Financial Innovation

RESEARCH **Open Access**

Blockchain-oriented approach for detecting cyber-attack transactions

Zhiqi Feng, Yongli Li* and Xiaochen Ma

*Correspondence:
liyongli@hit.edu.cn

School of Economics and Management, Harbin Institute of Technology, Harbin 150001, China

Abstract

With the high-speed development of decentralized applications, account-based blockchain platforms have become a hotbed of various financial scams and hacks due to their anonymity and high financial value. Financial security has become a top priority with the sustainable development of blockchain-based platforms because of an increasing number of cyber attacks, which have resulted in a huge loss of crypto assets in recent years. Therefore, it is imperative to study the real-time detection of cyber attacks to facilitate effective supervision and regulation. To this end, this paper proposes the weighted and extended isolation forest algorithms and designs a novel framework for the real-time detection of cyber-attack transactions by thoroughly studying and summarizing real-world examples. Furthermore, this study develops a new detection approach for locating the compromised address of a cyber attack to resolve the data scarcity of hack addresses and reduce time consumption. Moreover, three experiments are carried out not only to apply on different types of cyber attacks but also to compare the proposed approach with the widely used existing methods. The results demonstrate the high efficiency and generality of the proposed approach. Finally, the lower time consumption and robustness of our method were validated through additional experiments. In conclusion, the proposed blockchain-oriented approach in this study can handle real-time detection of cyber attacks and has significant scope for applications.

Keywords: Blockchain, Cyber-attack detection, Extended isolation forest, Decentralized application, Financial security, Fintech

Introduction

As a new decentralized infrastructure and disruptive core technology, the public blockchain technology has piqued the interest of researchers,[1] and the number of academic studies on blockchain is growing rapidly (Xu et al. 2019). According to Fang et al. (2022), Ethereum has become the mainstream blockchain platform for

[1] In order to facilitate readers understanding the concepts of blockchain, "Appendix 1" lists and explains the common concepts such as account, transaction, block, cryptocurrency, flash loan and decentralized exchange (DEX).

public blockchains, accounting for most of the total market capitalization. Currently, Ethereum is the largest decentralized open-source blockchain system that provides Turing-complete programming language to develop smart contracts. In consequence, several decentralized applications (*dapps*) based on smart contracts, such as Uniswap,[2] Aave,[3] and PanacakeSwap,[4] have emerged, and they have been applied to many areas, especially finance, arts, and collectibles. However, with the proliferation of dapps on the public blockchain, the account-based blockchain platforms have become a breeding ground for various financial scams and hacks due to their anonymity and enormous financial values. Cyber attacks and illegal activities are increasing on the account-based blockchain platforms (e.g., Ethereum,[5] Binance Smart Chain[6] (BSC), and SOLANA[7]). Furthermore, according to the Rekt database,[8] over $1.9 billion have been lost in 161 attacks on the decentralized application in 2021, indicating that cyber attacks have been a critical issue for the public blockchain. As the anonymity of blockchain provides convenience for hackers, an increasing number of financial regulators are attempting to strengthen blockchain supervision in various countries (Sebastião and Godinho 2021). Facing this issue, this paper aims to propose a general and real-time approach to detect cyber attacks to facilitate effective supervision and regulation in the field of the public blockchain.

In particular, focusing on the top 30 cyber attacks (ranked by the funds lost) appearing in recent years, as shown in Table 1, the account-based blockchain platforms have been targeted by three types of cyber attacks: Smart contract exploits, Flash loan attacks, and Identity theft. We will briefly introduce the three types of cyber attacks (Table 1), because they are the targets that the proposed new approach must detect. *Smart contract exploit* is the most frequent cyber attack in recent years since the decentralized applications are run on an open-source smart contract, which are written in programming languages and solely controlled by its own code. Hence, hackers have the opportunity to review code and probe the networks to look for *code vulnerabilities* of smart contracts, such as the vulnerabilities of re-entrancy, integer overflow, and multisig (Aspris et al. 2021; Efanov and Roschin 2018; Harvey et al. 2021). Note that the vulnerabilities mainly exist in smart contracts on account-based blockchain; therefore, the compromised addresses of Smart contract exploits belong to smart contracts. A real-world example related to the Smart contract exploit is described in "Appendix 2". Regarding *Flash loan attack*; hackers exploit *economic vulnerabilities* in the interaction between the decentralized applications of flash loans and other smart contracts. This enables hackers to borrow, arbitrage, and liquidate assets in an extremely short period, resulting in illegal profits (Qin et al. 2021), the most common method being arbitrage trading, in which the price of a crypto asset is manipulated on one decentralized exchange and quickly resold on another. In general, the compromised address of arbitrage trading based on flash

[2] https://docs.uniswap.org/.

[3] https://aave.com/.

[4] https://pancakeswap.finance/.

[5] https://ethereum.org/en/.

[6] https://www.bnbchain.org/en.

[7] https://solanaminer.com/.

[8] https://defiyield.app/rekt-database.

Table 1 Top 30 Cyber-attacks on the account-based platforms in recent years

Project name	Platform	Type of cyber-attack	Funds lost	Date
Ronin	RONIN	Smart contract exploit	$615,500,000	Mar 29, 2022
Poly Network	ETH	Smart contract exploit	$602,189,570	Aug 10, 2021
Wormhole	SOLANA	Smart contract exploit	$326,000,000	Feb 02, 2022
Beanstalk	ETH	Flash loan attack	$181,000,000	Apr 18, 2022
Parity	ETH	Smart contract exploit	$155,000,000	Nov 08, 2017
Vulcan Forged	ETH	Identity theft	$140,000,000	Dec 12, 2021
Boy X Highspeed	BSC	Smart contract exploit	$139,895,140	Oct 30, 2021
Cream Finance	ETH	Flash loan attack	$130,000,000	Oct 27, 2021
BadgerDAO	ETH	Smart contract exploit	$120,285,547	Dec 02, 2021
Horizon by Harmony	ETH	Identity theft	$100,000,000	Jun 23, 2022
Compound Labs	ETH	Smart contract exploit	$89,000,000	Nov 26, 2020
Qubit Finance	BSC	Smart contract exploit	$80,000,000	Jan 27, 2022
Fei Rari	ETH	Smart contract exploit	$79,749,026	Jan 05, 2021
Venus	BSC	Smart contract exploit	$77,000,000	May 18, 2021
Compound Labs	ETH	Smart contract exploit	$71,101,556	Sep 30, 2021
EasyFi	ETH	Smart contract exploit	$59,000,000	Apr 19, 2021
Uranium Finance	BSC	Smart contract exploit	$57,200,000	Apr 28, 2021
TheDAO	ETH	Smart contract exploit	$50,000,000	Jun 17, 2016
Cashio	SOLANA	Smart contract exploit	$48,000,000	Mar 23, 2022
bZx	BSC	Identity theft	$47,600,000	Nov 05, 2021
PancakeBunny	BSC	Flash loan attack	$45,000,000	May 19, 2021
Alpha Finance Lab	ETH	Flash loan attack	$37,500,000	Deb 13, 2021
Vee Finance	AVAX	Smart contract exploit	$36,000,000	Sep 21, 2021
Parity	ETH	Smart contract exploit	$34,000,000	Jul 19, 2017
MonoX	ETH	Smart contract exploit	$31,400,000	Dec 01, 2021
Spartan Protocol	BSC	Flash loan attack	$30,500,000	May 02, 2021
Paid Network	ETH	Smart contract exploit	$27,418,034	Mar 05, 2021
Lendf.Me	ETH	Smart contract exploit	$25,236,849	Apr 19, 2020
xToken	ETH	Flash loan attack	$24,000,000	May 12, 2021
Harvest Finance	ETH	Flash loan attack	$24,000,000	Oct 26, 2020

Platform represents the name of blockchain

loans is always the address of decentralized applications, which also belongs to smart contract. “Appendix 2” also provides detailed information about the arbitrage trading. Aside from the two types mentioned, *Identity theft* is a common type of cyber attacks (Fang et al. 2022; Xu 2016). It refers to a scenario in which a hacker gains unauthorized access to an individual or organization’s private key through phishing attacks, malware, or social engineering tactics, allowing them to access the associated blockchain account and transfer funds to the hacker’s account. According to the transaction definition (see “Appendix 1”), all transactions need to be signed using corresponding private keys before the transactions are submitted to the account-based blockchain. Therefore, losing authority of private keys is equal to losing funds on blockchain. The externally owned account (EOA), explained in “Appendix 1”, is thus a compromised address of Identity theft, since the private key on blockchain controls only an EOA. Based on the introduction of different cyber attacks, the compromised addresses of Smart contract exploit and Flash loan attacks belong to smart contracts, whereas those of Identity theft belong to EOAs. Meanwhile, both Smart contract exploit and Flash loan attack refer to exploiting code and economic vulnerabilities, respectively. In fact, the various types of cyber attacks imply that the key clues that must be detected differ. Accordingly, the first motivation for our work is to propose a general approach framework capable of dealing with multiple types of cyber attacks rather than just one or two types, especially since we believe that new types will emerge in the near future.

Although supervised machine learning methods (*SML*) have succeeded in numerous fields, they also suffer from several challenges when dealing with this paper's problem of detecting cyber-attack transactions in blockchain. The *first difficulty* stems from *insufficient data*. The adopted SML is part of the postmortem analysis technology, which means that the existing public transaction information is required to carry out the identity inference of the illegal addresses, such as the behavior analysis of the addresses and account identification. However, a dynamic and ever-changing cyber attack is hard to keep up. Only the historical types rather than the latest ones of cyber attacks can be learned, because not all labels can be available immediately (Carcillo et al. 2018; Dal Pozzolo et al. 2014). The *second difficulty* stems from the *data's imbalance*. In fact, the annotated information of blockchain addresses published on third-party sites is relatively scarce, resulting in data imbalance. As a consequence of the data imbalance issue, SML will be biased toward the majority class, resulting in poor classification performance of minority classes, because only the majority class of cyber attacks can be fully learned (Thabtah et al. 2020). The *third challenge* lies in *time consumption*. Real-time detection of cyber attacks is crucial for victims and managers to take corresponding measures to prevent potential losses. According to Etherscan[8] and Bscscan,[9] the average block time on the BSC and Ethereum is approximately 2.5 and 12 s, respectively. However, SML consumes far more time than average block times (Chen and Guestrin 2016). Therefore, completing the analysis of the real-time transactions within a short period is one of the most immense challenges for detecting cyber-attack transactions. Facing these challenges that SML is hard to cope with, the second motivation for this paper is to develop a new real-time method (i.e., with very low time consumption) that does not require adequate and balanced data.

Compared with the aforementioned SML, unsupervised machine learning (*UML*) methods can compensate for the deficiency of the aforementioned SML to some extent. Many existing studies demonstrate this point: (i) UML has been applied for credit card fraud detection by dealing with fraudsters' ability to invent novel fraud behaviors and changes in customer behaviors (Carcillo et al. 2021); (ii) for telecommunications fraud detection by correcting the misclassification of behavior types and recognizing the dynamic appearance of new fraud types (Hilas and Mastorocostas 2008); and (iii) for bot recognition in a web store by identifying more camouflaged agents (Rovetta et al. 2020), among others. Accordingly, UML can handle many types of cyber attacks, whether it has known them before or not, because many examples have illustrated that UML can discover patterns and information that may seem strange or suspicious. Fortunately, as a typical UML, the isolation forest (Liu

[9] https://bscscan.com/.

et al. 2012) has been demonstrated to be an effective method for anomaly detection with low time consumption and high efficiency in several fields, such as biostatistics and semiconductor manufacturing (Liu et al. 2012; Puggini and McLoone 2018), where the detection of cyber attacks falls under the category of anomaly detection. Furthermore, recent years' work has improved the traditional isolation forest into an extended isolation forest (EIF) by adjusting the way branch cuts are made (Hariri et al. 2019). However, in our experiment, the EIF also performs poorly. The third motivation of this paper is to develop a classic isolation forest and EIF for achieving satisfactory results in detecting cyber attacks on the account-based blockchain.

Facing the three listed motivations, the main work is introduced as follows. *First*, we propose a novel method for extracting real-time account-based blockchain transactions from open-source websites, such as Etherscan and Bscscan, and identifying the addresses with the highest expenditure as the target addresses based on the accumulated expenditures of various crypto assets. Because most target addresses have a long usage history, the data deficiency of the hack addresses can be solved in this manner, responding to the first motivation. *Second*, the target addresses will be filtered by the funds expenditure threshold, and only a few data points will be fed into the next stage of the detection system, resulting in a reduction in time consumption, which responds to the second motivation. *Third*, we extract historical transaction data of the target addresses using open-source websites and use various data preprocessing methods to process the original data feature, allowing more useful information to be extracted. Then, in response to the third motivation, we propose an improved algorithm that assigns an anomaly score to the depth of the isolation tree based on the traditional EIF, dubbed weighted and extended isolation forest (*WEIF*).

To summarize, the main contributions are listed. *The first contribution* is that this work is one of the first to conduct an in-depth study into the real-time detection of cyber attacks on account-based blockchains. This work not only considers various types of cyber-attack transactions by developing real-world cyber-attack examples, but it also develops a new general UML-based framework with low data request and computation costs. *The second contribution* is to propose an effective strategy for identifying suspected compromised addresses and filtering them using a fund expenditure threshold, which will significantly reduce the number of analyzed targets. *The third contribution* is the designed dynamic modeling technology, which refers to the development of an evolving model for mining behavioral differences between suspected compromised targets. As a result, the designed dynamic model can detect real-time and constantly changing cyber-attack transactions. Last but not least, by adding weight to the depth of EIF, a new algorithm called WEIF is created. When the weight is introduced, the gap between the average depth of the normal transaction and the average depth of the cyber-attack transaction grows larger than in the famous EIF. In fact, the larger the gap, the easier it will be to distinguish cyber-attack transactions. The results of three types of cyber attacks show its high efficiency and generality.

The remainder of this paper is organized as follows to present our work and contributions logically. "Related work" section examines SML and UML related works, and the detailed information on the traditional isolation forest and its extension.

"Methodology" section presents the overall framework for detecting cyber attacks, as well as our proposed algorithm and its validation results based on simulation data. "Experimental evaluation" section shows the detailed information about the training dataset and the analysis of experimental results. Finally, "Conclusion and future work" section concludes and discusses future work.

Related work

In essence, detecting cyber-attack transactions can be considered as identifying rare transactions that deviate significantly from most of the transactions. Because blockchain's openness makes it easier for researchers to access transaction data, an increasing number of researchers are working on developing new technologies to detect various types of cyber attacks on the blockchain. The majority of related studies focused on the use of SML, with only a few studies focusing on UML. Therefore, we introduce related studies of SML and UML. As aforementioned, our proposed algorithm is developed based on the traditional isolation forest and its extension. Thus, the theories of the standard isolation forest and its extension are also elaborated in this section. These mentioned methods reviewed in this section will be compared with our proposed algorithm in "Experimental evaluation" section.

SML

SML is a type of machine learning in which an algorithm is trained on a labeled dataset to recognize patterns and make predictions. The labeled dataset for SML consists of input data and corresponding output labels. SML evaluates its accuracy using the loss function and learns from training data until the error is decreased sufficiently. As an increasing number of cyber attacks of blockchain are provided on open-source websites, such as Etherscan, Bscscan, and Rekt database, most researchers attempt to collect malicious addresses via these open-source websites and focus on the use of SML to detect malicious transactions by learning the transaction behaviors of malicious addresses.

According to Farrugia et al. (2020), the XGBoost classifier was used on a balanced dataset (4,699 accounts) to detect malicious accounts on the Ethereum blockchain. Despite its 96% accuracy, the execution time of this model is more than 62 s, which is much longer than the average block times of Ethereum and BSC. Accordingly, it indicates that the XGBoost classifier is incapable of detecting cyber-attack transactions on account-based blockchains in real time. According to Aziz et al. (2022), various SML methods, including random forest, XGBoost, and the light gradient boosting machine (LGBM), have recently been used to detect fraud transactions by learning the transaction behavior of labeled accounts, and all of these models achieve more than 93% on the F1 score. All three mentioned models belong to ensemble learning techniques, which is introduced in Table 2.

Furthermore, some researchers have applied graph convolutional network (*GCN*) techniques (Shen et al. 2021; Yu et al. 2021) to the identity inference of phishing scams and Ponzi scheme based on the balanced dataset, and these techniques have also achieved good performance with around 90% on F1 score. A brief description of GCN is also provided in Table 2.

Table 2 A brief introduction of SML methods

Type	Description	Related methods
Techniques of ensemble learning	These techniques combine the predictions of two or more base models built with a given algorithm in order to improve overall accuracy and robustness of model	Random Forest (*RF*). It fits a number of decision trees classifiers on various sub-samples of training data and combines all prediction of classifiers to improve accuracy of model and solve the over-fitting problem (Breiman 2001) *XGBoost*. It is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable, which implements machine learning methods under the framework of Gradient Boosting (Chen and Guestrin 2016) LightGBM (*LGBM*). It is a gradient boosting framework that used decision tree-based learning algorithms. It adopts a leaf-wise tree growth strategy and introduce novel techniques including gradient-based one-side sampling and exclusive feature bundling (Ke et al. 2017)
Techniques of graph neural network	The aim of these techniques is to learn an embedding model that contains information of its neighborhood, which can be used to tackle a variety of issues, such as node and graph classification	Graph Convolutional Networks (*GCN*). the main idea of the GCN is to learn hidden layer representations that encode both local graph structure and features of nodes, and then the hidden layer representations will be passed through a neural network for node classification or graph classification (Kipf and Welling 2016)

According to the existing SML-related studies, SML methods are effective for the identification of malicious addresses based on balanced datasets. However, the datasets for anomaly detection of cyber attacks on account-based blockchain are extremely imbalanced, which may result in poor performance and the generality of SML methods. In recent years, various types of UML have been used to deal with the imbalanced datasets, and the detailed information of UML is introduced as follows.

UML

As mentioned above, the methods of SML are much more resource intensive because of the need for labeled data, but the methods of UML discover the hidden patterns or the data cluster without the human intervention. Currently, UML is the primary technology for detecting anomalies in many fields, including finance, telecommunications, and network administration (Carcillo et al. 2021; Hilas and Mastorocostas 2008; Rovetta et al. 2020). However, UML methods for detecting cyber attacks on the blockchain have received little attention. Only kernel-based techniques are used to detect abnormal addresses in the historical transaction network (Patel et al. 2020), and the kernel-based technique achieves a F1 score of approximately 80%. This section introduces the techniques of distance-based, clustering-based, histogram-based, kernel-based, neural network-based, and ensemble-based to fully understand how UML techniques work and

Table 3 A brief introduction of UML methods

Type	Description	Related methods
Distance-based techniques	These techniques apply the distance of an observation to its *k*th nearest neighbor and compute an anomaly score based on the density of each observation	*KNN/AvgKNN*. It assigns an anomaly score to an observation based on the distance to the k^{th} nearest neighbor and the average distance to *k* nearest neighbors respectively, according to Angiulli and Pizzuti (2002). The higher the anomaly score, the more abnormal it is Local Outliers Factors (*LOF*). It is similar with KNN. It assigns an anomaly score to a given instance based on computing its local density relative to the density of its neighbors (Breunig et al. 2000)
Clustering-based techniques	It assumes that the normal observations belong to large or dense clusters, whereas the abnormal observations belong to sparse or small clusters. It operates on the output of clustering methods	Cluster-Based Local Outlier Factor (*CBLOF*). It classifies a given observation as an outlier if the size or the density of its cluster is below a threshold (He et al. 2003)
Histogram-based techniques	The first step is the construction of a histogram based on the value of feature, and the second step is to compute an anomaly score for a given observation according to the height of bin in which it falls	Histogram-based Outlier Score (*HBOS*). It assigns an anomaly score by building a histogram with a fixed or a dynamic bin width for each attribute, and computes the outlier score based on the height of bin where the data point locates (Goldstein and Dengel 2012)
Kernel-based techniques	These algorithms apply a linear classifier to solve a non-linear problem, which is conducted by transforming a linearly inseparable data to a linearly separable one	One-class SVM (*OCSVM*): It is an unsupervised algorithm that learns a decision function for novelty detection by learning a distinct boundary around all normal observations (Schölkopf et al. 2001). By this way, any observation that does not fall inside the learned boundary is detected as outliers
Neural network-based techniques	It assumes that normal observations fall in high probability regions of the model, while the abnormal data points lie in the low-probability regions	Variational Autoencoder (*VAE*). Its architecture consists of both an encoder and a decoder with the optimization goal of reducing the reconstruction error between the encoded–decoded data and the initial data (Kingma and Welling 2013) Deep Support Vector Data Description (*DeepSVDD*). It is inspired by one-class SVM. Ruff et al. (2018) introduced a novel approach of anomaly detection by training a neural network and minimizing the volume of a hypersphere that encloses the network representations of normal samples
Ensemble-based techniques	These techniques combine multiple estimators in an anomaly detection via reducing the variance of model accuracy and making the algorithm to be more robust	Feature Bagging (*FB*). It combines the results from multiple individual outlier detection models trained by a small subset of features that are randomly selected from the original feature set (Lazarevic and Kumar 2005) Isolation Forest (*IF*). This technique isolates anomalies fast instead of normal points, which builds trees by splitting the randomly chosen features based on the random value from the range of the features (Liu et al. 2012)

apply these techniques to the detection of cyber-attack transactions on account-based blockchains, as shown in Table 3.

Among the UML methods shown in Table 3, the existing experiments on public datasets have shown that the isolation forest algorithm outperforms the other common UML techniques in terms of efficiency and accuracy while consuming significantly less memory (Falcão et al. 2019; Liu et al. 2012). However, the random isolation forest cuts are always horizontal or vertical, resulting in bias and artifacts in the anomaly score map (Hariri et al. 2019). Hariri et al. (2019) proposed the EIF to mitigate bias by using a random slope and a random intercept for branch cuts. Therefore, we use the traditional isolation forest and EIF as the base of our proposed model in this paper, and the detailed information on the isolation forest and EIF is further introduced as follows.

Isolation forest and its extensions

Isolation forest

An isolation forest, like a random forest, is built with decision trees, which belongs to UML methods because there are no predefined labels. The central idea behind isolation forest is to isolate anomalies by constructing a series of isolation trees with random attributes (Liu et al. 2012). The isolation tree is constructed using the algorithm shown in Table 12 ("Appendix 3") by splitting the subsample observations over a split value of a randomly selected attribute. In this manner, observations with corresponding attribute values less than the split value go left, whereas others go right, and the process is repeated recursively until the tree is fully constructed. The split value is randomly selected between the selected attribute's minimum and maximum values. Although the isolation forest is a typical method of UML, its random cuts are always straight lines, making the random cuts to be either horizontal or vertical. Therefore, several extensions of the isolation forest have been developed in recent years (Hariri et al. 2019).

EIF

Among the isolation forest algorithms developed, the EIF performs better (Hariri et al. 2019), which eliminates the disadvantage of isolation forest by adjusting the way of branch cuts. In contrast to the isolation forest, the EIF determines the information of random slope and intercept for the branch cut on a multidimensional dataset. The methods for generating the random slope and intercept for the branch cut will be briefly introduced here. In terms of the random slope for the EIF, it is a normal vector denoted as $\boldsymbol{n}$ by drawing a random number for each coordinate of $\boldsymbol{n}$ from the standard normal distribution $N(0{,}1)$. As a result, the branch cut is a hyperplane for the high-dimensional dataset rather than a straight line. In terms of the intercept denoted as $\boldsymbol{p}$, it is chosen from the value range of the training data. For a given point $\boldsymbol{x}$, the branching criteria for the data splitting are shown as follows:

$$(\boldsymbol{x}-\boldsymbol{p})\cdot\boldsymbol{n}\leq 0, \tag{1}$$

if the condition is not satisfied, the data point $\boldsymbol{x}$ moves down to the right branch, otherwise it will be passed to the left branch. By this way, the value of intercept will be restricted to available data at each branch point when we construct trees with larger

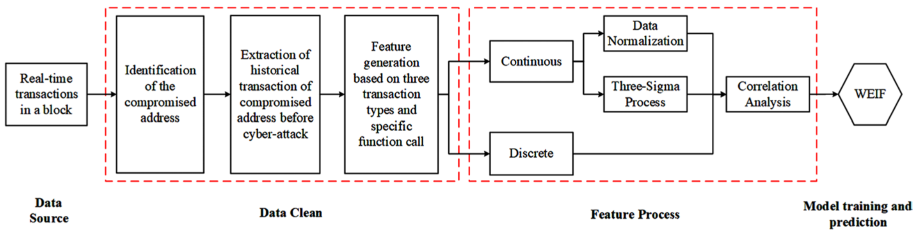

Real-time transactions in a block
Identification of the compromised address
Extraction of historical transaction of compromised address before cyber-attack
Feature generation based on three transaction types and specific function call
Continuous
Discrete
Data Normalization
Three-Sigma Process
Correlation Analysis
WEIF
Data Source
Data Clean
Feature Process
Model training and prediction

Fig. 1 The framework of anomaly detection proposed in this work

depths. The criteria for choosing intercept results in more possible branching options for areas of data concentration and less possible branching for areas of fewer observations.

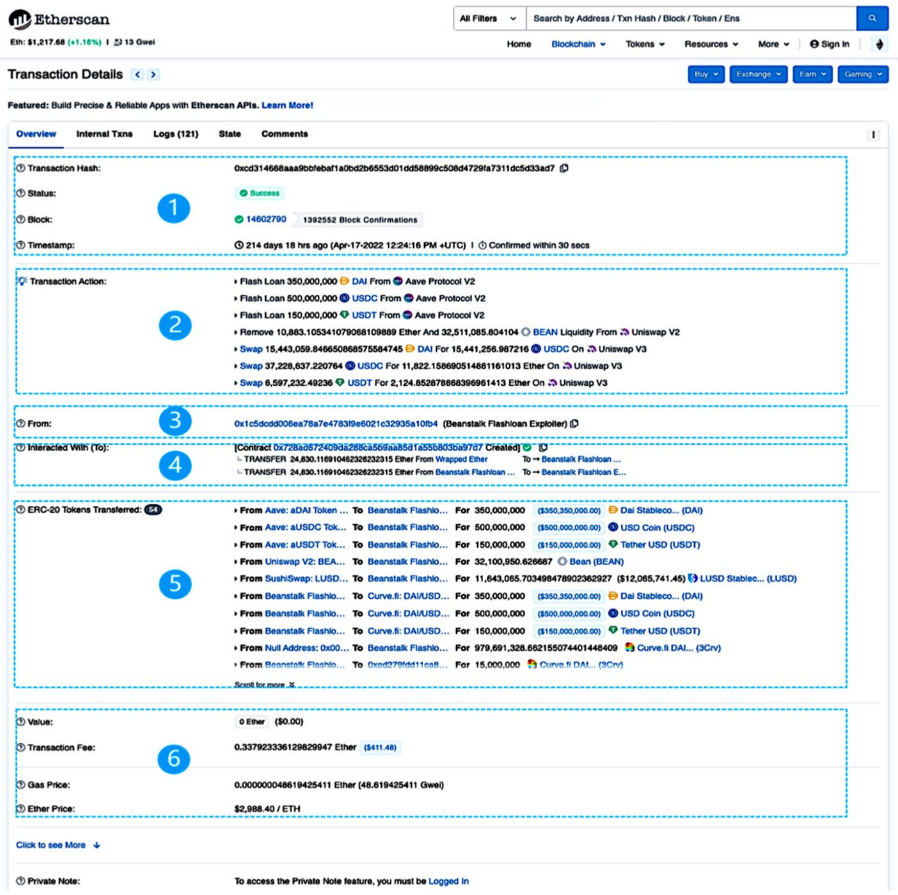
Except for the construction of isolation trees, there are few differences between the isolation forest algorithm and the architecture of EIF, which includes four procedures: isolation tree construction, depth computation, EIF construction, and anomaly score computation (Hariri et al. 2019). First, the isolation trees for the EIF are built using the Eq. (1) as described in Table 13 ("Appendix 3"). Secondly, how to compute the depth of an observation on a given extended isolation tree is elaborated in Table 14 ("Appendix 3"). Finally, Table 15 ("Appendix 3") shows the construction of an EIF and the computation of anomaly scores based on the isolation tree and depth computation.

Methodology

First of all, we propose a general framework for detecting cyber attacks in the context of the account-based blockchain. As shown in Fig. 1, this framework comprises four stages: the data source, data clean, feature process, model training and prediction. In the stage of *data source*, hundreds of real-time transactions in a block, referring to about 200 transactions every 12 s on Ethereum or about 150 transactions every 3 s on BSC, are extracted through open-source websites. For the stage of *data clean*, it is separated into the identification of compromised addresses, data extraction of historical transactions, and feature generation based on transaction behaviors, which will be executed in sequence. When it comes to the stage of the *feature process*, all the continuous features will be processed by data normalization and three sigma processes, and then all of the discrete features and the processed continuous features will be merged and fed into correlation analysis. During the final stage of *model training and prediction*, all the training data will be trained on our proposed new algorithm, WEIF, one of our main contributions in this work. The detailed information of each procedure in our proposed framework will be stated in the following subsections. Meanwhile, the evaluation metric is also introduced.

Data source

For the data source stage, all transactions are obtained from Etherscan and Bscscan, which are the leading platforms for Ethereum and BSC, respectively. A complex transaction on Etherscan is used as an example to demonstrate all of the detailed transaction information used in this paper (Fig. 2). The entire transaction details are divided into six parts, as shown in Fig. 2. In detail, the block number in Part 1 is the location where transactions are stored and encrypted, and it is generated every 2.5 s on



Transaction Details

Featured: Build Precise & Reliable Apps with Etherscan APIs. Learn More!

Transaction Hash:	0xcd314668aaa9bbfebaf1a0bd2b6553d01dd58899c508d4729fa7311dc5d33ad7
Status:	Success
Block:	14602790 1392552 Block Confirmations
Timestamp:	214 days 18 hrs ago (Apr-17-2022 12:24:16 PM +UTC) | Confirmed within 30 secs
Transaction Action:	Flash Loan 350,000,000 DAI From Aave Protocol V2 Flash Loan 500,000,000 USDC From Aave Protocol V2 Flash Loan 150,000,000 USDT From Aave Protocol V2 Remove 10,883.105341079068109889 Ether And 32,511,085.804104 BEAN Liquidity From Uniswap V2 Swap 15,443,059.846650868575584745 DAI For 15,441,256.987216 USDC On Uniswap V3 Swap 37,228,637.220764 USDC For 11,822.158690514861161013 Ether On Uniswap V3 Swap 6,597,232.49236 USDT For 2,124.852878868396961413 Ether On Uniswap V3
From:	0x1c5dcdd006ea78a7e4783f9e6021c32935a10fb4 (Beanstalk Flashloan Exploiter)
Interacted With (To):	[Contract 0x728ad672409da288ca5b9aa85d1a55b803ba97d7 Created] TRANSFER 24,830.116910462326232315 Ether From Wrapped Ether To → Beanstalk Flashloan ... TRANSFER 24,830.116910462326232315 Ether From Beanstalk Flashloan ... To → Beanstalk Flashloan E...
ERC-20 Tokens Transferred: 54	From Aave: aDAI Token ... To Beanstalk Flashlo... For 350,000,000 ($350,350,000.00) Dai Stableco... (DAI) From Aave: aUSDC Tok... To Beanstalk Flashlo... For 500,000,000 ($500,000,000.00) USD Coin (USDC) From Aave: aUSDT Tok... To Beanstalk Flashlo... For 150,000,000 ($150,000,000.00) Tether USD (USDT) From Uniswap V2: BEA... To Beanstalk Flashlo... For 32,100,950.626687 Bean (BEAN) From SushiSwap: LUSD... To Beanstalk Flashlo... For 11,643,065.703498478902362927 ($12,065,741.45) LUSD Stablec... (LUSD) From Beanstalk Flashlo... To Curve.fi: DAI/USD... For 350,000,000 ($350,350,000.00) Dai Stableco... (DAI) From Beanstalk Flashlo... To Curve.fi: DAI/USD... For 500,000,000 ($500,000,000.00) USD Coin (USDC) From Beanstalk Flashlo... To Curve.fi: DAI/USD... For 150,000,000 ($150,000,000.00) Tether USD (USDT) From Null Address: 0x00... To Beanstalk Flashlo... For 979,691,328.662155074401448409 Curve.fi DAI... (3Crv) From Beanstalk Flashlo... To 0xed279fdd11ca8... For 15,000,000 Curve.fi DAI... (3Crv) Scroll for more
Value:	0 Ether ($0.00)
Transaction Fee:	0.337923336129829947 Ether ($411.48)
Gas Price:	0.000000048619425411 Ether (48.619425411 Gwei)
Ether Price:	$2,988.40 / ETH
Private Note:	To access the Private Note feature, you must be Logged In

Fig. 2 The transaction details of account-based blockchains as an example

the BSC and every 12 s on Ethereum, respectively. This section contains all of the smart contract function calls for Part 2. Part 3 includes the transaction initiator, which refers to the EOA mentioned in "Appendix 1". For Part 4, all the transfers of the native crypto assets, referring to the ETH and BNB, are shown in this Part. For Part 5, all the token transfers, known as the nonnative crypto assets transfers, are contained. Apart from the parts mentioned above, the transaction value and transaction fee are included in Part 6, where the transaction fee is equal to the product of gas price and quantity of gas consumed in a transaction.

According to the definition of transaction on the blockchain, the transaction details can also be divided into four categories (i.e., external transaction, internal transfer, token transfer and transaction action). With regard to the external transaction, the transaction details of Part 1, Part 3, and Part 6 in Fig. 2 make up the basic information of external transactions. Regarding the internal transfer, the information in Part 4 contains all the internal transfer information. With regards to the token transfer, Part 5 represents the token transfer of the transaction. Apart from the three categories introduced above, Part 2 is named as the transaction action for transaction on the blockchain.

In particular, not all the transactions contain all six parts of the transaction details in Fig. 2. In fact, the more complicated transactions contains more parts of transaction details

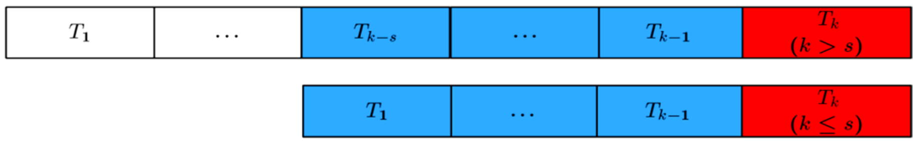


Fig. 3 Using dynamic window to construct the dataset from the historical transactions of compromised address. The cyber-attack transaction is marked as red, and the transactions marked with blue are recent transactions within the window size

on the account-based blockchain. For instance, the cyber-attack transactions of Identity theft are mainly composed of external transactions and token transfers, while cyber-attack transactions of Flash loan attack and Smart contract exploit usually involve all parts in Fig. 2.

Data clean

Three different procedures will be executed in sequence during the data clean stage, which are the identification of compromised addresses, data extraction, and feature generation.

Identification of compromised address

The compromised address is the target of hackers owing to a large quantity of crypto assets in the compromised address. Meanwhile, the majority of the compromised addresses exhibit long-term usage behavior. However, because the majority of hacker addresses are newly created for the cyber attacks, only a few transaction records are available for behavior analysis. As a result of more historical transactions than the hacker address, the behavior analysis of the compromised address for the cyber attack is easier to conduct.

As a result, a specific strategy is proposed in this paper to find the address with the highest spend volume as the suspected compromised address, despite the fact that the compromised addresses for real-time transactions are unknown. Based on the transaction details shown in Fig. 2, the suspected compromised addresses are located by computing the cumulative total expenditure of various crypto assets. The data deficiency of hacker addresses can thus be addressed to some extent.

Furthermore, over one million transactions are generated every day on the blockchain according to the statistics of Etherscan. In consequence, analyzing all the transactions in real-time is not feasible. To make our proposed framework to be suitable for the real-time analysis, we propose a novel approach by filtering the transactions according to the expenditures of the suspected compromised addresses. Obviously, with this transaction filtering approach, not too many transactions will be fed into the following analysis of the proposed framework. Thus, the time cost of the proposed framework can be significantly reduced in this way.

Data extraction

In fact, the real-time detection of cyber attacks only depends on the behavior of recent transactions due to the ever-changing behavior of transactions. Meanwhile, the suspected compromised address's numerous transactions make data extraction time consuming. Consequently, as shown in Fig. 3, we set a limit of transaction records, referring to the dynamic window, to construct the dataset of machine learning methods. For the compromised address of cyber-attack transaction, the window size is set as s, and T_k represents the cyber-attack transaction marked as red, where k is the transaction number of compromised address. Then, if the number of historical transactions before the cyber-attack transaction is larger than s, the transactions $[T_{k-s}, T_{k-s+1}, \ldots, T_k]$, marked as blue, will be extracted. Otherwise, the window size is equal to the number of historical transactions before the cyber attack and the transactions $[T_1, T_2, \ldots, T_k]$ will be extracted. This way, the time cost can be further reduced by the data extraction method. In particular, the different window sizes will be chosen to verify the stability and efficiency of our proposed algorithm.

Feature generation

According to the introduction of the transaction details, we establish a general feature set for all types of cyber attacks based on the transaction details of external transactions, internal transfer, token transfer, and transaction action. Therefore, all the features are separated into four categories (Table 4). Regarding the external transaction features, seven features are generated from the basic information of the external transactions, in which an EOA sends native crypto assets directly to another EOA or smart contract (see "Appendix 1"). Regarding the internal transfer features, these features, referring to transfer volume and transfer count, are computed from the internal transfer carried out through a smart contract as an intermediary. The token transfer features are extracted from the transactions with token transfers that refer to the transfers of ERC-20[10] or BEP-20[11] tokens in the transaction details. Apart from the three categories of features mentioned above, the transaction action features are generated based on the specific function call of the smart contract, as shown in the transaction action in Fig. 2.

Feature process

We process feature data using three methods (data normalization, three sigma process, and correlation matrix) to extract more useful information and speed up model training for anomaly detection.

Data normalization

Variables measured at different scales do not contribute equally to model fitting and may result in bias. To address this potential problem, Standard Scaler, also known as z score, is used for data normalization to speed up the model training and improve the model performance (Ioffe and Szegedy 2015). All feature values are rescaled to the new distribution so that the mean of observed values is 0 and the standard deviation is 1. The specification is expressed as

[10] The ERC-20 introduces a standard for Fungible Tokens on Ethereum, in other words, they have a property that makes each token be exactly the same (in type and value) as another token.

[11] The BEP-20 token standard serves pretty much the same function as the ERC-20 token standard, but it applies to tokens built on the Binance Smart Chain (BSC).

Table 4 Brief feature descriptions

Category	Feature name	Feature explanation
External transaction features	gas-fee	Transaction fee
	ex-tx-volume	Total volume of the native crypto asset within the external transaction
	action-type	Action type of the compromised address in the transaction
	spend-volume	Total volume of the native crypto asset spent by compromised address
	receive-volume	Total volume of the native crypto asset received by compromised address
	spend-anomaly	Anomaly index of the native crypto asset (spend volume)
	receive-anomaly	Anomaly index of the native crypto asset (receive volume)
Internal transfer features	inter-tx-volume	Total volume of the native crypto asset for the internal transfers
	inter-ts-volume-50%	Median volume of the native crypto asset for the internal transfers
	inter-ts-volume-max	The max volume of the native crypto asset for the internal transfers
	inter-ts-count	Number of the internal transfers within a transaction
	inter-ts-volume-min	Min volume of the native crypto asset for the internal transfers
	inter-spend-volume	Total volume of the native crypto asset spent by compromised address through internal transfer
	inter-spend-count	Number of the spend transfer operated by compromised address through the internal transfer
	inter-receive-volume	Total volume of the native crypto asset received by the compromised address through the internal transfer
	inter-receive-count	Number of the receive transfer operated by the compromised address through the internal transfer
Token transfer features	to-tx-degree	Degree of the token transfer within the transaction
	to-ts-count	Number of the token transfer within the transaction
	to-ts-count-anomaly	Anomaly index (the count of token transfer within the transaction)
	to-tx-outdegree	Out degree of the token transfer within the transaction
	spend-to-ts-count	Number of the spend transfer operated by compromised address through the token transfer
	to-tx-indegree	In degree of the token transfer within the transaction
	receive-to-ts-count	Number of the receive transfer operated by compromised address through the token transfer
	to-spend-anomaly	Anomaly index of the spend (spend volume of token transfer)
	to-receive-anomaly	Anomaly index of the receive (receive volume of token transfer)
Transaction action features	flash-currency-count	Number of the tokens related to flash loan within a transaction
	flash-action-count	Number of the flash loan action within the transaction
	swap-currency-count	Number of the tokens related to swap within a transaction
	swap-flash-count	Number of the flash loan action by means of swap
	swap-action-count	Number of the swap action within the transaction

$$z = \frac{x - \mu}{\sigma}, \tag{2}$$

where μ is the mean and σ is the standard deviation of the original data.

Three sigma process

In fact, the probability of occurrence decreases as the value deviates from the mean. Consequently, we apply the probabilistic rules of normal distribution to process transaction features. According to the normal distribution, the standard deviation, that is, sigma (σ), defines how far the normal distribution is spread around the mean. For an approximately normal distributed dataset, it follows a set of probabilistic rules described as follows: 68% of all values fall in [mean $-\ \sigma$, mean $+\ \sigma$], 95% of all values fall in [mean $-\ 2\sigma$, mean $+\ 2\sigma$], and 99.7% of all values fall in [mean $-\ 3\sigma$, mean $+\ 3\sigma$]. According to the rules, there are only 0.3% values falling outside three times the sigma range (3σ), and thus we can judge these values that fall outside [mean $-\ 3\sigma$, mean $+\ 3\sigma$] to be anomalous.

Correlation analysis

In fact, some features, while highly relevant to the specific type of cyber attacks, may be redundant. Meanwhile, if two independent features are highly correlated, they are considered redundant. Therefore, although eliminating redundant variables may not result in a significant loss of accuracy, it does result in a very efficient model under many constraints. In our proposed framework, the correlation analysis is used for feature selection by removing redundant features. The correlation coefficient of correlation analysis, denoted r, ranges from -1 to $+1$ and quantifies the direction and strength of the linear association between two features. Furthermore, the correlation coefficient is denoted as

$$r = \frac{\sum_{i=1}^{n} (p_i - \overline{p})(q_i - \overline{q})}{\sqrt{\sum_{i=1}^{n} (p_i - \overline{p})^2}\sqrt{\sum_{i=1}^{n} (q_i - \overline{q})^2}} \tag{3}$$

where n is the size of feature data, p_i and q_i are the individual features index with i, $\overline{p}$ and $\overline{q}$ are the mean value of two individual features.

WEIF

Proposed algorithm

According to the definition of EIF, the random slope and intercept for branch cuts should be determined before each branch cut during EIF construction, with a lower average depth indicating a more abnormal observation. The normal observation on a few trees may be close to the root due to the random selection of slope and intercept, whereas the abnormal observation on a few trees may be far away from the root based on EIF. As a result, an observation's anomaly score, calculated based on the average depth of the extended isolation trees, may deviate from its true depth range on the isolation tree, resulting in a bias. To mitigate the bias, we propose a novel algorithm, named as *WEIF*, by weighting the original depths of given observations in EIF, where the anomaly score

Table 5 Algorithm of weighted depth

```
Algorithm. WeightedDepth ($depth, \alpha$)
Input: The depths of all trees in iForest - D, the multiplier - $\alpha$
Output: an anomaly score of observation
1. Set Weighted_Depth = []
2. Compute the mean and quartile statistics of D
3. if $median > mean$ then
4.    for x in D do
5.       if $x \leq 1st\ quartile$ then
6.          $x = x \cdot (1 + \alpha \cdot abs(3rd\ quartile - mean)/mean)$
7.          Weighted_Depth.append(x)
8.       else if $x > 1st\ quartile$ then
9.          Weighted_Depth.append(x)
10.      end if
11.   end for
12.   return Weighted_Depth
13. else if $median < mean$ then
14.   for x in D do
15.      if $x \geq 3rd\ quartile$ then
16.         $x = x \cdot (1 - \alpha \cdot abs(1st\ quartile - mean)/mean)$
17.         Weighted_Depth.append(x)
18.      else if $x < 3rd\ quartile$ then
19.         Weighted_Depth.append(x)
20.      end if
21.   end for
22.   return Weighted_Depth
23. else if $median = mean$ then
24.   for x in D do
25.      Weighted_Depth.append(x)
26.   end for
27.   return Weighted_Depth
28. end if
```

of given observation is computed based on the average of weighted depths processed by the algorithm in Table 5.

In general, when the random trees of a forest produce shorter path lengths for some specific points, they are highly likely to be anomalies. Depths less than the first quartile of the original depths will be increased if the median of the original depths is greater than its mean, according to our proposed algorithm in Table 5. Depths greater than the third quartile of the original depths, on the other hand, will be decreased if the median of the original depths is less than its mean. Furthermore, if the median of the original depths is equal to its mean, the depths will not change. By this way, the depth difference between the normal observation and abnormal observations is becoming larger. The complexity of our proposed algorithm for training and prediction are $O\left(t\psi log(\psi)\right)$ and $O\left(ntlog(\psi)\right)$, respectively, where t is the number of trees, ψ is the subsample size of data and n is the number of observations in the dataset.

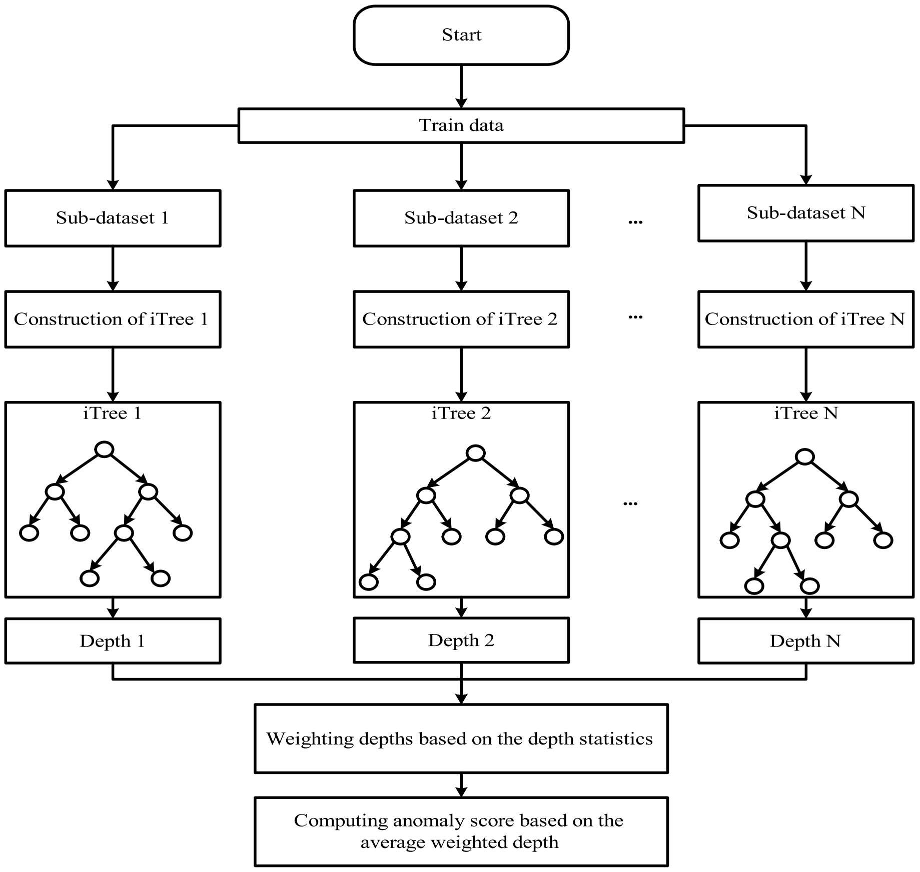


Fig. 4 Architecture of weighted and extended isolation forest. Firstly, several sub-datasets are generated by randomly sampling from the training dataset. Secondly, all the sub-datasets are passed into the construction of the isolation trees, and the original depths of the observations in the sub-datasets are computed for each isolation tree. Thirdly, all the original depths are weighted based on the algorithm in Table 5. Finally, the anomaly score is calculated on the average depth of the weighted depths

According to EIF, the architecture of WEIF contains four steps (Fig. 4): generating sub-datasets, constructing isolation trees, weighting the depths, and computing anomaly score. Compared with the architecture of the traditional isolation forest and its extension, there are few changes except for weighting the depths based on the algorithm shown in Table 5. Specifically, the definition of the anomaly score for an observation y is described as

$$S(y,n) = 2^{-\frac{E(h(y))}{c(n)}}, \tag{4}$$

where $E(h(y))$ is the mean value of weighted depths for a given observation in all trees, $c(n)$ is used to normalize the average path length $E(h(y))$ that is defined as the average path length of unsuccessful search in Binary Search Tree (Liu et al. 2012), i.e.,

$$c(n) = 2H(n-1) - \left(\frac{2(n-1)}{n}\right), \tag{5}$$

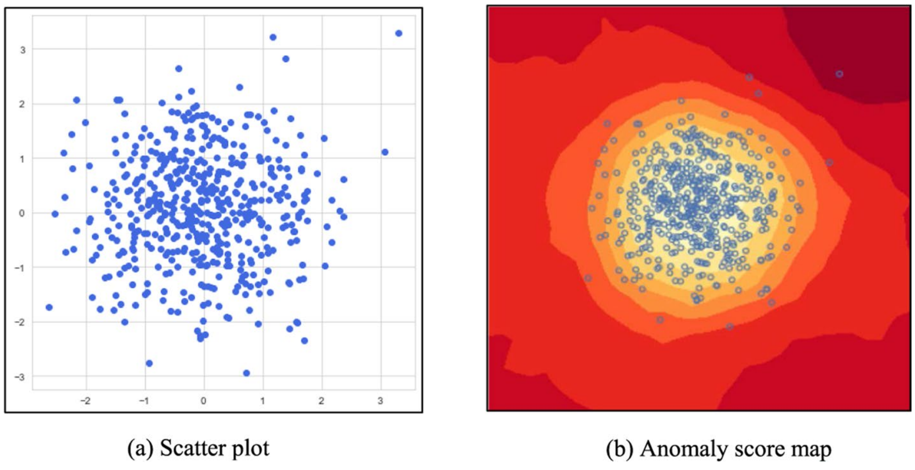


Fig. 5 Scatter plot and anomaly score map in this example. **a** The data of scatter plot are sampling from two-dimensional distribution with zero mean vector and identity covariance matrix. **b** The anomaly score map is plotted based on the weight depths processed by WEIF. A darker color means to be more anomalous

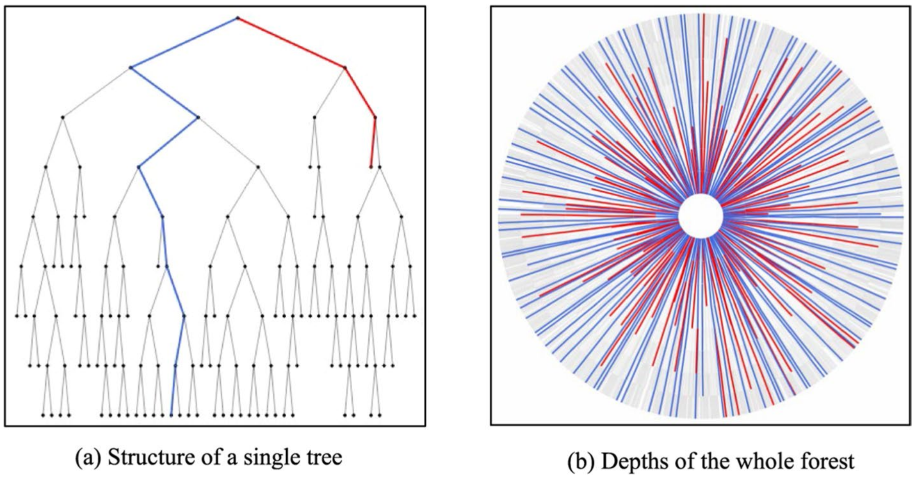

(a) Structure of a single tree
(b) Depths of the whole forest

Fig. 6 Structure of a single Tree and depths of WEIF. The results of normal observation are marked with blue, while the results of abnormal observation are marked with red. **a** The paths for a normal observation and an abnormal observation are plotted in a single isolation tree. **b** The depths of an observation in the whole forest are displayed as the radial line and the length of line represents the value of depth

where $H(i)$ can be calculated by $ln(i) + 0.5772156649$ (Euler's constant) and n is the number of observations in a given dataset (Liu et al. 2012).

Recalling Eq. (4), a smaller $E(h(y))$ means a higher anomaly score. By this way, all of the observations will be passed into the isolation trees and assigned an anomaly score. And the observation with a higher anomaly score is more anomalous based on Eq. (4). Specially, the threshold of the anomaly score shown in Fig. 4 is decided by the expected proportion of anomalies in the whole dataset, named as contamination in this paper.

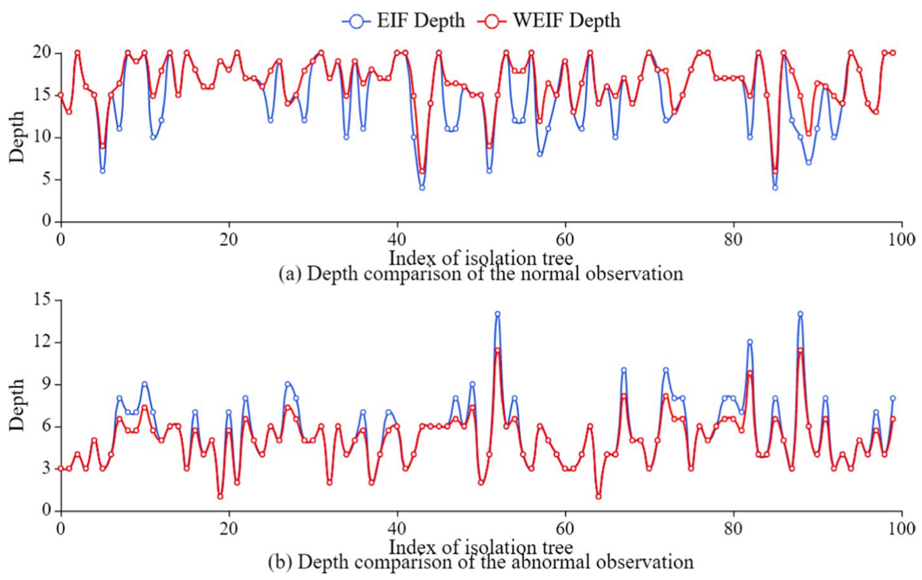


Fig. 7 Comparison of the depths processed by WEIF and EIF. The red and blue lines represent the depths computed by WEIF and EIF, respectively

Illustrative examples

To understand how WEIF works, we provide two illustrative examples of a two-dimensional dataset sampling from two-dimensional distribution with zero mean vector and an identity covariance matrix. The first example focuses on the comparison of normal and abnormal observations processed by our proposed algorithm, while the second example demonstrates the differences in the outputs of our proposed algorithm (WEIF) and EIF.

For the first example, the depth comparison of the normal and abnormal observations are presented with different forms. First, all the observations of the two-dimensional dataset are plotted in the scatter plot in Fig. 5a, where few samples exist further away from the center of the two-dimensional dataset. Second, a contour plot in Fig. 5b is achieved based on the average depths of all the observations computed by our proposed algorithm. Here, the color of the observation far away from the data center is darker than the observation close to the data center, indicating that the abnormal observation can be effectively identified by our proposed algorithm. Furthermore, the results of comparing the depths of the normal and abnormal observations with the depths computed by our proposed algorithm are shown in Fig. 6. As shown in Fig. 6a, the abnormal observation is quickly isolated, whereas the normal observation continues all the way to the bottom of the tree. All of the depths of the observations on the isolation trees are shown as straight lines in Fig. 6b, and it is obvious that the majority of the depths of the abnormal observations are shorter than the normal observations.

For the second example, the depths of a normal and abnormal observations are computed by WEIF and EIF, respectively. For comparing the results of WEIF and EIF, the depths of a normal observation for WEIF and EIF are plotted in Fig. 7a, while the depths of an abnormal for WEIF and EIF are plotted in Fig. 7b. The blue and red lines

Table 6 Descriptive statistics of transaction numbers for the compromised addresses

Statistics	Smart contract exploit	Flash loan attack	Identity theft	Normal transaction
Mean	1067.76	816.98	1493.38	126.89
Std	762.77	478.35	793.57	63.84
Min	22.00	19.00	11.00	13.00
Q1	384.50	503.75	522.25	102.00
Median	987.00	775.00	2000.00	112.50
Q3	1652.00	1019.50	2000.00	135.00
Max	2000.00	1876.00	2000.00	2000.00

Q1 and Q3 represent the first quartile and the third quartile of the historical transaction number, respectively

represent the depth of each tree processed by EIF and WEIF, respectively. Compared with EIF, the depths of a normal observation are increased by our proposed algorithm, as shown in Fig. 7a, whereas our proposed algorithm decreases the depths of an abnormal observation in Fig. 7b. In consequence, the difference in the average depth for the normal and abnormal observations is becoming larger for our proposed algorithm in contrast to EIF.

Evaluation metric

The F1 score is chosen to be the main metric of the models, which is the Harmonic Mean between precision and recall. The range for the F1 score is [0, 1]. It tells us how precise our classifier is and how robust it is. The greater F1 score indicates the better performance of our model. The precision, recall, and F1 score are formulated as follows:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{6}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{7}$$

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recal}. \tag{8}$$

True positive represents the number of real cyber attacks correctly detected, false positive represents the number of normal transactions wrongly detected as the cyber attacks, and false negative represents the number of real cyber attacks detected as the normal transactions.

Experimental evaluation

Several experiments on various types of cyber attacks are carried out to assess the efficiency and robustness of our proposed algorithm against all of the methods mentioned in "Related work" section.

Table 7 Summary statistics of features

Category	Feature name	Mean	Std	Min	Max
External transaction features	gas-fee	0.015	0.106	0.000	20.372
	ex-tx-volume	3393.605	25,488.039	0.000	342,000.000
	action-type	− 0.087	0.657	− 1.000	1.000
	spend-volume	15,393.631	56,557.212	0.000	523,208.005
	receive-volume	11,025.974	50,635.498	0.000	518,102.159
	spend-anomaly	0.273	0.447	0.000	1.000
	receive-anomaly	0.087	0.283	0.000	1.000
Internal transfer features	inter-tx-volume	109,507.400	315,690.942	0.000	2,088,140.768
	inter-ts-volume-50%	15,679.662	66,993.523	0.000	518,102.159
	inter-ts-volume-max	21,317.075	68,613.462	0.000	523,208.005
	inter-ts-count	4.623	11.677	0.000	67.000
	inter-ts-volume-min	1832.880	10,450.912	0.000	100,000.000
	inter-spend-volume	12,000.054	51,292.998	0.000	523,208.005
	inter-spend-count	0.530	2.612	0.000	32.000
	inter-receive-volume	11,025.974	50,635.498	0.000	518,102.159
	inter-receive-count	0.262	0.970	0.000	10.000
Token transfer features	to-tx-degree	3.322	4.639	0.000	28.000
	to-ts-count	78.984	185.409	0.000	700.000
	to-ts-count-anomaly	17.115	59.519	0.000	424.000
	to-tx-outdegree	0.634	1.178	0.000	8.000
	spend-to-ts-count	5.016	24.374	0.000	216.000
	to-tx-indegree	1.093	1.321	0.000	8.000
	receive-to-ts-count	30.721	92.054	0.000	350.000
	to-spend-anomaly	7.104	29.498	0.000	212.000
	to-receive-anomaly	1.317	5.466	0.000	58.000
Transaction action features	flash-currency-count	0.109	0.313	0.000	2.000
	flash-action-count	0.191	0.712	0.000	6.000
	swap-currency-count	0.858	2.207	0.000	15.000
	swap-flash-count	0.383	0.561	0.000	2.000
	swap-action-count	0.344	0.476	0.000	1.000

Full name and explanations of the listed features are shown in Table 4, hereafter

Dataset and experimental setup

Dataset

All the detailed information on cyber-attack transactions is extracted from the open-source websites, referring to Etherscan, Bscscan, and Rekt Dataset. First, all the labeled cyber-attack transactions are extracted from the Rekt Dataset and classified into Smart contract exploit, Flash loan attack, and Identity theft. Second, 66, 62, and 58 compromised addresses of cyber-attack transactions are extracted for the Smart contract exploit, Flash loan attack, and the Identity theft based on the identification strategy of compromised address. Specially, 200 addresses that hackers have not attacked are also randomly extracted from Etherscan and Bscscan for verifying the effectiveness of the model, which are made up of smart contracts and EOAs and are similar to the compromised addresses of all cyber-attack transactions. Finally, the historical transactions of each compromised address are extracted from Etherscan and Bscscan using the data extraction strategy shown in Fig. 3, with the window size set to 2000.

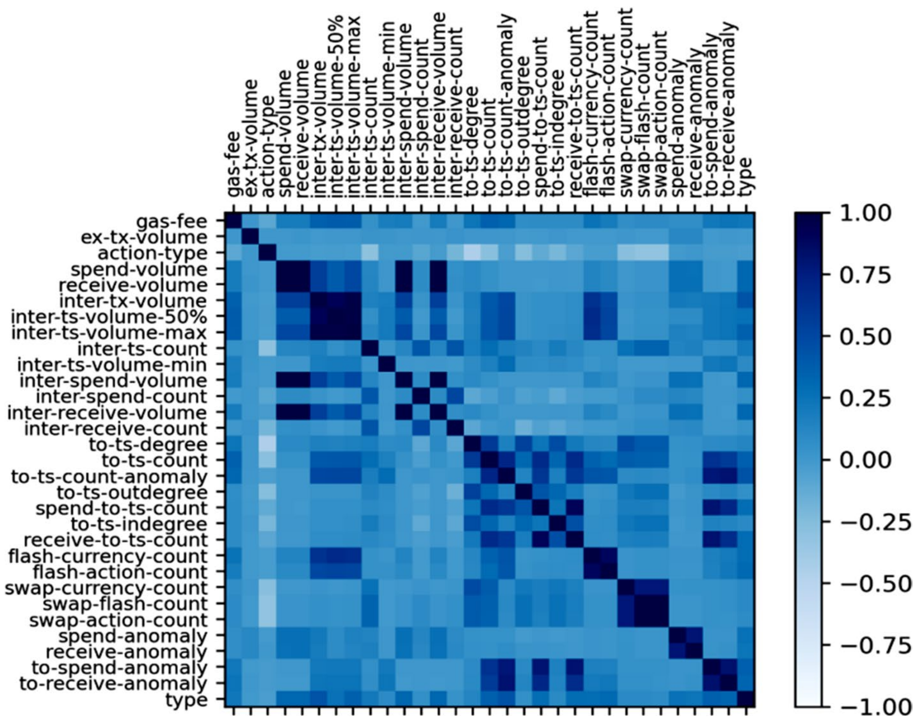


Fig. 8 Results of correlation analysis

To display the distribution of datasets for the compromised addresses and the differences between the three types of cyber attacks, this study computes the descriptive statistics of transaction quantity for the compromised addresses, as shown in Table 6. The descriptive statistics in Table 6 yield three findings. First, more than 50% compromised addresses of Identity theft have more than 2000 historical transactions, whereas more than 75% compromised addresses of other cyber attacks have less than 2000 historical transactions. The statistical results indicate that most of the Flash loan attacks and Smart contract exploits have been launched at the beginning of the decentralized applications, and that the compromised addresses of Identity theft have always been used for a long time or high-frequency trading. Second, the number of historical transactions on Identity theft, Smart contract exploit, and Flash loan attacks gradually decreases based on the values of median, third quartile, and max (see Table 6), demonstrating that the three types of cyber attacks are different from each other. Finally, as all values of the first quartile for all types of transactions are larger than 100, this indicates that the data deficiency of hack addresses is solved by extracting the historical transactions of the compromised addresses.

After the data extraction, several features (mentioned in "Methodology" section) are generated from external transactions, internal transfers, token transfers, transaction actions. To take full advantage of the transaction features, the descriptive statistics and correlation analyzes are carried out in this paper, whose result are shown in Table 7 and Fig. 8. As shown in Table 7, most of the min values of the features are equal to 0, since most of the normal transactions are simple transactions compared to the cyber-attack transactions.

Table 8 Result on Smart contract exploit

Category	Algorithm	Train			Test		
		Precision	Recall	F1 score	Precision	Recall	F1 score
SML	XGB	0.924 ± 0.039	0.572 ± 0.047	0.707 ± 0.047	0.653 ± 0.097	0.308 ± 0.045	0.419 ± 0.061
	RF	1.000 ± 0.000	0.924 ± 0.043	0.960 ± 0.023	0.633 ± 0.033	0.393 ± 0.077	0.483 ± 0.069
	LGBM	0.987 ± 0.013	0.813 ± 0.068	0.890 ± 0.035	0.619 ± 0.048	0.341 ± 0.130	0.430 ± 0.122
UML	CBLOF	0.830 ± 0.010	0.902 ± 0.011	0.865 ± 0.010	0.795 ± 0.023	0.875 ± 0.025	0.833 ± 0.024
	HBOS	0.882 ± 0.013	0.728 ± 0.011	0.798 ± 0.012	0.861 ± 0.028	0.775 ± 0.025	0.816 ± 0.026
	KNN	0.862 ± 0.011	0.880 ± 0.011	0.871 ± 0.011	0.833 ± 0.024	0.875 ± 0.025	0.854 ± 0.024
	Avg KNN	0.886 ± 0.032	0.848 ± 0.000	0.862 ± 0.015	0.871 ± 0.071	0.800 ± 0.000	0.832 ± 0.032
	LOF	0.846 ± 0.000	0.717 ± 0.000	0.776 ± 0.000	0.813 ± 0.000	0.650 ± 0.000	0.722 ± 0.000
	OCSVM	0.847 ± 0.007	0.902 ± 0.011	0.874 ± 0.001	0.834 ± 0.016	0.875 ± 0.025	0.854 ± 0.004
	FB	0.683 ± 0.017	0.750 ± 0.011	0.715 ± 0.014	0.582 ± 0.058	0.675 ± 0.125	0.624 ± 0.087
	DeepSVDD	0.866 ± 0.043	0.891 ± 0.022	0.877 ± 0.011	0.828 ± 0.120	0.875 ± 0.025	0.848 ± 0.075
	VAE	0.837 ± 0.000	0.891 ± 0.000	0.863 ± 0.000	0.850 ± 0.000	0.850 ± 0.000	0.850 ± 0.000
	IF	0.868 ± 0.021	0.859 ± 0.011	0.863 ± 0.016	0.884 ± 0.067	0.859 ± 0.011	0.869 ± 0.027
	EIF	0.837 ± 0.033	0.880 ± 0.011	0.857 ± 0.012	0.830 ± 0.037	0.880 ± 0.033	0.853 ± 0.004
	WEIF	0.875 ± 0.071	0.880 ± 0.022	0.876 ± 0.032	0.861 ± 0.089	0.900 ± 0.050	0.880 ± 0.070

Each metric is formed with its mean value and corresponding range. The corresponding range of each metric is the max difference between the mean value and its real value of the metric, hereafter

According to the result of the correlation analysis in Fig. 8, the type of cyber attack is heavily related to the external features and token transfer features (i.e., gas fee, spend volume, spend-anomaly, to–ts-count and to-ts–count-anomaly). Meanwhile, the redundant features exist in the feature data as a result of the correlation analysis. For example, the correlation coefficients between internal transfer features (i.e., inter-tx-volume, inter-ts-volume-50%, and inter-ts-volume-max) are extremely high, resulting in a lower model efficiency. To improve the detection model's efficiency, we remove redundant features prior to model training.

Experimental setup

To provide sound results, the dataset of the compromised address is split into a train dataset and a test dataset according to the ratios of 70% and 30%. Furthermore, the expected proportion of anomalies in the entire dataset is set at 5% for all UML methods, as mentioned in "Methodology" section. For our proposed algorithm, many parameters should be decided before model training, such as the multiplier in Table 5, extension level, and the number of isolation trees in Table 15 ("Appendix 3"). The grid search technique is used to estimate the parameters of WEIF in order to find the optimal parameters, as it has always been used to find the optimal parameters for several algorithms such as SVM and neural network algorithm (Pontes et al. 2016; Syarif et al. 2016). Based on parameter estimation using sklearn's GridSearchCV,[12] the multiplier, the extension level, and the number of isolation trees of WEIF are equal to 2, 4, and 100, respectively.

[12] https://scikit-learn.org/stable/modules/g0.013enerated/sklearn.model_selection.GridSearchCV.html

Table 9 Result on Flash loan attack

Category	Algorithm	Train			Test		
		Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
SML	XGB	0.934 ± 0.007	0.803 ± 0.086	0.861 ± 0.053	0.677 ± 0.010	0.557 ± 0.079	0.608 ± 0.044
	RF	0.991 ± 0.009	0.991 ± 0.009	0.991 ± 0.000	0.767 ± 0.045	0.578 ± 0.013	0.658 ± 0.008
	LGBM	0.981 ± 0.000	0.944 ± 0.018	0.962 ± 0.010	0.719 ± 0.031	0.512 ± 0.034	0.598 ± 0.034
UML	CBLOF	0.825 ± 0.009	0.909 ± 0.000	0.865 ± 0.005	0.809 ± 0.017	0.974. ± 0.027	0.894 ± 0.021
	HBOS	0.893 ± 0.012	0.852 ± 0.011	0.872 ± 0.016	0.881 ± 0.024	0.928 ± 0.019	0.904 ± 0.022
	KNN	0.857 ± 0.009	0.886 ± 0.000	0.872 ± 0.019	0.838 ± 0.019	0.947 ± 0.000	0.889 ± 0.011
	Avg KNN	0.919 ± 0.032	0.886 ± 0.000	0.902 ± 0.005	0.900 ± 0.005	0.909 ± 0.000	0.904 ± 0.009
	LOF	0.785 ± 0.008	0.500 ± 0.023	0.611 ± 0.019	0.833 ± 0.009	0.789 ± 0.053	0.810 ± 0.032
	OCSVM	0.842 ± 0.009	0.909 ± 0.000	0.874 ± 0.005	0.845 ± 0.019	0.928 ± 0.019	0.884 ± 0.019
	FB	0.628 ± 0.006	0.614 ± 0.023	0.620 ± 0.009	0.612 ± 0.053	0.789 ± 0.053	0.689 ± 0.022
	DeepSVDD	0.860 ± 0.047	0.886 ± 0.000	0.872 ± 0.024	0.839 ± 0.108	0.974 ± 0.026	0.896 ± 0.051
	VAE	0.833 ± 0.000	0.898 ± 0.012	0.864 ± 0.005	0.864 ± 0.000	0.947 ± 0.000	0.903 ± 0.000
	IF	0.870 ± 0.019	0.909 ± 0.000	0.889 ± 0.010	0.884 ± 0.068	0.909 ± 0.000	0.895 ± 0.035
	EIF	0.835 ± 0.035	0.909 ± 0.000	0.870 ± 0.019	0.827 ± 0.043	0.909 ± 0.000	0.865 ± 0.023
	WEIF	0.880 ± 0.055	0.909 ± 0.000	0.894 ± 0.029	0.888 ± 0.063	0.928 ± 0.019	0.906 ± 0.045

Table 10 Result on Identity theft

Category	Algorithm	Train			Test		
		Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
SML	XGB	0.962 ± 0.038	0.571 ± 0.000	0.717 ± 0.011	0.829 ± 0.060	0.529 ± 0.059	0.641 ± 0.026
	RF	1.000 ± 0.000	0.893 ± 0.012	0.943 ± 0.007	0.867 ± 0.049	0.588 ± 0.059	0.701 ± 0.058
	LGBM	1.000 ± 0.000	0.798 ± 0.036	0.887 ± 0.022	0.787 ± 0.059	0.559 ± 0.088	0.652 ± 0.081
UML	CBLOF	0.672 ± 0.032	0.427 ± 0.037	0.522 ± 0.037	0.583 ± 0.083	0.382 ± 0.088	0.461 ± 0.091
	HBOS	0.809 ± 0.017	0.463 ± 0.000	0.589 ± 0.005	0.739 ± 0.039	0.412 ± 0.088	0.528 ± 0.010
	KNN	0.790 ± 0.016	0.598 ± 0.012	0.681 ± 0.014	0.767 ± 0.033	0.676 ± 0.000	0.719 ± 0.031
	Avg KNN	0.877 ± 0.049	0.598 ± 0.012	0.711 ± 0.025	0.828 ± 0.095	0.676 ± 0.029	0.744 ± 0.056
	LOF	0.477 ± 0.023	0.134 ± 0.012	0.209 ± 0.017	0.325 ± 0.075	0.088 ± 0.029	0.139 ± 0.043
	OCSVM	0.732 ± 0.008	0.500 ± 0.012	0.594 ± 0.006	0.683 ± 0.017	0.441 ± 0.029	0.535 ± 0.017
	FB	0.407 ± 0.037	0.268 ± 0.024	0.324 ± 0.029	0.391 ± 0.083	0.382 ± 0.029	0.383 ± 0.117
	DeepSVDD	0.642 ± 0.051	0.268 ± 0.049	0.373 ± 0.040	0.647 ± 0.186	0.324 ± 0.147	0.417 ± 0.017
	VAE	0.719 ± 0.005	0.500 ± 0.012	0.590 ± 0.010	0.714 ± 0.014	0.441 ± 0.029	0.545 ± 0.026
	IF	0.794 ± 0.027	0.561 ± 0.000	0.657 ± 0.009	0.829 ± 0.094	0.598 ± 0.029	0.692 ± 0.025
	EIF	0.763 ± 0.049	0.622 ± 0.012	0.685 ± 0.027	0.764 ± 0.054	0.659 ± 0.000	0.707 ± 0.023
	WEIF	0.819 ± 0.063	0.744 ± 0.012	0.778 ± 0.022	0.814 ± 0.109	0.706 ± 0.029	0.753 ± 0.047

Results

Based on the experimental setup, this section aims to provide two types of results: one is about the efficiency and generality of our proposed algorithm (WEIF) and the other is about the importance of the generated features in detecting cyber-attack transactions on blockchain.

The performance of WEIF

Our proposed algorithm, as well as the comparative algorithms mentioned in "Related work" section, are used in three different types of cyber attacks. The results of the Smart contract exploit, the Flash loan attack, and the Identity theft are shown in Tables 8, 9 and

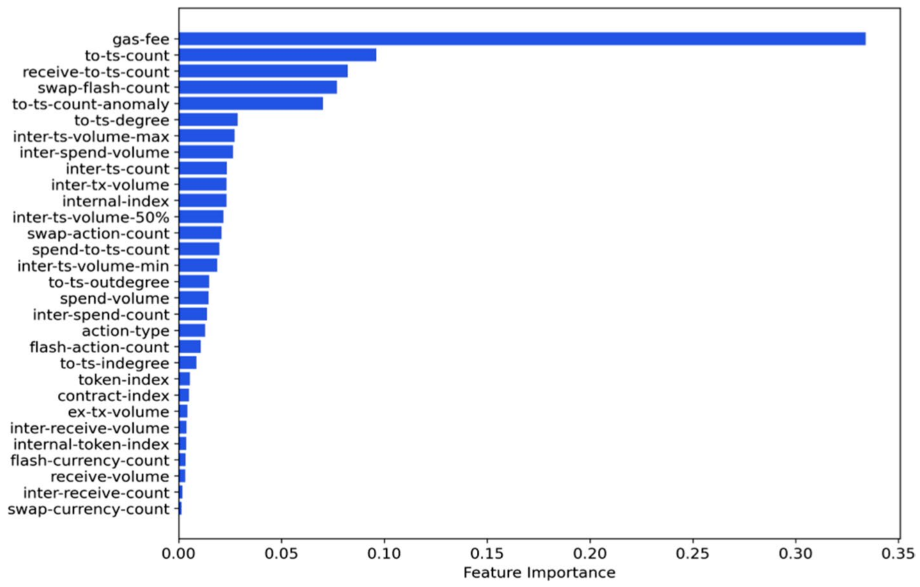


Fig. 9 Feature importance of Smart contract exploit

10, where all the recalls of GCN are equal to zero because of the heavily data imbalance and hence, its result is not included. According to the experimental results, we have the following finds.

According to the results of the Smart contract exploit in Table 8, WEIF has the highest F1 score on the test dataset. In Table 8, the difference in F1 score between the two results of datasets is small, and the average F1 score of UML is around 0.83 for the test dataset. In Table 8, all SML methods perform well on the train dataset when compared to UML methods, but the average F1 score of SML on the test dataset is less than 0.5. As for the result of the Flash loan attack in Table 9, WEIF outperforms all comparative models with 0.906 on the F1 score for the test dataset, and the average F1 score of UML is about 0.87 on the test dataset. However, although the methods of SML have a good performance on the train dataset, these methods work poorly on the test dataset, as shown in Table 9. As for the results of Identity theft in Table 10, WEIF achieved the best performance with 0.753 on F1 score. Most of the methods belonging to UML have a poor performance on cyber-attack detection of Identity theft except for the Avg k-nearest neighbors (KNN), KNN, EIF, and WEIF, and its average differences in F1 score for both types of datasets are about 0.05. Be similar to the results of SML in Tables 8 and 9, the SML has a good performance on the train dataset, but poor performance on the test dataset, as shown in Table 10.

To summarize, three types of cyber attacks can be detected based on the machine learning methods. First, for all three types of cyber attacks, the proposed algorithm achieves the highest F1 score on the test datasets, demonstrating the efficiency and generality of WEIF on different types of cyber attacks. Second, all of the F1 scores of SML on train datasets are significantly higher than those on test datasets, whereas the differences in F1 scores of UML for the two types of data sets are negligible. As a result, SML performs less effectively than UML in detecting cyber attacks on account-based

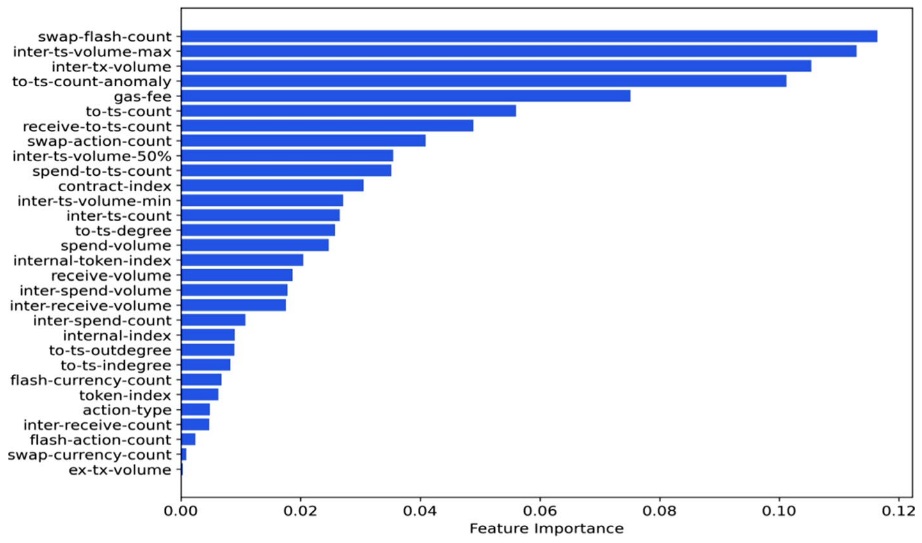


Fig. 10 Feature importance of Flash loan attack

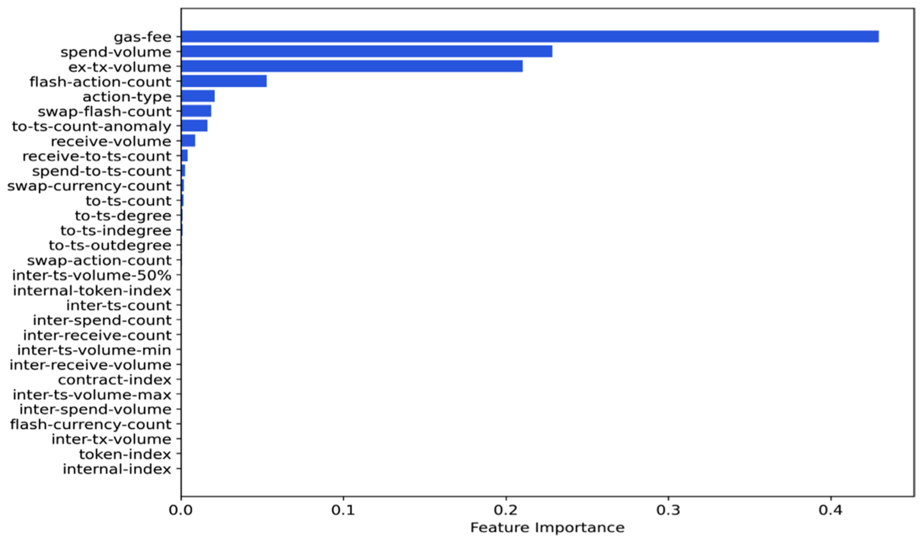


Fig. 11 Feature importance of Identity theft

blockchains. Last but not least, when compared to EIF, our proposed WEIF achieves F1 score improvements of about 0.03, 0.04 and 0.05, respectively, indicating that computing the anomaly score based on the weighted depths of the EIF makes our proposed model more efficient.

Feature importance

Generally speaking, a higher feature importance score means that the specific feature will have a larger effect on the model. In order to determine out which feature is significant for the cyber-attack detection, random forest is applied in the analysis of feature importance owning to its best performance of cyber-attack classification among all methods of SML, as shown in Tables 8, 9 and 10. Figures 9, 10 and 11 show the results of feature importance for three types of cyber attacks. As shown in Figs. 9, 10 and 11,

Table 11 Robustness test of WEIF

Type of cyber-attack	Precision	Recall	F1 score
Smart contract exploit	0.881	0.894	0.887
Flash loan attack	0.868	0.937	0.901
Identity theft	0.843	0.741	0.789

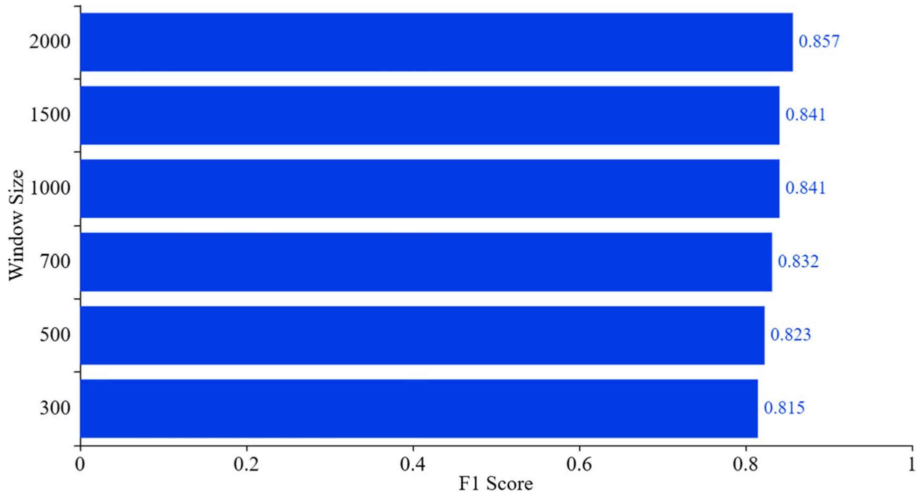


Fig. 12 Results of F1 score with different window sizes of training data

there are some differences and similarities in the analysis of feature importance for three types of cyber attacks.

In terms of the differences between the three types of cyber attacks, the number of significant features differs based on the results of feature importance. In particular, the top five features account for more than 70% of the impact in detecting Smart contract exploits, as shown in Fig. 9, whereas the cumulative importance score of the top ten features is about 60% for the detection of Flash loan attack in Fig. 10 and the top three features account for about 90% of impact for the detection of Identity theft in Fig. 11. In terms of similarity between three types of cyber attacks, some features are shared by all types of cyber attacks, as illustrated in Figs. 9, 10 and 11. For example, the feature of gas fee is included in the top ten significant features for all types of cyber attacks, demonstrating that more complicated operations within the cyber-attack transaction result in more gas consumed when compared to normal transactions. To sum up, although the cyber-attack transactions are different, the features designed in this paper are general for all types of cyber-attack detection based on the high F1 scores achieved by WEIF, shown in Tables 8, 9 and 10. Furthermore, the significant features for the three types of cyber attacks are almost the same as the results of feature importance for LGBM, as shown in Figs. 16, 17, 18 (see "Appendix 4"). This demonstrates that the result of feature importance does not significantly depend on the selection of SML methods.

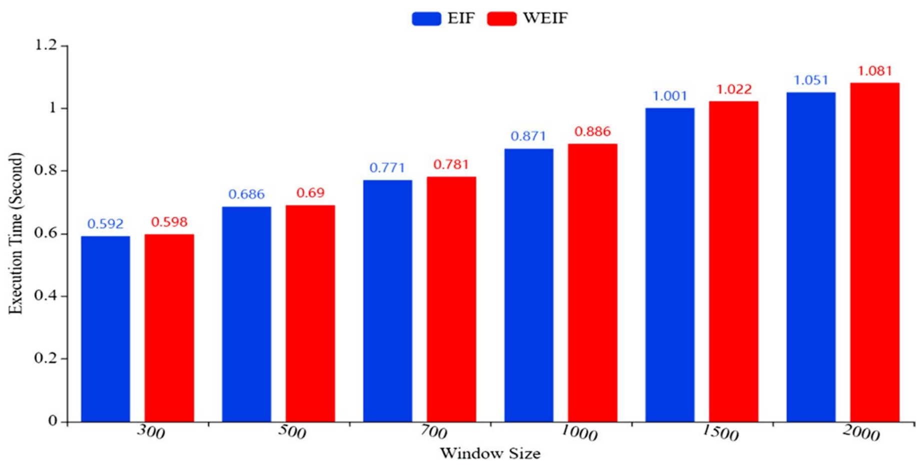


Fig. 13 Results of execution time on different window sizes of training data. The blue and red bars represent the execution time of EIF and WEIF, respectively

Additional validations

Two additional validations are carried out to evaluate our proposed algorithm's efficiency and robustness. The first validation is to conduct a robustness test of WEIF based on a new dataset. The second validation is to conduct experiments on different window sizes of training data, mentioned in the data extraction, to determine whether our proposed approach is robust and suitable for the real-time detection of cyber-attack transactions.

The first validation is performed on the new dataset, which contains all of the compromised addresses from cyber attacks as well as an alternate set of 100 random addresses that were not attacked by hackers. The robustness tests on WEIF for all three types of cyber attacks follow the same process as the previous section, and the results are listed in Table 11. Compared to the results shown in Tables 8, 9 and 10, the performances for all three types do not decrease, which indicates the robustness of WEIF.

Six experiments are designed in the second validation to test the robustness and execution time of WEIF by using different sizes of training data mentioned in "Methodology" section, i.e., different window size of historical transactions of the compromised address for model training. Figures 12 and 13 show both results. As shown in Fig. 12, although the training data window size decreases from 2000 to 300, the F1 score of WEIF decreases by only 0.05, with the lowest F1 score being 0.815, *indicating the robustness of WEIF again that is not sensitive to the size of the training data*. According to Fig. 13, the average execution time of EIF and WEIF, referring to the average time consumption of model training and prediction for a single transaction, are almost the same, and the gap in the average execution time between WEIF and EIF is also narrowing as the size decrease. This finding implies that, when compared to EIF, WEIF does not significantly increase time consumption. Furthermore, WEIF's lowest execution time is about 0.6 s, which is significantly less than the average block time mentioned above. As a result of the stability and robustness of our proposed algorithm, the window size can be set to a lower value for the blockchain with lower block time. Otherwise, it can be increased for the blockchain with longer block time.

Discussion

Based on the results of the experiments above, our study provides important theoretical values and high practical application values for the researchers and industry practitioners, respectively. *For the theoretical values*, we propose a general framework for cyber-attack detection that incorporates the compromised address recognizer, real-time transaction filter system, general feature generator, and detection model. Our proposed framework addresses the issue of data imbalance and data scarcity of hackers' addresses. Furthermore, within our proposed framework of cyber-attack detection, we propose a novel algorithm named WEIF that is based on the isolation forest and its extension as the detection model and outperforms all methods on three types of cyber attacks based on experimental results. Based on its strong performance against various types of cyber attacks, our proposed framework and algorithm can serve as a theoretical foundation for improving the supervision and regulation of public blockchains. *For the practical application values*, the generality, robustness, and low time consumption of our proposed algorithm on different types of cyber attacks have been proved by the experimental results above. First, our proposed algorithm's generality and robustness to various types of cyber attacks makes it perfectly capable of detecting dynamic and ever-changing cyber attacks on account-based blockchains. Second, because of its low time consumption, our proposed algorithm can detect all real-time transactions in a short period of time. Meanwhile, to reduce the number of real-time transactions analyzed by our proposed algorithm, a novel approach of filtering real-time transactions based on the expenditures of compromised addresses is proposed. Finally, the technology of multiprocesses or multithreads can be used to accelerate the process of detecting cyber attacks. All of these evidences show that our proposed algorithm can be directly applied to the detection of cyber attacks in the blockchain industry in real time.

Conclusion and future work

The dynamic and ever-changing cyber attacks frequently happen on account-based blockchain in recent years. However, only a few technologies of machine learning have been applied for the real-time detection of cyber attacks. To this end, we propose a systematic and comprehensive anomaly detection method for coping with this problem. First, a novel algorithm namely, WEIF, is developed for anomaly detection based on the standard isolation forest and its extended model. Then, we propose a general framework on the basis of our proposed algorithm through a comprehensive study of real-world examples of cyber attacks. Within this general framework, a novel approach of identifying the compromised address is created to solve the hack addresses' data deficiency and reduce the time consumption of our proposed framework. Next, several experiments are carried out on different types of cyber attacks to verify our proposed algorithm's efficiency and generality. As expected, the experimental results demonstrate the advantage of our proposed method in contrast to many widely used state-of-the-art techniques. Besides, the result also indicates that the techniques of SML are not suitable for real-time detection of cyber attacks, owing to data imbalance and data deficiency. Finally, the results of additional experiments provide more evidence for supporting the lower time consumption and the robustness of our proposed approach, illustrating that our proposed approach is capable of real-time detection of cyber attacks on the account-based blockchain.

In the future, we plan to extend our work from three aspects. First, we will try to apply the multivariate time-series analysis algorithms to the anomaly detections in the context of the account-based blockchain, since the historical transactions belong to the dataset with a time-series format. Second, crypto exchanges are the main gateway for connecting real-world user information to pseudonymous addresses on account-based blockchain, but few studies related to crypto exchanges have been conducted. Therefore, we plan to thoroughly study different types of crypto exchanges. Finally, we will develop applications to analyze the fund flows of illegal activities and automatically extract the funds' path from the large transaction networks.

Appendix 1: Common concepts on account-based blockchain

In this appendix, the common concepts on account-based blockchain, related to the analysis of cyber-attack, are shown as follow.

Account. There are two types of accounts: externally owned account (*EOA*) controlled by private key, and smart contract account controlled by their codes, described in white-paper of Ethereum and BSC. For the EOA, it is made up of a cryptographic pair of keys: public and private. The public key and private key are similar to an online bank account and the corresponding password, and losing the private key is equivalent to losing the funds of the corresponding account. For the smart contract account, it is written in programming languages such as Solidity, and all of the smart contracts are executed on the blockchain. Applications can call the smart contract functions, change their state, and initiate transactions. Both account types have the ability to receive, hold and send the crypto assets, and interact with the deployed smart contracts.

Transaction. The transaction on the account-based blockchain refers to an action initiated by an externally owned account. In other words, an account is managed by a human, not a contract. A transaction is a signed data message sent from an externally owned account to another account on blockchain, e.g., so the recipient of a transaction has more crypto asset and the sender has less. It contains the information of the transaction sender and recipient, which are the amount of cryptocurrency to be transferred and the transaction fee the sender is willing to pay. Generally, an internal transfer is the consequence of smart contract logic that is triggered by an external transaction, where the transaction is transmitted from the EOA to the smart contract. Meanwhile, the execution of transaction with a smart contract may result in more transactions depending on the code of smart contract.

Transaction Fee. Transaction fee is also known as gas fee, and it is the fee paid to the nodes (miner) for executing transaction. When we transfer money on the account-based blockchain, the miner must pack our transaction and put it on the blockchain to complete the transaction. In this process, the nodes will consume computing resources, and the miner should be compensated. The gas fee only depends on the complexity of the transaction. Overall, the higher gas fee is consumed in the more complicated transaction. Meanwhile, the price of gas can be set by users, and the set price will affect the

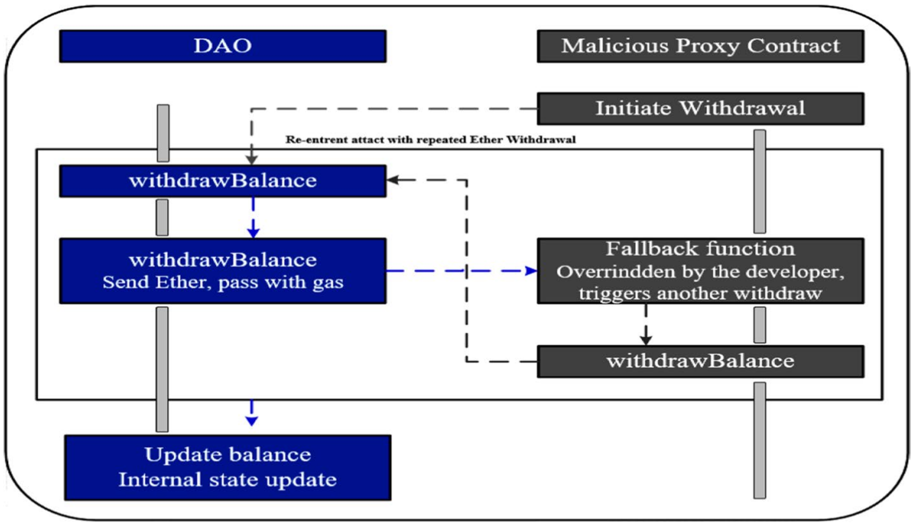


Fig. 14 An illustration of the re-entrancy attack

transaction speed. As miners give priority to transactions with high gas prices. If the transaction party sets the gas price too low, the speed of the transaction will slow down.

Block. A block is mined when added to the account-based network, once the consensus is reached. A transaction is said to be mined when it is included to the blockchain in a new block. Therefore, each block has several transactions. In order to preserve the historical transaction records, every new block contains a unique identifier of its parent block, this is how all of the blocks are linked in the blockchain. As a result, all of the blocks on the account-based blockchain are strictly ordered as well as the transactions within the blocks. According the Etherscan and BscScan, the average block times, referring to the times it takes to mine a new block of BSC and Ethereum, are 2.5 and 12 s respectively.

Cryptocurrency**.** The Ether (*ETH*) and *BNB* are the digital fuel for Ethereum and BSC, respectively, which is similar to the gasoline for our cars. For instance, transaction fee is an amount of computer power required in order to execution of the transaction, which is paid by ETH or BNB. Compared with the ETH and BNB, ERC-20 and BEP-20 Tokens are the most commonly used tokens on the Ethereum and BSC network, which are supported by the smart contract. According to the report of Etherscan and BscScan, there are more than 500,000 types of ERC20 Tokens on Ethereum and 2,568,483 types of BEP-20 Tokens on BSC, with over 100 billion dollars market capitalization.

Flash loan**.** Flash loan is one of the decentralized applications based on smart contracts. Because of the state reverting feature of Ethereum and BSC, the tools of flash loan are developed to enable the uncollateralized lending service. This type of loan service provides users an unsecured loan from lenders without intermediaries. The rule of flash loan is that the borrower must pay back the loan before the transaction ends. Otherwise, the transaction will be rejected and the smart contract reverses the transaction, and it's like the loan never happened in the first place on Ethereum and BSC.

Decentralized exchange (DEX). DEXs are blockchain-based applications that provide users the trading of crypto assets without intermediaries. It works entirely

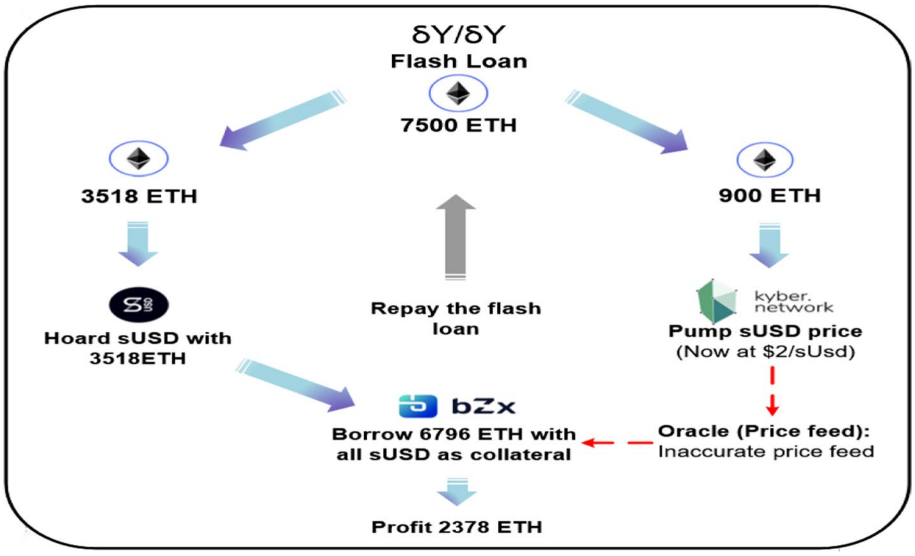

δY/δY
Flash Loan
7500 ETH
3518 ETH
900 ETH
Repay the flash loan
Hoard sUSD with 3518ETH
kyber. network
Pump sUSD price (Now at $2/sUsd)
bZx
Borrow 6796 ETH with all sUSD as collateral
Oracle (Price feed): Inaccurate price feed
Profit 2378 ETH

Fig. 15 Flowchart of bZx Flash loan attack scheme

through automated algorithms based on a set of smart contracts. Unlike the centralized exchanges, DEXs do not allow for the exchange between crypto assets and fiat. Meanwhile, DEXs do not hold users' crypto assets. Instead, users hold all their assets directly in their wallets all the time.

Appendix 2: Real-world examples of cyber-attack transaction

In this section, two real-world examples of cyber-attack transaction, referring to Flash loan attack and re-entrancy attack, are elaborated below.

Real-world example of Smart contract exploit. Taking the re-entrancy as an example, its brief process is shown in Fig. 14, DAO is the victim contract which is marked with blue, and the malicious proxy contract is marked with gray. The detail process of re-entrancy attack contains several steps:

Step 1. Malicious contract calls the withdrawBalance function of DAO attempting to withdraw a certain amount A of ETH from an account containing a large amount B of ETH;
Step 2. The withdrawBalance function of DAO check that the withdrawal is valid if B > A.
Step 3. The withdrawBalance function of DAO transfers the requested A ETH to the malicious contract.
Step 4. This transfer triggers Fallback function of the malicious contract, which calls the withdrawBalance function of DAO again requesting a withdrawal of A ETH.
Step 5. The withdrawBalance function of DAO checks that the withdrawal is valid, since account balance of Malicious contract is still B and B > A.
Step 6. The withdrawBalance function of DAO transfers the requested A ETH to the malicious contract.
Step 7. The Fallback function of malicious contract returns without performing any action.

Step 8. The withdrawBalance function of DAO updates its state to reflect the withdrawal in step 6, reducing account balance of malicious contract to (B–A) ETH.

Real-world example of Flash loan attack. For example, the Flash loan attack happened on 18 February 2020 which is shown in Fig. 15. First, the hacker obtained a flash loan of 7500 ETH from the bZx protocol and split the total amount of ETH into three parts (3518, 900 and 3082). Second, the hacker converted 3518 ETH to sUSD on Synthetix, the synthetic USD token (sUSD) is enabled by the Syntetix protocol and the sUSD were bought at the price of $1. Third, the hacker swapped 900 ETH in two batches for sUSD through Kyber. The first batch was sold for 540 ETH in KyberSwap and the second batch was sold 18 times for 20 ETH each in Kyber, effectively inflating the price of sUSD up to $2 in Kyber (By this way, the supply of sUSD will be decreased, and the total supply of ETH is increasing in Kyber. The ratio of the supply of ETH and the supply of sUSD in Kyber will rise, which makes the price of sUSD go up). After this operation, the hacker has finished all the preparatory work, such as accumulation of sUSD and inflating the price to the certain price in Kyber. Fourth, since bZx relies on Kyber for the real-time price feed, with the spiked sUSD/ETH price (This price is higher than the actual prices), the hacker started to attack the bZx by borrowing 6796 ETH with all the collection of sUSD. Finally, the hacker repaid the 7500 ETH flash loan back to bZx with a profit of 2378 ETH.

Appendix 3: Algorithms of the standard isolation forest and its extension

In this section, the algorithms of the standard isolation forest and its extension are introduced as follow (See Tables 12, 13, 14, 15).

Table 12 Algorithm of isolation tree (IF)

```
Algorithm. $iTree(X, e, l)$
Input: The input data - $X$, current tree height - $e$, height limit - $l$
Output: an iTree
1.  If $e \geq l$ or $|X| \leq 1$ then
2.      return $exNode\{Size \longleftarrow |X|\}$
3.  else
4.      Let $F$ be a list of attributes in $X$
5.      Random select an attribute $f \in F$
6.      Randomly select a split point $p$ from max and min
7.      Values of attribute $f$ in $X$
8.      $X_l \longleftarrow filter(X, f < p)$
9.      $X_r \longleftarrow filter(X, f \geq p)$
10.     return $inNode\{Left \longleftarrow iTree(X_l, e+1, l),$
                        $Right \longleftarrow iTree(X_r, e+1, l),$
                        $FeaturesSplit \longleftarrow f,$
                        $SplitValue \longleftarrow p\}$
11. end if
```

Table 13 Algorithm of constructing isolation tree (EIF)

Algorithm. $iTree(X, e, l)$

Input: The input data - X, current tree height - e, height limit - l

Output: an iTree

```
1.  if $e \geq l$ or $|X| \leq 1$ then
2.      return $exNode\{Size \longleftarrow |X|\}$
3.  else
4.      Randomly select a normal vector $\vec{n} \in IR^{|X|}$ by drawing each coordinate of $\vec{n}$ from a standard Gaussian distribution
5.      Randomly select a normal vector $\vec{p} \in IR^{|X|}$ in the range of $X$
6.      set coordinates of $\vec{n}$ to zero according to extension level
7.      $X_l \longleftarrow filter(X, (X - \vec{p}) \cdot \vec{n} \leq 0)$
8.      $X_r \longleftarrow filter(X, (X - \vec{p}) \cdot \vec{n} > 0)$
9.      return $inNode\{Left \longleftarrow iTree(X_l, e + 1, l),$
                    $Right \longleftarrow iTree(X_r, e + 1, l),$
                    $Normal \longleftarrow \vec{n},$
                    $Intercept \longleftarrow \vec{p}\}$
10. end if
```

Table 14 Algorithm of depth computation (EIF)

Algorithm. $PathLength(x, T, e)$

Input: The observation - x, an iTree - T, current path length - e; to be initialized to zero when first called

Output: path length of x

```
1.  if $T$ is an external node then
2.      return $e + c(T.size)$ {$c(.)$is defined in isolation forest}
3.  end if
4.  $\vec{n} \longleftarrow T.Normal$
5.  $\vec{p} \longleftarrow T.Intercept$
6.  if $(x - \vec{p}) \cdot \vec{n} \leq 0$ then
7.      return $PathLength(x, T.left, e + 1)$
8.  else if $(x - \vec{p}) \cdot \vec{n} > 0$ then
9.      return $PathLength(x, T.right, e + 1)$
10. end if
```

Table 15 Algorithm of constructing isolation forest (EIF)

Algorithm. $iForest(X, t, \psi)$

Input: The input data - X, number of tree - t, sub-sampling size - ψ

Output: a set of t *iTrees*

```
1.  Initialize $Forests$
2.  set height limit $l = ceiling(log_2\psi)$
3.  for $i = 1$ to $t$ do
4.      $X' \leftarrow sample(X, \psi)$
5.      $Forest \leftarrow Forest \cup iTree(X', 0, l)$
6.  end for
```

Appendix 4: Analysis of feature importance based on LGBM

In this section, LGBM is also selected to carry out the analysis of feature importance for verifying the whether or not the result of feature importance depends the selection of SML methods. And the results of feature importance for LGBM are show in Figs. 16, 17 and 18.

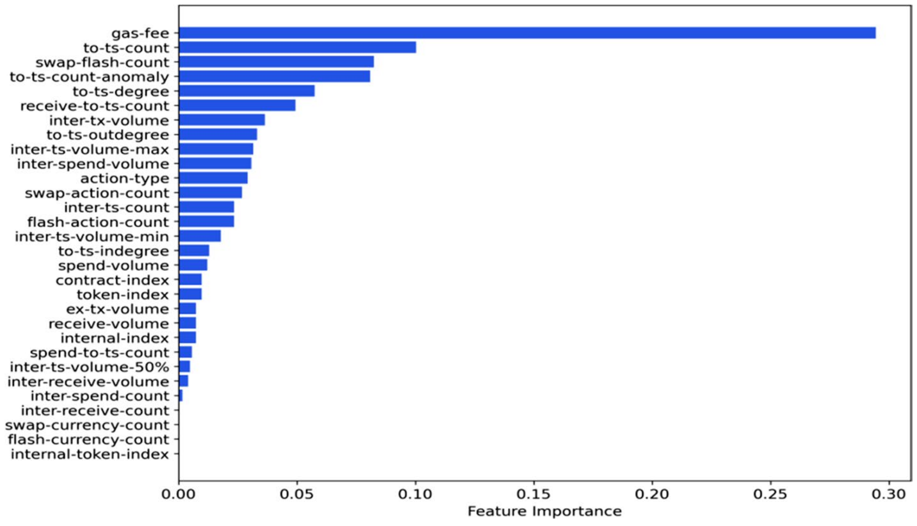


Fig. 16 Feature importance of Smart contract exploit based on LGBM

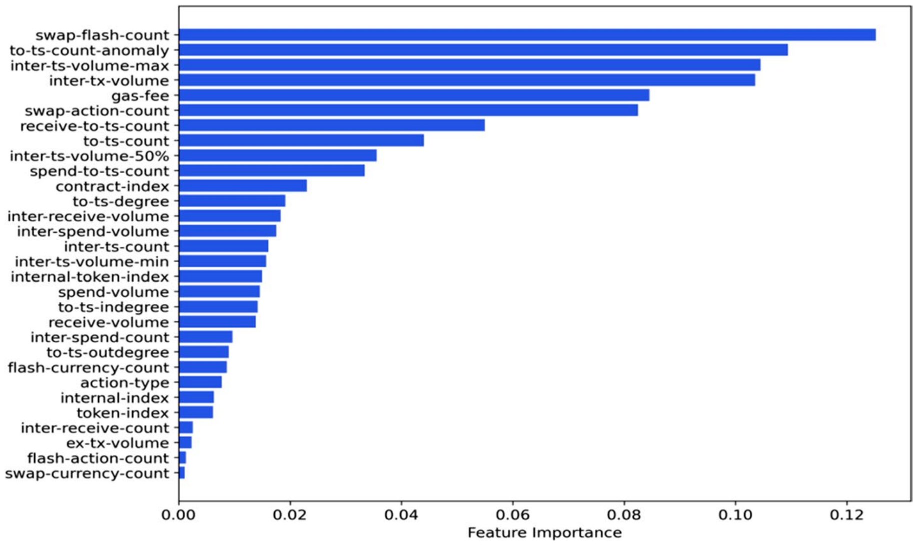


Fig. 17 Feature importance of Flash loan attack based on LGBM

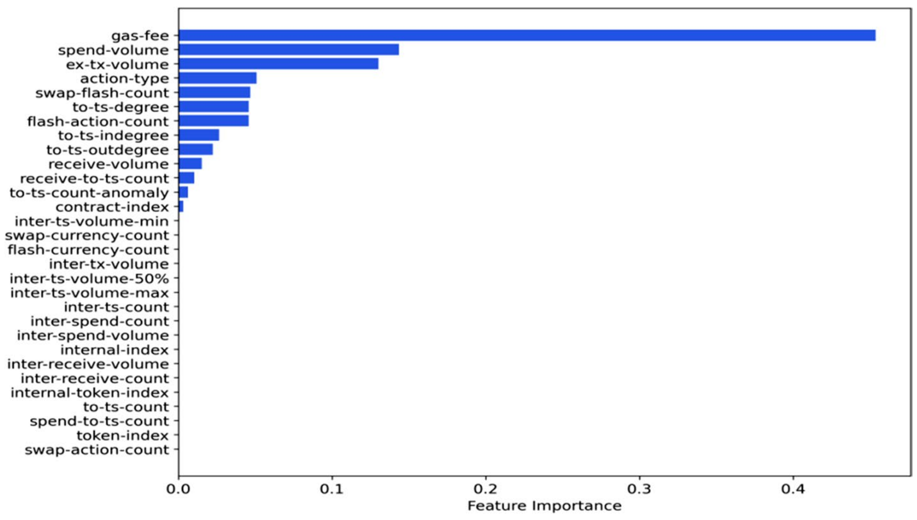


Fig. 18 Feature importance of Identity theft based on LGBM

Abbreviations

dapps	Decentralized applications
BSC	Binance smart chain
DEX	Decentralized exchange
SML	Supervised machine learning methods
PTA	Postmortem analysis technology
UML	Unsupervised machine learning methods
RF	Random forest
LGBM	Light gradient boosting machine
GCN	Graph convolutional network
LOF	Local outliers' factors
CBLOF	Cluster-based local outlier factor
HBOS	Histogram-based outlier score
OCSVM	One-class SVM
VAE	Variational autoencoder
DeepSVDD	Deep support vector data description
FB	Feature bagging
IF	Isolation forest
EIF	Extended isolation forest
WEIF	Weighted and extended isolation forest
EOA	Externally owned account
ETH	Ether

Acknowledgements
We greatly thank the discussions from Dr. Chao Liu and Dr. Baoqiang Zhan.

Author contributions
ZF: Conceptualization, methodology, visualization, software, validation, and data curation, and writing—original draft. YL: Writing—review and editing, conceptualization, supervision, validation. XM: Writing—review and editing, visualization, and data curation. All authors read and approved the final manuscript.

Funding
This work was supported by the National Natural Science Foundation of China (72171059, 71771041, 72121001), the Fundamental Research Funds for the Central Universities (FRFCU5710000220) and the Natural Science Foundation of Heilongjiang Province, China (No. YQ2020G003).

Availability of data and materials
Data and codes are available at https://github.com/fung2022/A-blockchain-oriented-approach-for-detecting-cyber-attack-transactions.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 17 August 2022 Accepted: 29 March 2023
Published online: 02 May 2023

References

Angiulli F, Pizzuti C (2002) Fast outlier detection in high dimensional spaces. In: European conference on principles of data mining and knowledge discovery, pp 15–27
Aspris A, Foley S, Svec J, Wang L (2021) Decentralized exchanges: the "wild west" of cryptocurrency trading. Int Rev Financ Anal 77:101845
Aziz RM, Baluch MF, Patel S, Ganie AH (2022) LGBM: a machine learning approach for Ethereum fraud detection. Int J Inf Technol 1–11
Breiman L (2001) Random forests. Mach Learn 45(1):5–32
Breunig MM, Kriegel H-P, Ng RT, Sander J (2000) LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data, pp 93–104
Carcillo F, Dal Pozzolo A, Le Borgne Y-A, Caelen O, Mazzer Y, Bontempi G (2018) Scarff: a scalable framework for streaming credit card fraud detection with spark. Inf Fus 41:182–194
Carcillo F, Le Borgne Y-A, Caelen O, Kessaci Y, Oblé F, Bontempi G (2021) Combining unsupervised and supervised learning in credit card fraud detection. Inf Sci 557:317–331
Chen T, Guestrin C (2016) Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 785–794
Dal Pozzolo A, Caelen O, Le Borgne Y-A, Waterschoot S, Bontempi G (2014) Learned lessons in credit card fraud detection from a practitioner perspective. Expert Syst Appl 41(10):4915–4928
Efanov D, Roschin P (2018) The all-pervasiveness of the blockchain technology. Procedia Comput Sci 123:116–121
Falcão F, Zoppi T, Silva CBV, Santos A, Fonseca B, Ceccarelli A, Bondavalli A (2019) Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection. In: Proceedings of the 34th ACM/SIGAPP symposium on applied computing, pp 318–327
Fang F, Ventre C, Basios M, Kanthan L, Martinez-Rego D, Wu F, Li L (2022) Cryptocurrency trading: a comprehensive survey. Financ Innov 8(1):1–59
Farrugia S, Ellul J, Azzopardi G (2020) Detection of illicit accounts over the Ethereum blockchain. Expert Syst Appl 150:113318
Goldstein M, Dengel A (2012) Histogram-based outlier score (HBOS): a fast unsupervised anomaly detection algorithm. In: KI-2012: poster and demo track, pp 59–63
Hariri S, Kind MC, Brunner RJ (2019) Extended isolation forest. IEEE Trans Knowl Data Eng 33(4):1479–1489
Harvey CR, Ramachandran A, Santoro J (2021) DeFi and the future of finance. Wiley
He Z, Xu X, Deng S (2003) Discovering cluster-based local outliers. Pattern Recogn Lett 24(9–10):1641–1650
Hilas CS, Mastorocostas PA (2008) An application of supervised and unsupervised learning approaches to telecommunications fraud detection. Knowl-Based Syst 21(7):721–726
Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, pp 448–456
Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu T-Y (2017) Lightgbm: a highly efficient gradient boosting decision tree. In: Advances in neural information processing systems, vol 30
Kingma DP, Welling M (2013) Auto-encoding variational Bayes. arXiv preprint http://arxiv.org/abs/1312.6114
Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint http://arxiv.org/abs/1609.02907
Lazarevic A, Kumar V (2005) Feature bagging for outlier detection. In: Proceedings of the eleventh ACM SIGKDD international conference on knowledge discovery in data mining, pp 157–166
Liu FT, Ting KM, Zhou Z-H (2012) Isolation-based anomaly detection. ACM Trans Knowl Discov Data (TKDD) 6(1):1–39
Patel V, Pan L, Rajasegarar S (2020) Graph deep learning based anomaly detection in ethereum blockchain network. In: International conference on network and system security, pp 132–148
Pontes FJ, Amorim G, Balestrassi PP, Paiva A, Ferreira JR (2016) Design of experiments and focused grid search for neural network parameter optimization. Neurocomputing 186:22–34
Puggini L, McLoone S (2018) An enhanced variable selection and Isolation Forest based methodology for anomaly detection with OES data. Eng Appl Artif Intell 67:126–135
Qin K, Zhou L, Livshits B, Gervais A (2021) Attacking the defi ecosystem with flash loans for fun and profit. In: International conference on financial cryptography and data security, pp 3–32
Rovetta S, Suchacka G, Masulli F (2020) Bot recognition in a Web store: an approach based on unsupervised learning. J Netw Comput Appl 157:102577
Ruff L, Vandermeulen R, Goernitz N, Deecke L, Siddiqui SA, Binder A, Müller E, Kloft M (2018) Deep one-class classification. In: International conference on machine learning, pp 4393–4402
Schölkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC (2001) Estimating the support of a high-dimensional distribution. Neural Comput 13(7):1443–1471
Sebastião H, Godinho P (2021) Forecasting and trading cryptocurrencies with machine learning under changing market conditions. Financ Innov 7(1):1–30
Shen J, Zhou J, Xie Y, Yu S, Xuan Q (2021) Identity inference on blockchain using graph neural network. In: International conference on blockchain and trustworthy systems, pp 3–17
Syarif I, Prugel-Bennett A, Wills G (2016) SVM parameter optimization using grid search and genetic algorithm to improve classification performance. TELKOMNIKA (telecommun Comput Electron Control) 14(4):1502–1509

Thabtah F, Hammoud S, Kamalov F, Gonsalves A (2020) Data imbalance in classification: experimental evaluation. Inf Sci 513:429–441

Xu JJ (2016) Are blockchains immune to all malicious attacks? Financ Innov 2(1):1–9

Xu M, Chen X, Kou G (2019) A systematic review of blockchain. Financ Innov 5(1):1–14

Yu S, Jin J, Xie Y, Shen J, Xuan Q (2021) Ponzi scheme detection in ethereum transaction network. In: International conference on blockchain and trustworthy systems, pp 175–186

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.