạn quyền rút Ether của người dùng. Điều này cho phép bất kỳ ai cũng có thể gọi hàm rút Ether mà không cần xác thực.

Trong trường hợp contract lưu trữ số lượng lớn Ether, hacker có thể dễ dàng gọi hàm rút tiền và chuyển tất cả về ví của mình. Đây là một lỗ hổng nghiêm trọng vì nó đe dọa trực tiếp đến tài sản của chủ sở hữu lẫn người dùng.

Nguyên nhân của lỗ hổng thường là do các nhà phát triển bỏ qua việc kiểm soát quyền truy cập vào hàm rút tiền. Hoặc họ cho rằng việc xác thực đã được thực hiện ở giai đoạn trước đó. Tuy nhiên không có cơ chế nào ngăn chặn hacker gọi trực tiếp đến hàm sensitive. **Hình 2.12** đã mô phỏng lại cuộc tấn công trong smart contract.

```

pragma solidity ^0.8.0;

contract UnprotectedWithdrawalExample {
    mapping(address => uint256) public balances;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdraw() public {
        // Unprotected Ether Withdrawal vulnerability
        uint256 amount = balances[msg.sender];
        msg.sender.transfer(amount);

        // Set balance to 0 after withdrawal
        balances[msg.sender] = 0;
    }
}

```

Hình 2.12: Mô phỏng lỗ hổng Unprotected Ether Withdrawal

2.2.3.4. Timestamp Dependence

Lỗ hổng Timestamp Dependence xảy ra khi smart contract phụ thuộc quá nhiều vào dữ liệu thời gian. Trong môi trường blockchain, thời gian có thể bị giả mạo. Nếu không cẩn thận, điều này có thể dẫn tới những hệ quả nghiêm trọng như lừa đảo tài chính. Kẻ gian có thể tấn công bằng cách làm sai lệch giá trị thời gian. Nếu hợp đồng dựa vào đó để xử lý các hoạt động then chốt, hậu quả có thể rất lớn. Cần có thêm các biện pháp kiểm soát khác để ngăn chặn. **Hình 2.13** đã mô phỏng lại cuộc tấn công trong smart contract.

```

pragma solidity ^0.8.0;

contract TimestampDependenceExample {
    mapping(address => uint256) public lastAccessTime;

    function accessRestrictedResource() public {
        require(block.timestamp - lastAccessTime[msg.sender] > 1 days, "Access allowed once per day");

        // Do something important here

        lastAccessTime[msg.sender] = block.timestamp;
    }
}

```

Hình 2.13: Mô phỏng lỗ hổng Timestamp Dependence

Thứ thách lớn nhất là khả năng hacker làm giả dữ liệu thời gian. Chúng có thể thay đổi thời gian của các nút mạng để đánh lừa smart contract. Điều đó ảnh hưởng tới quyết định của hợp đồng và mở ra cơ hội gian lận.

Như vậy, lỗ hổng này liên quan tới việc smart contract quá phụ thuộc vào dữ liệu thời gian dễ bị làm giả, dẫn tới những hậu quả tiêu cực.

2.2.3.5. Front Running

Front Running đang trở thành mối đe dọa lớn đối với an ninh của các ứng dụng tài chính phi tập trung trên nền tảng blockchain. Bản chất của vấn đề này là hacker theo dõi, nhận diện các giao dịch lớn của người dùng, sau đó cố tình đẩy lệnh giao dịch của mình lên trước. Kết quả là giá và điều kiện thực thi giao dịch ban đầu bị ảnh hưởng, hacker hưởng lợi còn người dùng bị thiệt.

Cụ thể, hacker sẽ liên tục giám sát những giao dịch tiềm năng có thể tác động mạnh đến thị trường. Khi phát hiện một giao dịch lớn, chúng nhanh tay đặt lệnh trước với giá khác biệt để thu lợi từ sự chênh lệch. Điều này không chỉ khiến hacker hưởng lợi bất chính mà còn gây rủi ro, thiệt hại cho người dùng thực sự do phải mua bán với giá không công bằng. **Hình 2.14** đã mô phỏng lại cuộc tấn công trong smart contract.

```

pragma solidity ^0.8.0;

contract FrontRunningVulnerableContract {
    address public owner;
    uint256 public price;

    constructor() {
        owner = msg.sender;
        price = 1 ether;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Not the contract owner");
        _;
    }

    function setPrice(uint256 newPrice) public onlyOwner {
        price = newPrice;
    }

    function buy() public payable {
        require(msg.value >= price, "Insufficient funds");

        // Simulate front running vulnerability
        // In a real scenario, this vulnerable part could be more complex
        if (msg.value > price) {
            // Front running vulnerability: Refund the excess value
            payable(msg.sender).transfer(msg.value - price);
        }

        // Complete the purchase
        owner = msg.sender;
        price = msg.value;
    }

    function withdraw() public onlyOwner {
        // Owner can withdraw the balance
        payable(owner).transfer(address(this).balance);
    }
}

```

Hình 2.14: Mô phỏng lỗ hổng Front Running

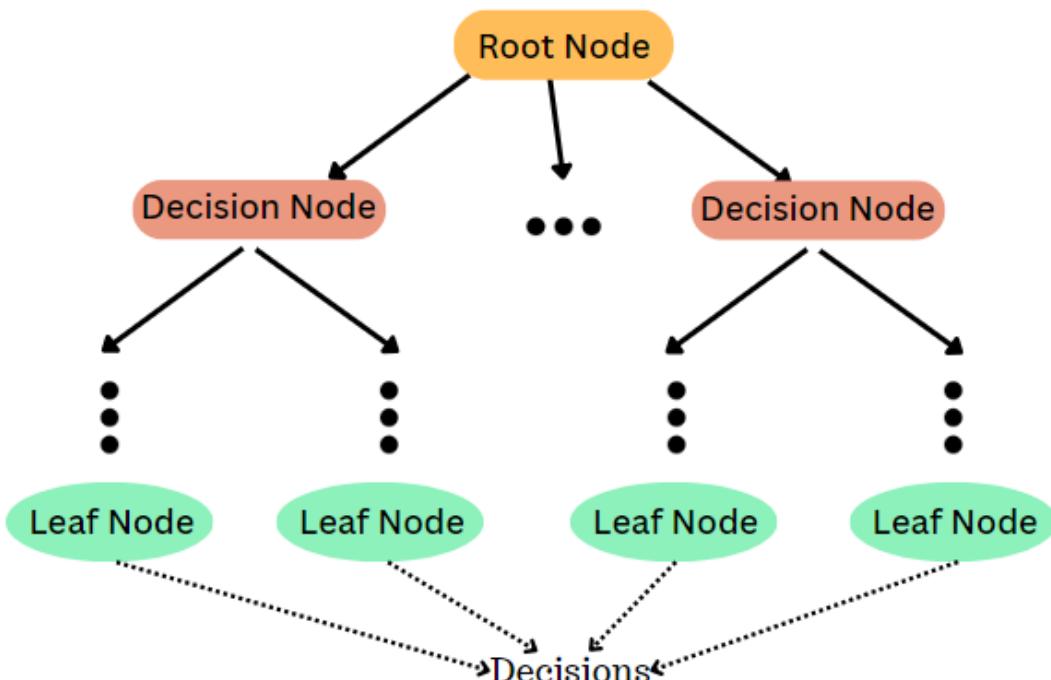
2.3. Các phương pháp phát hiện lỗ hổng tự động

2.3.1. Phương pháp học máy

2.3.1.1. Decision Tree

Decision Tree là một trong những kỹ thuật phân loại và dự báo phổ biến nhất trong học máy, với ưu điểm nổi bật là khả năng giải thích và dễ hiểu. Cơ chế hoạt động của nó dựa trên việc xây dựng một cấu trúc cây, trong đó mỗi nút đại diện cho một thuộc tính, mỗi nhánh đại diện cho một giá trị của thuộc tính. Quá trình huấn luyện cây sử dụng các thước đo độ thuần khiết như Entropy hay Gini Index để liên tục phân chia dữ liệu cho đến khi đạt được điểm dừng.

Hình 2.15 đã mô phỏng lại kiến trúc của mô hình.



Hình 2.15: Mô hình học máy Decision Tree

Ưu điểm nổi bật của Decision Tree bao gồm: Có khả năng xử lý dữ liệu phi tuyến tính và tương tác giữa các thuộc tính; Không yêu cầu xử lý dữ liệu trước

khi huấn luyện; Có khả năng xử lý cả dữ liệu định danh lẫn số liệu; Cho phép trực quan hóa và giải thích quá trình ra quyết định. Tuy nhiên, mô hình cũng có nhược điểm như dễ bị overfitting nếu cây quá phức tạp và nhạy cảm với nhiễu trong dữ liệu.

Về mặt ứng dụng, Decision Tree được sử dụng rộng rãi trong nhiều lĩnh vực như: y tế (chẩn đoán bệnh, dự báo biến chứng), tài chính (đánh giá rủi ro tín dụng, gian lận thẻ tín dụng), bán lẻ (dự đoán hành vi khách hàng), viễn thông (nhận diện cuộc gọi rác), v.v. Các ứng dụng điển hình bao gồm hệ thống hỗ trợ ra quyết định, phát hiện bất thường, dự đoán xu hướng, phân khúc khách hàng, lọc các trường hợp quan tâm, v.v.

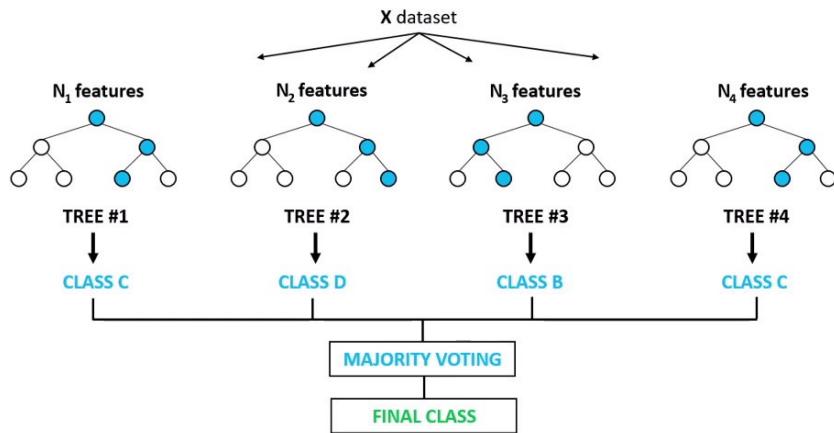
Nhìn chung, nhờ tính đơn giản, dễ hiểu và hiệu quả, Decision Tree là một trong những mô hình cốt lõi không thể thiếu trong học máy và thống kê, đã và đang được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau. Kỹ thuật này dự kiến sẽ còn phát triển và mở rộng ứng dụng trong tương lai.

2.3.1.2. Random Forest

Random Forest là một kỹ thuật học máy mạnh mẽ dựa trên nền tảng của Decision Tree. Ý tưởng chính là xây dựng một tập hợp gồm nhiều cây quyết định thay vì chỉ một cây đơn lẻ. Mỗi cây đều được huấn luyện trên một tập con ngẫu nhiên rút từ tập dữ liệu ban đầu. Điều này tạo ra độ đa dạng và giúp nâng cao độ chính xác so với một decision tree đơn lẻ.

Xét về quá trình huấn luyện, đầu tiên các tập dữ liệu con được rút ra ngẫu nhiên từ tập dữ liệu gốc. Sau đó, với mỗi tập con, một cây quyết định sẽ được huấn luyện sử dụng thuật toán decision tree tiêu chuẩn. Bước này được lặp lại cho đến khi xây dựng đủ số lượng cây mong muốn. Khi dự đoán, các cây quyết định đưa ra kết quả riêng của mình, kết quả cuối cùng là đa số bình chọn của tất cả các cây. **Hình 2.16** đã mô phỏng lại kiến trúc của mô hình.

Random Forest Classifier



Hình 2.16: Mô hình học máy Random Forest

Ưu điểm nổi trội của Random Forest là khả năng giảm overfitting và tăng khả năng tổng quát hóa so với các thuật toán cơ bản như decision tree. Sự đa dạng của các cây quyết định giúp tránh hiện tượng quá khớp với tập huấn luyện. Bên lại, nhược điểm là Random Forest tốn nhiều tài nguyên tính toán và khó giải thích kết quả hơn.

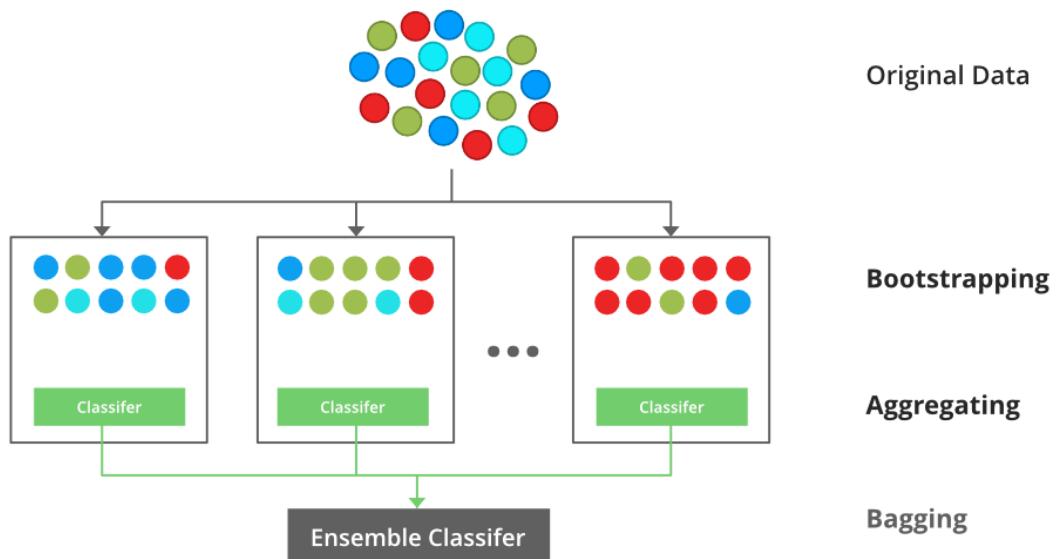
Ứng dụng của Random Forest rất rộng rãi, từ bài toán phân loại, dự đoán cho đến phân tích hình ảnh, xử lý ngôn ngữ tự nhiên. Các lĩnh vực như tài chính, y tế, gen, hình ảnh đều sử dụng rộng rãi kỹ thuật này. Sự vượt trội về độ chính xác và khả năng chịu nhiễu sót khiến Random Forest trở thành một trong những mô hình học máy phổ biến và hiệu quả nhất.

Nhìn chung, với những ưu điểm về độ chính xác, khả năng chống nhiễu và tính linh hoạt, Random Forest là một công cụ mạnh mẽ cho nhiều tác vụ học máy và được ứng dụng rộng rãi trong thực tiễn. Đây chắc chắn là một mô hình không thể thiếu của học máy hiện đại.

2.3.1.3. XGBoost

XGBoost là một trong những thuật toán học máy tiên tiến và mạnh mẽ nhất hiện nay, vượt trội hơn hẳn so với các thuật toán cơ bản thông thường. Đây là phiên bản nâng cao của Gradient Boosting, kết hợp nhiều kỹ thuật tối ưu hóa để cải thiện đáng kể độ chính xác và hiệu suất.

Về cơ bản, XGBoost xây dựng mô hình dựa trên nhiều cây quyết định yếu. Mỗi cây mới được thêm vào để khắc phục nhược điểm của các cây trước, dần dần nâng cao độ chính xác của mô hình. Quá trình huấn luyện có hàm mục tiêu là tối thiểu hóa lỗi dự đoán bằng cách tối ưu hóa hàm loss. Để tránh overfitting, XGBoost sử dụng kỹ thuật regularization và early stopping. **Hình 2.17** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.17: Mô hình học máy XGBoost

Ưu điểm nổi bật của XGBoost là độ chính xác rất cao, vượt trội hơn hẳn so với các thuật toán phổ biến khác. Bên cạnh đó, nó cũng rất nhanh trong quá trình huấn luyện, có thể xử lý hiệu quả các tập dữ liệu lớn và phức tạp. Hạn chế lớn nhất là đòi hỏi nhiều tài nguyên tính toán và khó khăn trong việc giải

thích mô hình.

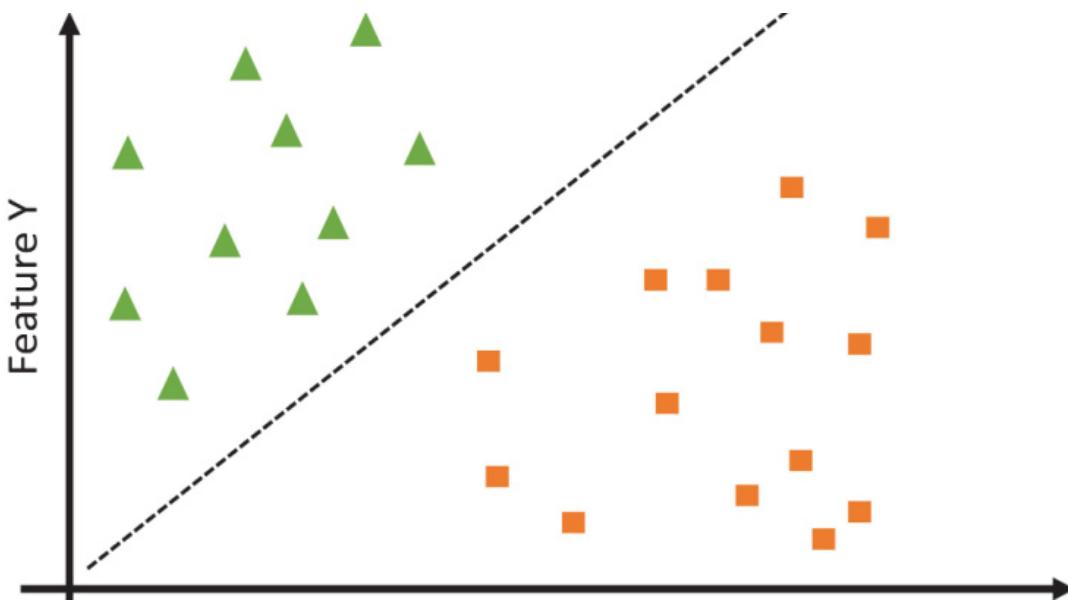
Các lĩnh vực ứng dụng của XGBoost rất đa dạng, từ dự đoán tài chính, phân tích dữ liệu khách hàng, nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên cho tới các bài toán dự báo trong nông nghiệp, giao thông. Thậm chí trong nhiều cuộc thi AI và dự án nghiên cứu, XGBoost là sự lựa chọn hàng đầu nhờ khả năng mô hình hóa tuyệt vời.

Nhìn chung, với sự kết hợp hoàn hảo giữa nhiều kỹ thuật tối ưu hiện đại cùng khả năng cải thiện đáng kể hiệu suất, XGBoost chắc chắn là một trong những thuật toán học máy quan trọng và phổ biến nhất trong thập kỷ gần đây.

2.3.1.4. Support Vector Machine

Support Vector Machine (SVM) là một thuật toán máy học phỏng biển được sử dụng cho các nhiệm vụ phân loại và hồi quy. SVM nổi bật với khả năng làm việc hiệu quả trong không gian chiều cao, đặc biệt là trong trường hợp dữ liệu không gian lớn và phức tạp.

Quá trình hoạt động của SVM là xác định ranh giới phân loại tối ưu giữa các nhóm dữ liệu bằng cách tìm ra siêu phẳng (hyperplane) chia không gian thành hai phần. Đối với trường hợp phân loại tuyến tính, siêu phẳng này cố gắng tối đa hóa khoảng cách giữa các điểm dữ liệu thuộc các lớp khác nhau. Nếu dữ liệu không thể phân loại tuyến tính, SVM sử dụng các hàm đặc trưng (kernel functions) để ánh xạ dữ liệu vào không gian chiều cao hơn, nơi mà việc tạo ra siêu phẳng phân loại trở nên khả thi. **Hình 2.18** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.18: Mô hình học máy Support Vector Machine

Ưu điểm chính của SVM là khả năng hoạt động hiệu quả trong không gian chiều cao, đồng thời có khả năng xử lý tốt với các tập dữ liệu lớn. SVM cũng linh hoạt với sự thay đổi của hàm đặc trưng, giúp nó làm việc với nhiều loại dữ liệu khác nhau.

Tuy nhiên, SVM có thể trở nên phức tạp và yêu cầu nhiều tài nguyên khi áp dụng cho các tập dữ liệu lớn. Việc lựa chọn hàm đặc trưng và tham số có thể đòi hỏi sự chuyên sâu trong lĩnh vực và hiểu biết sâu rộng về dữ liệu.

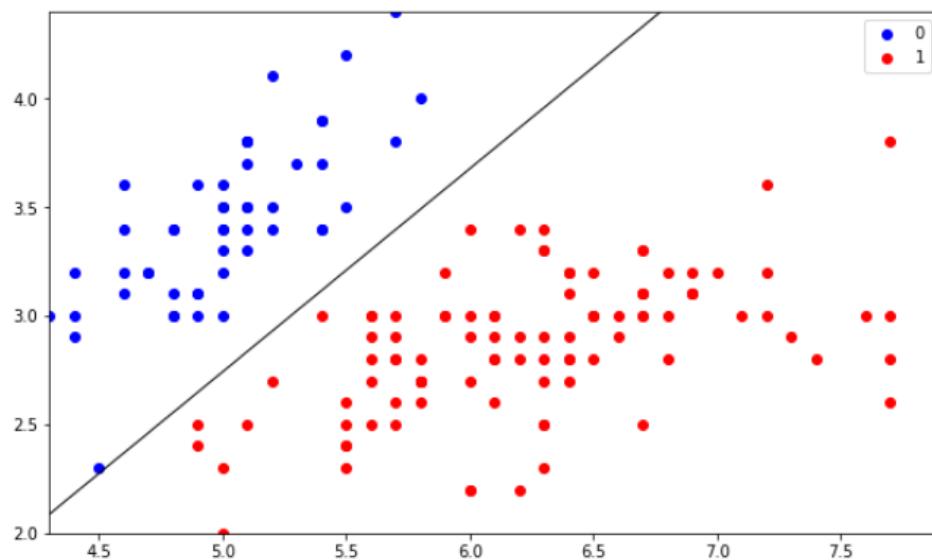
Ứng dụng của SVM rất đa dạng, từ phân loại văn bản, nhận diện khuôn mặt, đến dự báo giá chứng khoán. Đối với các tập dữ liệu có cấu trúc rõ ràng và không gian lớn, SVM thường là một lựa chọn mạnh mẽ với khả năng phân loại chính xác và hiệu suất ổn định.

2.3.1.5. Logistic Regression

Logistic Regression là một trong những kỹ thuật phân loại cơ bản và quan trọng nhất trong lĩnh vực học máy, cho phép dự đoán xác suất một đối tượng thuộc vào một nhãn cụ thể dựa trên các đặc trưng của nó.

Cụ thể, Logistic Regression mô hình hóa mối quan hệ giữa các biến độc lập,

là các thuộc tính của đối tượng, với xác suất dự đoán là yếu tố đầu ra, thông qua hàm logistic. Các tham số của mô hình được cập nhật trong quá trình huấn luyện để phản ánh mối liên hệ này. Cuối cùng mô hình được sử dụng để dự đoán xác suất cho các trường hợp mới. **Hình 2.19** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.19: Mô hình học máy Logistic Regression

Ưu điểm lớn nhất của Logistic Regression là đơn giản, dễ hiểu và dễ giải thích cách thức hoạt động. Nó cũng khá linh hoạt, có thể mở rộng để giải quyết nhiều loại bài toán phân loại khác nhau, từ nhị phân tới đa lớp. Tuy vậy, giả định tuyến tính của mô hình là một điểm yếu lớn khi dữ liệu phức tạp.

Về mặt ứng dụng, Logistic Regression rất phổ biến trong y học để dự đoán các bệnh lý, phân loại giai đoạn ung thư; trong ngân hàng để đánh giá rủi ro tín dụng; trong marketing để dự đoán xác suất khách hàng mua sắm trực tuyến. Nói chung đây là một công cụ rất hữu ích cho hầu hết các tác vụ phân loại cơ bản, khi mà độ giải thích của mô hình là quan trọng.

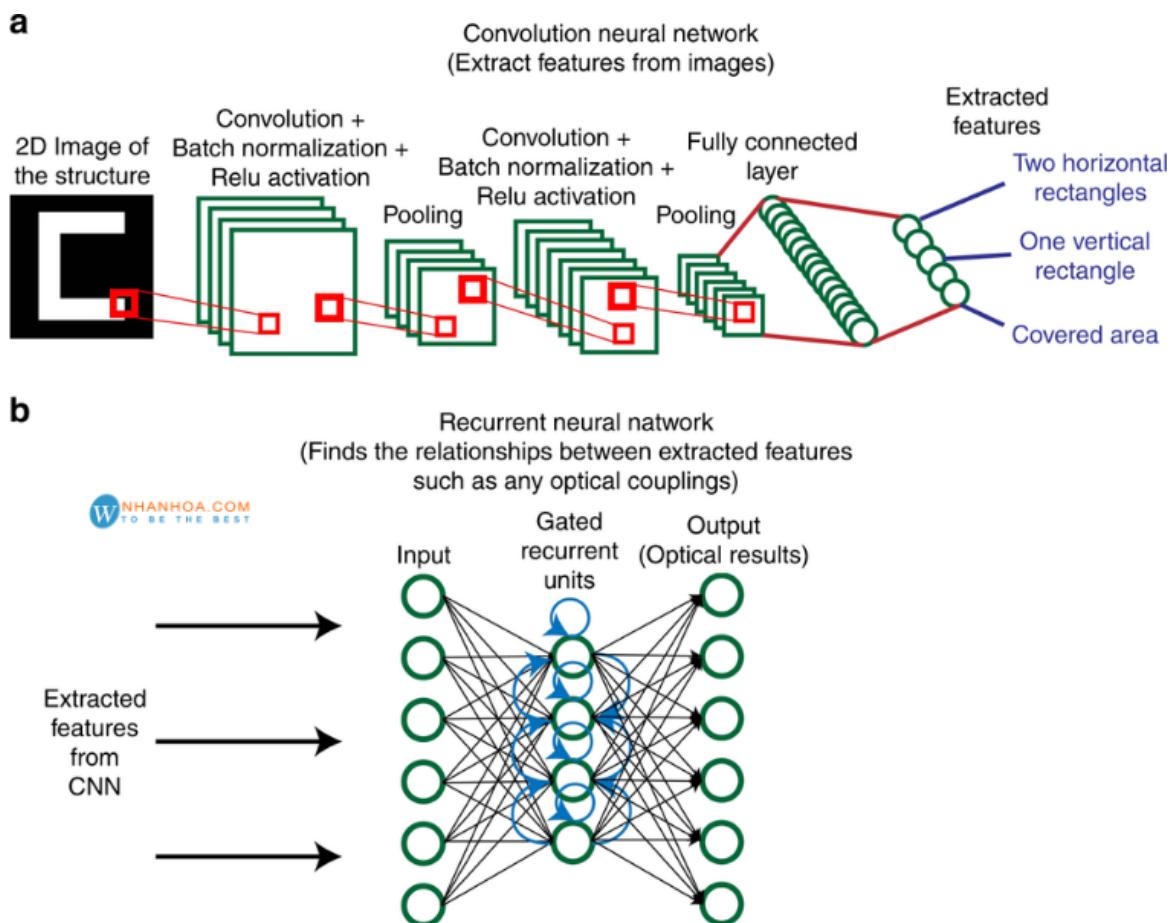
Nhìn chung, với ưu điểm về tốc độ huấn luyện nhanh, dễ triển khai và khả năng giải thích tốt, Logistic Regression được xem là một trong những kỹ thuật học máy quan trọng và không thể thiếu cho hầu hết các tác vụ phân loại.

2.3.2. Phương pháp học sâu

2.3.2.1. Convolutional Neural Network

Convolutional Neural Network (CNN) đã trở thành mô hình nhịp tim của công nghệ nhận dạng hình ảnh hiện đại, với khả năng học các đặc trưng phức tạp từ dữ liệu hình ảnh và video một cách hiệu quả.

CNN là một kiến trúc mạng nơ-ron sâu chuyên biệt cho bài toán xử lý ảnh, với các lớp tích chập và lớp gộp là hai thành phần cốt lõi. Các lớp tích chập sử dụng các bộ lọc để trích xuất các đặc trưng cục bộ như cạnh, góc, kết cấu. Các lớp gộp sau đó giảm kích thước ảnh xuống để loại bỏ thông tin dư thừa, đồng thời làm tăng tốc độ xử lý. **Hình 2.20** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.20: Mô hình học sâu Convolutional Neural Network

Ưu điểm vượt trội của CNN là khả năng học các đặc trưng trừu tượng cấp cao, biểu diễn đa chiều của hình ảnh mà không cần kỹ thuật trích chọn đặc trưng thủ công phức tạp. Điều này giúp CNN vượt trội trong các tác vụ phân loại, phát hiện đối tượng trên ảnh. Song, CNN đòi hỏi lượng dữ liệu lớn và tài nguyên tính toán cao.

Ứng dụng của CNN bao trùm hầu như mọi lĩnh vực liên quan đến hình ảnh: nhận dạng khuôn mặt, phân loại đối tượng trong ảnh vệ tinh, dịch chữ viết tay, dự đoán bệnh trên ảnh y khoa,... CNN là “con ngựa chiến” cho mọi hệ thống thị giác máy tính hiện đại.

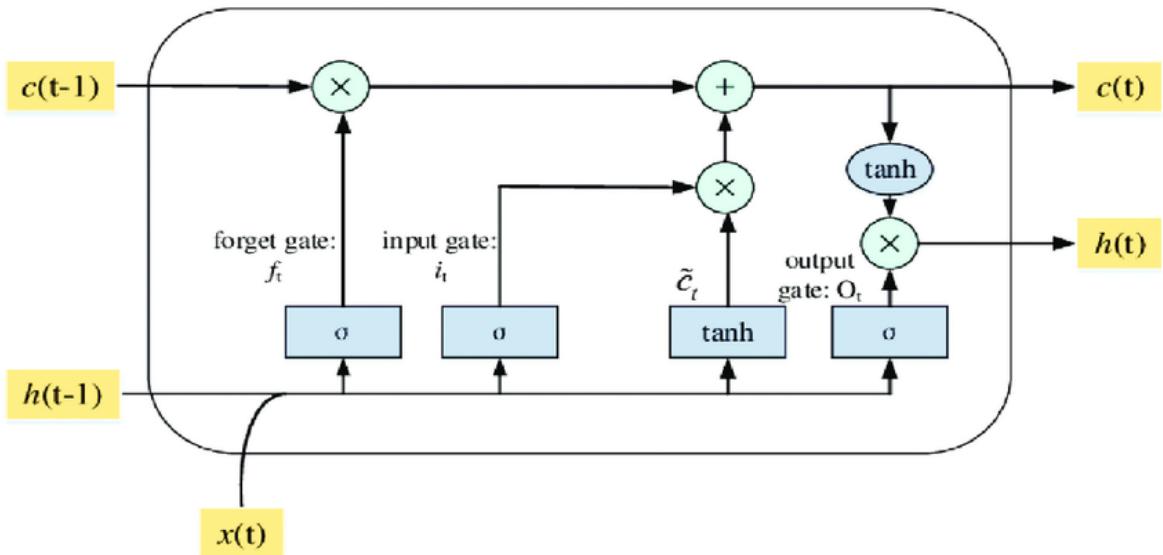
Với sự bùng nổ của dữ liệu hình ảnh và video trong thập kỷ qua, cùng nhu cầu nhận dạng ngày một cao, có thể nói CNN chính là công cụ đầy quyền năng đã giúp cách mạng hóa khả năng hiểu hình ảnh của máy móc. Với tốc độ phát triển chóng mặt của CNN cùng các ứng dụng trên diện rộng, có thể khẳng định vai trò then chốt không thể thay thế của nó trong tương lai.

2.3.2.2. Long Short-Term Memory

Long Short-Term Memory (LSTM) là một trong những kiến trúc mạng nơ-ron học sâu đột phá nhất trong lĩnh vực xử lý ngôn ngữ tự nhiên và chuỗi dữ liệu. Với các cơ chế cảng kiểm soát thông tin, LSTM giải quyết được vấn đề biến mất gradient khi huấn luyện mô hình trên các chuỗi dữ liệu dài.

Cụ thể, LSTM sử dụng 3 loại cổng để quyết định xem nên lưu giữ, cập nhật hay loại bỏ thông tin từ các bước trước đó của chuỗi. Cổng quên quyết định những gì nên loại bỏ; cổng đầu vào quyết định những gì nên cập nhật; cổng đầu ra xác định thông tin nào truyền tới các tế bào LSTM tiếp theo. Nhờ đó, LSTM có thể duy trì và khai thác thông tin liên quan trong những chuỗi dữ liệu dài.

Hình 2.21 đã mô phỏng lại kiến trúc của mô hình.



Hình 2.21: Mô hình học sâu Long Short-Term Memory

Ưu điểm vượt trội của LSTM là khả năng xử lý các phụ thuộc xa - một thách thức lớn mà các mạng nơ-ron truyền thống gặp phải. LSTM cũng có khả năng học các mối quan hệ phức tạp trong dữ liệu chuỗi. Tuy vậy, LSTM đòi hỏi khối lượng dữ liệu lớn và tốn nhiều tài nguyên tính toán hơn so với các mô hình đơn giản.

Ứng dụng chính của LSTM nằm ở các hệ thống xử lý ngôn ngữ tự nhiên như dịch máy, tổng hợp văn bản, nhận dạng giọng nói. LSTM cũng được sử dụng rộng rãi cho các bài toán dự báo chuỗi thời gian trong tài chính, thời tiết, sản xuất.

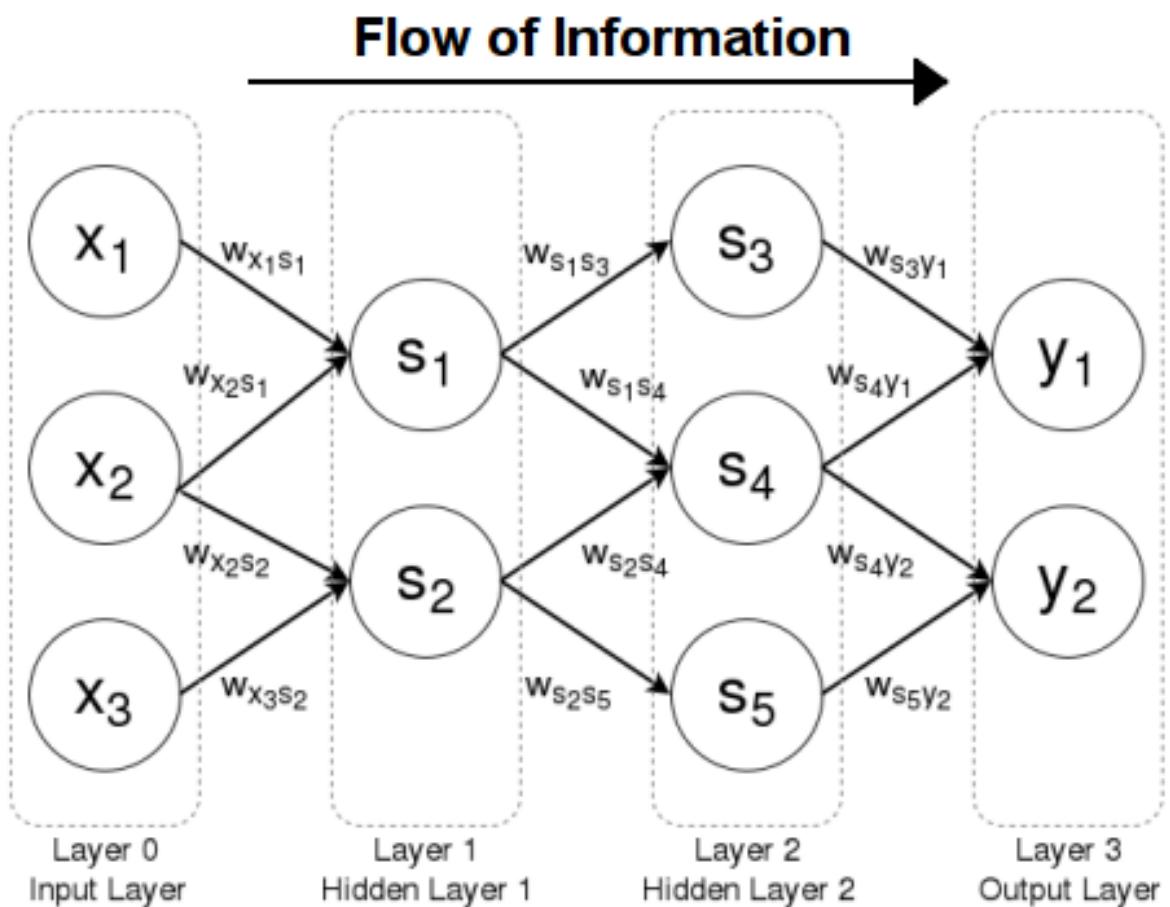
Nhìn chung, với khả năng xử lý hiệu quả các phụ thuộc xa trong dữ liệu chuỗi, LSTM thực sự là một đột phá quan trọng giúp mở ra kỷ nguyên mới cho các ứng dụng AI về ngôn ngữ và dự báo chuỗi. Sự phát triển mạnh mẽ gần đây của lĩnh vực xử lý ngôn ngữ tự nhiên có sự đóng góp rất lớn từ kiến trúc LSTM.

2.3.2.3. Feed forward network

Feedforward Neural Network (FNN) là kiến trúc mạng nơ-ron nhân tạo cơ bản, với các tính chất then chốt là không có vòng lặp và thông tin chuyển tiếp

một chiều từ lớp đầu vào qua các lớp ẩn đến lớp đầu ra. Đây vẫn là một mô hình cực kỳ quan trọng và được ứng dụng rộng rãi hiện nay.

Cụ thể, FNN bao gồm các lớp nơ-ron kết nối với nhau theo chiều thẳng từ trước ra sau. Mỗi lớp bao gồm nhiều nơ-ron, mỗi nơ-ron được kết nối với tất cả nơ-ron ở lớp liền kề. Trong quá trình lan truyền thông tin xuyên suốt mạng, các trọng số kết nối giữa các nơ-ron được điều chỉnh để tối ưu hóa mục tiêu. **Hình 2.22** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.22: Mô hình học sâu Feedforward neuron network

Ưu điểm lớn nhất của FNN là đơn giản, dễ hiểu và dễ triển khai. FNN cũng có thể xử lý nhiều dạng dữ liệu khác nhau và thích hợp với nhiều bài toán. Tuy nhiên, FNN lại kém hiệu quả trong việc nắm bắt các mối quan hệ phức tạp hoặc không tuyến tính trong dữ liệu.

Các lĩnh vực ứng dụng phổ biến của FNN rất đa dạng, như nhận dạng chữ số viết tay, phân loại văn bản/email, phân loại ảnh, dự đoán chuỗi thời gian tài chính. Nói chung, đối với các bài toán đơn giản hóa hay dữ liệu có cấu trúc phân lớp rõ ràng, FNN thường đem lại hiệu quả cao.

Nhìn chung, với ưu điểm về tốc độ huấn luyện nhanh, khả năng triển khai đơn giản cùng hiệu năng tốt trên nhiều bài toán, FNN vẫn đóng vai trò cốt lõi trong công nghệ mạng nơ-ron nhân tạo hiện đại. Sự phát triển mạnh mẽ của học sâu trong thập kỷ qua cũng có nền tảng từ những ý tưởng ban đầu về FNN.

2.3.2.4. Robustly Optimized BERT Pretraining Approach

Robustly Optimized BERT Pretraining Approach (RoBERTa) đại diện cho một bước tiến quan trọng trong công nghệ xử lý ngôn ngữ tự nhiên, với khả năng hiểu và mô hình hóa ngôn ngữ ở cấp độ sâu sắc chưa từng có. Đây là phiên bản nâng cấp và tối ưu hóa của kiến trúc ngôn ngữ BERT.

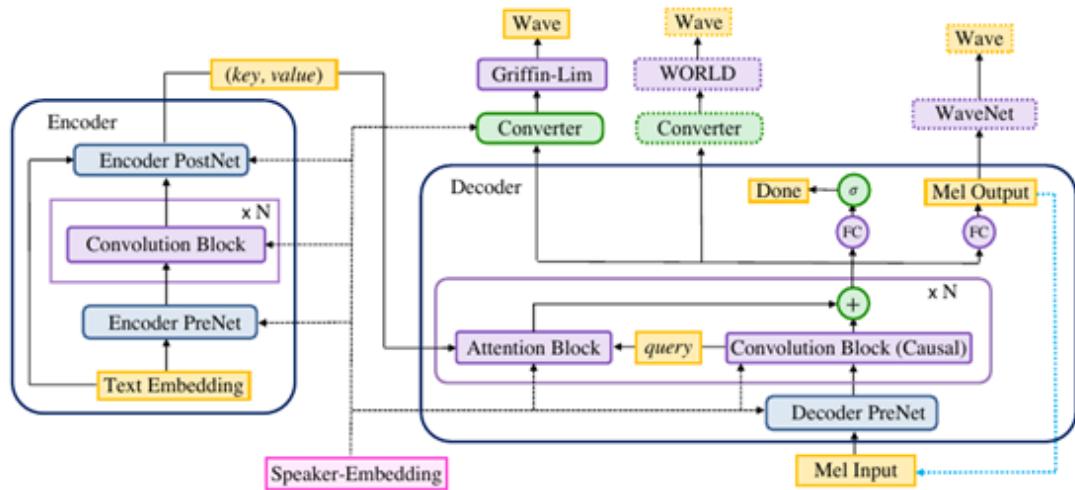
Cụ thể, RoBERTa sử dụng cơ chế encoder-decoder tương tự BERT để trích xuất các đặc trưng ngôn ngữ nghĩa phong phú từ văn bản. Tuy nhiên, một số thành phần không cần thiết đã được loại bỏ và thay vào đó là các kỹ thuật tối ưu hóa trong quá trình huấn luyện giúp tăng hiệu suất.

Ưu điểm vượt trội nhất của RoBERTa so với các mô hình trước đó là khả năng hiểu và xử lý ngôn ngữ tự nhiên chính xác và sâu sắc hơn. Kết quả, RoBERTa đạt kết quả state-of-the-art trên nhiều tác vụ xử lý ngôn ngữ phức tạp. Tuy vậy, mô hình lại đòi hỏi dung lượng lớn và tốn kém tài nguyên tính toán.

Các ứng dụng điển hình của RoBERTa rất đa dạng, từ trả lời câu hỏi, đối thoại chatbot, dịch máy đến phân loại văn bản, tóm tắt văn bản tự động. Với độ chính xác và khả năng hiểu ngôn ngữ sâu sắc, RoBERTa thực sự là bước đột phá cho nhiều ứng dụng AI ngôn ngữ.

Nhìn chung, với sự cải tiến về kiến trúc và kỹ thuật tối ưu hóa, RoBERTa đại diện cho hướng đi mới của các mô hình xử lý ngôn ngữ tự nhiên, mở ra kỷ

nguyên mới cho khả năng đọc, hiểu và suy luận ngôn ngữ của máy móc. **Hình 2.23** đã mô phỏng lại kiến trúc của mô hình.



Hình 2.23: Mô hình học sâu RoBERTa

CHƯƠNG 3. MÔ HÌNH ĐỀ XUẤT

Trong phần này, chúng tôi trình bày về phương pháp trong việc xây dựng mô hình ChainSniper trong việc thực hiện phát hiện lỗ hổng trong hợp đồng thông minh trong mạng liên chuỗi một cách tự động. Bên cạnh đó, chúng tôi thực hiện việc xây dựng tập dữ liệu nhằm đánh giá các mô hình học máy trong hệ thống ChainSniper.

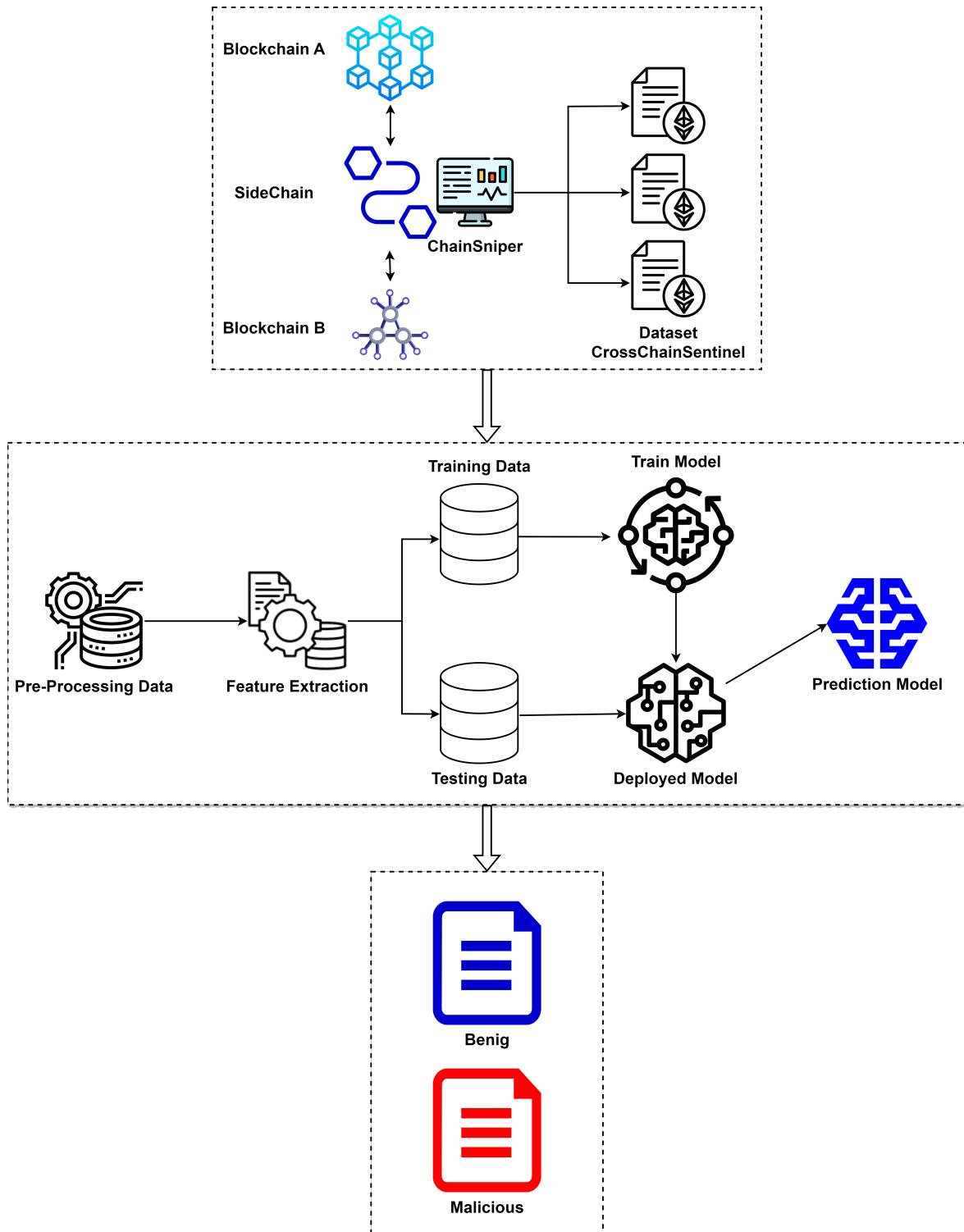
3.1. Tổng quan mô hình

3.1.1. Các thành phần chính của mô hình

Mô hình chính của ChainSniper bao gồm 5 thành phần:

1. Ethereum Sepolia đóng vai trò là Blockchain A
2. Quorum đóng vai trò là Blockchain B
3. Sidechain chuyển dữ liệu và ghi lại các hợp đồng thông minh
4. Mô hình Học máy phát hiện lỗ hổng
5. Module phân loại các hợp đồng thông minh trên sidechain

Hệ thống bao gồm hai mạng blockchain được kết nối với nhau thông qua sidechain bridge giúp chuyển dữ liệu và ghi lại thông tin của các hợp đồng thông minh. Dữ liệu này được xử lý trước và đưa vào các mô hình học máy được huấn luyện trên dữ liệu có nhãn để phát hiện các hợp đồng độc hại. Các mô hình được đánh giá để đo lường hiệu suất. Cuối cùng, một module sử dụng các dự đoán của mô hình để phân loại các hợp đồng thông minh trên sidechain thành lành tính hoặc độc hại. **Hình 3.1** minh họa mô hình chính:



Hình 3.1: Mô hình tổng quan

3.1.2. Mô hình cầu nối

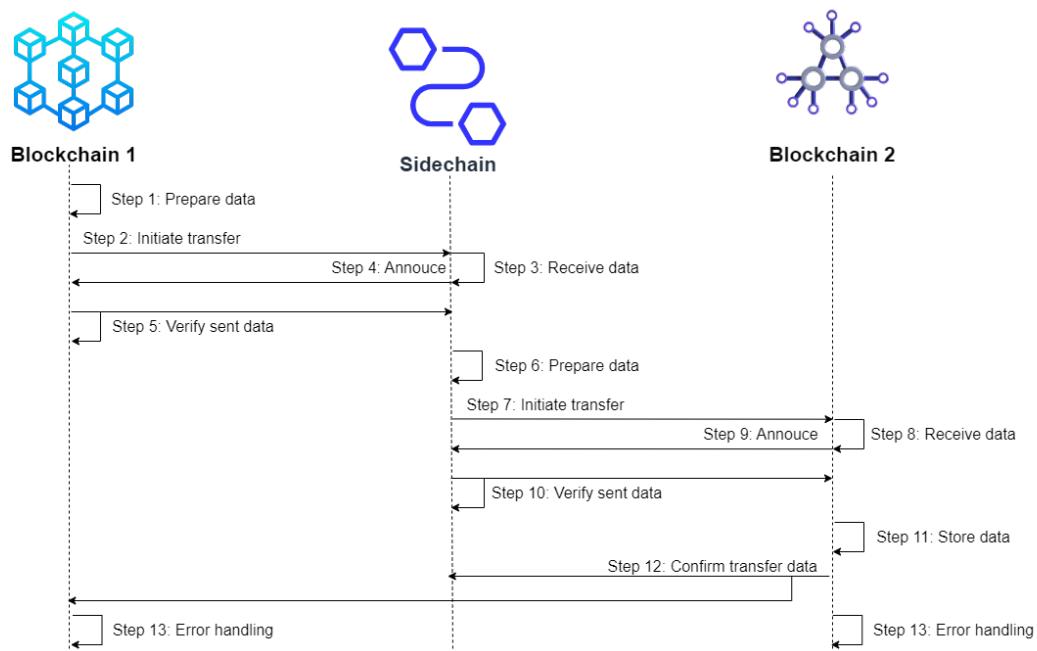
Hệ thống ChainSniper cung cấp khả năng tương tác giữa các blockchain không đồng nhất thông qua cầu nối sidechain. Cụ thể, sidechain hoạt động như một lớp trung gian xử lý và chuyển dữ liệu giữa các mạng blockchain được kết nối với nhau. Để có thể truyền tải dữ liệu, tài sản giữa các chuỗi được khóa lại ở một chuỗi này và giải khóa những biểu diễn tương đương ở chuỗi kia thông qua cơ chế neo hai chiều dựa trên hợp đồng đa chữ ký.

Khi có các giao dịch xuyên chuỗi diễn ra, các nút của sidechain sẽ ghi lại các thông tin, dữ liệu của giao dịch đó như địa chỉ của các hợp đồng thông minh tham gia, dấu thời gian diễn ra, các lời gọi hàm, tham số truyền vào, giá trị trả về cũng như các ngoại lệ nếu có. Nhật ký giao dịch này sau đó sẽ được tổng hợp, xử lý để tạo thành một tập dữ liệu cung cấp cái nhìn sâu sắc về các hành vi, cách thức hoạt động của các hợp đồng thông minh cũng như các mẫu thực thi của chúng.

Các bước thực hiện việc chuyển đổi dữ liệu qua sidechain (**Hình 3.2**):

1. Chuẩn bị dữ liệu: Xác định định dạng và tính toàn vẹn của dữ liệu cần chuyển đổi.
2. Khởi tạo quá trình chuyển đổi: Kết nối với sidechain để khởi tạo quá trình chuyển đổi.
3. Nhận dữ liệu ở sidechain: Sidechain nhận dữ liệu đến từ hệ thống blockchain A sau khi quá trình chuyển đổi được khởi tạo.
4. Sidechain thông báo nhận dữ liệu: blockchain A nhận thông báo từ sidechain về sự kiện chuyển đổi dữ liệu.
5. Xác minh dữ liệu đã gửi đến sidechain: Sidechain xác minh tính chính xác và toàn vẹn của dữ liệu đã nhận từ hệ thống chính.
6. Chuẩn bị dữ liệu (lần 2): Chuẩn bị lại dữ liệu cho lần chuyển đổi tiếp theo

7. Khởi tạo quá trình chuyển đổi (lần 2): Blockchain B tiếp tục kết nối với sidechain để khởi tạo lần chuyển đổi tiếp theo
8. Nhận dữ liệu (lần 2): Sidechain gửi dữ liệu mới đến Blockchain B.
9. Thông báo: Blockchain B nhận thông báo từ sidechain về sự kiện chuyển đổi dữ liệu lần thứ hai.
10. Xác minh dữ liệu đã gửi từ sidechain (lần 2): Sidechain tiếp tục xác minh tính chính xác và toàn vẹn của dữ liệu mới nhận được.
11. Lưu trữ dữ liệu: Dữ liệu được lưu trữ trong Blockchain B để sử dụng trong các quy trình tiếp theo.
12. Xác nhận chuyển đổi dữ liệu: Blockchain B xác nhận rằng quá trình chuyển đổi đã hoàn thành thành công và dữ liệu đã được chuyển đổi một cách chính xác.
13. Xử lý lỗi: Nếu có lỗi xuất hiện, các mạng blockchain cần xử lý chúng, bao gồm việc thông báo lỗi và ghi log để theo dõi và giải quyết vấn đề.

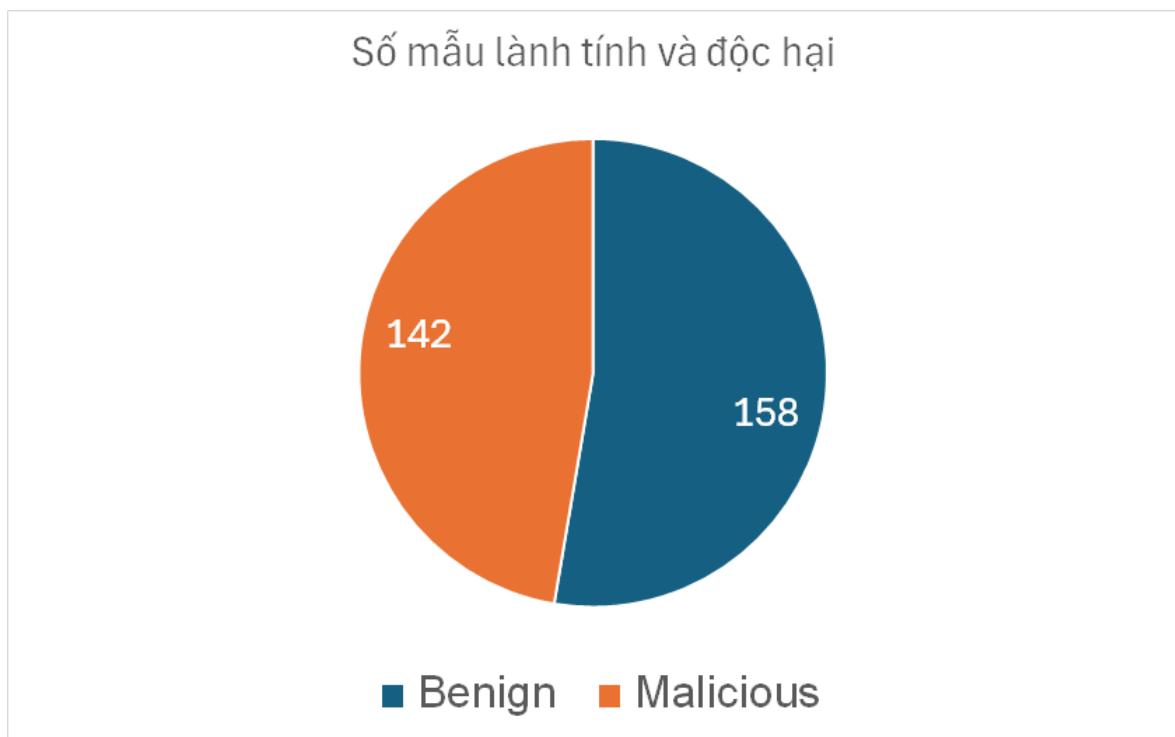


Hình 3.2: Các bước chuyển đổi dữ liệu qua cầu nối Sidechain

3.2. Phát hiện các lỗ hổng bằng phương pháp học máy và học sâu

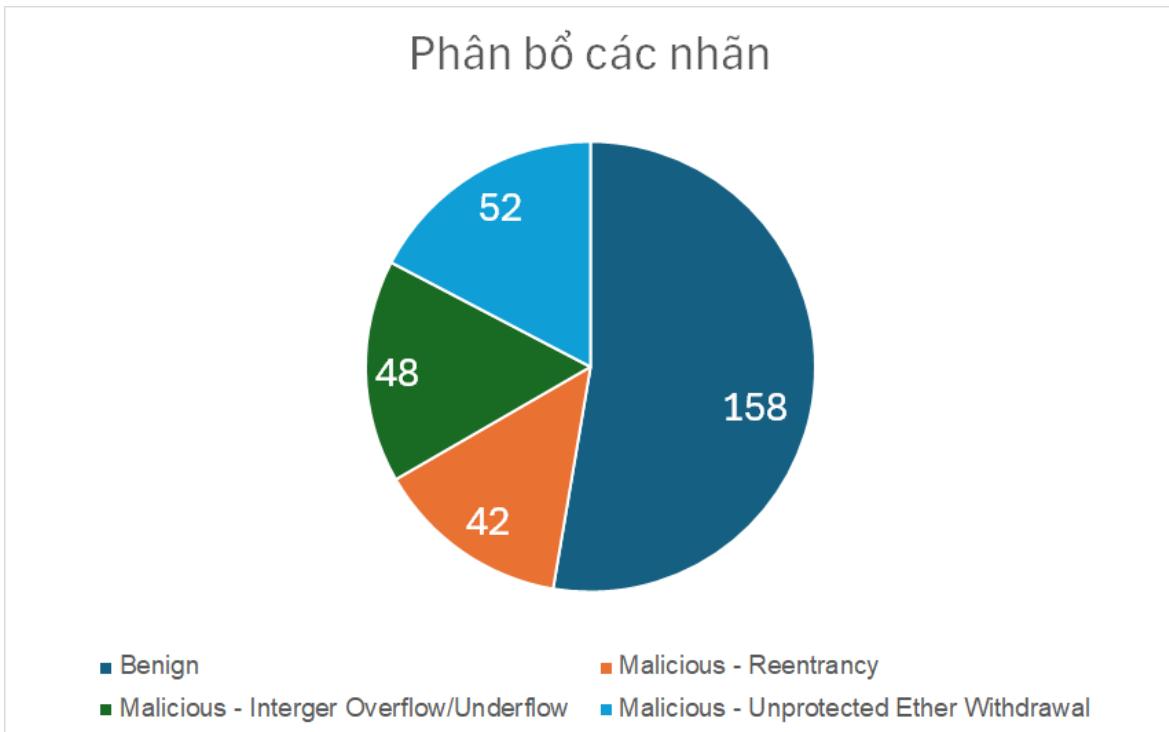
3.2.1. Xây dựng tập dữ liệu

Tập dữ liệu CrossChainSentinel¹ được tạo ra với mục đích phân tích các lỗ hổng bảo mật tiềm ẩn trong các hợp đồng cầu nối sidechain. Tập dữ liệu này chứa 300 tệp hợp đồng thông minh, trong đó 158 mẫu được xác định là lành tính, 142 mẫu còn lại được xếp vào loại độc hại (**Hình 3.3**). Cụ thể, số mẫu độc hại bao gồm 42 hợp đồng bị lỗ hổng tái tham chiếu (Reentrancy), 48 hợp đồng chứa lỗi tràn/cạn số nguyên (Integer Overflow/Underflow) và 52 hợp đồng có vấn đề rút Ether ra mà không được bảo vệ (Unprotected Ether Withdrawal) (**Hình 3.4**).



Hình 3.3: Phân bố mẫu lành tính và độc hại

¹<https://github.com/anhkiet1227/CrossChainSentinel>



Hình 3.4: Phân bố mẫu lành tính và độc hại tương ứng với các lỗ hổng

Những lỗ hổng trên được xác định dựa trên kỹ thuật từ các nghiên cứu trước như Smartbugs-wild², SolidiFi-benchmark³ cùng Danh sách đen các địa chỉ Ethereum liên quan tới các cuộc tấn công đa chuỗi. Dữ liệu về cầu nối chuỗi chéo được mô phỏng theo 15 nhà cung cấp thực tế khác nhau gồm Commos, Avalanche, Chainlink... Tập dữ liệu CrossChainSentinel tập trung vào những lỗ hổng phổ biến có thể xảy ra khi tài sản được chuyển đổi giữa các blockchain thông qua hợp đồng sidechain.

Việc đưa ra các rủi ro tiềm ẩn đó trong một tập dữ liệu có nhãn rộng lớn giúp CrossChainSentinel hỗ trợ tăng cường khả năng kiểm toán, thử nghiệm và phát hiện những lỗ hổng nguy hiểm trong cơ chế cầu nối chuỗi chéo trước khi chúng được triển khai trên mạng chính. Sự đa dạng về nguồn chuỗi chéo và các loại lỗ hổng khiến đây là tập dữ liệu lý tưởng để đánh giá công cụ, mô hình nhằm tăng cường bảo mật chuỗi

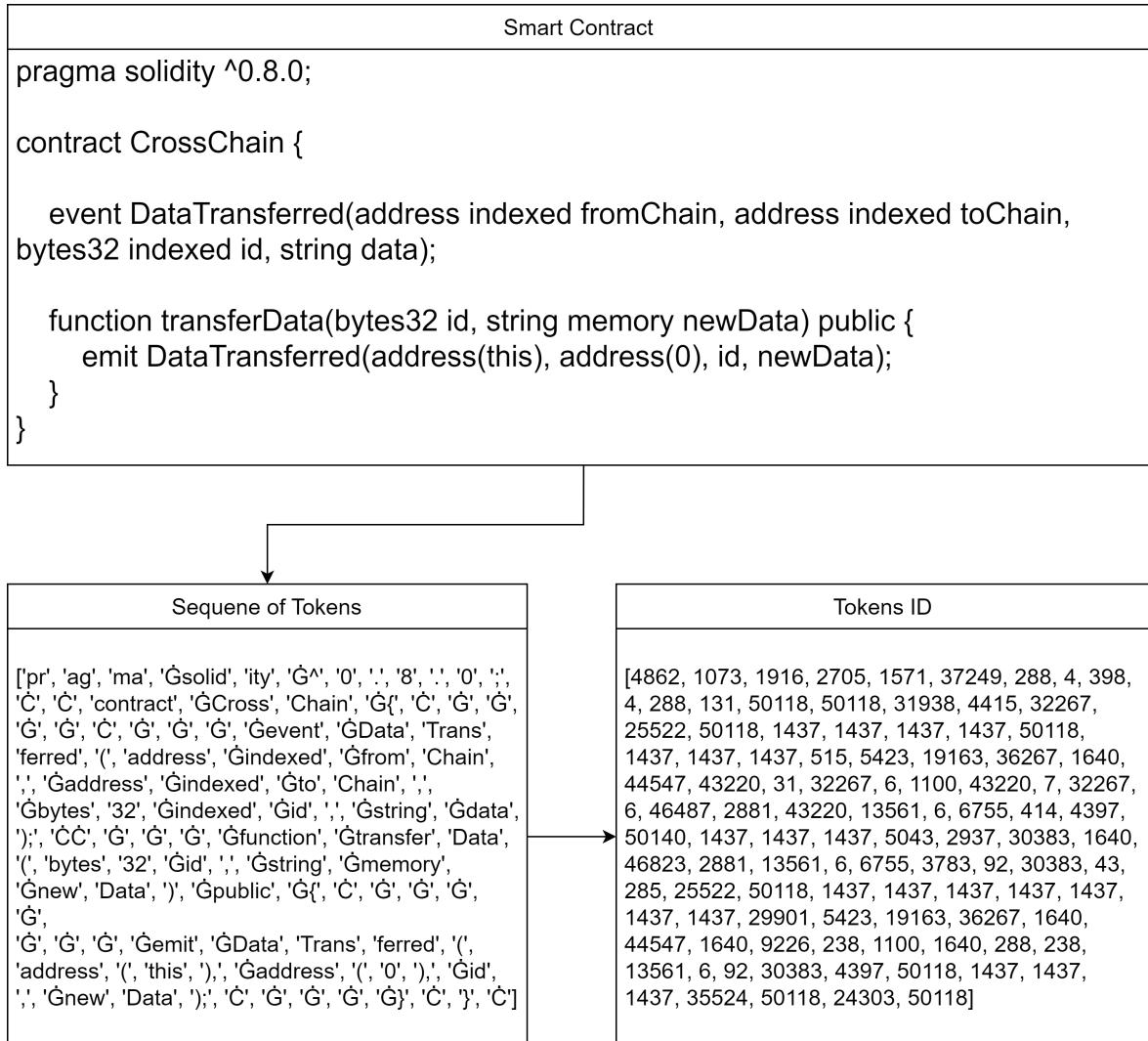
²<https://github.com/smartbugs/smartbugs-wild>

³<https://github.com/DependableSystemsLab/SolidiFI-benchmark>

3.2.2. Dán nhãn các mẫu và xử lý dữ liệu

Tập dữ liệu chứa hai loại nhãn để phân loại hợp đồng thông minh. Tập nhãn đầu tiên là một tập nhãn nhị phân, xác định xem một hợp đồng có đặc điểm an toàn hay nguy hiểm. Tập nhãn thứ hai chi tiết hóa các nhãn đa lớp, phân loại các hợp đồng độc hại thành các loại lỗ hổng cụ thể như an toàn, có khả năng bị Reentrancy, Integer Overflow/Underflow và Unprotected Ether Withdrawal. Dữ liệu được mô tả bằng các đặc trưng như tên file dự án, ID commit, nhãn mục tiêu, cấu trúc và nội dung của hợp đồng, cũng như số thứ tự file. Trong tập nhãn nhị phân, giá trị 0 biểu thị an toàn và 1 là nguy hiểm. Trong tập nhãn đa lớp, giá trị 0 là an toàn, 1 là lỗ hổng Reentrancy, 2 là lỗ hổng Integer Overflow/Underflow và 3 là lỗ hổng Unprotected Ether Withdrawal.

Để xử lý dữ liệu, chúng tôi bắt đầu bằng việc chuyển đổi các file mã nguồn hợp đồng thông minh thành một chuỗi các token thông qua quá trình tokenization (**Hình 3.5**). Mỗi file code smart contract được phân chia thành các token, đơn vị nhỏ nhất của mã nguồn, như từ vựng, ký tự đặc biệt, hoặc biểu thức. Việc này giúp chúng tôi biểu diễn cấu trúc và ý nghĩa của mã nguồn một cách có cấu trúc. Tiếp theo, mỗi token được chuyển đổi thành một tokenID, là một số nguyên duy nhất đại diện cho loại cụ thể của token đó. Quá trình này giúp giảm kích thước của dữ liệu và tạo ra một biểu diễn số hóa hiệu quả cho mã nguồn hợp đồng. Điều này làm cho việc đưa dữ liệu vào mô hình học máy trở nên hiệu quả hơn và giúp mô hình "hiểu" cấu trúc và ngữ cảnh của mã nguồn. Qua quá trình xử lý dữ liệu này, chúng tôi tạo ra một tập hợp các tokenID, mỗi tokenID đại diện cho một phần của mã nguồn hợp đồng. Điều này sẽ giúp chúng tôi xây dựng vector đặc trưng cho mỗi hợp đồng, và cuối cùng, đưa dữ liệu này vào mô hình học máy để phân loại các loại lỗ hổng một cách chính xác.



Hình 3.5: Quá trình xử lý dữ liệu

3.2.3. Ứng dụng các mô hình học máy tổng việc phát hiện các lỗ hổng

Việc chọn các phương pháp học máy bởi chúng cho phép tự động phát hiện lỗ hổng bảo mật trong các hợp đồng thông minh bằng cách phân biệt các mẫu trong mã hợp đồng và cấu trúc. So với kiểm toán thủ công, các mô hình học máy có thể phân tích hợp đồng một cách hiệu quả và nhanh hơn. Chúng tôi thử nghiệm vài mô hình học máy cổ điển, bao gồm Cây quyết định, Rừng ngẫu nhiên, Máy vector hỗ trợ, XGBoost và Hồi quy Logistic. Nhờ khả năng

giải thích, các mô hình này cho phép hiểu được lý do tại sao một số hợp đồng bị gán cờ là dễ bị tấn công.

Cây quyết định (Decision tree) và rừng ngẫu nhiên (Random forest) xây dựng các quy tắc phân cấp dựa trên các thuộc tính mã để phân loại hợp đồng. Trong khi đó, SVM tìm ranh giới tối ưu giữa hợp đồng dễ bị tấn công và an toàn trong không gian đặc trưng. XGBoost tiếp tục tăng độ chính xác thông qua một tập hợp các học viên yếu. Chúng tôi trích xuất các đặc trưng số liệu có giá trị thông tin như độ phức tạp hàm, luồng điều khiển và các mẫu cú pháp.

Bằng cách huấn luyện trên dữ liệu đại diện thích hợp của cả ví dụ tích cực và tiêu cực, các mô hình học cách phát hiện bền vững các lỗ hổng bảo mật. Chúng tôi áp dụng các kỹ thuật này trong ChainSniper để xây dựng một máy quét tự động cho các lỗ hổng như Reentrancy, Interger Overflow/Underflow và Unprotected Ether Withdrawal. Tính linh hoạt của học máy cung cấp một phương pháp luận để phát hiện lỗ hổng trong khi tránh nỗ lực thủ công mở rộng.

3.2.4. Ứng dụng các mô hình học sâu trong việc phát hiện các lỗ hổng

Việc chọn các phương pháp học sâu bởi chúng có thể mô hình hóa các mối quan hệ phức tạp của logic mã và phụ thuộc mà không cần trích xuất thủ công các đặc trưng. Các kỹ thuật như CNN, RNN, LSTM và các mô hình Transformer như RoBERTa cho phép học từ đầu đến cuối trực tiếp từ mã nguồn hợp đồng thông minh thô để phát hiện các mẫu cú pháp và ngữ nghĩa phức tạp mà đặc trưng cho các lỗ hổng.

Việc thử nghiệm các mạng nơ-ron sâu bao gồm các mô hình tuần tự dựa trên LSTM có khả năng diễn giải các luồng điều khiển, CNN trích xuất các mẫu cú pháp cục bộ và bộ mã hóa Transformer như RoBERTa cung cấp các vector từ ngữ cảnh phong phú. Nhờ xử lý phân cấp theo tầng, các mạng này tự động học quan hệ tiềm ẩn giữa các token và cấu trúc dẫn đến sự hiểu biết về logic cho

thấy lỗ hổng bảo mật.

Hơn thế nữa, các phụ thuộc dài hạn được mô hình hóa bởi LSTM có thể truy tìm các luồng thông tin trong hợp đồng để xác định các vấn đề kiểm tra cuộc gọi không kiểm soát. Các vector ngữ cảnh của RoBERTa có khả năng phân biệt các sắc thái giữa các cấu trúc có vẻ tương tự nhưng có thể hoặc không dễ bị tấn công. Các lớp chú ý có thể giải thích được của học sâu cũng hỗ trợ xác định các thành phần gây vấn đề. Việc áp dụng các kỹ thuật này trong ChainSniper để xây dựng các máy quét tự động có thể cờ các nguy cơ Reentrancy, Integer Overflow/Underflow và Unprotected Ether Withdrawal.

CHƯƠNG 4. THÍ NGHIỆM VÀ ĐÁNH GIÁ

Ở phần này, chúng tôi trình bày về việc thiết lập thực nghiệm bao gồm môi trường, các chỉ số đánh giá và kịch bản thực nghiệm, đồng thời kết quả về mặt hiệu quả và thời gian thực thi của mô hình

4.1. Thiết lập thực nghiệm

4.1.1. Môi trường thực nghiệm

Đối với việc chạy mô hình liên chuỗi, yêu cầu công việc phải được thực hiện chạy trong môi trường hiệu năng cao về CPU, dung lượng lưu trữ và bộ nhớ. Do đó, để đáp ứng được những yêu cầu, nhóm đã thực hiện thiết lập môi trường như **Bảng 4.1**:

	Blockchain A	SideChain	Blockchain B
CPU	4 cores	4 cores	4 cores
GPU	N/A	N/A	N/A
RAM	8 GB	8 GB	8 GB
Hard Drive	60 GB	60 GB	60 GB
OS	Ubuntu 22.04	Ubuntu 22.04	Ubuntu 22.04

Bảng 4.1: Môi trường thực nghiệm mạng liên chuỗi

Đối với việc huấn luyện mô hình học máy, yêu cầu công việc phải được thực hiện chạy liên tục trong thời gian dài với hiệu suất cao từ CPU, GPU cùng dung lượng lưu trữ và bộ nhớ lớn. Do đó, để đáp ứng được những yêu cầu tính toán khắt khe đó, nhóm đã thực hiện thiết lập môi trường như **Bảng 4.2**:

Thiết bị	Machine 1	Machine 2	Máy Google Collab
CPU	4 cores	4 cores	4 cores
GPU	N/A	N/A	T4
RAM	8 GB	16 GB	12.7 GB
Hard Drive	60 GB	60 GB	166 GB
OS	Ubuntu 22.04	Window 10	Ubuntu 20.04

Bảng 4.2: Môi trường thực nghiệm các phương pháp học máy

4.1.2. Chỉ số đánh giá

Trong nghiên cứu này, chúng tôi sử dụng 5 chỉ số để đánh giá hiệu suất mô hình, gồm: Accuracy, Precision, Recall, F1-score và Thời gian dự đoán. Để tính toán 4 chỉ số đầu tiên, chúng tôi áp dụng confusion matrix. Đây là một công cụ quan trọng trong các bài toán phân loại, cho phép tính nhanh các chỉ số trên thông qua các khái niệm True/False Positive/Negative. Nó thể hiện số lượng các điểm dữ liệu thực tế thuộc vào một lớp nào đó nhưng bị dự đoán thuộc lớp khác. Từ đó ta có được các định nghĩa:

- True Positive (TP) là những mẫu được dự đoán đúng thuộc lớp positive và thực tế cũng thuộc lớp positive.
- True Negative (TN) là những mẫu được dự đoán đúng thuộc lớp negative và thực tế cũng thuộc lớp negative.
- False Positive (FP) là những mẫu được dự đoán nhầm thuộc lớp positive nhưng thực tế lại thuộc lớp negative.
- False Negative (FN) là những mẫu được dự đoán nhầm thuộc lớp negative nhưng thực tế lại thuộc lớp positive.

Các độ đo đánh giá như Accuracy, Precision, Recall và F1-Score được định nghĩa và tính toán dựa trên số lượng mẫu được xác định là True Positive, True Negative, False Positive và False Negative. Từ đó ta có được các định nghĩa và công thức:

- Accuracy là chỉ số đo lường độ chính xác tổng thể của mô hình. Nó được tính bằng tỷ lệ phần trăm số lượng dự đoán đúng (bao gồm cả true positives và true negatives) trên tổng số quan sát. Giá trị Accuracy càng cao cho thấy mô hình dự đoán càng chính xác.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.1)$$

- Precision đánh giá mức độ chính xác của các dự đoán dương tính. Nó được tính bằng cách lấy tỷ lệ giữa số true positives và tổng số các dự đoán dương tính. Giá trị Precision cao chỉ ra rằng mô hình có độ tin cậy lớn khi dự đoán các quan sát là dương tính.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (4.2)$$

- Recall (hay Sensitivity) phản ánh khả năng của mô hình trong việc tìm ra tất cả các trường hợp dương tính. Nó được tính dựa trên tỷ lệ true positives trên tổng số quan sát thực sự dương tính. Giá trị Recall càng cao cho thấy mô hình càng có khả năng phát hiện hết các trường hợp dương tính mà không bỏ sót.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4.3)$$

- F1-Score là sự cân bằng giữa Precision và Recall. Nó được tính bằng công thức kết hợp Precision và Recall. F1-Score cho phép đánh giá tổng thể về độ chính xác và độ đầy đủ của mô hình. Mô hình có F1-Score cao được coi

là có hiệu suất tốt.

$$\text{F1-score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \quad (4.4)$$

Ngoài ra, thời gian dự đoán (Inference Time) là một trong những chỉ số quan trọng để đánh giá hiệu suất của mô hình máy học. Thời gian dự đoán phản ánh khoảng thời gian mà mô hình mất để đưa ra dự đoán cho một hoặc một tập các quan sát mới. Thông thường, thời gian dự đoán càng ngắn thì mô hình càng có hiệu suất cao, đặc biệt là trong các ứng dụng thời gian thực. Mô hình có thời gian dự đoán nhanh sẽ trả về kết quả nhanh hơn, cho phép hệ thống đưa ra quyết định hoặc hành động kịp thời. Ngược lại, thời gian dự đoán chậm có thể khiến trải nghiệm người dùng kém hơn. Vì vậy, khi thiết kế và đánh giá mô hình, thời gian dự đoán là một yếu tố then chốt cần xem xét.

4.1.3. Kịch bản thực nghiệm

Trước khi chúng tôi đi sâu vào các giai đoạn cụ thể của quá trình, hãy xem xét quá trình toàn bộ mà chúng tôi đã thiết kế để kiểm tra và phân loại lỗi trong các smart contract trên blockchain. Đầu tiên, chúng tôi xây dựng một mô hình cầu nối blockchain, tạo ra một môi trường mô phỏng cho việc tạo ra các smart contract. Tiếp theo, chúng tôi thu thập và nhãn mã smart contract để tạo ra một tập dataset đủ lớn để huấn luyện và kiểm tra mô hình. Sau đó, chúng tôi sử dụng các thuật toán học máy để xây dựng các mô hình phân loại lỗi smart contract, và quá trình huấn luyện được thực hiện trên tập dataset đã chuẩn bị trước đó. Để đánh giá hiệu suất, chúng tôi sử dụng các chỉ số như Accuracy, Precision, Recall, và F1-Score, và cuối cùng, mô hình tốt nhất sẽ được triển khai và kiểm tra trên giao diện ứng dụng để đảm bảo tính thực tế và hiệu quả trong môi trường thực tế. Bây giờ, chúng ta sẽ xem xét chi tiết từng bước trong quá trình này.

1. Xây dựng mô hình cầu nối blockchain: Mục tiêu của giai đoạn này là mô

phỏng quá trình sáng tạo các smart contract trên blockchain. Chúng tôi thiết lập một mô hình cầu nối để tái tạo các bước quan trọng trong việc tạo ra smart contract, tạo nên một môi trường mô phỏng cho nghiên cứu tiếp theo.

2. Thu thập và nhãn mác smart contract: Sau khi mô phỏng, chúng tôi thu thập các smart contract được tạo ra từ mô hình cầu nối. Các smart contract này sau đó được nhãn mác để tạo thành tập dataset cho quá trình huấn luyện và kiểm tra mô hình.
3. Áp dụng thuật toán học máy: Với tập dataset đã có, chúng tôi áp dụng các thuật toán học máy như máy học và học sâu để xây dựng các mô hình phân loại lỗi trong smart contract. Các thuật toán này được chọn để hiểu và dự đoán các lỗi có thể xuất hiện trong smart contract.
4. Huấn luyện mô hình: Quá trình huấn luyện được thực hiện trên tập dataset đã được chuẩn bị. Mô hình học từ dữ liệu để hiểu các đặc trưng quan trọng và mối quan hệ giữa chúng để có khả năng dự đoán lỗi một cách chính xác.
5. Đánh giá và so sánh mô hình: Sau khi huấn luyện, chúng tôi đánh giá hiệu suất của các mô hình bằng cách sử dụng các chỉ số như Accuracy, Precision, Recall, và F1-Score. Sự so sánh giữa các mô hình giúp chúng tôi chọn ra mô hình có hiệu suất tốt nhất.
6. Lựa chọn mô hình tốt nhất: Dựa trên kết quả đánh giá, mô hình có hiệu suất tốt nhất được lựa chọn để tiếp tục triển khai và kiểm tra trong các tình huống thực tế.
7. Triển khai trên giao diện ứng dụng: Mô hình tốt nhất được triển khai trên giao diện ứng dụng để kiểm tra và xác nhận tính khả thi của nó trong môi trường thực tế. Điều này đảm bảo rằng mô hình có thể được tích hợp một cách hiệu quả vào các ứng dụng blockchain thực tế.

4.2. Kết quả thực nghiệm

4.2.1. Kết quả về mặt hiệu suất

Bảng kết quả cung cấp cái nhìn tổng thể về khả năng phát hiện lỗ hổng bảo mật smart contract liên chuỗi của các mô hình. Các chỉ số Accuracy, Precision, Recall và F1 Score giúp đánh giá toàn diện hiệu suất theo nhiều khía cạnh. Mỗi mô hình có đặc điểm riêng và cách tiếp cận khác nhau. Thông qua các độ đo trên, chúng ta có thể phân tích kỹ lưỡng hiệu quả của từng mô hình, **Bảng 4.3** thể hiện kết quả về mặt hiệu suất. RoBERTa nổi bật là mô hình dẫn đầu về khả năng phân loại lỗ hổng smart contract, với độ chính xác cao nhất 96.67% so với các mô hình còn lại. Điểm nổi trội của RoBERTa so với các mô hình khác là độ đo và F1 score lần lượt đạt 100% và 93.33% - cao nhất trong tất cả các mô hình, thể hiện sự cân bằng tốt giữa độ chính xác và độ nhạy. Lý do RoBERTa thành công có thể là nhờ khả năng xử lý ngôn ngữ tự nhiên và ngữ cảnh của kiến trúc transformer, giúp phân tích hiệu quả các đoạn mã phức tạp trong hợp đồng thông minh.

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.8519	0.8209	0.8730	0.8462
Random Forest	0.7312	0.6324	1.0000	0.7748
XGBoost	0.6211	0.5485	0.9924	0.7065
SVM	0.8458	0.7551	0.9911	0.8571
Logistic Regression	0.9000	0.9071	0.8864	0.8967
CNN	0.9000	0.8235	1.0000	0.9032
LSTM	0.5500	0.5500	1.0000	0.7100
FNN	0.8125	0.8636	0.7037	0.7755
RoBERTa	0.9667	1.0000	0.8750	0.9333

Bảng 4.3: Hiệu suất của các mô hình học máy trong việc phát hiện lỗ hổng bảo mật trong các hợp đồng thông minh liên chuỗi

4.2.2. Kết quả về mặt thời gian

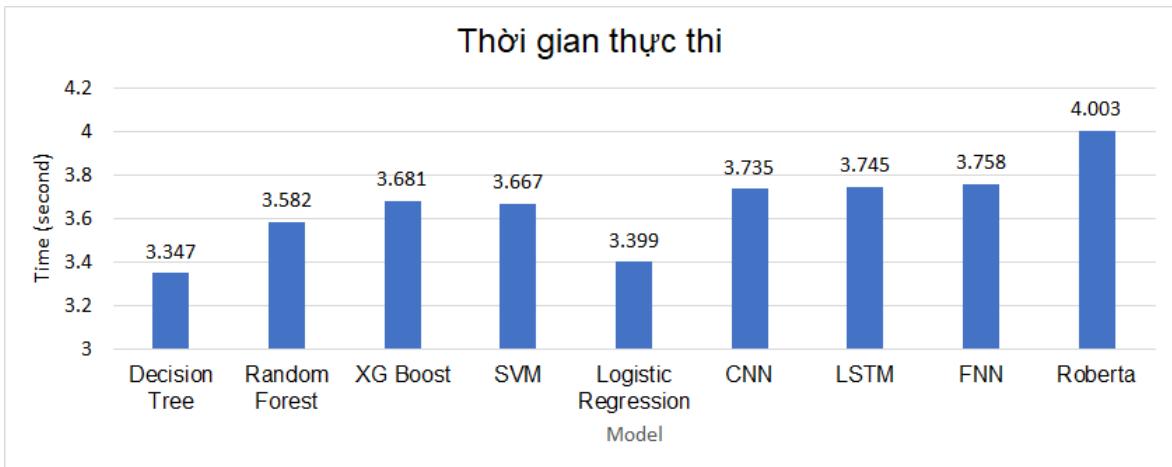
Có sự đa dạng đáng kể trong thời gian thực hiện của các mô hình. **Bảng 4.4** thể hiện kết quả về mặt thời gian thực thi của các mô hình. Các mô hình như Decision Tree, Random Forest và SVM có thời gian thực hiện tương đối ngắn, trong khoảng từ 3.3 - 3.7 giây. Điều này có thể là do tính đơn giản của các mô hình này, đặc biệt là Decision Tree và Random Forest.

Ngược lại, mô hình RoBERTa có thời gian thực hiện lớn nhất, với 4.003 giây. Điều này có thể là do sự phức tạp của mô hình transformer, nhưng thời gian vẫn ở mức chấp nhận được trong bối cảnh khả năng dự đoán cao của nó.

Model	Time Performance (seconds)
Decision Tree	3.347
Random Forest	3.582
XGBoost	3.681
SVM	3.667
Logistic Regression	3.399
CNN	3.735
LSTM	3.745
FNN	3.758
RoBERTa	4.003

Bảng 4.4: Thời gian thực thi của các mô hình học máy trong việc phát hiện lỗ hổng bảo mật trong các hợp đồng thông minh liên chuỗi

Thời gian thực hiện cần phải được xem xét kỹ lưỡng, đặc biệt là khi quyết định về việc triển khai mô hình trong các ứng dụng thời gian thực hoặc yêu cầu đào tạo định kỳ. Đối với các ứng dụng đòi hỏi hiệu suất thời gian nhanh, việc lựa chọn mô hình như Decision Tree hoặc SVM có thể là lựa chọn hợp lý, trong khi với các nhiệm vụ yêu cầu độ chính xác cao, RoBERTa vẫn là một lựa chọn mạnh mẽ mặc dù thời gian thực hiện cao nhất. **Hình 4.1** cho thấy về sự chênh lệch về mặt thời gian thực thi ở các mô hình.



Hình 4.1: Thời gian thực thi của các mô hình học máy trong việc phát hiện lỗ hổng bảo mật trong các hợp đồng thông minh liên chuỗi

4.3. Thảo luận

Hệ thống được đánh giá về khả năng phát hiện bảo mật các lỗ hổng và hành vi độc hại trong các hợp đồng thông minh xuyên chuỗi. Các mô hình học máy, đặc biệt là Roberta, có hiệu quả rất cao trong việc phân loại và xác định lỗ hổng, với độ chính xác vượt 96%. Điều này cho thấy khả năng của hệ thống trong việc diễn giải ngữ nghĩa hợp đồng để học các mô hình dự đoán chính xác phần mã độc hại.

Khả năng phát hiện bảo mật xuất phát từ cách tiếp cận độc đáo trong việc tạo tập dữ liệu xuyên chuỗi của hệ thống. Theo đó, dữ liệu hợp đồng sẽ được biến đổi thành các đặc trưng mạnh mẽ. Tổng thể, kết quả khẳng định độ bảo mật của hệ thống cho các mạng blockchain liên kết thông qua các mô hình học máy được huấn luyện trên tập dữ liệu xuyên chuỗi mới.

Việc tích hợp ChainSniper vào hệ thống cũng tăng cường thêm lớp bảo vệ thông qua khả năng phân tích động các phụ thuộc và tương tác giữa các hợp đồng xuyên blockchain. Bằng cách giám sát theo thời gian thực, ChainSniper có thể xác định các dấu hiệu của nỗ lực khai thác lỗ hổng. Sự kết hợp này cho phép giám sát bảo mật chủ động trên các mạng blockchain liên kết.

Về mặt thời gian giao động, các mô hình học máy đạt mức 3.3 - 3.7 giây. Trong khi đó các mô hình học sâu cho thời gian 3.7 - 4..0 giây. Cao nhất là mô hình RoBERTa với 4.003 giây. Mặc dù vậy, độ chính xác của các mô hình này đều rất cao và phù hợp ứng dụng thực tế.

CHƯƠNG 5. KẾT LUẬN

Trong phần này, chúng tôi đưa ra kết luận về công trình nghiên cứu và hướng phát triển trong tương lai của mô hình.

5.1. Kết luận

Nghiên cứu này cung cấp một giải pháp quan trọng nhằm tăng cường tính bảo mật cho hệ sinh thái smart contract liên kết chuỗi. Cụ thể, nhóm đã đề xuất và xây dựng một mô hình có tên gọi ChainSniper dựa trên công nghệ sidechain. Cùng với đó, nhóm nghiên cứu cũng đã xây dựng một tập dữ liệu mang tên CrossChainSentinel bao gồm 300 đoạn mã nguồn smart contract có gán nhãn. Trên nền tảng dữ liệu này, ChainSniper được thiết kế kết hợp một loạt các mô hình máy học tiên tiến, trong đó có các bộ phân loại học sâu, nhằm mục đích phát hiện các smart contract độc hại trong môi trường blockchain liên kết. Sau quá trình thử nghiệm và đánh giá chuyên sâu, mô hình đề xuất đạt kết quả rất khả quan với độ chính xác lên tới 96% trong nhiệm vụ phát hiện lỗ hổng smart contract dựa trên bộ dữ liệu được tạo mới. Các kết quả trên chứng minh tiềm năng lớn của việc kết hợp các công nghệ sidechain và máy học trong việc tăng cường bảo mật cho các hệ thống hợp đồng thông minh liên kết chuỗi. Công trình cung cấp một mô hình hiệu quả với độ chính xác cao, mang lại hướng tiếp cận triển vọng để nâng cấp tính an toàn cho các hợp đồng thông minh hoạt động trong môi trường phức tạp với nhiều blockchain kết nối với nhau. Như vậy, nghiên cứu đã mở ra cơ hội và hướng đi mới để cải thiện tính bảo mật cho hệ sinh thái smart contract, từ đó góp phần thúc đẩy sự phát triển mạnh mẽ của công nghệ blockchain trong tương lai.

5.2. Hướng phát triển

Nghiên cứu này mở ra nhiều hướng phát triển tiềm năng để nâng cao hiệu quả của mô hình ChainSniper trong tương lai. Một trong những khả năng chính là mở rộng tập dữ liệu CrossChainSentinel bằng cách bổ sung thêm các mẫu ví dụ lành tính (benign) và độc hại (vulnerable). Việc làm giàu dữ liệu đào tạo sẽ nâng cao khả năng khai quật hóa và độ chính xác của các mô hình máy học được áp dụng. Bên cạnh đó, việc tìm hiểu và ứng dụng các kiến trúc tiên tiến như mô hình Transformer hay chuyển giao kiến thức từ các ngôn ngữ tự nhiên sẽ giúp tăng khả năng hiểu ngữ nghĩa của mã smart contract, từ đó nâng cao chất lượng phát hiện lỗ hổng. Sự kết hợp này cho phép khai thác hiệu quả ngữ liệu tự nhiên để phân tích mã máy, mở ra hướng tiếp cận mới cho an ninh smart contract. Cuối cùng, mục tiêu lâu dài là nâng cấp liên tục cho ChainSniper để đạt được khả năng phát hiện hoàn toàn tự động các điểm yếu trong smart contract trước khi chúng được triển khai. Điều này cho phép thực hiện kiểm định tích cực để tăng độ tin cậy và bảo mật, thúc đẩy sự trưởng thành của các ứng dụng phi tập trung trên nhiều công nghệ blockchain. Nhìn chung, kết hợp giữa mở rộng dữ liệu, tối ưu thuật toán và nâng cao tính năng sẽ giúp mô hình ChainSniper ngày càng hoàn thiện, trở thành giải pháp đắc lực trong việc kiểm tra toàn diện smart contract nhằm giảm thiểu rủi ro bảo mật cho các hệ thống phức tạp liên kết chuỗi.

TÀI LIỆU THAM KHẢO

Tài liệu tham khảo

- [1] P. Patel and H. Patel, “Achieving a secure cloud storage mechanism using blockchain technology,” p. 130–142, 2023. [Online]. Available: <http://dx.doi.org/10.7763/IJCTE.2023.V15.1342>
- [2] M. Kumarathunga, R. N. Calheiros, and A. Ginige, “Sustainable microfinance outreach for farmers with blockchain cryptocurrency and smart contracts,” p. 9–14, 2022. [Online]. Available: <http://dx.doi.org/10.7763/IJCTE.2022.V14.1304>
- [3] T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song, “zkbridge: Trustless cross-chain bridges made practical,” in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3003–3017.
- [4] P. Han, Z. Yan, W. Ding, S. Fei, and Z. Wan, “A survey on cross-chain technologies,” *Distributed Ledger Technologies: Research and Practice*, vol. 2, no. 2, pp. 1–30, 2023.
- [5] Y. Hei, D. Li, C. Zhang, J. Liu, Y. Liu, and Q. Wu, “Practical agentchain: A compatible cross-chain exchange system,” *Future Generation Computer Systems*, vol. 130, pp. 207–218, 2022.
- [6] R. Lan, G. Upadhyaya, S. Tse, and M. Zamani, “Horizon: A gas-efficient, trustless bridge for cross-chain transactions,” *arXiv preprint arXiv:2101.06000*, 2021.

- [7] K. Qin and A. Gervais, “An overview of blockchain scalability, interoperability and sustainability,” *Hochschule Luzern Imperial College London Liquidity Network*, pp. 1–15, 2018.
- [8] T. Hardjono, “Blockchain gateways, bridges and delegated hash-locks,” *arXiv preprint arXiv:2102.03933*, 2021.
- [9] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghanianha, and K.-K. R. Choo, “Sidechain technologies in blockchain networks: An examination and state-of-the-art review,” *Journal of Network and Computer Applications*, vol. 149, p. 102471, 2020.
- [10] M. H. Miraz and D. C. Donald, “Atomic cross-chain swaps: development, trajectory and potential of non-monetary digital token swap facilities,” *arXiv preprint arXiv:1902.04471*, 2019.
- [11] J. Zhang, J. Gao, Y. Li, Z. Chen, Z. Guan, and Z. Chen, “Xscope: Hunting for cross-chain bridge attacks,” in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–4.
- [12] S. Sayeed, H. Marco-Gisbert, and T. Caira, “Smart contract: Attacks and protections,” *IEEE Access*, vol. 8, pp. 24 416–24 427, 2020.
- [13] N. Atzei, M. Bartoletti, and T. Cimoli, “A survey of attacks on ethereum smart contracts (sok),” in *Principles of Security and Trust: 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22–29, 2017, Proceedings 6*. Springer, 2017, pp. 164–186.
- [14] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F.-Y. Wang, “An overview of smart contract: architecture, applications, and future trends,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 108–113.

- [15] T. H.-D. Huang, “Hunting the ethereum smart contract: Color-inspired inspection of potential attacks,” *arXiv preprint arXiv:1807.01868*, 2018.
- [16] L. Zhang, W. Chen, W. Wang, Z. Jin, C. Zhao, Z. Cai, and H. Chen, “Cb-gru: A detection method of smart contract vulnerability based on a hybrid model,” *Sensors*, vol. 22, no. 9, p. 3577, 2022.
- [17] E. Lai and W. Luo, “Static analysis of integer overflow of smart contracts in ethereum,” in *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*, 2020, pp. 110–115.
- [18] M. Staderini, C. Palli, and A. Bondavalli, “Classification of ethereum vulnerabilities and their propagations,” in *2020 Second International Conference on Blockchain Computing and Applications (BCCA)*. IEEE, 2020, pp. 44–51.
- [19] H. Mao, T. Nie, H. Sun, D. Shen, and G. Yu, “A survey on cross-chain technology: Challenges, development, and prospect,” *IEEE Access*, 2022.
- [20] M. Rodler, W. Li, G. O. Karame, and L. Davi, “Sereum: Protecting existing smart contracts against re-entrancy attacks,” *arXiv preprint arXiv:1812.05934*, 2018.
- [21] P. Praitheeshan, L. Pan, J. Yu, J. Liu, and R. Doss, “Security analysis methods on ethereum smart contract vulnerabilities: a survey,” *arXiv preprint arXiv:1908.08605*, 2019.
- [22] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H.-N. Lee, “Systematic review of security vulnerabilities in ethereum blockchain smart contract,” *IEEE Access*, vol. 10, pp. 6605–6621, 2022.
- [23] J.-W. Liao, T.-T. Tsai, C.-K. He, and C.-W. Tien, “Soliaudit: Smart contract vulnerability assessment based on machine learning and fuzz testing,” in

2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS). IEEE, 2019, pp. 458–465.

- [24] S. Badillo, B. Banfai, F. Birzele, I. I. Davydov, L. Hutchinson, T. Kam-Thong, J. Siebourg-Polster, B. Steiert, and J. D. Zhang, “An introduction to machine learning,” *Clinical pharmacology & therapeutics*, vol. 107, no. 4, pp. 871–885, 2020.
- [25] M. Krichen, “Strengthening the security of smart contracts through the power of artificial intelligence,” *Computers*, vol. 12, no. 5, p. 107, 2023.
- [26] Y. Xu, G. Hu, L. You, and C. Cao, “A novel machine learning-based analysis model for smart contract vulnerability,” *Security and Communication Networks*, vol. 2021, pp. 1–12, 2021.
- [27] T. Haugum, B. Hoff, M. Alsadi, and J. Li, “Security and privacy challenges in blockchain interoperability-a multivocal literature review,” in *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*, 2022, pp. 347–356.
- [28] D. He, Z. Deng, Y. Zhang, S. Chan, Y. Cheng, and N. Guizani, “Smart contract vulnerability analysis and security audit,” *IEEE Network*, vol. 34, no. 5, pp. 276–282, 2020.
- [29] W. Deng, H. Wei, T. Huang, C. Cao, Y. Peng, and X. Hu, “Smart contract vulnerability detection based on deep learning and multimodal decision fusion,” *Sensors*, vol. 23, no. 16, p. 7246, 2023.
- [30] J. Huang, K. Zhou, A. Xiong, and D. Li, “Smart contract vulnerability detection model based on multi-task learning,” *Sensors*, vol. 22, no. 5, p. 1829, 2022.

- [31] B. Jiang, Y. Liu, and W. K. Chan, “Contractfuzzer: Fuzzing smart contracts for vulnerability detection,” in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 259–269.
- [32] R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, and A. Singh, “Empirical vulnerability analysis of automated smart contracts security testing on blockchains,” *arXiv preprint arXiv:1809.02702*, 2018.