# DeFiScanner: Spotting DeFi Attacks Exploiting Logic Vulnerabilities on Blockchain

Bin Wang, Xiaohan Yuan, Li Duan, Hongliang Ma, Bin Wang, Chunhua Su, and Wei Wang, *Member, IEEE*

***Abstract*— With the rapid development of decentralized financial (DeFi), the total value locked (TVL) in DeFi continues to increase. A big number of adversaries exploit logic vulnerabilities to attack DeFi applications for profit, such as flash loan attacks and price manipulation attacks. However, the current vulnerability detection tools for smart contracts cannot be directly used to detect the logic vulnerabilities generated by the combination of different protocols. How to characterize and detect DeFi attacks that exploited logic vulnerabilities is a big challenge. In this work, we propose a deep-learning-based attack detection system on DeFi, called DeFiScanner, in which we design a novel neural network that includes a global model, a local model, and a fusion model to characterize DeFi attacks. First, the unstructured emitted events are automatically and efficiently normalized. Second, the transaction-related features of normalized emitted events are enriched with the global model and the semantic features of emitted events are extracted with the local model. Finally, the transaction-related features and the semantic features of emitted events are fused efficiently with the fusion model to detect DeFi attacks. We collect a dataset that consists of 50 910 real-world DeFi transactions on Ethereum (ETH). The extensive experimental results demonstrate the effectiveness of DeFiScanner. The true positive rate (TPR) and the area under the receiver operating characteristic (ROC) curve of the system reach 0.91 and 0.97, respectively.**

***Index Terms*— Attacks detection, blockchain, decentralized finance, deep learning.**

## I. Introduction

UNLIKE traditional finance, decentralized financial (DeFi) takes full advantage of the transparency and openness of a decentralized network (i.e., blockchain) to provide diverse financial services without relying on the third-party intermediaries. As of the end of May 2022, the total value locked (TVL) in DeFi applications reaches to $55 billion. With the rapid growth of DeFi applications on public blockchain ecosystems, such as Ethereum (ETH) [1], external attacks against DeFi applications have also emerged, which seriously threaten the financial security of DeFi participants. Many security issues related to DeFi have been reported, including pump-and-dump (P&D) scams [2], [3], front-running [4], [5], [6], and flash loan attacks [7], which have brought many security incidents [8], [9], [10], [11], [12]. Attackers compromise DeFi applications through code and logic vulnerabilities, causing huge losses to the DeFi ecosystem.

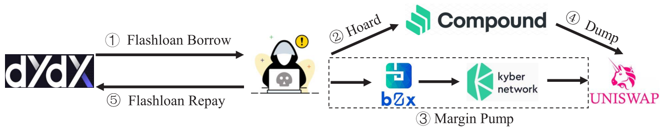


Fig. 1. Five composable DeFi protocols in bZx hack.

To illustrate the attacks, we take an example that took place in the bZx project [10]. The bZx is a DeFi project for lending and margin trading on Ethereum. The platform uses many other DeFi protocols to provide these services. Anyone can add funds to the fund pool of bZx to borrow other tokens and take advantage of long or short positions by trading other assets on margin. Due to the complex combination of protocols, attackers can exploit the dependence of bZx's oracle with other DeFi projects to manipulate the exchange rate of crypto assets and profit.

As shown in Fig. 1, the attack transaction (called external transaction) can be broken down into five steps (called internal transactions): flashloan borrow, hoard, margin pump, dump, and flashloan repay. First, the attacker takes advantage of the flashloan service of dYdX to borrow a large amount of ETH. Second, the attacker deposits parts of ETH into compound to borrow wrapped bitcoin (WBTC). After hoarding, the attacker uses the margin trade service of bZx to short ETH in favor of WBTC. Specifically, bZx forwards the order to KyberSwap to complete the transaction. KyberSwap consults its reserves and finds the best exchange rate. The price of WBTC in Uniswap is tripled by the attacker. Finally, the attacker swaps the WBTC borrowed from compound back for WETH in Uniswap to gain profits and repays the flashloan to dYdX.

The core step of arbitrage (i.e., profiting by buying and selling commodities at different prices) is that attackers

Manuscript received 2 September 2022; revised 24 October 2022 and 26 November 2022; accepted 29 November 2022. Date of publication 21 December 2022; date of current version 3 April 2024. This work was supported in part by the National Natural Science Foundation of China under Grant U21A20463 and Grant 61902021. *(Corresponding authors: Wei Wang; Bin Wang.)*

Bin Wang, Xiaohan Yuan, Li Duan, and Wei Wang are with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, Beijing 100044, China (e-mail: bin.wang@bjtu.edu.cn; xiaohan.yuan@bjtu.edu.cn; duanli@bjtu.edu.cn; wangwei1@bjtu.edu.cn).

Hongliang Ma is with the School of Information Science and Technology, Shihezi University, Shihezi 832003, China (e-mail: mhl_inf@shzu.edu.cn).

Bin Wang is with the Zhejiang Key Laboratory of Multi-Dimensional Perception Technology, Application and Cybersecurity, Hangzhou 310053, China (e-mail: bin_wang@zju.edu.cn).

Chunhua Su is with the Department of Computer Science and Engineering and the Division of Computer Science, University of Aizu, Aizuwakamatsu 965-8580, Japan (e-mail: chsu@u-aizu.ac.jp).

Digital Object Identifier 10.1109/TCSS.2022.3228122

successfully control the exchange rate of a crypto asset pair by exploiting the price dependencies of bZx.

The existing research work and tools [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28] mainly focus on privacy protection, malicious code detection, and detecting the vulnerabilities of smart contracts, such as the re-entrancy, timestamps dependency, and short address attack. Specifically, there have been a lot of works [29], [30], [31], [32], [33], [34], [35], [36], [37] on detecting abnormal behaviors, such as money laundering, frauding, and Ponzi scheme. However, on one hand, since many attacks against DeFi projects use the combinations of different DeFi protocols, these tools cannot be used to detect the logic vulnerabilities in DeFi projects as shown in Fig. 1. On the other hand, internal transactions contain high-level semantic features, e.g., a user deposits an amount of ETH into a decentralized exchange (DEX) to borrow WBTC. We cannot extract these features from the code level of smart contracts.

To detect the attacks against DeFi applications, we need to analyze internal transactions and extract semantic features of the combination of different DeFi protocols. There are two challenges in detecting attacks against DeFi applications. First, internal transactions are very complicated. Attackers exploit the logic vulnerabilities generated by the combinations of protocols to launch flash loan attacks for profit. The diverse and complex combinations between protocols bring great challenges to attack detection. External transactions usually contain a large number of internal transactions. In addition, since the data of internal transactions are unstructured, it is difficult to directly input them into a model for detection. We need to efficiently process large amounts of unstructured data and transform them into structured (semi-structured) data. Second, extracting the multilevel DeFi semantic information is difficult. Complex internal transactions often contain different levels of semantic information, such as external transaction level and internal transaction level. The external transaction level contains the block number, timestamp, and other global information of the transaction. High-level semantic information is contained at the internal transaction level. It usually takes a lot of human efforts to define features and obtain semantic information. We need to build a neural network that can extract effective features from different levels of semantic information and perform attack detection.

To address the above challenges, we propose a system, called DeFiscanner, to perform large-scale attack detection on DeFi applications with the deep learning methods. Considering the complexity of internal transactions, we do not directly process all the internal transactions. Instead, we use the events generated by internal transactions to extract high-level features. The events filter many unnecessary internal transactions. To automatically and efficiently process unstructured data, we adapt the word2vec [38] method to vectorize events and feed the vectorizes into a deep learning model. We also define a series of features from the account level and external transaction level guided by prior knowledge to enrich high-level features. With the designed neural network, the features of different levels are fused efficiently to detect attacks. To evaluate the effectiveness of DeFiScanner, we collect 50 910 real-world external transactions of DeFi applications from January 2021 to April 2022. We use seven models, namely, support vector machine (SVM) [39], $k$-means [40], autoencoder [41], deepautoencoder, long short-term memory (LSTM) [42], convolutional neural networks (CNN) [43], and LSTM-CNN to detect attack transactions.

We make the following contributions.

1) We propose a system called DeFiScanner for large-scale attack detection on DeFi applications with the deep learning methods. DeFiScanner can process unstructured emitted events generated by DeFi applications and fuse the multilevel features to extract high-level semantic features. To the best of our knowledge, our work is the first deep-learning-based system to detect DeFi attacks.
2) We design a neural network in DeFiScanner. It can fuse different kinds of features from external transactions and emitted events to accommodate multilevel features. It provides an effective fused method for deep-learning-based attack detection on DeFi.
3) We evaluate the effectiveness of DeFiScanner by collecting 50 910 real-world transactions on DeFi applications. The experimental results indicate that DeFiScanner can detect DeFi attacks effectively. The true positive rate (TPR) and the area under the receiver operating characteristic (ROC) curve of the system reach 91.11% and 0.97, respectively.

The remainder of this article is organized as follows. In Section II, we introduce the relevant works. We briefly introduce some concepts about Ethereum and DeFi applications in Section III. Next, we descript the details of DeFiScanner in Section IV. The setup of experiment is shown in Section V. In Section VI, we evaluate the performance of DeFiScanner. We conclude our work in Section VII.

## II. Related Work

### A. Ethereum DeFi Ecosystem

*1) Order Book:* Order book is the most common way to realize transactions on DeFi. The purpose of order book is to match the trading intentions of buyers and sellers. Some platforms such as Coinbase [44] and Binance [45] are implemented based on a centralized order book. Coinbase is the largest trading volume in the world and the first centralized exchange listed on Nasdaq [46]. dYdX [47] is a classic implementation of DEXs. Serum [48] implements a relatively special DEX. 0× Relayer [49] adopts the order matching method of the open order book, which continuously accepts and broadcasts orders with all 0 addresses. Waves Exchange [50] is a platform for exchanging stable currencies, which provides exchange services between Waves tokens and stable currencies that are anchored to the legal currencies of various countries. The exchange between the digital currency and the stablecoin is carried out through the first-in-first-out (FIFO) mechanism, and the stable currency maintains the stability of the legal currency based on the Neutrino [51] protocol. Injective [52] proposes an order matching mechanism based on frequent batch auctions (FBAs).

*2) Automated Market Maker:* The DEXs based on the automatic market maker (AMM) mechanism are currently the most popular choice in the industry. Uniswap [53] is one of the classics among market maker mechanisms. Uniswap v3 [54] is an improvement of the Uniswap protocol. The main difference between them is the constant product function. Similar to Waves Exchange in the transaction order book, AMMs also have mechanisms to support stablecoin transactions, such as StableSwap [55]. SushiSwap [56] is a protocol almost the same as Uniswap, with the only difference being the structure of community governance. SushiSwap plundered the liquidity of Uniswap through a "vampire attack" [57] in August 2020. Raydium [58] is a mechanism that combines AMMs and transaction order books.

The relationship between DEXs is complex and an external transaction often involves a large number of DeFi protocols. With the combination of these protocols, it is easy to generate logical vulnerabilities that can be exploited by attackers.

### B. Flash Loan Attacks

Due to the convenience of flash loans, attackers often use flash loans to boost profits, which brings serious financial security problems to the development of the DeFi ecosystem. Since the flash loan attack has just appeared in the past two years, there are only some initial research works about it. Wenkai et al. [59] conduct a systematic examination of the security issues of the DeFi ecosystem built on blockchain. Qin et al. [7] analyze the two initial flash loan attacks and redistribute the funds used in each step to maximize the revenue. Wang et al. [60] propose a flash loan transaction identification method to identify flash loan transactions on Uniswap, Aave, and dYdX. Cao et al. [61] propose a system to visualize capital flows of flash loan attacks. Wang et al. [62] implement a simple system to detect attack transactions on DeFi, which identified transactions with extremely high yields as attack transactions. Most of these works only introduce the principle of flash loan attacks and cannot automatically and accurately detect attacks on DeFi.

## III. Background

We briefly introduce Ethereum transactions, cryptocurrencies, DeFi, and flash loan attacks in this section to better understand our work.

### A. Ethereum Transactions

The accounts in Ethereum are divided into two types: external accounts and contract accounts. External accounts are created and remain unchanged when users join Ethereum. Each external account has a pair of public–private keys, the private key is used to sign transactions, and the public key is used to generate an Ethereum address. There are three types of transactions in Ethereum: one is a transfer transaction, which is the general exchange of funds between two accounts, and there is often no additional information in the transaction. The second is a contract creation transaction. Users can publish contracts to the blockchain through the transaction. The third is the contract invocation transaction. External accounts or other contracts can invocate certain functions in the contract with this kind of transaction, as shown in Fig. 2. In this article, we refer to external transactions and internal transactions collectively as transactions.

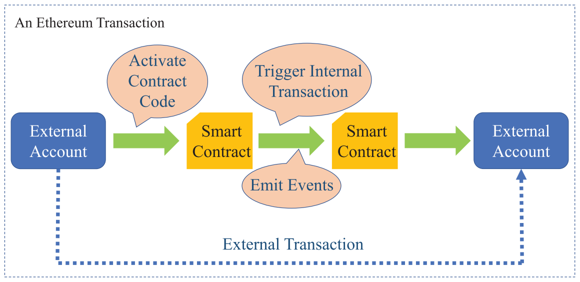


Fig. 2. Ethereum transaction.

Events are inheritable members of contracts. They are declared in the codes of smart contracts and can be emitted with the emit keywords. Events record the related parameters in the transaction logs. These logs are stored on the blockchain and can be accessed using the address until the contract appears on the blockchain.

### B. Cryptocurrencies

In Ethereum, digital cryptocurrencies are divided into two categories: Ether (the native token) and ERC20 (Ethereum Request for Comment 20) token. Ether is generated along with Ethereum, and transaction fees in Ethereum have to be paid with Ether. ERC20 tokens are attached to Ethereum and created through smart contracts. DEXs can deploy a smart contract that conforms to the ERC20 standard to publish an ERC20 token. Since there are kinds of ERC20 tokens in Ethereum, we only introduce the stablecoins and liquidity provider (LP) tokens.

Stablecoin is designed to maintain the stability of the market, which relates to an anchored fiat currency (usually the U.S. dollar). Dai [63] of MarkerDAO is one of the notable implementations. It maintains price stability by over-collateralizing assets and triggering liquidations when the value of the collateralized assets falls below the collateral line. USD coin (USDC) [64] and tetherUS (USDT) [65] are also popular stablecoins. The stablecoins provide a valuable reference for transactions. The value of digital cryptocurrencies and other equity tokens can be linked with anchored fiat currency through stablecoins.

LP tokens are issued to crypto liquidity pools on DEXs with AMM protocols. Amounts of platforms on DeFi provide the service, such as Uniswap and Balancer. The Balancer is a popular DEX that distributes LP tokens to their LPs. LPs can sell LP tokens and use LP tokens to exchange for other tokens. The corresponding liquidities can be redeemed at any time.

### C. Decentralized Finance

DeFi applications involve multilevel protocols. The most important are protocols of the on-chain asset exchange. The design of the transaction mechanism is related to the stability of complex applications. There are two main components of DeFi, one is the stablecoins represented by Bitcoin and Ethereum, and the other is the smart contract that realizes trading, lending, and investment.

1) *Decentralized Exchange:* DEX is a peer-to-peer marketplace that allows trading cryptocurrencies directly between two parties. DEX is designed to solve problems inherent in a centralized exchange, such as centralized custody of assets, geographic restrictions, and asset selection. DEX uses AMMs instead of the traditional order books. Without matching individual buy and sell orders, users can deposit assets into a pool, and the price is determined based on the ratio between the assets in the pool. DEX also allows users to passively provide liquidity, market any asset on Ethereum, and provide traders with always-available liquidity.
2) *Lending:* DeFi protocols allow users to lend assets. Users must provide collateral assets that exceed the amount they borrowed. Similar to the mortgage loan in traditional finance, users can post various assets as collateral and borrow other crypto assets including stablecoins with DeFi protocols. Moreover, once the value of the collateral falls below a specified loan-to-value ratio, the collateral will be liquidated to ensure that the loan can be repaid.
3) *Flash Loan:* Flash loan is a new type of noncollateral loan and many DeFi apps have provided this kind of protocol [66], [67]. All the operations with the flash loan have to be accomplished in one transaction or one block. It allows users to borrow a considerable amount of capital without collateralizing assets (but paying extra fees). The loan should be repaid within a certain amount of time (about 13 s in Ethereum); otherwise, the transaction is reverted to ensure the safety of the fund, i.e., undoing all operations performed previously. The operations between borrowing and repaying are called circuits, and users can flexibly combine various DeFi protocols. But it has been abused by hackers to launch attacks [61].

## D. Flash Loan Attacks and Price Manipulation

Generally, many DEXs require real-time information on the market price of assets to provide borrowing, lending, and redemption. To enable this functionality, DeFi protocols have introduced oracles, which report the prices of asset from real-world (off-chain) sources, such as on-chain DEXs (e.g., Uniswap, dYdX) and decentralized oracle networks (e.g., Chainlink).

In practice, flash loan attacks are often combined with price (or oracle) manipulation. An arbitrageur executes a strategy and makes a profit by lowering or increasing a specific asset. A simple successful flash loan attack involves the following steps, as shown in Fig. 3.

1) The attacker searches for DeFi protocols that support flash loans and borrows a large amount of token $X$ as flash loans.
2) The attacker uses token $X$ in exchange for token $Y$. Due to a large amount of exchange, the price of token $X$ decreases, and the price of token $Y$ increases.

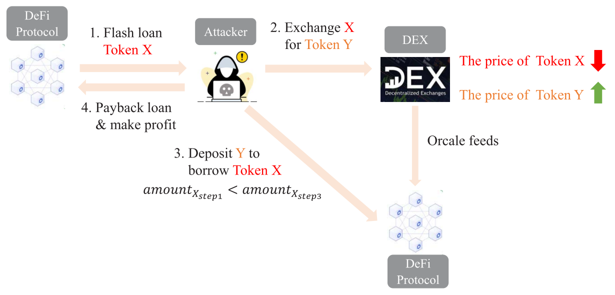

DeFi Protocol
1. Flash loan Token X
Attacker
2. Exchange X for Token Y
DEX
DEX Decentralized Exchanges
The price of Token X
The price of Token Y
4. Payback loan & make profit
Orcale feeds
3. Deposit Y to borrow Token X
$amount_{X_{step1}} < amount_{X_{step3}}$
DeFi Protocol

Fig. 3. Attacker manipulates the price of particular assets.

3) The attacker chooses another DeFi lending protocol as a target and deposits token $Y$ as collateral to lend more token $X$. Note that the criterion for protocol selection is that the DeFi protocol uses the above DEX as its only oracle data source.
4) Ultimately, the attacker uses the borrowed token $X$ to repay the previous flash loan. Due to the large amount of token $X$ lent, the attacker can profit from the remaining token $X$.

Besides the above case on bZx, in the past two years, flash loan attacks have compromised the multiple DEXs resulting in a loss of \$200 M.

## IV. Detection Models

The target of our work is to design an attack detection system for DeFi applications, called DeFiScanner. It can automatically analyze the unstructured emitted events and output "attack" or "normal." In the DeFiScanner, we design a neural network that can extract efficient multilevel features and achieve good performance without consuming a lot of human effort. We describe the design of DeFiScanner in this section.

As shown in Fig. 4, DeFiScanner has two phases: the training (i.e., learning) phase and the detection phase.

1) The learning phase has five steps, as shown in Fig. 4(a).

*Step 1:* Extract the global information $G$, emitted events $E$, related accounts, balances, and transferred tokens with transaction hashes. The emitted events of a raw transaction are shown in Fig. 5. The global information include block number, timestamp, transaction cost, and gas used.

*Step 2:* Extract the global features for each transaction with global information. The global features have a total of 37 dimensions, denoted by $G(g_1, g_2, \ldots, g_{37})$, most of which are statistical features, such as the number of transfer functions, withdraw functions, and the related tokens. We show the details in Table I.

*Step 3:* Generate the ground-truth labels of training programs. To better evaluate the DeFiScanner, we label the ground truth of raw transactions. Specifically, we label each raw transaction as "0" (i.e., normal transaction) or "1" (i.e., attack). The ground-truth labels of raw transactions are available with the Etherscan [68] and the public information from the official websites of DEXs.

*Step 4:* Transform emitted events into vectors to extract semantic features. This step has two substeps, which are highlighted below.

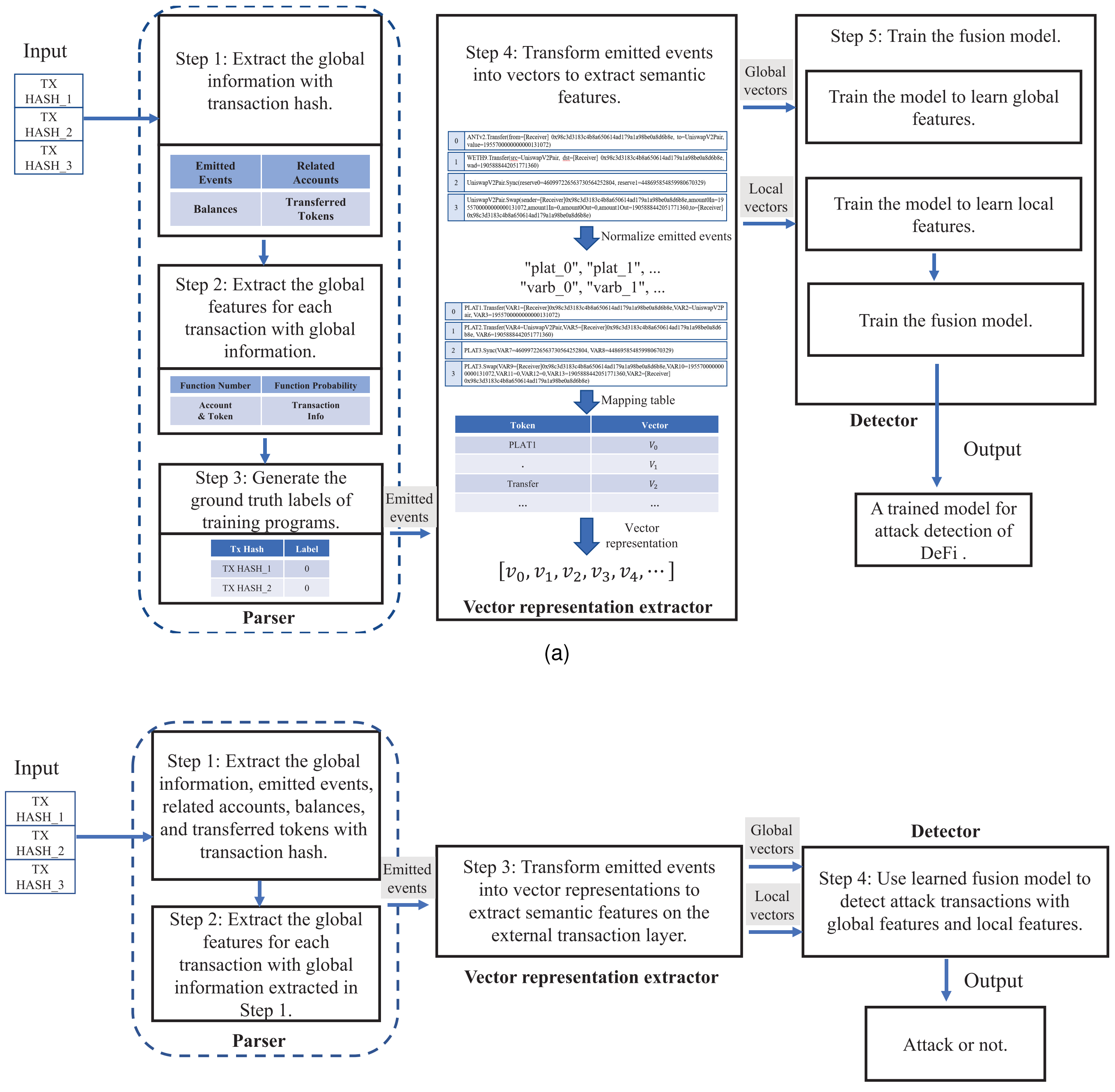
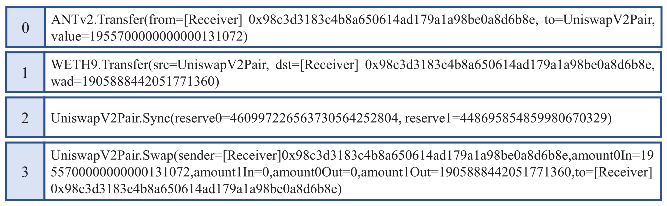

Input
TX HASH_1
TX HASH_2
TX HASH_3
Step 1: Extract the global information with transaction hash.
Emitted Events
Related Accounts
Balances
Transferred Tokens
Step 2: Extract the global features for each transaction with global information.
Function Number
Function Probability
Account & Token
Transaction Info
Step 3: Generate the ground truth labels of training programs.
Tx Hash
Label
TX HASH_1
0
TX HASH_2
0
Parser
Emitted events
Step 4: Transform emitted events into vectors to extract semantic features.
Normalize emitted events
"plat_0", "plat_1", ...
"varb_0", "varb_1", ...
Mapping table
Token
Vector
PLAT1
Transfer
Vector representation
$[v_0, v_1, v_2, v_3, v_4, \cdots]$
Vector representation extractor
Global vectors
Local vectors
Step 5: Train the fusion model.
Train the model to learn global features.
Train the model to learn local features.
Train the fusion model.
Detector
Output
A trained model for attack detection of DeFi .
(a)
Input
TX HASH_1
TX HASH_2
TX HASH_3
Step 1: Extract the global information, emitted events, related accounts, balances, and transferred tokens with transaction hash.
Step 2: Extract the global features for each transaction with global information extracted in Step 1.
Parser
Emitted events
Step 3: Transform emitted events into vector representations to extract semantic features on the external transaction layer.
Vector representation extractor
Global vectors
Local vectors
Detector
Step 4: Use learned fusion model to detect attack transactions with global features and local features.
Output
Attack or not.
(b)

Fig. 4. Overview of DeFiScanner: the training phase generates patterns of DeFi normal transactions, and the detection phase uses learned patterns to determine whether a target transaction is an attack or not. (a) Training phase. It can be divided into three parts, including a parser, a vector representation extractor, and a detector. (b) Detection phase.



| | |
|---|---|
| 0 | ANTv2.Transfer(from=[Receiver] 0x98c3d3183c4b8a650614ad179a1a98be0a8d6b8e, to=UniswapV2Pair, value=1955700000000000131072) |
| 1 | WETH9.Transfer(src=UniswapV2Pair, dst=[Receiver] 0x98c3d3183c4b8a650614ad179a1a98be0a8d6b8e, wad=1905888442051771360) |
| 2 | UniswapV2Pair.Sync(reserve0=460997226563730564252804, reserve1=448695854859980670329) |
| 3 | UniswapV2Pair.Swap(sender=[Receiver]0x98c3d3183c4b8a650614ad179a1a98be0a8d6b8e,amount0In=1955700000000000131072,amount1In=0,amount0Out=0,amount1Out=1905888442051771360,to=[Receiver] 0x98c3d3183c4b8a650614ad179a1a98be0a8d6b8e) |

Fig. 5. Emitted events of a raw transaction.

Transform emitted events into certain symbolic representations, denoted by $S_1 = \mathrm{Map}(S)$. The purpose of this function is to filter some noise of internal transactions, as follows:

$$\mathrm{Map}\left(\begin{pmatrix} \text{dYdX} \\ \text{Uniswap} \\ \vdots \end{pmatrix}, \begin{pmatrix} \text{from} \\ \text{to} \\ \vdots \end{pmatrix}\right) = \begin{pmatrix} \text{plat\_0} \\ \text{plat\_1} \\ \vdots \end{pmatrix}, \begin{pmatrix} \text{var\_0} \\ \text{var\_1} \\ \vdots \end{pmatrix}. \tag{1}$$

We observe that different transactions involve different DEX platforms, and the combinations of protocols are also different, (e.g., platform names, and variables), which may affect the effectiveness of extracted semantic features and

TABLE I
GLOBAL FEATURES OF ACCOUNT LEVEL AND EXTERNAL TRANSACTION LAYER

| Function Number | | Function Probability | | Account & Token | Transaction Info |
|---|---|---|---|---|---|
| transferNum | withdrawNum | transferPr | withdrawPr | addressNum | txCostETH |
| flashloanNum | collectfeeNum | flashloanPr | collectfeePr | tokenNum | txCostUSD |
| depositNum | mintNum | depositPr | mintPr | amountTokenMax | gasUsed |
| borrowNum | collectNum | borrowPr | collectPr | senderMaxOutdegree | gasUsedGwei |
| approvalNum | rewardNum | approvalPr | rewardPr | senderMaxIndegree | |
| swapNum | liquidityNum | swapPr | liquidityPr | | |
| unknownNum | sumNum | unknownPr | funEntropy | | |

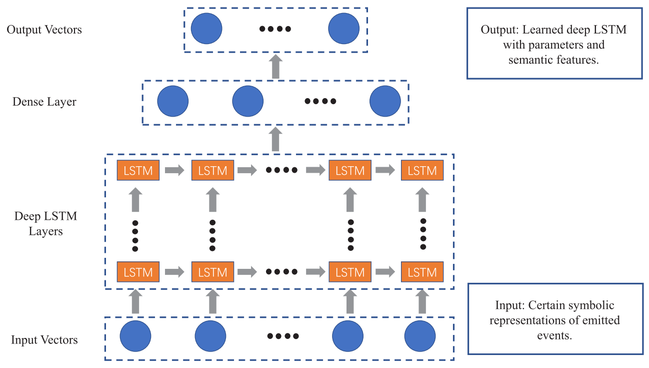
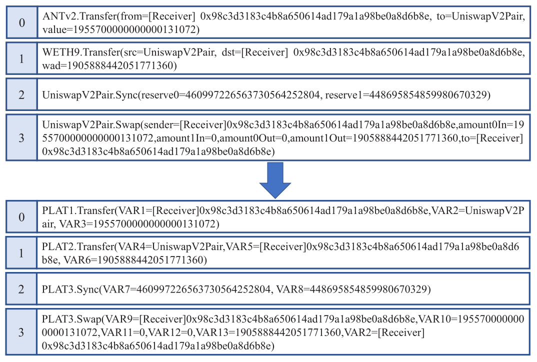

Output Vectors
Output: Learned deep LSTM with parameters and semantic features.
Dense Layer
Deep LSTM Layers
LSTM
Input Vectors
Input: Certain symbolic representations of emitted events.

Fig. 6. LSTM neural network.



```
0  ANTv2.Transfer(from=[Receiver] 0x98c3d3183c4b8a650614ad179a1a98be0a8d6b8e, to=UniswapV2Pair,
   value=195570000000000131072)
1  WETH9.Transfer(src=UniswapV2Pair, dst=[Receiver] 0x98c3d3183c4b8a650614ad179a1a98be0a8d6b8e,
   wad=1905888442051771360)
2  UniswapV2Pair.Sync(reserve0=460997226563730564252804, reserve1=448695854859980670329)
3  UniswapV2Pair.Swap(sender=[Receiver]0x98c3d3183c4b8a650614ad179a1a98be0a8d6b8e,amount0In=19
   5570000000000131072,amount1In=0,amount0Out=0,amount1Out=1905888442051771360,to=[Receiver]
   0x98c3d3183c4b8a650614ad179a1a98be0a8d6b8e)

0  PLAT1.Transfer(VAR1=[Receiver]0x98c3d3183c4b8a650614ad179a1a98be0a8d6b8e,VAR2=UniswapV2P
   air, VAR3=195570000000000131072)
1  PLAT2.Transfer(VAR4=UniswapV2Pair,VAR5=[Receiver]0x98c3d3183c4b8a650614ad179a1a98be0a8d6
   b8e, VAR6=1905888442051771360)
2  PLAT3.Sync(VAR7=460997226563730564252804, VAR8=448695854859980670329)
3  PLAT3.Swap(VAR9=[Receiver]0x98c3d3183c4b8a650614ad179a1a98be0a8d6b8e,VAR10=195570000000
   0000131072,VAR11=0,VAR12=0,VAR13=1905888442051771360,VAR2=[Receiver]
   0x98c3d3183c4b8a650614ad179a1a98be0a8d6b8e)
```

Fig. 7. Example of illustrating emitted event normalization.

reduce the performance of DeFiScanner. This motivates us to normalize emitted events by mapping the same variables and platform names to constants. We rename different platform names and variables according to the order of their appearance (i.e., "plat_0," "plat_1," ..., "varb_0," "varb_1," ... ). However, the functions and parameter values are not mapped because the function names are standard in the DeFi smart contracts, such as "Transfer," "Swap," and "FlashLoan." Then we encode symbolic emitted events and train a neural network, as shown in Figs. 6 and 7 gives an example of emitted events' normalization. As follows:

$$V = \text{word2vec}(S_1) = (v_1, v_2, \ldots, v_m). \tag{2}$$

The output $h_t^l$ of the each layer in LSTM at time $t$ can be denoted as

$$h_t^l = o_t^l * \tanh\left(C_t^l\right) \tag{3}$$

where $o_t^l$ is the output gate, and $C_t^l$ is the state of LSTM cell

$$o_t^l = \sigma\left(W_o \cdot \left[h_{t-1}^l \oplus V_t^l\right] + b_o\right) \tag{4}$$

and

$$\begin{aligned} C_t^l &= f_t^l * C_{t-1}^l + i_t^l * \tilde{C}_t^l \\ \tilde{C}_t^l &= \tanh\left(W_C \cdot \left[h_{t-1}^l \oplus V_t^l\right] + b_C\right) \\ i_t^l &= \sigma\left(W_i \cdot \left[h_{t-1}^l \oplus V_t^l\right] + b_i\right) \\ f_t^l &= \sigma\left(W_f \cdot \left[h_{t-1}^l \oplus V_t^l\right] + b_f\right) \end{aligned} \tag{5}$$

where $*$ denotes the elementwise multiplication, $\oplus$ denotes the concatenation of vectors, $b_i$ is the bias of input gate, $b_f$ is the bias of the forget gate, $b_C$ is the bias of the cell, $\tanh x = (\sinh x / \cosh x) = (e^x - e^{-x}/e^x + e^{-x})$ is the hyperbolic tangent function, and $\sigma(x) = (1/1 + e^{-x})$ denotes the sigmoid function. $W_C = [w_{\text{hc}}^l, w_{\text{vc}}^l]$, $W_i = [w_{\text{hi}}^l, w_{\text{vi}}^l]$, and $W_f = [w_{\text{hf}}^l, w_{\text{vf}}^l]$ are the weight matrices of neural network. $V_t^l$ denotes the input of each layer $l - 1 (l > 1)$ and also denotes the vectorized emitted events that inputted to the first layer $(l = 1)$. The output of the last layer is $L$. We divide the symbolic representation of the emitted events into a series of tokens with lexical analysis, including the platform names, the function names, the variables, and the values. But it results in a large corpus of tokens. With the idea of text mining [69], we consider using the word2vec to convert these tokens into vectors with a fixed length [70].

*Step 5:* Train the fusion model with global features and local features. Inspired by the multifeature fusion method in code vulnerability detection [71], we propose a new neural network architecture that uses the LSTM as a building block, as shown in Fig. 8. Because emitted events are time-dependent and LSTM is shown to be capable of extracting temporal features.

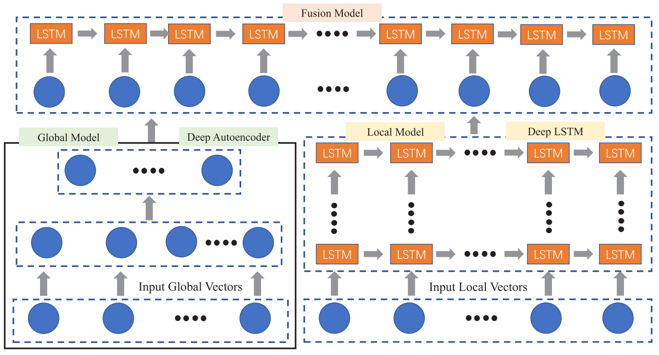


Fig. 8. Neural network architecture in DeFiScanner.

The network is able to extract two kinds of features: global features learned from Table I and local features learned from emitted events. Since global features and local features are extracted from different layers, we use a feature fusion layer to learn comprehensive features

$$f^f = M^g(G) \oplus M^l(L) \tag{6}$$

where $f^f$ denotes the fusion features, $M^g$ denotes the global model, and $M^l$ denotes the local model.

The network consists of a deep autoencoder network and two LSTM networks. The deep autoencoder network is regarded as learning the global features, and the two LSTM networks are regarded as the local-feature learning model and the feature-fusion model, as shown in Fig. 8. To learn the local features from emitted events, the model uses preprocessing layers and deep LSTM layers. The preprocessing layer mainly filters "0" to fix the length of vectors, and the deep LSTM layer is mainly used to extract local features. We build the fusion model with a fusion layer, deep LSTM layers, and a dense layer. The global features and local features are fused with the fusion layer. The deep LSTM layers are mainly used to extract features at the semantic level. To detect attack transactions, we use the reconstruction error of the fused features as follows:

$$L\left(f_i{}^f, M^f\left(f_i{}^f\right)\right) = \sum_{i=1}^{n} \left\| f_i{}^f - M^f\left(f_i{}^f\right)\right\|_2^2 \tag{7}$$

where $i$ denotes the number of samples used to train the fusion model.

2) As shown in Fig. 4(b), the detection phase contains the following steps.

*Step 1:* Extract the global information (similar to step 1 in the training phase).

*Step 2:* Extract the global features (similar to step 2 in the training phase).

*Step 3:* Transform emitted events into vectors (the same as Step 4 in the training phase).

*Step 4:* Use the learned fusion model to detect attack transactions with global features and local features.

## V. Experiment Setup

### A. Evaluation Dataset

We collect 50 910 real-world transactions on DeFi that have been verified between January 2021 and April 2022. For the attacks in the dataset, we limit our collection to the DeFi attacks appearing on Ethereum. We obtain detailed data through the public transaction hashes of attacks and construct ground-truth labels by manually inspecting the files. Especially, the testing dataset contains some realistic complex transactions that have hundreds of emitted events. Since normal transactions account for a large proportion of all the transactions in the real environment, the proportion of positive and negative samples in the dataset is unbalanced. With the idea of anomaly detection, we train the fusion model with normal transactions to learn the distribution of normal transactions. To fit the distribution of real-world data, we set the ratio of normal transactions to attacks in the testing dataset to 10:1.

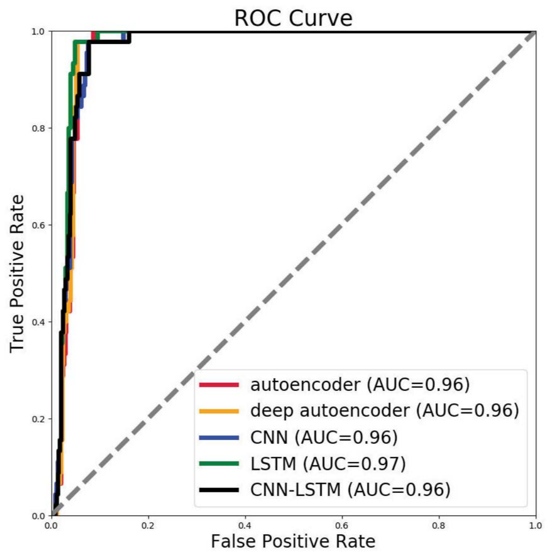


Fig. 9. ROC curve of the deep learning models.

In the dataset, the global information of external transactions and internal transactions are saved respectively. The global information of external transactions includes the gas used, the timestamp, and so on. For the global information of internal transactions, we collect the numbers of each function called, the amount of each token related, and the degree of each account. We also collect the corresponding emitted events of each transaction. Due to the transaction hashes being unique to each transaction, we only label the transaction hashes.

### B. Training Phase

The training phase is divided into the following four steps.

*Step 1:* To generate global features from training data, we input the features shown in Table I into the global model.

*Step 2:* We identify and rename the noise to normalize emitted events, such as the names of platforms. Then we use word2vec to generate the vectors of local features. Each vector represents an emitted event with 50 vector dimensions. Let $l_1$, $l_2$, and $l_3$, respectively, denote the length of global features, local features, and fusion features. In the training phase, we tune $l_1$, $l_2$ and $l_3$ for different performances. We set $l_1 = 37$, $l_2 = 50$, and $l_3 = l_1 + l_2$. Since the length of the input is fixed by the local model, for emitted events with fewer words and the length of vectors is less than 50, we pad 0 at the end to fix the length of vectors to $l_2$. We abandon the tail of vectors to match length $l_2$ of which the length is more than 50.

*Step 3:* We get the representation of global features with the global model and extract semantic features (called local features) of emitted events with the local model.

*Step 4:* We continuously fine-tune the parameters of the model training phase (such as optimizer, learning rate, batch size, dropout, number of layers, number of neurons, and epoch). We also use expert knowledge to guide an exhaustive search to quickly find the best values of these hyperparameters. We first make the global model and the local model perform best by adjusting the parameters. Second, we freeze the

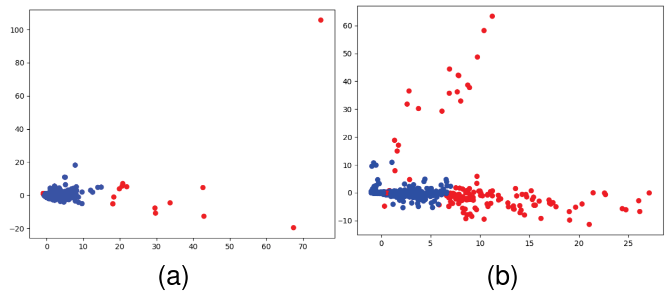


Fig. 10. Performance of $k$-means with global features and fusion features, respectively. (a) Global features. (b) Fusion features.

TABLE II
DETECTION PERFORMANCE COMPARISON OF SEVEN MODELS

| Model | FPR | TPR | precision | F1-socre |
|---|---|---|---|---|
| SVM | 98.63% | **99.45%** | 6.70% | 12.56% |
| K-means | 19.37% | 42.22% | 31.67% | 58.91% |
| Autoencoder | 9.11% | 97.78% | 51.76% | 67.69% |
| DeepAutoencoder | 8.89% | 97.78% | 52.38% | 68.22% |
| LSTM | **4.44%** | 91.11% | **67.21%** | **77.36%** |
| CNN | 6.44% | 93.33% | 59.15% | 72.41% |
| LSTM-CNN | 5.78% | 91.11% | 61.19% | 73.21% |

parameters of the global model and the local model to tune the hyperparameters of the fusion model. Finally, we get the attack detection model.

Unlike classification models that use a softmax activation function at the output layer and train the model with a categorical cross-entropy loss function, we use a sigmoid activation function at the output layer and train the fusion model with the reconstruction loss function.

### C. Detection Phase

We perform four steps to detect attacks. The first three steps are consistent with the corresponding steps in the training phase. In step 4, the reconstruction loss is used to distinguish whether a transaction is an attack with the trained model.

### D. Evaluation Metrics

We use the TPR or recall, false positive rate (FPR), precision (P), and F1-measure (F1) to evaluate DeFiScanner. The TPR or recall metric $\mathrm{TPR} = (\mathrm{TP}/\mathrm{TP} + \mathrm{FN})$ measures the ratio of true positive (TP) attacks to the entire population of samples that are attacks. The FPR metric $\mathrm{FPR} = (\mathrm{FP}/\mathrm{FP} + \mathrm{TN})$ measures the ratio of false positive (FP) attacks to the entire population of samples that are not attacks. The precision metric $\mathrm{P} = (\mathrm{TP}/\mathrm{TP} + \mathrm{FP})$ measures the correctness of the detected attacks. The F1-score metric $\mathrm{F1} = (2 \cdot \mathrm{P} \cdot \mathrm{TPR}/\mathrm{P} + \mathrm{TPR})$ takes consideration of both precision and TPR. FP is the number of normal transactions detected as attacks, false negative (FN) is the number of samples for which real attacks are not detected, TP is the number of samples for which real attacks are detected, and true negative (TN) is the number of samples for which normal transactions are not detected.

It would be ideal that an attack detection system neither misses attacks (i.e., $\mathrm{TPR} \approx 1$) nor triggers false alarms (i.e., $\mathrm{FPR} \approx 0$ and $\mathrm{P} \approx 1$).

## VI. EXPERIMENTS AND RESULTS

Our experiments focus on answering the following two questions (RQs).

*RQ1:* Is DeFiScanner an effective attack detection system?

The attack detection systems for logic vulnerabilities on DeFi should detect both normal transactions and attacks. To answer this question, we conduct experiments involving normal transactions and attacks.

*RQ2:* Can DeFiScanner extract contextual information?

If the fusion model plays an important role in the system? To answer this question, we investigate the effectiveness of using global features without the fusion model.

### A. Experiments for Answering RQ1

Table II summarizes the experimental results. We use seven models to detect attacks, namely, SVM, $k$-means, autoencoder, deep autoencoder, LSTM, CNN, and LSTM-CNN. These models are evaluated on the same testing dataset. $k$-means is a representative method of unsupervised learning. SVM is the representative method of supervised learning. For autoencoder, deep autoencoder, LSTM, CNN, and LSTM-CNN, these models are representatives of the self-supervised methods in deep learning. We observe that the FPR and TPR of $k$-means are 19.37%, and 42.22%, respectively. The precision and F1-score of $k$-means are 31.76%, and 58.91%, respectively. The FPR, TPR, precision, and F1-score of SVM are 98.63%, 99.45%, 6.70%, and 12.56%, respectively. Compared with SVM, $k$-means is 75% lower in terms of FPR and 55% lower in TPR. This can be explained that SVM cannot detect normal transactions and attacks at the same time, because both the FPR and TPR reach 90%, which means that regardless of whether the input is an attack or a normal transaction, the model judges it as an attack.

It can be seen that the self-supervised learning methods perform best in detecting both normal transactions and attack transactions simultaneously. However, autoencoder performs the worst among the self-supervised learning methods. The FPR, TPR, precision, and F1-score of autoencoder are 9.11%, 97.78%, 51.76%, and 67.69%, respectively. Compared with $k$-means, the autoencoder is 10.26% lower in terms of FPR, 55.56% higher in terms of TPR, 20.09% higher in terms of precision, and 8.78% higher in terms of F1-score. This may be due to the fact that the self-supervised learning model can learn the distribution of normal transactions, resulting in a smaller FPR and a larger TPR. Deep autoencoder, LSTM, CNN, and LSTM-CNN also perform well. Among the self-supervised learning methods, the LSTM model performs the best. The FPR of the LSTM is 4.67% lower than that of the autoencoder, which performs better on the TPR metric. But in practice, we will prioritize reducing the FPR. In addition, the accuracy and F1-score also show that the LSTM model is more effective than other self-supervised learning methods.

Due to the internal transactions containing specific logic and being executed sequentially, the emitted events have time dependencies among them. The emitted events of each external transaction can be viewed as time series data. It can be

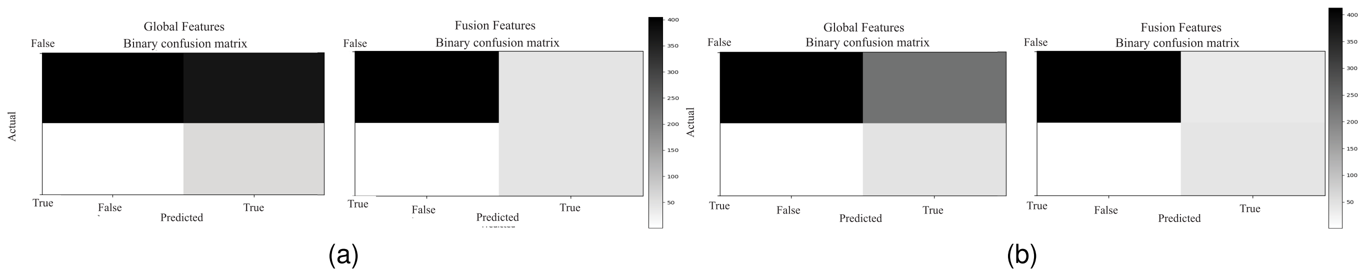


Fig. 11. Confusion matrix of (D)AE with global features and fusion features, respectively. (a) Autoencoder. (b) DeepAutoencoder.

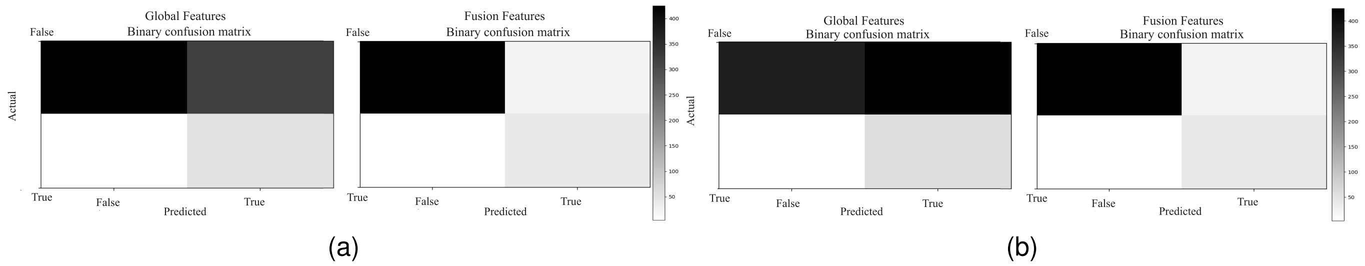


Fig. 12. Confusion matrix with global features and fusion features, respectively. (a) CNN. (b) LSTM.

explained why LSTM performs more effectively than other self-supervised learning models.

We further compare the ROC curves of all the models. The ROC curve can remain unchanged though the distribution of positive and negative samples in the testing set changes. So the ROC curve is an important metric when the samples in the dataset are imbalanced. The larger the area under the ROC curve, the better the detection effect of the model. Fig. 9 shows plots of the ROC curve of autoencoder, deep autoencoder, CNN, LSTM, and CNN-LSTM, respectively. The abscissa of the plane is the FPR and the ordinate is the TPR. We can observe that LSTM performs better than CNN and CNN-LSTM. The area under the ROC curve of LSTM reaches 0.97.

The ROC results also confirm that the emitted events are one kind of time series data. The performance is not good as using LSTM alone when we combine the CNN with LSTM in DeFiScanner.

Insight 1: The deep learning model that is trained with fusion features can perform well in attack transaction detection on DeFi.

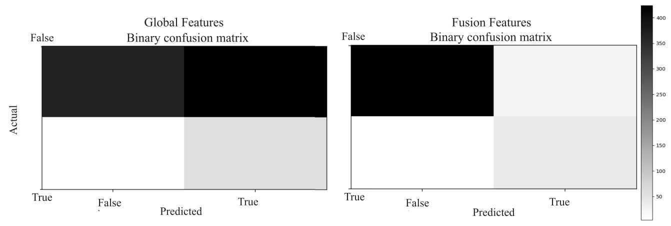


Fig. 13. Confusion matrix of CNN-LSTM with features and fusion features, respectively.

TABLE III
DETECTION PERFORMANCE COMPARISON OF SEVEN MODELS WITH GLOBAL FEATURES

| Model | FPR | TPR | precision | F1-socre |
|---|---|---|---|---|
| SVM | 98.63% | **<u>99.99%</u>** | 3.67% | 7.07% |
| K-means | **<u>19.60%</u>** | 26.67% | 2% | **<u>37.21%</u>** |
| Autoencoder | 40.89% | 84.44% | 17.11% | 28.46% |
| DeepAutoencoder | 37.78% | 84.44% | **<u>18.27%</u>** | 30.04% |
| LSTM | 56.89% | 86.67% | 13.22% | 22.94% |
| CNN | 63.33% | 88.89% | 12.31% | 21.62% |
| LSTM-CNN | 53.56% | 84.44% | 13.62% | 23.46% |

## B. Experiments for Answering RQ2

To answer whether DeFiScanner is able to extract contextual information of events effectively, and evaluate the importance of the fusion model in the system, we conduct experiments with the global features to detect attacks.

Table III summarizes the experimental results. These models do not perform as well as models trained by fusion features. We observe that the FPR of LSTM trained by global features is 56.89%. Compared with LSTM trained with fusion features, the LSTM trained with global features has 52.45% higher in terms of FPR, 53.99% lower in terms of precision, and 54.42% lower in terms of F1-score.

Without the local features, the performance of DeFiScanner is significantly lower. We can conclude that there is much useful information in emitted events, and the features extracted from emitted events are important to DeFiScanner to realize attack detection. The local model in DeFiScanner can extract the key features to improve the performance.

Fig. 10 shows plots of the detection result of $k$-means with global features and the detection result of $k$-means with fusion features. The red points represent attacks and the blue points represent normal transactions of DeFi. It can be seen that when we fuse global features and local features, the model can effectively improve the performance of separating the attacks from the normal transactions.

Insight 2: DeFiScanner can effectively extract the high-level semantic features of internal transactions. The model trained

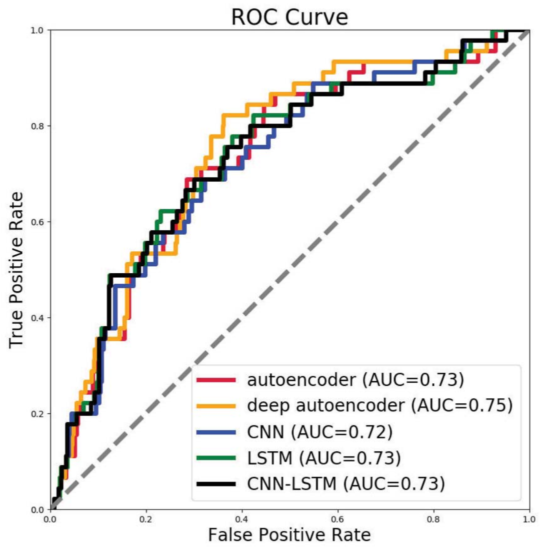


Fig. 14. ROC curve of the deep learning models without local features.

with global features cannot have a good performance in the task of DeFi attack transaction detection.

Finally, we compare the confusion matrices generated using global features and fusion features to detect attacks, as shown in Figs. 11–13. The confusion matrix is a specific matrix used to visualize the performance of an algorithm. Each column represents the predicted class, and each row represents the actual class. All the correct predictions are on the diagonal. The experimental results also validate the effectiveness of the DeFiScanner.

We also plot the ROC curves of these deep learning models trained by global features, respectively, as shown in Fig. 14. We can observe that the area under the ROC curve is 0.2 lower than the fusion models.

In a word, the local features and global features are all useful for detecting DeFi attacks, but the local features play a more important role in DeFiScanner than global features. The global features include block number, timestamp, transaction cost, the gas used, and the statistical features from the external transaction level. We often use these features to detect money laundering, frauding, and Ponzi schemes. Because the anomalies of these behaviors perform on the external transaction level and account level. Due to the profit of attacks reflected on the account level, and the complicated internal transactions often cause more transaction costs and the gas used, these global features can also assist DeFiScanner to detect DeFi attacks. The local features are extracted from the emitted events, and the detailed steps of attacks are the key to detecting DeFi attacks. The experimental results also demonstrate the discussion.

There exist limitations of DeFiScanner. First, DeFiScanner is limited to detecting attacks on DeFi based on the assumption of the availability for emitted events of raw transactions. Second, the present implementation of DeFiScanner only deals with DeFi attacks on the transaction layer. It cannot discover what codes cause the logic vulnerabilities or analyze the flow of funds stolen by attackers. We will focus on this in our future work.

## VII. Conclusion

In this work, we are motivated to detect attacks on DeFi applications such as flash loan attacks and price manipulation attacks. We present the first deep-learning-based attack detection system on DeFi, named DeFiScanner. It processes the unstructured emitted events generated by DeFi applications with word2vec and uses the fusion model to extract various high-level semantics. First, the unstructured emitted events generated by DeFi applications are automatically processed with word2vec. Second, a series of features from the account level and external transaction level are extracted to enrich high-level features. Finally, the features of different levels are fused efficiently with the designed fusion model. We use the representative methods of an unsupervised learning, namely, $k$-means, a supervised classification algorithm, namely, SVM, and five self-supervised methods in deep learning, namely, autoencoder, deep autoencoder, CNN, LSTM, and CNN-LSTM, to conduct comparative experiments. Comprehensive experimental results show that DeFiScanner can extract high-level DeFi semantics with the deep learning models and perform well in the detection of DeFi attacks.

## References

[1] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, Apr. 2017.

[2] J. Kamps and B. Kleinberg, “To the moon: Defining and detecting cryptocurrency pump-and-dumps,” *Crime Sci.*, vol. 7, no. 1, pp. 1–18, Dec. 2018.

[3] J. Xu and B. Livshits, “The anatomy of a cryptocurrency pump-and-dump scheme,” in *Proc. 28th USENIX Secur. Symp. (USENIX Secur.)*, 2019, pp. 1609–1625.

[4] P. Daian et al., “Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 910–927.

[5] S. Eskandari, S. Moosavi, and J. Clark, “SoK: Transparent dishonesty: Front-running attacks on blockchain,” in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2019, pp. 170–189.

[6] L. Zhou, K. Qin, C. F. Torres, D. V. Le, and A. Gervais, “High-frequency trading on decentralized on-chain exchanges,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 428–445.

[7] K. Qin, L. Zhou, B. Livshits, and A. Gervais, “Attacking the DeFi ecosystem with flash loans for fun and profit,” in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2021, pp. 3–32.

[8] *88mph Incident*. Accessed: Nov. 2021. [Online]. Available: https://peckshield.medium.com/88mph-incident-root-cause-analysis-ce477e00a74d

[9] *Akropolis Incident*. Accessed: Nov. 2021. [Online]. Available: https://peckshield.medium.com/akropolis-incident-root-cause-analysis-c11ee59e05d4

[10] *BZX Hack I*. Accessed: Nov. 2021. [Online]. Available: https://peckshield.medium.com/bzx-hack-full-disclosure-with-detailed-profit-analysise6b1fa9b18fc

[11] *Cheese Bank Incident*. Accessed: Nov. 2021. [Online]. Available: https://peckshield.medium.com/cheese-bank-incident-root-cause-analysis-d076bf87a1e72020

[12] *Opyn Incident*. Accessed: Nov. 2021. [Online]. Available: https://peckshield.medium.com/opyn-hacks-root-cause-analysis-c65f3fe249db

[13] L. Cheng et al., “SCTSC: A semicentralized traffic signal control mode with attribute-based blockchain in IoVs,” *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 6, pp. 1373–1385, Dec. 2019.

[14] C. F. Torres, M. Baden, R. Norvill, B. B. F. Pontiveros, H. Jonker, and S. Mauw, “ÆGIS: Shielding vulnerable smart contracts against attacks,” in *Proc. 15th ACM Asia Conf. Comput. Commun. Secur.*, Oct. 2020, pp. 584–597.

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

[16] L. Li et al., “CreditCoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2204–2220, Jan. 2018.

[17] X. Liu, J. Liu, S. Zhu, W. Wang, and X. Zhang, “Privacy risk analysis and mitigation of analytics libraries in the Android ecosystem,” *IEEE Trans. Mobile Comput.*, vol. 19, no. 5, pp. 1184–1199, May 2020.

[18] W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu, “BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors,” *Inf. Sci.*, vol. 511, no. 2, pp. 284–296, Feb. 2020.

[19] W. Wang, M. Zhao, and J. Wang, “Effective Android malware detection with a hybrid model based on deep autoencoder and convolutional neural network,” *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 8, pp. 3035–3043, 2019.

[20] W. Wang et al., “HGATE: Heterogeneous graph attention auto-encoders,” *IEEE Trans. Knowl. Data Eng.*, early access, Dec. 28, 2021, doi: 10.1109/TKDE.2021.3138788.

[21] Z. Wang, H. Jin, W. Dai, K.-K.-R. Choo, and D. Zou, “Ethereum smart contract security research: Survey and future research opportunities,” *Frontiers Comput. Sci.*, vol. 15, no. 2, pp. 1–18, Apr. 2021.

[22] C. Wang et al., “Demystifying Ethereum account diversity: Observations, models and analysis,” *Frontiers Comput. Sci.*, vol. 16, no. 4, pp. 1–12, Aug. 2022.

[23] Q. Zhenquan, J. Ye, J. Meng, B. Lu, and L. Wang, “Privacy-preserving blockchain-based federated learning for marine Internet of Things,” *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 1, pp. 159–173, Feb. 2021.

[24] X. Xu, Q. Liu, X. Zhang, J. Zhang, L. Qi, and W. Dou, “A blockchain-powered crowdsourcing method with privacy preservation in mobile environment,” *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 6, pp. 1407–1419, Dec. 2019.

[25] P. Li, J. Lai, and Y. Wu, “Accountable attribute-based authentication with fine-grained access control and its application to crowdsourcing,” *Frontiers Comput. Sci.*, vol. 17, no. 1, pp. 1–14, Feb. 2023.

[26] P. Zhang and M. Zhou, “Security and trust in blockchains: Architecture, key technologies, and open issues,” *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 3, pp. 790–801, Jun. 2020.

[27] G. Cheng, Y. Chen, S. Deng, H. Gao, and J. Yin, “A blockchain-based mutual authentication scheme for collaborative edge computing,” *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 1, pp. 146–158, Feb. 2022.

[28] Y. Zhang, X. Xu, A. Liu, Q. Lu, L. Xu, and F. Tao, “Blockchain-based trust mechanism for IoT-based smart manufacturing system,” *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 6, pp. 1386–1394, Dec. 2019.

[29] Y. Xie, G. Liu, C. Yan, C. Jiang, M. Zhou, and M. Li, “Learning transactional behavioral representations for credit card fraud detection,” *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 5, 2022, doi: 10.1109/TNNLS.2022.3208967.

[30] Z. Li, M. Huang, G. Liu, and C. Jiang, “A hybrid method with dynamic weighted entropy for handling the problem of class imbalance with overlap in credit card fraud detection,” *Expert Syst. Appl.*, vol. 175, Aug. 2021, Art. no. 114750.

[31] C. Yang, G. Liu, C. Yan, and C. Jiang, “A clustering-based flexible weighting method in AdaBoost and its application to transaction fraud detection,” *Sci. China Inf. Sci.*, vol. 64, no. 12, pp. 1–11, Dec. 2021.

[32] L. Zheng, G. Liu, C. Yan, C. Jiang, and M. Li, “Improved TrAdaBoost and its application to transaction fraud detection,” *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 5, pp. 1304–1316, Oct. 2020.

[33] C. Jiang, J. Song, G. Liu, L. Zheng, and W. Luan, “Credit card fraud detection: A novel approach using aggregation strategy and feedback mechanism,” *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3637–3647, Oct. 2018.

[34] Q. Bai, C. Zhang, N. Liu, X. Chen, Y. Xu, and X. Wang, “Evolution of transaction pattern in Ethereum: A temporal graph perspective,” *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 3, pp. 851–866, Jun. 2022.

[35] D. Lin, J. Chen, J. Wu, and Z. Zheng, “Evolution of Ethereum transaction relationships: Toward understanding global driving factors from microscopic patterns,” *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 2, pp. 559–570, Apr. 2022.

[36] S. Li, Y. Yuan, J. J. Zhang, B. Buchanan, E. Liu, and R. Ramadoss, “Guest editorial special issue on blockchain-based secure and trusted computing for IoT,” *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 6, pp. 1369–1372, Dec. 2019.

[37] Y. Zhang, W. Yu, Z. Li, S. Raza, and H. Cao, “Detecting Ethereum Ponzi schemes based on improved LightGBM algorithm,” *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 2, pp. 624–637, Apr. 2022.

[38] J. Bhatta, D. Shrestha, S. Nepal, S. Pandey, and S. Koirala, “Efficient estimation of Nepali word representations in vector space,” *J. Innov. Eng. Educ.*, vol. 3, no. 1, pp. 71–77, Mar. 2020.

[39] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1997.

[40] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases,” *AI Mag.*, vol. 17, no. 3, p. 37, 1999.

[41] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[42] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[43] K. Fukushima and S. Miyake, “Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition,” in *Competition and Cooperation in Neural Nets*. Cham, Switzerland: Springer, 1982, pp. 267–285.

[44] *Coinbase*. Accessed: Nov. 2021. [Online]. Available: https://www.coinbase.com/

[45] C. Busayatananphon and E. Boonchieng, “Financial technology DeFi protocol: A review,” in *Proc. Joint Int. Conf. Digit. Arts, Media Technol. ECTI Northern Sect. Conf. Electr., Electron., Comput. Telecommun. Eng. (ECTI DAMT NCON)*, Jan. 2022, pp. 267–272.

[46] Oxford Analytica, “Coinbase listing will pave way for other exchanges,” Emerald Expert Briefings, no. oxan-es, Dec. 2021.

[47] *dYdX Matching Specification*. Accessed: Nov. 2021. [Online]. Available: https://legacy-docs.dydx.exchange/

[48] *Serum White Paper*. Accessed: Nov. 2021. [Online]. Available: https://projectserum.com/

[49] W. Warren and A. Bandeali. *0x: An Open Protocol for Decentralized Exchange on the Ethereum Blockchain*. [Online]. Available: https://github.com/0xProject/whitepaper

[50] *Waves Exchange*. Accessed: Nov. 2021. [Online]. Available: https://docs.waves.exchange/en

[51] S. Ivanov and A. Pupyshev. *Neutrino Protocol White Paper*. Accessed: Nov. 2021. [Online]. Available: https://wp.neutrino.at/

[52] E. Chen and A. Chon. *Injective Protocol: A Collision Resistant Decentralized Exchange Protocol*. Accessed: Nov. 2021. [Online]. Available: https://coinpare.io/whitepaper/injective-protocol.pdf

[53] A. A. Aigner and G. Dhaliwal, “UNISWAP: Impermanent loss and risk profile of a liquidity provider,” 2021, *arXiv:2106.14404*.

[54] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson, “Uniswap v3 core,” Uniswap, New York, NY, USA, Tech. Rep., 2021.

[55] M. Egorov. (2019). *StableSwap-Efficient Mechanism for Stablecoin Liquidity*. [Online]. Available: https://medium.com/yield-protocol/introducing-ydai-43a727b96fc7

[56] *The Sushiswap Project*. Accessed: Nov. 2021. [Online]. Available: https://sushichef.medium.com/the-sushiswap-project-dd6eb80c6ba2

[57] D. Stone, “Trustless, privacy-preserving blockchain bridges,” 2021, *arXiv:2102.04660*.

[58] Raydium Team. *Raydium Protocol Litepaper*. Accessed: Nov. 2021. [Online]. Available: https://raydium.io/Raydium-Litepaper.pdf

[59] W. Li, J. Bu, X. Li, and X. Chen, “Security analysis of DeFi: Vulnerabilities, attacks and advances,” 2022, *arXiv:2205.09524*.

[60] D. Wang et al., “Towards a first step to understand flash loan and its applications in DeFi ecosystem,” in *Proc. 9th Int. Workshop Secur. Blockchain Cloud Comput.*, 2021, pp. 23–28.

[61] Y. Cao, C. Zou, and X. Cheng, “Flashot: A snapshot of flash loan attack on DeFi ecosystem,” 2021, *arXiv:2102.00626*.

[62] B. Wang et al., “BLOCKEYE: Hunting for DeFi attacks on blockchain,” in *Proc. IEEE/ACM 43rd Int. Conf. Softw. Eng., Companion Proc. (ICSE-Companion)*, May 2021, pp. 17–20.

[63] M. Team. (2020). *The Maker Protocol: Makerdao’s Mulfi-Collateral DAI (MCD) System*. [Online]. Available: https://makerdao.com/en/whitepaper/

[64] Centre. *Centre Whitepaper*. Accessed: Nov. 2021. [Online]. Available: https://www.centre.io/pdfs/centre-whitepaper.pdf

[65] Tether. (2016). *Fiat Currencies on the Bitcoin Blockchain*. Accessed: Nov. 2021. [Online]. Available: https://tether.to/wp-content/uploads/2016/06/TetherWhitePaper.pdf

[66] Aave. *Aave*. Accessed: Nov. 2021. [Online]. Available: https://aave.com/

[67] dYdX. *dYdX*. Accessed: Nov. 2021. [Online]. Available: https://dydx.exchange/

[68] Etherscan. *Etherscan*. Accessed: Nov. 2021. [Online]. Available: https://etherscan.io/

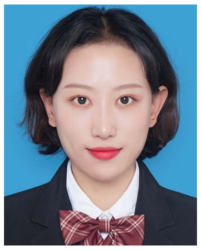
[69] L. Wolf, Y. Hanani, K. Bar, and N. Dershowitz, “Joint word2vec networks for bilingual semantic representations,” *Int. J. Comput. Linguistics Appl.*, vol. 5, no. 1, pp. 27–42, 2014.

[70] G. Hinton, J. McClelland, and D. Rumelhart, *Distributed Representations. Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge, MA, USA: MIT Press, 1986, ch. 3, pp. 77–109.

[71] Z. Li et al., “VulDeePecker: A deep learning-based system for vulnerability detection,” 2018, *arXiv:1801.01681*.

**Bin Wang** is currently pursuing the Ph.D. degree with Beijing Jiaotong University, Beijing, China.

His research interests include blockchain, data security, and machine learning.



**Xiaohan Yuan** is currently pursuing the Ph.D. degree with Beijing Jiaotong University, Beijing, China.

Her research interests include blockchain security and applications.

**Li Duan** received the Ph.D. degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2016.

She was a Research Fellow with Nanyang Technological University, Singapore, and the University of Science and Technology Beijing, Beijing. She is currently an Assistant Professor with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing. Her research interests are services computing and the Internet of Things, data security and privacy protection, and blockchain security and applications.

Dr. Duan received the National Natural Science Foundation of China, the Post-Doctoral Fund, and the Basic Scientific Research Project.

**Hongliang Ma** received the Ph.D. degree from Beijing Jiaotong University, Beijing, China, in 2018.

He is currently an Associate Professor with Shihezi University, Shihezi, China. His research interests mainly include artificial intelligence security, web security, and anomaly detection.

**Bin Wang** received the Ph.D. degree from the China National Digital Switching System Engineering and Technological Research and Development Center, Zhengzhou, China, in 2010.

He is currently a Professor of Zhejiang Key Laboratory of Multi-Dimensional Perception Technology, Application and Cybersecurity, Hangzhou, China, and Zhejiang University, Hangzhou. His research interests mainly include the Internet of Things security, cryptography, artificial intelligence security, and new network security architecture.

**Chunhua Su** received the B.S. degree from the Beijing Electronic and Science Institute, Beijing, China, in 2003, and the M.S. and Ph.D. degrees in computer science from the Faculty of Engineering, Kyushu University, Fukuoka, Japan, in 2006 and 2009, respectively.

He has worked as a Research Scientist with the Cryptography and Security Department, Institute for Infocomm Research, Singapore, from 2011 to 2013. From 2013 to 2016, he has worked as an Assistant Professor with the School of Information Science, Japan Advanced Institute of Science and Technology, Asahidai, Nomi-shi, Ishikawa. From 2016 to 2017, he worked as an Assistant Professor with the Graduate School of Engineering, Osaka University, Suita, Japan. He is currently working as a Senior Associate Professor with the Division of Computer Science, University of Aizu, Aizuwakamatsu, Japan. His research interests include cryptanalysis, cryptographic protocols, privacy-preserving technologies in data mining and the IoT security and privacy.

**Wei Wang** (Member, IEEE) received the Ph.D. degree from Xi'an Jiaotong University, Xi'an, China, in 2006.

He was a Post-Doctoral Researcher with the University of Trento, Trento, Italy, from 2005 to 2006. He was a Post-Doctoral Researcher with TELECOM Bretagne, Brest Cedex, France, and with INRIA, Paris, France, from 2007 to 2008. He was also a European ERCIM Fellow with the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, and with the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Esch-sur-Alzette, Luxembourg, from 2009 to 2011. He visited INRIA, ETH, Switzerland, Zurich, NTNU, Trondheim, South-Trondelag, CNR, New York, America, New York University Polytechnic, New York, and King Abdullah University of Science and Technology, Thuwal, Saudi Arabia. He is a Full Professor with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China. He is an Elsevier highly cited Chinese researchers. He has authored or coauthored over 100 peer-reviewed papers in various journals and international conferences. His main research interests include mobile, computer, and network security.

Dr. Wang is an Editorial Board Member of *Computers & Security* and a Young AE of *Frontiers of Computer Science*.