Mục lục các minh chứng

ChainSniper: A Machine Learning Approach for Auditing Cross-Chain Smart Contracts

Tuan-Dung Tran
dungtran@uit.edu.vn
University of Information Technology, Vietnam National University
Ho Chi Minh city, Vietnam

Kiet Anh Vo
20520605@gm.uit.edu.vn
University of Information Technology, Vietnam National University
Ho Chi Minh city, Vietnam

Phan The Duy
duypt@uit.edu.vn
University of Information Technology, Vietnam National University
Ho Chi Minh city, Vietnam

Nguyen Tan Cam
camnt@uit.edu.vn
University of Information Technology, Vietnam National University
Ho Chi Minh city, Vietnam

Van-Hau Pham
haupv@uit.edu.vn
University of Information Technology, Vietnam National University
Ho Chi Minh city, Vietnam

ABSTRACT

Smart contracts are autonomous programs stored on blockchain networks that self-execute agreed terms in a transparent and accurate manner. Within cross-chain platforms, smart contracts facilitate interaction and exchange of data between diverse blockchains. However, the presence of vulnerabilities in smart contracts renders them susceptible to exploitation, jeopardizing security. Considerable research has focused on identifying and detecting such vulnerabilities, though existing approaches have yet to achieve comprehensive coverage. This paper presents ChainSniper, a sidechain-based framework integrating machine learning to automatically appraise vulnerabilities in cross-chain smart contracts. A comprehensive dataset, denoted "CrossChainSentinel", was compiled comprising 300 manually labeled code snippets. This dataset was leveraged to train machine learning models discerning vulnerable versus secure smart contracts. Experimental findings demonstrate the viability of machine learning methodologies for enhancing smart contract auditing within decentralized applications spanning multiple networks. Notable detection precision was achieved, substantiating ChainSniper's potential to strengthen security analysis through an automated and expansive evaluation of smart contract code.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**.

KEYWORDS

Cross-chain, Smart Contract, Vulnerability, Machine Learning

ACM Reference Format:
Tuan-Dung Tran, Kiet Anh Vo, Phan The Duy, Nguyen Tan Cam, and Van-Hau Pham. 2024. ChainSniper: A Machine Learning Approach for Auditing Cross-Chain Smart Contracts. In *2024 9th International Conference on Intelligent Information Technology (ICIIT 2024), February 23–25, 2024, Ho Chi Minh City, Vietnam.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3654522.3654577

ICIIT 2024, February 23–25, 2024, Ho Chi Minh City, Vietnam

ACM ISBN 979-8-4007-1671-3/24/02
https://doi.org/10.1145/3654522.3654577

1 INTRODUCTION

The idea behind blockchain is to build a decentralized network that provides capabilities for storing, exchanging, and verifying data and most importantly, makes services and transactions on the network more reliable and authentic [13, 21]. However, to have these powerful features, blockchain systems are often built independently with their own rules and protocols, leading to blockchains having no interoperability across chains. In order that, Cross-chain technology refers to a set of protocols and systems that enable interoperability and communication between multiple blockchain networks [29]. Cross-chain technology aims to connect these isolated ecosystems, allowing assets and data to be transferred and shared seamlessly across various blockchains [16]. This innovation is crucial in addressing the limitations of scalability, speed, and functionality that individual blockchains often face. By facilitating cross-chain communication, users can harness the advantages of different blockchains, enhance liquidity, and promote a more interconnected and efficient blockchain ecosystem. Different techniques like atomic swaps, wrapped tokens, sidechains, and decentralized bridges are used to allow interoperability between blockchain networks. This interoperability enables blockchain networks to work together seamlessly, moving toward a future where they can coexist in harmony.

Smart contracts are self-executing programs that automatically enforce the rules of an agreement. This automatizes the process of ensuring transactions follow predefined logic [19]. However, a cross-chain attack leveraging a smart contract vulnerability involves exploiting weaknesses in the code of a smart contract that operates across multiple blockchains. Such an attack can result in significant financial losses. For example, a recent incident witnessed a flaw in a cross-chain smart contract that managed assets worth over $600 million [31]. In addition, The DAO was a crowdfunding effort on the Ethereum blockchain that raised over $150 million in Ether before it was hacked in 2016 due to a smart contract flaw that allowed an attacker to steal more than $50 million worth of Ether, necessitating a hard fork to restore the funds. In July 2017, the

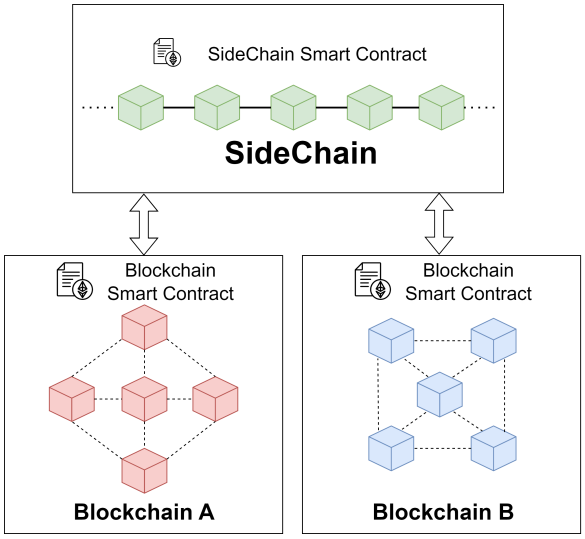


Figure 1: The interchain communication between two blockchains through a sidechain.

prominent Ethereum wallet Parity was hacked due to a vulnerability in a smart contract library, resulting in over $30 million worth of Ether being stolen. Furthermore, in August 2017, the Ethereum-based gambling platform KingDice was compromised due to a smart contract exploit, leading to almost $300,000 worth of Ether being stolen. Additionally, in 2021, various smart contract exploitation occurred on Binance Smart Chain leading to the theft of millions of dollars worth of cryptocurrency, including over $200 million stolen through a hacked Venus Protocol smart contract [1, 25].

With the escalating proliferation of cross-chain adoption, a comprehensive vulnerability analysis of smart contract code before deployment is imperative. Manual code auditing, though crucial, is labor-intensive and susceptible to errors [10]. This task is further complicated within the realm of cross-chain networks, exemplified by Polkadot, where communication among diverse blockchains is facilitated. The Ethereum blockchain ecosystem is significantly marred by critical vulnerabilities, namely reentrancy, overflow underflow attacks, and exposed Ether withdrawal within smart contracts and decentralized applications (dApps). Reentrancy epitomizes an assailant's capability to iteratively invoke a vulnerable contract's function prior to the completion of the preceding invocation, consequently engendering unexpected and undesirable behavioral outcomes [24]. In parallel, overflow and underflow attacks manifest when numeric values transcend their designated bounds, resulting in unpredictable consequences and potentially furnishing malevolent actors with exploitable opportunities [22]. Additionally, a stark concern materializes in the form of unprotected Ether withdrawal vulnerabilities, characterized by instances where a smart contract inadequately verifies withdrawal requests, thereby empowering malicious entities to siphon Ether from the contract without the requisite authorization [14]. The pernicious repercussions of these vulnerabilities reverberate throughout the Ethereum ecosystem, substantiated by a multitude of reported cases and their ensuing financial toll [25]. This pressing reality underscores the paramount importance of stringent security protocols in the sphere of blockchain development.

Machine learning is a field of artificial intelligence that enables computers to learn and improve from experience without being explicitly programmed. Recent advances in machine learning have enabled more complex neural network architectures like deep learning that can process massive datasets to make predictions and decisions, surpassing human capabilities in some tasks. Machine learning, having showcased notable promise in software code analysis and classification, assumes a pivotal role in this context [17]. Based on the concept of the previous study, we curate a dataset featuring snippets of Solidity code, meticulously annotated for vulnerabilities. Employing this dataset, we proceed to train a spectrum of classifiers, including Decision Trees, Random Forests, XGBoost, CNN (Convolutional Neural Network), Long Short-Term Memory (LSTM) networks, SVM with kernel (Linear, RBF, Poly, Sigmoid), Logistic Regression, Feed Forward Network (Neural Network), Roberta culminating in adept vulnerability detection.

While cross-chain networks allow for more complex decentralized apps by connecting multiple blockchains, they also increase the number of potential vulnerabilities that attackers could exploit [18]. Our machine learning-driven approach presents an automated and scalable framework for the discernment of vulnerabilities within this burgeoning paradigm. This paper introduces ChainSniper, a sidechain-based approach integrating machine learning to advance security in cross-chain environments. Our main contributions are as follows:

- We developed ChainSniper, encompassing robust logging, automated smart contract scanning, and runtime monitoring to facilitate secure inter-blockchain data transfer via a sidechain bridge.
- A novel dataset termed "CrossChainSentinel" was compiled, consisting of 300 manually labeled smart contract samples. This comprised 158 benign contracts and 142 contracts containing injected vulnerabilities including reentrancy flaws, overflow and underflow bugs, and unprotected ether withdrawal issues.
- We implement and evaluate various machine learning models on their ability to accurately detect malicious contracts in the cross-chain environment using the constructed dataset. Experimental findings demonstrate the viability of machine learning techniques for strengthening smart contract auditing across interconnected blockchain networks.

2 RELATED WORKS

Within this section, we delve into the realm of prior research pertaining to cross-chain technology, security vulnerabilities inherent in smart contracts, and the methodologies employed for the detection of said vulnerabilities.

2.1 Cross-chain Technology

Cross-chain technology aims to make it easier to transfer assets and share information between different blockchain networks [4]. Independent blockchain networks often have unique protocols and architectures that prevent interoperability between chains so that

comprehensive cross-chain solutions enable separate blockchain systems to share data and operate seamlessly together are required to fully realize the potential of blockchain technology [8]. Pioneering endeavors such as those documented in [6] have conducted a multivocal literature review (MLR) on security and privacy challenges in blockchain interoperability. The authors identified key vulnerabilities including wormhole attacks, denial of service, timing attacks, incompatible cryptography, and identifier leaks in hashed timelock contracts.

Currently, there are three main approaches for enabling interoperability and asset transfers between different blockchains: notary schemes, hash locking, and relays/sidechains. A notary scheme in blockchain refers to a consensus method where trusted third-party entities called notaries validate transactions by cryptographically signing and attesting to their legitimacy before addition to the blockchain, which helps prevent double spending [23]. Based on the authors of the study [5], hash locking is mentioned briefly as a potential function that can be implemented by a gateway to access a Hash Time Lock Contract (HTLC) at a third-party remote blockchain. This can help ensure payment is received before regenerating an asset representation on the destination blockchain. Another study [26] indicates that Sidechains are third blockchains connected to primary blockchains via a two-way peg, allowing assets to be transferred between chains; this lets new features be added without modifying the main blockchain protocol, helping to address limitations like scalability and privacy, but current implementations face issues around centralization risks and need for more decentralized options. However, this blockchain interoperability protocol allows different blockchain networks to communicate and transfer data but raises concerns about security and privacy vulnerabilities.

2.2 Smart Contract Vulnerabilities

Smart contracts are digital protocols that aim to simplify, verify, or enforce the negotiation or performance of a contract. They have a wide range of uses, including financial services, prediction markets, and the Internet of Things [28]. These contracts seamlessly operate on blockchain platforms, autonomously effecting actions contingent upon predetermined conditions, obviating the need for intermediaries. Thus, they facilitate trustless transactions and automate processes within the blockchain network.

Among the vulnerabilities that compromise their integrity, reentrancy manifests as a significant threat [32]. This flaw allows an external malevolent contract to repetitively invoke the vulnerable contract, disrupting its intended operation and granting unauthorized access to assets. Consequently, financial losses and unforeseen outcomes may transpire. Furthermore, the peril of integer overflow/underflow arises from mathematical operations yielding values surpassing the representation capacity of the data type. Malicious entities can exploit this weakness to manipulate arithmetic computations, potentially leading to unanticipated consequences and financial detriment[15]. Another vulnerability of concern is the 'Unprotected Ether Withdraw,' whereby a contract inadequately implements access control or validation, enabling unprotected Ether withdrawal by any entity [27]. Exploitation of this weakness by nefarious actors may result in the depletion of contract funds.

In a seminal work [7], He et al. elucidate the spectrum of smart contract vulnerabilities and propose remedies to mitigate these risks. They shortly summarize and draw comparisons among extant tools tailored to address these vulnerabilities. Additionally, Parizi et al. [20] contribute significantly to this domain by presenting a comprehensive testing tool designed to detect vulnerabilities, demonstrating the outcomes of their tool, and providing insightful comparisons with existing counterparts. However, these studies do not apply to the cross-chain concept with the vulnerabilities in smart contracts.

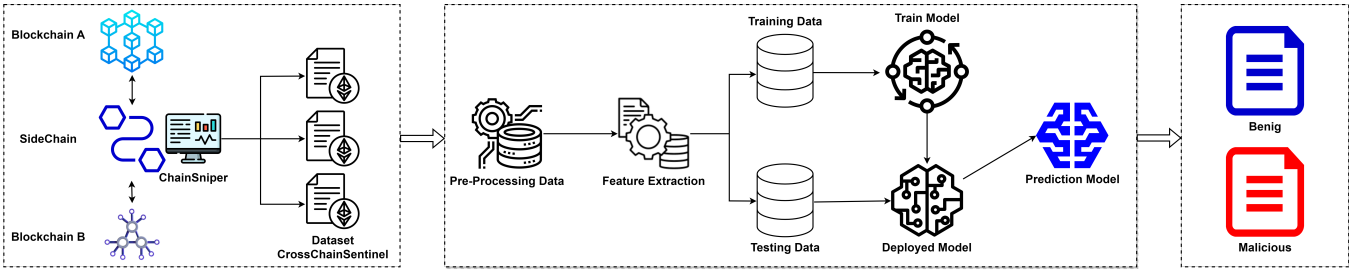
2.3 Vulnerabilities detection techniques

Recently, machine learning has experienced revived interest driven by the massive and constantly growing amount of data and computing power now available, as well as the development of more advanced learning techniques [2]. It involves training algorithms on datasets to recognize patterns and make predictions without explicit programming. For smart contracts, machine learning models can analyze the code to identify issues like reentrancy bugs, Integer overflow/underflow attacks, and Denial-of-Service (DoS) attacks [12]. By learning from examples of both vulnerable and secure smart contracts, machine learning models can verify contracts efficiently at scale. This automated vulnerability detection enables developers to identify and fix problems early on, leading to more powerful decentralized applications built on blockchain platforms. At the moment, machine learning models are showing promise as a way to enhance smart contract security in an automated manner.

Xu et al. [30] present a comprehensive overview of machine-learning techniques for detecting vulnerabilities in smart contracts. Their approach employs shared child nodes for analysis and incorporates the K-Nearest Neighbors model to efficiently detect vulnerabilities. They conduct comparative experiments, demonstrating superior accuracy over tools like Oyente and SmartCheck. A limitation is the model's focus on Solidity, necessitating refinements to pinpoint vulnerable lines of code. Also, Deng et al. [3] propose a new method for detecting vulnerabilities in smart contracts using deep learning and multimodal decision fusion. It extracts five distinct features to represent contracts and achieves high accuracy with this sophisticated process. However, the study does not employ unsupervised learning, which merits further explanation given its potential relevance for this problem.

Huang et al. [9] propose a smart contract vulnerability detection model using multi-task learning. It contains a shared bottom layer to extract features from opcodes, and task branches for detection and recognition. By setting an auxiliary recognition task, it learns directed vulnerability features to improve capabilities. Experiments show it can effectively detect issues like arithmetic bugs, reentrancy, and unknown calls. Overall, multi-task learning appears promising for enhancing automated security analysis of contracts. Jiang et al. [11] also demonstrate the efficacy of fuzzing and runtime monitoring for vulnerability detection in contracts. Their tool ContractFuzzer analyzes ABI specifications to generate test inputs defines oracles to check vulnerabilities, instruments the EVM to log details, and monitors execution to flag potential bugs. It detected over 450 vulnerabilities precisely in experiments, including the DAO bug.

Figure 2: The conceptual architecture of the ChainSniper



3 PROPOSED METHOD

In this section, we first present an elaborate exposition of the ChainSniper system and its constituent components, meticulously outlining their architectural intricacies. Additionally, we expound upon three distinct categories of vulnerabilities that afflict cross-chain smart contracts, providing a comprehensive elucidation of their nature and potential ramifications. Finally, we explicate the meticulous construction process employed in curating the CrossChainSentinel dataset.

3.1 Main model

The main model of the ChainSniper has 5 components: 1) Ethereum Sepolia plays a role in Blockchain A 2) Quorum plays a role in Blockchain B 3)The Sidechain transfers data and captures smart contracts, 4) The machine Learning model detects vulnerabilities, and 5) The module classifies the sidechain smart contracts. The system consists of two blockchain networks connected via a sidechain bridge that facilitates data transfer and captures smart contract information. This data is preprocessed and input to machine learning models trained on labeled data to detect malicious contracts. The models are validated to evaluate their performance. Finally, a module leverages the model predictions to classify sidechain smart contracts as either benign or malicious. Figure 2 shows the main model:

3.2 Cross-Chain and Potential vulnerabilities

3.2.1 Cross-Chain Networks Overview. The ChainSniper system enables interoperability between heterogeneous blockchains through a sidechain bridge. The sidechain processes and transfers data between the interconnected networks via a two-way peg mechanism that locks assets on one chain and unlocks equivalent representations on the other using multi-signature contracts. When cross-chain transactions occur, the sidechain nodes log the transaction's data such as contract addresses, timestamps, function calls, parameters, return values, and exceptions. This transaction log is aggregated into a dataset that provides insights into contract behaviors and execution patterns.

To identify vulnerabilities, static analysis extracts semantic security features from contract source code related to authentication, access control, and encryption. Dynamic analysis also extracts runtime trace features by executing test cases through the sidechain and analyzing the logs. The combined static and dynamic features train machine learning models to accurately detect vulnerable and malicious contracts across interconnected networks. Overall, the sidechain powers seamless interoperability while its logs feed an integrated security pipeline to enhance protection across chains. Figure 3 illustrates the main steps to simulate the transfer of data using a sidechain bridge:

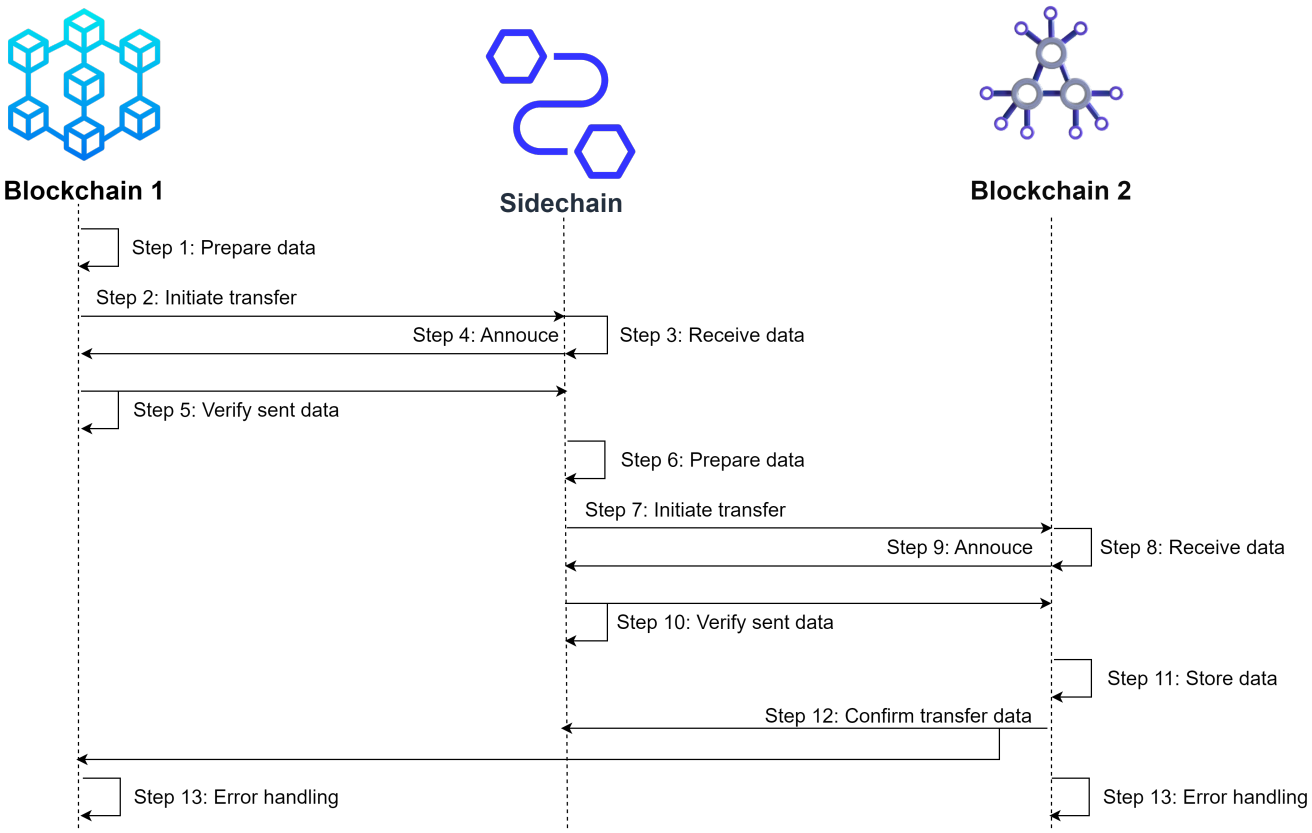
3.2.2 Overflow and Underflow attack in CrossChain. Overflow and underflow vulnerabilities can arise in CrossChain systems when values exceed or go below the permitted range during math operations, often in smart contracts. This can happen when proper input validation and safe math functions are not used. Attackers exploit these arithmetic bugs to intentionally cause overflow/underflow that lead to unexpected outcomes they can benefit from, like draining excess tokens from a contract. In CrossChain bridges, a lack of protection against these attacks could enable attackers to manipulate asset values as they transfer between chains and steal funds. The algorithm 1 simulates the Overflow/Underflow attack:

```
Function generate SourceChainContract,
DestinationChainContract
    sourceBalance =
    GetBalanceFromSourceChain(SourceChainContract)
    amountToSend = MAX_INT - sourceBalance
    if CheckTheValueIsValid(amountToSend) then
        SendFundsToDestinationChain(DestinationChainContract,
        amountToSend)
        destinationBalance = GetBalanceFromDestination-
        Chain(DestinationChainContract)
        newBalance = destinationBalance + amountToSend
        SetBalanceOnDestinationChain(DestinationChainContract,
        newBalance)
    end
end
```

Algorithm 1: The overflow/underflow attack algorithm

3.2.3 Reentrancy attack in CrossChain. Reentrancy vulnerabilities can be exploited in CrossChain systems when contract functions make external calls before updating the state. Attackers can manipulate execution flow to recursively call back into a function before its first invocation completes. For example, in a token transfer function, an attacker could call back in before the sender's balance is updated, allowing them to drain excess tokens. These recursive call attacks can allow hackers to steal funds by repeatedly calling functions before the balance state is updated. The algorithm 2 simulates the Reentrancy attack:

Figure 3: The sequence diagram of the interchain transfer data



Function ***generate*** *SourceChainContract, DestinationChainContract*
    LockContract()
    CallDestinationChain(DestinationChainContract)
    if *CallbackReceived()* **then**
        ProcessCallback()
        UpdateState()
        UnlockContract()
    end
end

Algorithm 2: The Reentrancy attack algorithm

Function ***generate*** *SourceChainContract, DestinationChainContract*
    withdrawalAmount = GetWithdrawalAmount()
    if *CheckTheValueIsValid(withdrawalAmount)* **then**
        WithdrawEtherFromSourceChain(SourceChainContract, withdrawalAmount)
        ReceiveEtherOnDestinationChain(DestinationChainContract, withdrawalAmount)
    end
end

Algorithm 3: Unprotected Ether Withdrawal attack algorithm

3.2.4 Unprotected Ether Withdrawal attack in CrossChain. CrossChain contracts that hold Ether can be vulnerable to attacks if access controls around withdrawals are not properly implemented. Attackers may exploit unchecked effects or lack of access restrictions in withdrawal functions to arbitrarily drain Ether funds. For example, a withdraw function without validation checks could allow any user to steal Ether by simply calling it. Other risks stem from calling self-destruct or using reentrancy to manipulate Ether transfers. Improper access controls and validations provide avenues for attackers to exploit and steal Ether funds. The algorithm 3 simulates the Unprotected Ether Withdrawal attack:

3.3 The CrossChainSentinel Dataset

3.3.1 Dataset Construction. A new dataset called CrossChainSentinel[1] was created to analyze vulnerabilities in sidechain bridge contracts. The dataset contains 300 smart contract files, with 158 benign and 142 malicious samples. The malicious contracts include 42 with reentrancy vulnerabilities, 48 with integer overflow/underflow bugs, and 52 with unprotected ether withdrawal issues. These were identified using techniques from prior research like Smartbugs-wild[2],

[1]https://github.com/anhkiet1227/CrossChainSentinel
[2]https://github.com/smartbugs/smartbugs-wild

SolidiFi-benchmark[3], and the Blacklists Ethereum address linked to multichain attacks. The cross-chain bridge data was modeled after 15 different real-world providers such as Commos, Avalanche, and Chainlink. The CrossChainSentinel dataset includes common vulnerabilities that can exist when assets are transferred between chains via sidechain contracts. By surfacing these risks in a large labeled dataset, CrossChainSentinel enables more robust auditing, testing, and detection of dangerous flaws in cross-chain bridging mechanisms before mainnet deployment. The diversity of cross-chain sources and vulnerability types makes this dataset well-suited for evaluating tools and models aimed at enhancing cross-chain security.

3.3.2 Dataset labeling and pre-processing. The data contains two sets of labels. The first set has binary labels indicating whether smart contracts are benign or malicious. The second set has multi-class labels categorizing malicious contracts into more specific vulnerability types: benign, mal-reentrancy, mal-overflow, and mal-unprotect. The data can be extracted into features including the file name (project), commit ID (commitID), label (target), contract structure and content (func), and file number (idx). The benign label is encoded as 0 and the malicious label as 1 for the binary classification set. The multi-class labels are encoded as 0 for benign, 1 for mal-reentrancy, 2 for mal-overflow, and 3 for mal-unprotect. Finally, change the smart contract files to feature vectors and put it into machine learning models to classify vulnerabilities in the contracts.

3.4 Machine Learning and Deep Learning Models in ChainSniper

3.4.1 Machine Learning Models. We choose machine learning approaches because they allow us to automatically detect vulnerabilities in smart contracts by discerning patterns in the contract code and structure. Compared to manual auditing, machine learning models can analyze contracts more efficiently and consistently. We experiment with several classical machine learning models, including Decision Trees, Random Forests, Support Vector Machines, XGBoost, and Logistic Regression. Owing to their interpretability features, these models enable an understanding of why certain contracts get flagged as vulnerable. Decision trees and random forests build hierarchical rule sets based on code properties to classify contracts. Meanwhile, SVMs find optimal boundaries between vulnerable and benign contracts in feature space. XGBoost further boosts accuracy through an ensemble of weak learners. We extract informative numerical features like function complexity, control flow, and syntax patterns. By training on properly represented data of both positive and negative examples, the models learn robust detectors of security flaws. We apply these techniques in ChainSniper to build an automated scanner for vulnerabilities like Reentrancy, Integer overflows/underflows, and Unprotected Ether Withdrawal. The flexibility of machine learning provides a methodology to detect exploits while avoiding extensive manual effort.

3.4.2 Deep Learning Models. We choose deep learning approaches because they can model the intricacies of code logic and dependencies without simplifying manual feature extraction. Techniques like CNNs, RNNs, LSTMs, and Transformer models such as RoBERTa enable end-to-end learning directly from raw smart contract source code to uncover complex syntactic and semantic patterns that characterize vulnerabilities. We experiment with deep neural networks including LSTM-based sequential models that interpret control flows, CNNs that extract local syntax patterns, and Transformer encoders like RoBERTa that provide rich contextual word representations. Owing to their layered hierarchical processing, these networks automatically learn latent relationships between tokens and structure that lead to an understanding of logic indicative of security flaws. For example, the long-range dependencies modeled by LSTMs can trace information flows in a contract to identify unchecked call issues. RoBERTa's contextual representations discern nuances between seemingly similar constructs that may or may not be vulnerable. The deep learning interpretable attention layers also facilitate identifying the problematic components detected. We apply these techniques in ChainSniper to build automated scanners that can flag potential Overflows/Underflows, Reentrancy, and Unprotected Ether Withdrawal. Avoiding manual feature engineering, deep neural networks can unlock intrinsic comprehension capabilities surpassing classical ML or human abilities for identifying intricate exploit patterns in smart contracts across different blockchains.

[3]https://github.com/DependableSystemsLab/SolidiFI-benchmark

4 EXPERIMENTS AND SECURITY ANALYSIS

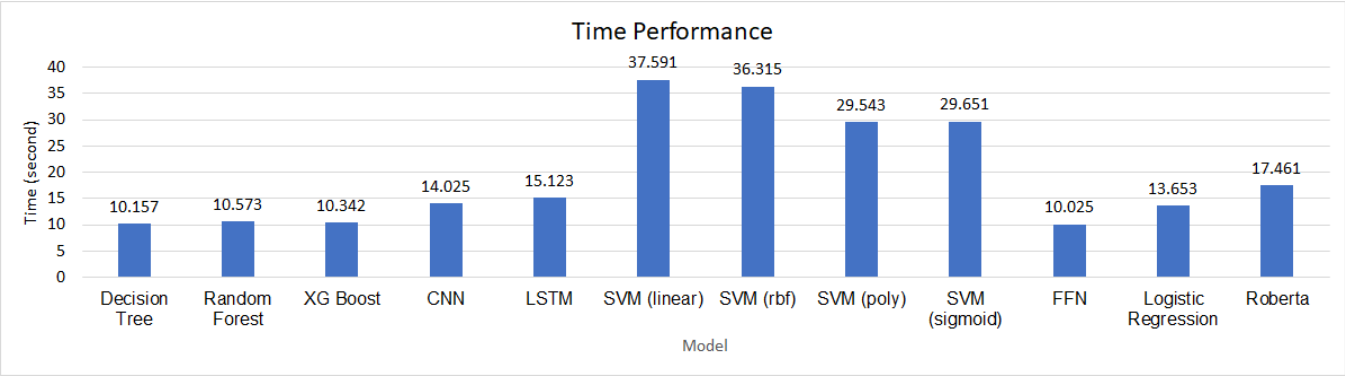
In this section, we perform a series of meticulous experiments, wherein we instantiate the ChainSniper system within a sidechain environment. Leveraging the power of machine learning and deep learning models, we diligently endeavor to unveil vulnerabilities ingrained within cross-blockchain smart contracts. Moreover, we subject the system to rigorous scrutiny, evaluating its performance through a quartet of metrics, namely Accuracy, Precision, Recall, and F1 Score. Additionally, we meticulously compute the average processing time for each machine learning model employed in the audit of cross-chain smart contracts, meticulously elucidating the temporal characteristics of the auditing process. Finally, we engage in an in-depth discourse on system security, expounding upon our findings and offering insightful commentary within the confines of the security analysis segment.

4.1 Experiments and Results

For the experiments, the sidechain system was implemented on 3 machines running Ubuntu 22.04, each with a 4-core CPU, 8 GB RAM, and 60 GB hard drive. The machine learning experiments utilized 2 machines - machine 1 ran Ubuntu 22.04 on a 4-core CPU with 8 GB RAM and 60 GB storage. Machine 2 used Google Colab Pro with an Ubuntu 20.04 environment, T4 GPU, 12.7 GB RAM, and 166 GB hard drive.

The CrossChain phase utilizes Sepolia Ethereum and Quorum as blockchain A and blockchain B to transfer data through a sidechain connecting the two blockchains. After that, the data was collected and split 80/20 into training and test sets. Accuracy, precision, recall, and F1 score were used as evaluation metrics. The data was tested on Random Forest, XGBoost, CNN, LSTM, SVMs with various kernels, Logistic Regression (LR), Feedforward Neural Network (FFN), and Roberta. Roberta achieved the best performance with an accuracy

Figure 4: The performance of the ChainSniper in detecting cross-chain smart contract vulnerabilities



of 0.967, precision of 0.999, recall of 0.875, and F1 score of 0.933. The next follow-up about performances was Logistic Regression and XGBoost. Overall, models like Roberta performed the best for classifying and identifying vulnerabilities in smart contracts, demonstrating their ability to effectively interpret the semantics and learn predictive patterns from the contract text and metadata. The table 1 gives information about the result of the experiment:

Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.598	0.723	0.618	0.666
Random Forest	0.603	0.620	0.735	0.672
XGBoost	0.725	0.720	0.996	0.836
CNN	0.721	0.715	0.993	0.831
LSTM	0.711	0.721	0.995	0.836
SVM (linear)	0.720	0.720	0.994	0.835
SVM (rbf)	0.722	0.724	0.954	0.823
SVM (poly)	0.726	0.727	0.996	0.840
SVM (sigmoid)	0.602	0.729	0.731	0.730
FFN	0.724	0.721	0.991	0.834
Logistic Regression	0.729	0.729	**0.999**	0.843
Roberta	**0.967**	**0.999**	0.875	**0.933**

Table 1: A Comparative Analysis: Performance of Machine Learning Models in Detecting Vulnerabilities within Cross-Chain Smart Contracts

The evaluation tested a variety of machine learning models on a classification task, measuring the inference time per sample in seconds. The fastest model was a simple feedforward neural network (FFN), completing inferences in 10.025 seconds on average. Other very fast models included decision trees (10.157s), random forests (10.573s), and XGBoost (10.342s). The linear support vector machine (SVM) was the slowest, taking 39.591 seconds per inference. In general, the neural network models like CNNs, LSTMs, and Roberta took longer than classical machine learning techniques like SVMs, random forests, and boosted trees. The figure 4 gives information about the time performance of the experiment:

4.2 Security Analysis

The system was evaluated for its ability to securely detect vulnerabilities and malicious behavior in cross-chain smart contracts. The machine learning models, especially Roberta, were highly effective at classifying and identifying vulnerabilities with accuracy exceeding 96% and precision nearing 99%. This demonstrates the system's capability to interpret contract semantics and metadata to learn predictive patterns that pinpoint malicious code. The secure detection capabilities stem from the system's unique cross-chain dataset creation approach that structures raw contract data into robust feature representations. Overall, the results validate the system's security strength for interoperable blockchain networks through its machine-learning models trained on a novel cross-chain dataset. Further security enhancements could involve expanding the diversity and size of training data, tuning additional model hyperparameters, and testing against new types of attacks.

The integration of ChainSniper into the system further supports its security capabilities for detecting vulnerabilities and malicious behavior in cross-chain smart contracts. ChainSniper provides an extra layer of protection through its ability to dynamically analyze real-time dependencies and interactions between contracts across multiple blockchains. By monitoring cross-chain activities as they occur, ChainSniper can identify various behaviors and transaction flows that may be indicative of exploitation attempts. Together, the synergistic combination of ChainSniper's dynamic tracing capabilities and the system's robust machine-learning models enables proactive security monitoring across interoperable blockchain networks.

5 CONCLUSION AND FUTURE WORKS

This study proposed and implemented a sidechain-based framework, termed ChainSniper, to enable secure inter-blockchain exchange of data for the automated analysis of smart contract security properties. A novel dataset 'CrossChainSentinel' comprising 300 labeled smart contract snippets was introduced. ChainSniper encompassed machine learning models including deep learning classifiers trained on CrossChainSentinel to detect malicious contracts across interconnected blockchain networks. Experimental findings demonstrate the potential of machine learning-driven approaches and sidechain architectures for strengthening smart contract auditing within multi-platform ecosystems and this vulnerability detection system achieved a high accuracy rate of 96%.

Going forward, avenues for improvement include enlarging CrossChainSentinel with additional benign and vulnerable samples. This would enhance model generalization capabilities. Further exploring advanced architectures such as Transformers leveraging transfer learning from pre-trained language models may boost semantic comprehension for vulnerability identification. Ultimately, iterative upgrades to ChainSniper aim to realize fully automated detection of smart contract weaknesses prior to deployment. This would allow proactive auditing to reinforce reliability and security, advancing the maturity of decentralized applications spanning multiple distributed ledger technologies. Combined with continued data augmentation and algorithm optimization, the proposed framework holds promise for comprehensively screening smart contracts to ameliorate exploitation risks within complex blockchain networks.

ACKNOWLEDGMENTS

This research is funded by the Faculty of Computer Networks and Communications, University of Information Technology, Vietnam National University Ho Chi Minh City, Vietnam.

REFERENCES

[1] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. 2017. 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, 164–186.

[2] Solveig Badillo, Balazs Banfai, Fabian Birzele, Iakov I Davydov, Lucy Hutchinson, Tony Kam-Thong, Juliane Siebourg-Polster, Bernhard Steiert, and Jitao David Zhang. 2020. An introduction to machine learning. *Clinical pharmacology & therapeutics* 107, 4 (2020), 871–885.

[3] Weichu Deng, Huanchun Wei, Teng Huang, Cong Cao, Yun Peng, and Xuan Hu. 2023. Smart contract vulnerability detection based on deep learning and multimodal decision fusion. *Sensors* 23, 16 (2023), 7246.

[4] Panpan Han, Zheng Yan, Wenxiu Ding, Shufan Fei, and Zhiguo Wan. 2023. A survey on cross-chain technologies. *Distributed Ledger Technologies: Research and Practice* 2, 2 (2023), 1–30.

[5] Thomas Hardjono. 2021. Blockchain gateways, bridges and delegated hash-locks. *arXiv preprint arXiv:2102.03933* (2021).

[6] Terje Haugum, Bjørnar Hoff, Mohammed Alsadi, and Jingyue Li. 2022. 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*. 347–356.

[7] Daojing He, Zhi Deng, Yuxing Zhang, Sammy Chan, Yao Cheng, and Nadra Guizani. 2020. Smart contract vulnerability analysis and security audit. *IEEE Network* 34, 5 (2020), 276–282.

[8] Yiming Hei, Dawei Li, Chi Zhang, Jianwei Liu, Yizhong Liu, and Qianhong Wu. 2022. Practical AgentChain: A compatible cross-chain exchange system. *Future Generation Computer Systems* 130 (2022), 207–218.

[9] Jing Huang, Kuo Zhou, Ao Xiong, and Dongmeng Li. 2022. Smart contract vulnerability detection model based on multi-task learning. *Sensors* 22, 5 (2022), 1829.

[10] TonTon Hsien-De Huang. 2018. Hunting the ethereum smart contract: Color-inspired inspection of potential attacks. *arXiv preprint arXiv:1807.01868* (2018).

[11] Bo Jiang, Ye Liu, and Wing Kwong Chan. 2018. Contractfuzzer: Fuzzing smart contracts for vulnerability detection. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 259–269.

[12] Moez Krichen. 2023. Strengthening the security of smart contracts through the power of artificial intelligence. *Computers* 12, 5 (2023), 107.

[13] Malni Kumarathunga, Rodrigo N. Calheiros, and Athula Ginige. 2022. Sustainable Microfinance Outreach for Farmers with Blockchain Cryptocurrency and Smart Contracts. , 9–14 pages. https://doi.org/10.7763/ijcte.2022.v14.1304

[14] Satpal Singh Kushwaha, Sandeep Joshi, Dilbag Singh, Manjit Kaur, and Heung-No Lee. 2022. Systematic review of security vulnerabilities in ethereum blockchain smart contract. *IEEE Access* 10 (2022), 6605–6621.

[15] Enmei Lai and Wenjun Luo. 2020. Static analysis of integer overflow of smart contracts in ethereum. In *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*. 110–115.

[16] Rongjian Lan, Ganesha Upadhyaya, Stephen Tse, and Mahdi Zamani. 2021. Horizon: A gas-efficient, trustless bridge for cross-chain transactions. *arXiv preprint arXiv:2101.06000* (2021).

[17] Jian-Wei Liao, Tsung-Ta Tsai, Chia-Kang He, and Chin-Wei Tien. 2019. 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, 458–465.

[18] Hanyu Mao, Tiezheng Nie, Hao Sun, Derong Shen, and Ge Yu. 2022. A survey on cross-chain technology: Challenges, development, and prospect. *IEEE Access* (2022).

[19] Mahdi H Miraz and David C Donald. 2019. Atomic cross-chain swaps: development, trajectory and potential of non-monetary digital token swap facilities. *arXiv preprint arXiv:1902.04471* (2019).

[20] Reza M Parizi, Ali Dehghantanha, Kim-Kwang Raymond Choo, and Amritraj Singh. 2018. Empirical vulnerability analysis of automated smart contracts security testing on blockchains. *arXiv preprint arXiv:1809.02702* (2018).

[21] Parin Patel and Hiren Patel. 2023. Achieving A Secure Cloud Storage Mechanism Using Blockchain Technology. , 130–142 pages. https://doi.org/10.7763/ijcte.2023.v15.1342

[22] Purathani Praitheeshan, Lei Pan, Jiangshan Yu, Joseph Liu, and Robin Doss. 2019. Security analysis methods on ethereum smart contract vulnerabilities: a survey. *arXiv preprint arXiv:1908.08605* (2019).

[23] Kaihua Qin and Arthur Gervais. 2018. An overview of blockchain scalability, interoperability and sustainability. *Hochschule Luzern Imperial College London Liquidity Network* (2018), 1–15.

[24] Michael Rodler, Wenting Li, Ghassan O Karame, and Lucas Davi. 2018. Sereum: Protecting existing smart contracts against re-entrancy attacks. *arXiv preprint arXiv:1812.05934* (2018).

[25] Sarwar Sayeed, Hector Marco-Gisbert, and Tom Caira. 2020. Smart contract: Attacks and protections. *IEEE Access* 8 (2020), 24416–24427.

[26] Amritraj Singh, Kelly Click, Reza M Parizi, Qi Zhang, Ali Dehghantanha, and Kim-Kwang Raymond Choo. 2020. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. *Journal of Network and Computer Applications* 149 (2020), 102471.

[27] Mirko Staderini, Caterina Palli, and Andrea Bondavalli. 2020. Classification of ethereum vulnerabilities and their propagations. In *2020 Second International Conference on Blockchain Computing and Applications (BCCA)*. IEEE, 44–51.

[28] Shuai Wang, Yong Yuan, Xiao Wang, Juanjuan Li, Rui Qin, and Fei-Yue Wang. 2018. An overview of smart contract: architecture, applications, and future trends. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 108–113.

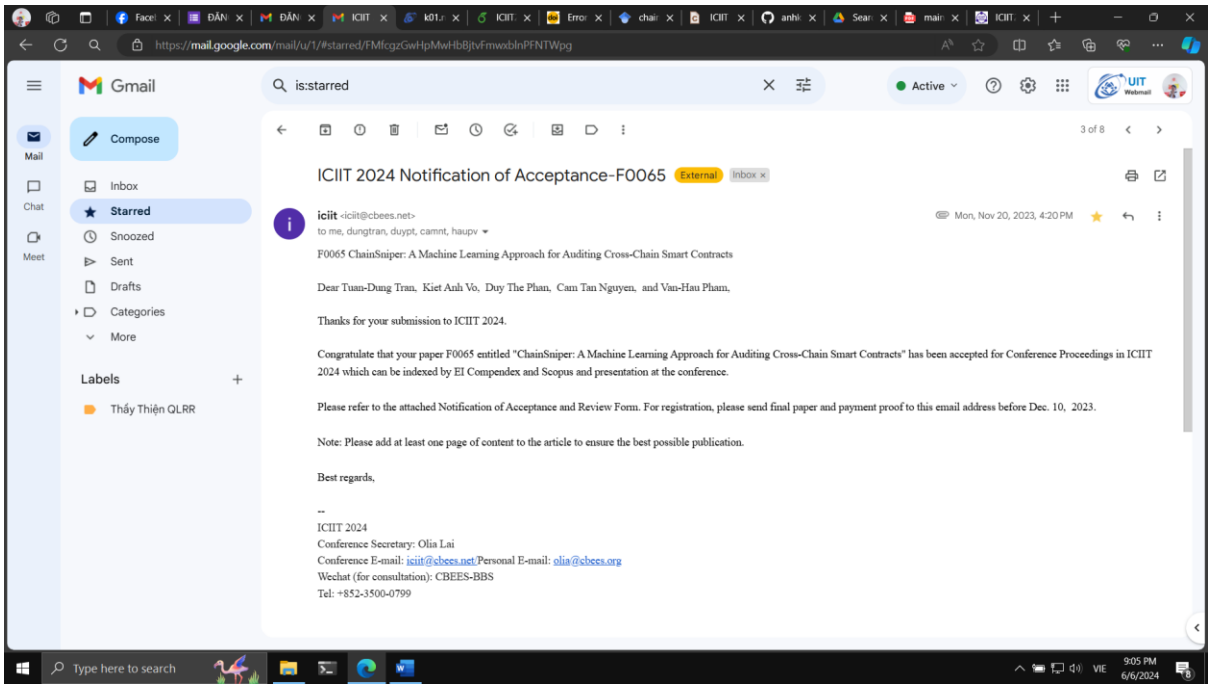
[29] Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, Fan Zhang, Yupeng Zhang, Yongzheng Jia, Dan Boneh, and Dawn Song. 2022. zkbridge: Trustless cross-chain bridges made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 3003–3017.

[30] Yingjie Xu, Gengran Hu, Lin You, and Chengtang Cao. 2021. A novel machine learning-based analysis model for smart contract vulnerability. *Security and Communication Networks* 2021 (2021), 1–12.

[31] Jiashuo Zhang, Jianbo Gao, Yue Li, Ziming Chen, Zhi Guan, and Zhong Chen. 2022. Xscope: Hunting for cross-chain bridge attacks. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–4.

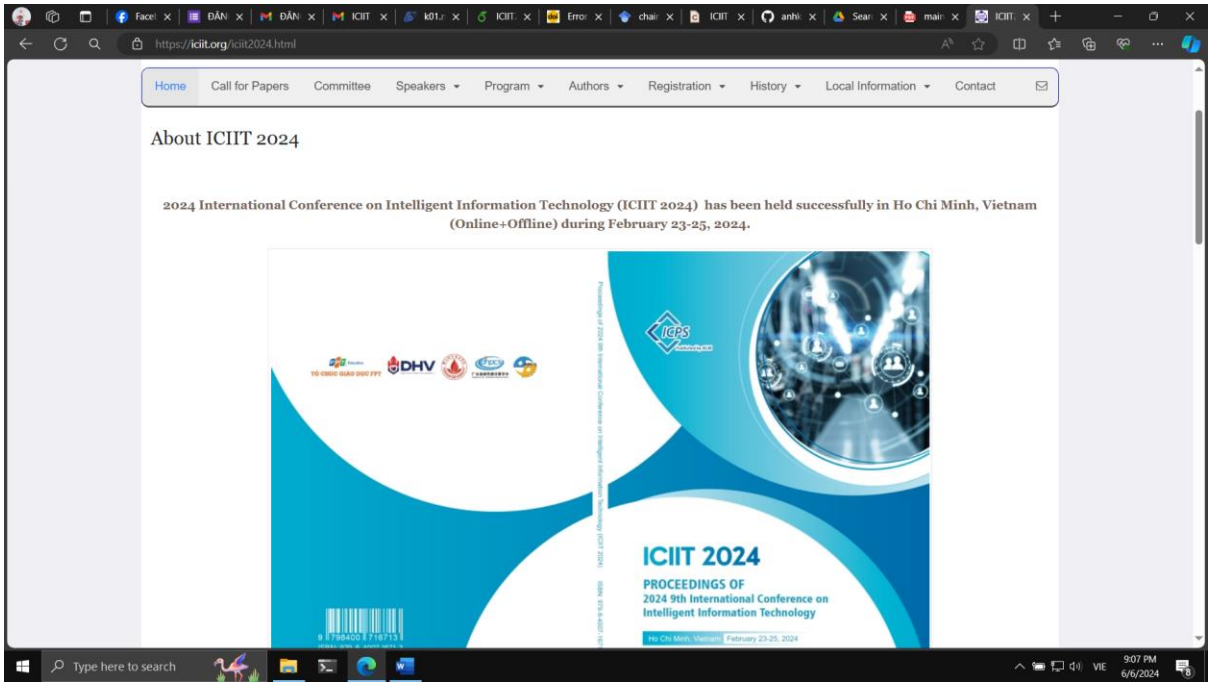
[32] Lejun Zhang, Weijie Chen, Weizheng Wang, Zilong Jin, Chunhui Zhao, Zhennao Cai, and Huiling Chen. 2022. Cbgru: A detection method of smart contract vulnerability based on a hybrid model. *Sensors* 22, 9 (2022), 3577.

Thư chấp nhận

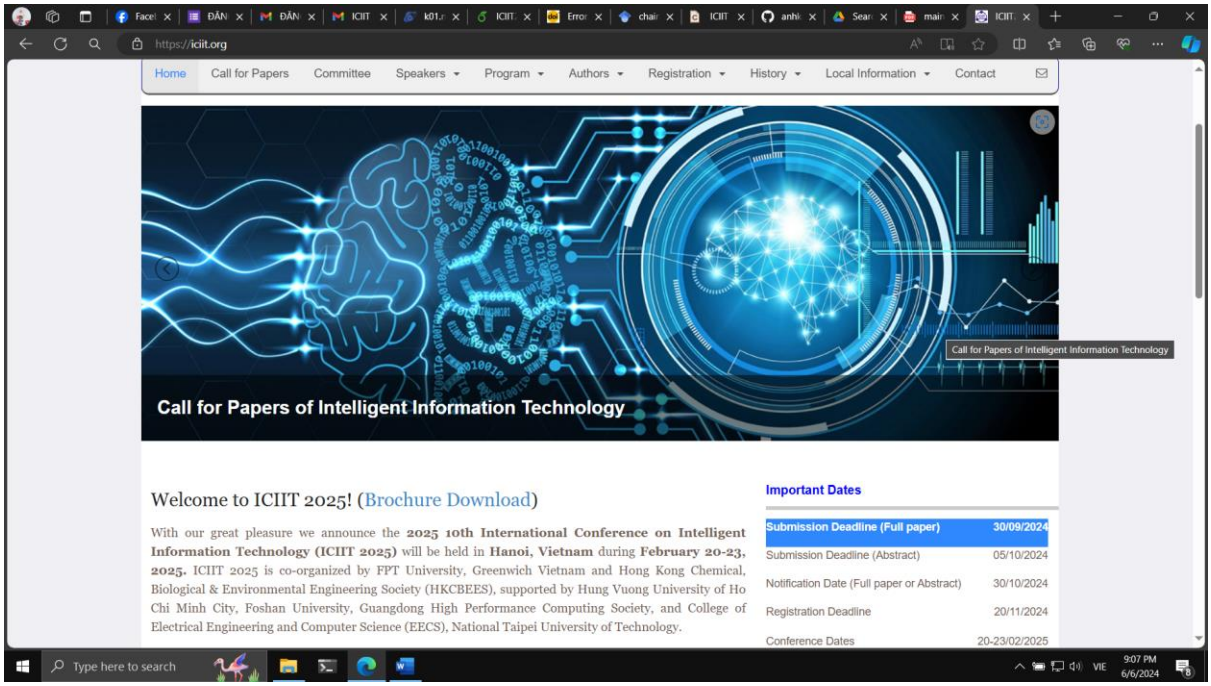


Trang homepage hội nghị

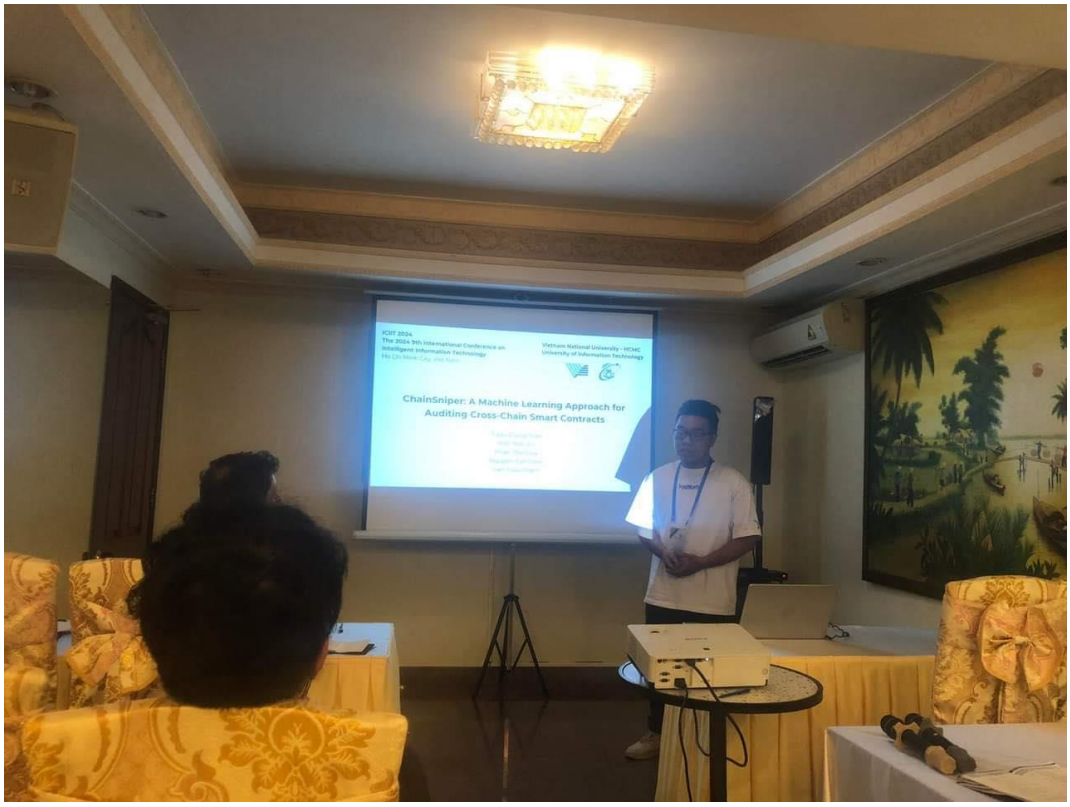
https://iciit.org/iciit2024.html



https://iciit.org/



Hình ảnh tại buổi hội nghị



Giấy chứng nhận tham gia hội nghị