Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 2983605, 10 pages
https://doi.org/10.1155/2021/2983605

Research Article

Malicious Mining Behavior Detection System of Encrypted Digital Currency Based on Machine Learning

Mu Bie[1] **and Haoyu Ma**[2]

[1]*Information Technology Center, Chongqing Jianzhu College, Chongqing 400072, China*
[2]*School of Information Science and Engineering, Chongqing Jiaotong University, Chongqing 400074, China*

Correspondence should be addressed to Haoyu Ma; mlsxeby@163.com

Received 26 July 2021; Revised 16 August 2021; Accepted 16 September 2021; Published 18 November 2021

Academic Editor: Yong Chen

Copyright © 2021 Mu Bie and Haoyu Ma. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the gradual increase of malicious mining, a large amount of computing resources are wasted, and precious power resources are consumed maliciously. Many detection methods to detect malicious mining behavior have been proposed by scholars, but most of which have pure defects and need to collect sensitive data (such as memory and register data) from the detected host. In order to solve these problems, a malicious mining detection system based on network timing signals is proposed. When capturing network traffic, the system does not need to know the contents of data packets but only collects network flow timing signals, which greatly protects the privacy of users. Besides, we use the campus network to carry out experiments, collect a large amount of network traffic data generated by mining behavior, and carry out feature extraction and data cleaning. We also collect traffic data of normal network behavior and combine them after labeling. Then, we use four machine learning algorithms for classification. The final results show that our detection system can effectively distinguish the normal network traffic and the network traffic generated by mining behavior.

1. Introduction

With the rapid popularity of the Internet and the continuous emergence of information technology, the attack means for emerging technologies are also constantly upgraded and evolving. With the help of various means, the media and carriers for the implementation of network security threats are unpredictable, and the network security situation is always not optimistic. In recent years, with the development of encryption currency trading market, and encryption of monetary value, malicious attack has become the most widely used means in mining and a kind of network security threats, affecting the enterprises, organizations, and individuals at the end of 2008, in the hearing on the network publishing a paper on the currency, called the currency: peer-to-peer electronic cash system [1]. In Satoshi Nakamoto's white paper, the authors describe a new kind of "currency" trading system that can work in a nontrust environment. That "currency" is cryptocurrency. Subsequent researchers then extended this concept and introduced the mining system into the mining of Bitcoin, which led to the rapid development of Bitcoin. In addition to Bitcoin, other types of cryptocurrencies have been released, bringing broader market space [2]. Mining itself is not a malicious activity. For Bitcoin mining, it is a computational activity, using hardware resources to perform mathematical computations in the Bitcoin network, which are paid in Bitcoins. As a result, countless people began to join the Bitcoin mining team [1]. In addition, with the more and more types of electronic currency, mining currency is also expanding, and mining forms and algorithms are also constantly evolving. At first, the mining behavior mainly uses CPU, GPU, and other resources, and the cost of these hardware resources is relatively low. Later, as the cryptocurrency market gradually expanded, people were no longer satisfied with using these hardware resources for mining but began to design various professional mining tools, such as programmable array for mining. Such professional mining tools greatly improved the mining speed [3].

Malicious mining or mining hijacking is a malicious practice that uses infected devices to mine cryptocurrency. The attacker exploits the computing power and bandwidth of the victim (the computer) to mine (in the vast majority of cases, this is done without the victim's knowledge or consent). Typically, malicious mining software responsible for such activities is designed to use sufficient system resources to perform mining operations without being noticed or detected for as long as possible. Since cryptocurrency mining requires a lot of computing resources, hackers try to break into multiple devices on the same LAN, so they can make more money by increasing computing power. Malicious cryptocurrency mining usually takes place in two ways: first, mining with browser-based cryptography, typically by embedding a script containing the mining code into a website. Take Coinhive scripting as an example, a cryptocurrency mining service that relies on a small piece of code embedded in a website that uses some or all of the computing power of a browser visiting a particular website to mine Monero cryptocurrency [4]. Due to the special ring signature scheme [5] used in the Monero coin protocol, it is difficult for law enforcement officers to determine the identity of the attacker through the collection address. Many attackers use Coinhive's services to turn multiple compromised sites and routers into mining machines for malicious mining operations.

The second way is to use binary-based malicious mining software for mining. When a user inadvertently clicks an e-mail containing mining software or a related malicious advertising link, the malicious mining software will be downloaded to the user's host computer, and malicious mining will be carried out without the victim's knowledge [6]. The earliest mining Trojan appeared in 2012. With the price of cryptocurrency skyrocketing from 2017, mining Trojan has become a major security threat in the Internet since 2018. Malicious mining attack not only brings performance loss to users, but also may greatly waste power resources and increase carbon emissions. Therefore, it is of great significance to conduct an in-depth study on mining software and find a more effective detection method for malicious cryptocurrency mining software by analyzing its characteristics.

In order to deal with the security threats brought by malicious cryptocurrency mining software, researchers have proposed a variety of protection schemes against cryptocurrency mining attacks; it is found that these existing solutions have certain limitations through comprehensive analysis [7].

In 2018, Hong [8] from Fudan University proposed a method to detect web mining behavior based on hash function. The author uses the Chrome Remote Interface (based on the Chrome debugging protocol, which supports debugging the Chrome browser) to carry out remote debugging of the web page and obtains the function call information of the web mining operation. Then, select the commonly used hash operation function for monitoring, according to the total time spent on hash operation to judge whether it is mining software.

In the same year, Konoth et al. from VU Amsterdam proposed Minesweeper [9], which is a defense strategy for web mining based on the characteristics of mining algorithm itself. Through inspection, the author finds that the current malicious web page mining operations mostly adopt WebAssembly (WASM) technology to improve the efficiency of web page encryption mining and use WASM code to realize the mining algorithm Cryptonight. Therefore, by analyzing the WASM code, the author matches the mining algorithm based on the detected encryption and hash operations, so as to determine whether a web page operation belongs to mining behavior.

Conti et al. [10] proposed the method of detecting mining software by using Hardware Performance Counter (HPC). The change of HPC value is used as the characteristic of the mining algorithm, and the curve conforming to the mining algorithm is fitted with the machine learning algorithm, so as to judge whether the program conforms to the rules of the mining algorithm and determine whether it is mining software [11]. However, this approach is highly controversial.

Based on the existing research work, this paper proposes a set of mining flow detection systems based on machine learning, aiming at the deficiency of the existing malicious mining behavior detection schemes. The method presented in this paper has three obvious advantages: first, it purely utilizes network stream timing signals for detection with strong privacy, without the need to install any local software and the need to read user data. Second, for network traffic transmitted with SSL encryption, there is no need to decrypt, as well as to detect; it is a warning, and classification can be realized through packet flow. Third, the scheme is highly adaptable and can be adapted to different mining algorithms and pool strategies at any time through new data training.

2. Network Traffic Classification Detection Algorithm

Various machine learning algorithms are used in this paper, including support vector machines (SVM), K-Nearest Neighbor (KNN), AdaBoost, and Convolutional Neural Networks (CNN). The performance differences of various algorithms were compared through experiments, and the detection efficiency was evaluated.

2.1. Linearly Separable Support Vector Machines. First, our task is to distinguish between normal network traffic and malicious mining traffic. And it is a typical binary classification task; the basic idea of support vector mechanism model is to create an optimal decision hyperplane and make the plane on both sides of the plane maximize the distance between the two classes of samples recently, and support vector machine was used first, in order to solve the problem of sample classification binary classification problems, so the algorithm has a good applicability for such tasks.

We assume that the sample network traffic data set to be trained is as follows:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}, \quad y_i \in \{-1, +1\}, \tag{1}$$

wherein x_j, $i = 1, \ldots, k$ represents the training sample data, and y_i represents the class symbol corresponding to each training sample.

The method of network traffic classification is to find a partition hyperplane in the sample space based on the collected training set M. The ultimate purpose of support vector machine is to construct a decision function, which can correctly classify every information. In two-dimensional space, intuitiveness is said to exist in the two-dimensional space of different points corresponding to the coordinates of separation, and the line can be a straight line and can also be a curve; for this task, characteristics exist in multidimensional space, thread classification obviously cannot meet the requirements, and this time, structure can only be used for classification of plane to classify a point in space, And such lines and planes are classification lines and classification hyperplanes, which can be expressed as

$$(\omega \cdot x) + b = 0. \tag{2}$$

In Equation (2), ω is the direction of the normal of the hyperplane, and b is the distance between the hyperplane and the origin. The location of the hyperplane is determined by these two unknowns. It is assumed that the distance between any point x in the network traffic training set and the hyperplane (ω, b) can be expressed as

$$r = \frac{|\omega \cdot x + b|}{\|\omega\|}. \tag{3}$$

As shown in Figure 1, in the SVM schematic diagram, the training sample points falling on the dotted line in the figure are the "support vectors" of the SVM algorithm, namely, the points with correct classification. We can calculate the sum of the distances from any two support vectors belonging to different classes to the hyperplane as

$$\gamma = \frac{2}{\|\omega\|}. \tag{4}$$

Through analysis, it can be found that there are many hyperplanes that can separate the two types of data, but there is only one optimal hyperplane that meets the two conditions at the same time, and only the points closest to the hyperplane are closely related to the hyperplane, and the other points will not directly affect the classification results. We give the basic lemma for support vector machines [12]:

$$\begin{aligned} &\min_{\omega,b} \frac{1}{2}\|\omega\|^2 \\ &\text{s.t. } y_i(\omega \cdot x_i + b) \geq 1, \\ &i = 1, 2, \ldots, k. \end{aligned} \tag{5}$$

Through observation, it can be found that Equation (5) is a convex quadratic programming problem with a unique minimum point. Formula (5) is transformed into a "duality problem" by Lagrange multiplier method. By introducing

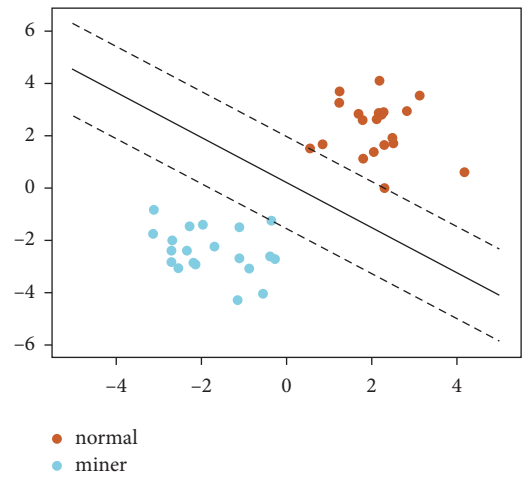

Figure 1: Use SVM to classify network traffic.

Lagrange multiplier, the formula can be processed more easily. Although the effect is not clear in linear cases, such change can be generalized to solve nonlinear problems and can be converted into kernel function for calculation. In the algorithm in this paper, we use the linear kernel function, which is defined as follows:

$$\kappa(x_1, x_2) = \langle x_1, x_2 \rangle. \tag{6}$$

2.2. Suitable AdaBoost Algorithm. Adaptive Boosting is the full name of AdaBoost algorithm, and Boosting, as a typical algorithm, was proposed in 1995 [13]. Boosting algorithm's main intention is to enhance the "weak classifier" into "strong classifier," and each "weak classifier" has its own functions and deficiencies. The advantages of each "weak classifier" will be brought into play, and the deficiencies of one "weak classifier" will be compensated by another or more "weak classifiers" to enhance the whole process into a "strong classifier." The AdaBoost algorithm enhances several "weak classifiers" into "strong classifiers" by an adaptive way. AdaBoost algorithm is reflected in practical application, mostly used in dichotomy. Recently, AdaBoost algorithm is mostly used in image recognition technology with remarkable effect and excellent performance in feature selection technology [14].

The algorithm ideas applicable to this task are as follows:

Assign a set of labeled network traffic data sets $(x_1, y_1), \ldots, (x_n, y_n)$ and $y_i \in \{+1, -1\}$, y as a lable, and x as a feature. Then, initialize weight:

$$D_i(i) = \frac{1}{n}(i = 1, \ldots, n). \tag{7}$$

Select a weak classifier h_t to reduce the weight of the error:

$$h_t = \text{argmin}_{h_{j \in H}} e_j = \sum_{i=1}^{n} D_t(i) I\left[y_i \neq h_j(x_i)\right]. \tag{8}$$

And calculate the proportion of $h_t(x)$ in the strong classifier:

$$a_t = \frac{1}{2}\ln\frac{1-e_i}{e_i}. \tag{9}$$

Until $e_t = \min_j e_j > 1/2$, set $T = t-1$ and then stop the loop. Update a new weight:

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t} \tag{10}$$

where Z_t is the normalized factor to ensure that D_{t+1} could be allocated.

Finally, we got a strong classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} a_t h_t(x)\right). \tag{11}$$

2.3. Unsupervised Learning K-Nearest Neighbor Algorithm. K-Nearest neighbor algorithm is a famous statistical method of pattern recognition, which plays an important role in machine learning classification algorithms. In the KNN algorithm, the distance between the given test object and each object in the training set is firstly calculated; then, the K training objects nearest to the test object are selected as the nearest neighbors of the test object; and finally, the test objects are classified according to the main categories belonging to the K nearest neighbors. In general, KNN uses the "voting method" in the classification task; that is, it chooses the marker category with the most occurrence times in k instances as the prediction result. Euclidean distance is used to calculate the similarity when measuring the sample distance:

Assume that the category of network traffic is determined by n attribute; that is, the sample to be tested is an n-dimensional vector. Assume that the representation of sample i is $X^i = (x_1^i, x_2^i, \ldots, x_n^i)$. x_j^i represents the $j-$th attribute of sample i, so the Euclidean distance between any two samples can be obtained:

$$d\left(x^i, x^j\right) = \sqrt{\sum_{p=1}^{n}\left(x_p^i - x_p^j\right)^2}. \tag{12}$$

In a feature space as shown in Figure 2, a blue circle represents normal traffic, an orange rectangle represents abnormal traffic, and a red star represents an unknown anomaly detection point to be detected. According to the anomaly detection principle of KNN algorithm, the comparison of Euclidean distances will be used to judge whether a new unknown point is abnormal. Thus, the intrusion points A and B can be marked as abnormal traffic by the KNN algorithm, because the Euclidean distance from these two points to the abnormal class is closer than that to the normal class.

2.4. End-to-End Convolutional Neural Network Algorithm. In our previous research [15], deep learning has been applied to the detection of dark web traffic, and because network traffic can extract hundreds of relevant features, the feature extraction and analysis are also a question worth considering; for the first three algorithms, we use the principal causes analysis (PCA) [14] for dimension reduction; for deep learning, we can not undertake feature extraction or reduce it; this is called the end-to-end learning solutions. In Figure 3, we show the steps of the end-to-end learning algorithm.

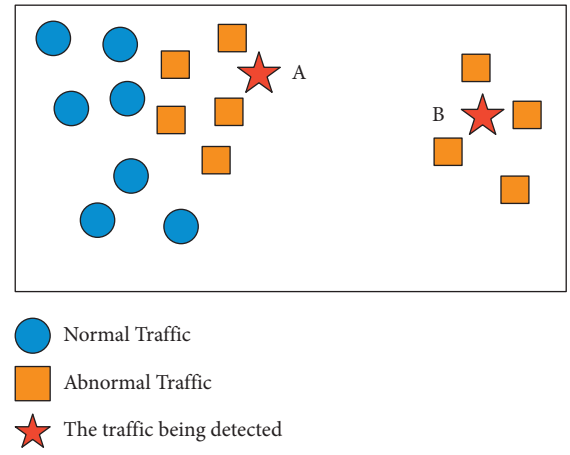

Figure 2: KNN.

Deep learning can automatically extract high-level features from original data and learn the internal rules and levels of samples. It has a high adaptability to massive high-dimensional data and well solves the problem of feature engineering in traditional machine learning.

In the overall structure of the model, the convolutional layer operation is the core part of CNN, responsible for automatically extracting a variety of abstract features from the original flow generated images, and different convolutional kernel learning features in the layer are different. The size parameters of the convolution kernel are specified manually, and the internal weight parameters need to be adjusted continuously in round training. The convolution kernel completes the convolution operation with each feature channel in the form of sliding window on the input matrix. Let us now define the $i-$th traffic data in the data set that is represented by k-dimensional vector, and a traffic with a characteristic length of n is represented as follows:

$$x_{i:n} = x_1 \oplus x_2, \ldots, \oplus x_n. \tag{13}$$

Convolutional operation: we define a convolution kernel $w \in \mathbb{R}^{hk}$, and the filter operates on a set of network traffic bytes of window width k and outputs a new feature. Refer to formula (14); a set of network traffic bytes generates characteristics by operating the following:

$$c_i = f\left(w \cdot x_{i:\,i+h-1} + b\right). \tag{14}$$

In formula (14), b is the bias term, and f is a nonlinear function.

In the dichotomy task, Sigmoid function is often used as the activation function [14]. In the actual training comparison, we find that the effect of this activation function is better than other activation functions.

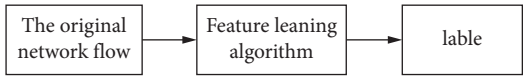

Figure 3: The end-to-end learning scheme.

$$f(x)_{\text{Sigmoid}} = \frac{1}{1+e^{-x}}, \tag{15}$$

$$c = [c_1, c_2, \ldots, c_{n-h+1}]. \tag{16}$$

Feature mapping: in formula (16), filters operate in every possible traffic byte window and generate a corresponding feature map, where $c \in \mathbb{R}^{n-h+1}$.

Pooling operation: we use maximum pooling for sampling. After the pooling layer completes the convolution operation between the image and the feature map in the convolution layer, the output data will enter the pooling layer for subsampling. By reducing the data dimension, overfitting is reduced, and the original feature information abstracted from the traffic data is extracted.

$$\hat{c} = \max\{c\}. \tag{17}$$

After the above operation, the resulting data will be entered into the full connection layer for classification. The operation of the full connection layer is described as follows: in formula (18), we define a linear function.

$$f(x)_{\text{probability}} = \sum_{i=1}^{d} w_i x_i + b. \tag{18}$$

We assume that there is a network traffic with d attributes described; we need to continuously update w and b for the purpose of training. And we choose Root Mean Square Prop (RMSProp) algorithm [13] as the optimization algorithm to update parameters:

$$s_{dw} = \beta v_{dw} + (1-\beta) dw^2, \tag{19}$$

$$s_{db} = \beta v_{db} + (1-\beta) db^2, \tag{20}$$

$$W = W - \alpha \frac{dW}{\sqrt{s_{dw}} + \varepsilon}, \tag{21}$$

$$b = b - \alpha \frac{db}{\sqrt{s_{db}} + \varepsilon}. \tag{22}$$

In formulas (19) and (20), s_{dw} and s_{db} are, respectively, the gradient momentum accumulated by the loss function in the first $t-1$ round of iteration. And β is an index of gradient accumulation, which we can define freely. In formulas (21) and (22), α is the learning rate in the training process, and to prevent the denominator from becoming zero, a small value ε is used for smoothing.

And we continue to use the sigmoid function as the activation function of the neuron. Then, in order to overcome the Sigmoid function brought by the characteristics of slow parameter update, we use the cross-entropy loss function to speed up the training process:

$$L(\hat{y}, y)_{\text{cross-entropy-loss}} = -y \log(\hat{y}) - (1-y)\log(1-\hat{y}). \tag{23}$$

Figure 4 shows the structure of the neural network. After the above processing, the trained neural network model can give the classification results of the input network traffic data.

3. Design of Malicious Mining Flow Detection System

In the early 1990s, the concept of abnormal traffic detection technology was first proposed in the literature [16]. Relevant scholars classified large-scale network traffic and divided it into normal traffic and abnormal traffic. Abnormal traffic detection technology can model network traffic characteristics to find abnormal patterns to predict the occurrence of abnormal, and at the same time, it can dig out new attack types for prevention. Abnormal traffic detection technology mainly includes five aspects, namely, network data traffic collection, traffic feature extraction, behavior modeling, abnormal detection, and result presentation and feedback. The specific technical process of abnormal traffic detection is shown in Figure 5.

In this paper, the detection model we proposed is roughly the same as that in Figure 5, and the specific implementation process will be introduced below.

3.1. Attacker Model. In this paper, it is assumed that the attacker will use the existing vulnerability exploitation program to remotely attack and exploit the hosts and services with vulnerabilities on the public network to achieve the purpose of planting malicious mining programs. Attackers may also be targeted at the target Server and host open Web services and applications for brute force cracking access, such as brute force cracking of Nginx Server or SQL Server, violent guess of SSH, VNC login credentials, and so on. There may also be unauthorized access vulnerabilities due to incorrectly configured application services and components deployed on the server. The hacker group carries out batch scanning of relevant service ports, and when the host and server with unauthorized access vulnerability are detected, further download and implant malicious mining programs by injecting execution scripts and commands [17]. Malicious mining attacks usually use remote code execution vulnerabilities or unauthorized vulnerabilities to execute commands and download and release subsequent malicious mining scripts or Trojans. Secondly, according to the our understanding and research of mining software, all mining software has network behavior, and this is an important basis for the study of mining software in this paper, but also an important support for the detection of mining software in this paper. Therefore, this paper assumes that mining software can access the network unconstrained and carry out network communication. Figure 6 describes in detail the communication flow between the miner and the mining pool.

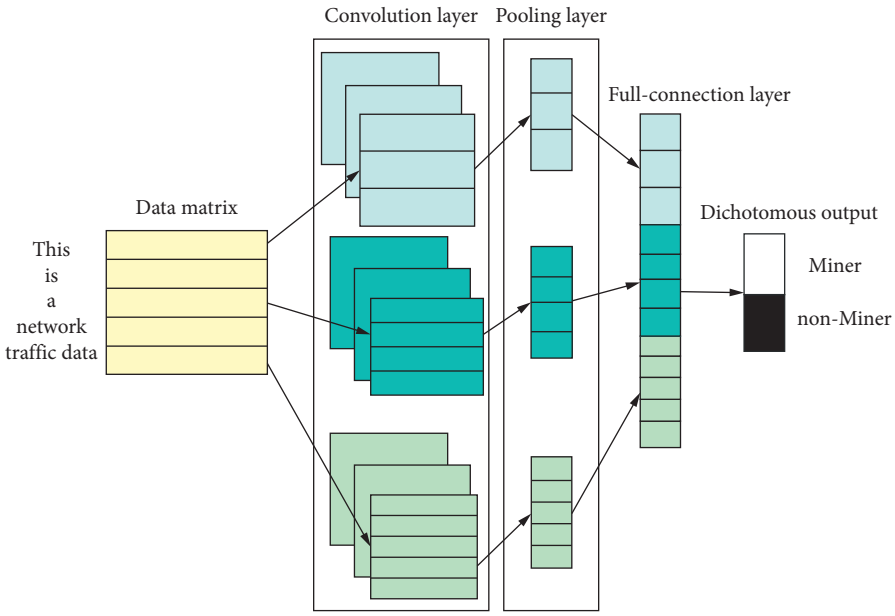

Figure 4: Neural network diagram.

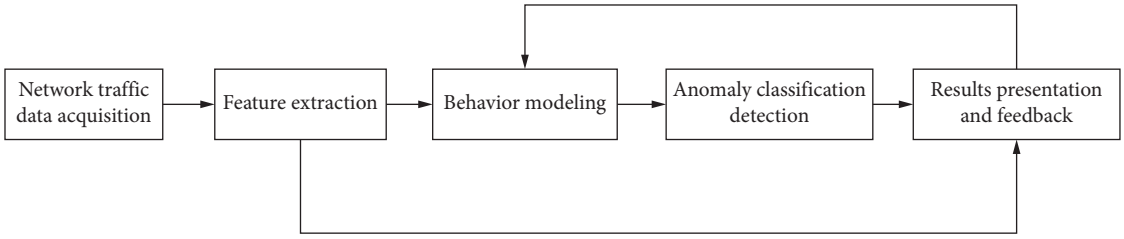

Figure 5: Flow for detecting abnormal traffic.

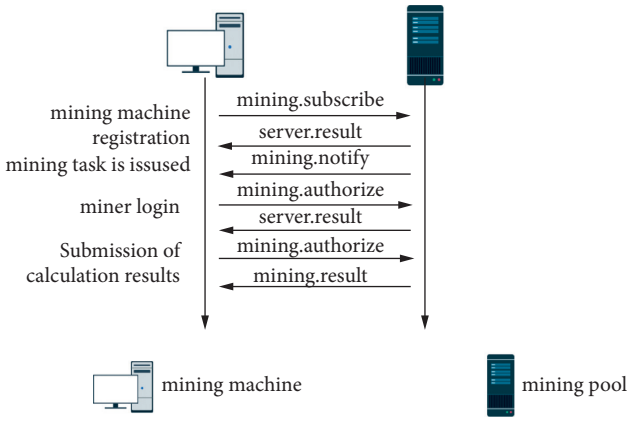

Figure 6: The communication flow between the controlled host and the mine pool.

We assume that all communication sessions in the figure above are transmitted using the Transport Layer Security protocol and cannot be decrypted. And the invaded host cannot be scanned for virus or Trojan because of objective reasons. We set that the attacker does not use a proxy server to forward network traffic, so that we can quickly lock the original IP address for blocking.

3.2. Detection and Defense Models. Anomaly traffic detection technology [18] is done by comparing the differences between normal mode and abnormal mode to judge whether it, apparently many kinds of attacks on attack, along with the rapid change of the large-scale traffic data, using the characteristics of these changes, can detect abnormal flow, so as to do a series of measures, including closing the relevant network port or shielding attack IP; this is undoubtedly a guarantee to the network environment. Anomaly traffic detection technology collected various hosts and servers of a large number of discrete traffic; the traffic data contains the network connection characteristics, building the model according to the data characteristics, analysis, and calculation of the secondary data, getting the multidimensional data flow characteristics, and achieving the normal behavior pattern, with the similar degree of normal behavior patterns to determine whether the behavior is unusual [19]. Finally, the feedback analysis of the test results is carried out. In this paper, we define abnormal traffic as the network traffic that mining software (Trojan) and mining pool communicate through.

Figure 7 shows the overall architecture diagram of the malicious mining flow detection system we proposed. In an environment where detection and warning are desired, network traffic can be collected from the interfaces of various

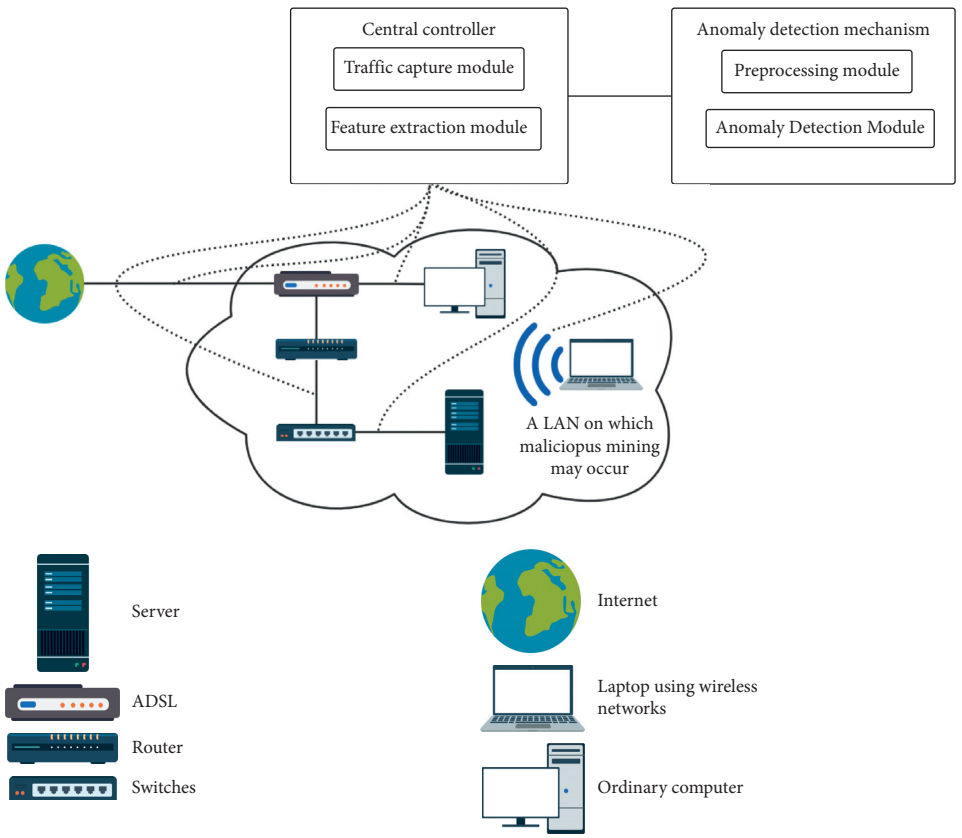

Figure 7: System architecture diagram.

devices for detection and analysis, or the underlying network can be analyzed directly through the device's operating system. The central controller will undertake this function. When the feature extraction module completes the feature extraction and feature selection of the data stream, it will send the selected optimal feature subset to the anomaly detection mechanism for processing and detection, so as to improve the efficiency and accuracy of the abnormal traffic detection mechanism and reduce the burden of the central controller.

Anomaly detection mechanism is a centralized data flow information analysis and anomaly detection processing mechanism. It obtains the stream feature vectors after feature selection through the central controller, preprocesses these vectors, and classifies them through the anomaly detection algorithm in Chapter 3 to determine whether there is an anomaly. Anomaly detection mechanism is divided into preprocessing module and anomaly detection module. The preprocessing module is mainly responsible for standardizing and normalizing the stream feature vectors sent by the controller. Suppose that there are n stream feature vectors, each of which contains t features, denoted as $X_{ij}\,(1 \le i \le n, 1 \le j \le t)$. The preprocessing of X_{ij} includes the following processes:

3.2.1. Standardized.

$$X_{ij}^{*} = \frac{X_{ij} - \text{Mean}_j}{\text{AvgDev}_j} \tag{24}$$

Among them,

$$\text{Mean}_j = \frac{X_{1j} + X_{2j} + \cdots + X_{nj}}{n},$$
$$\text{AvgDev}_j = \frac{\left|X_{1j} - \text{Mean}_j\right| + \left|X_{2j} - \text{Mean}_j\right| + \cdots + \left|X_{nj} - \text{Mean}_j\right|}{n} \tag{25}$$

3.3. Normalized. Normalization refers to the normalization of standardized data to the interval $[0, 1]$. Let X'_{ij} be the normalized value of X_{ij}^{*}:

$$X'_{ij} = \frac{X_{ij}^{*} - X_{\min}}{X_{\max} - X_{\min}}. \tag{26}$$

Among them,

$$X_{\min} = \min\left\{X_{ij}^{*}\right\},$$
$$X_{\max} = \max\left\{X_{ij}^{*}\right\}. \tag{27}$$

After the normalization and normalization of the preprocessing module of the anomaly detection module, the preprocessing stream feature vector will be handed over to the anomaly detection module for anomaly detection. The anomaly detection module uses the detection algorithm proposed in Chapter 3 to detect the traffic. Figure 8 shows the overall flow chart of the detection system we proposed.

4. Experiment

In order to simulate the real attack scenario, we used two hosts in the same campus network for experiments. One server was the target machine, and the other was Wireshark to capture the network traffic of the target machine. In order to make the collected data as fair and reliable as possible, we used the TLS protocol for the mine pool address, and all the features related to privacy in the data set were deleted (such as IP address).

4.1. Experiment Conditions. We use Wireshark [20] for network traffic capture, and then we use CICFlowMeter [21] for flow feature extraction. The specific experimental environment parameters are listed in Table 1.

4.2. Model Parameters. For the machine learning algorithm in Chapter 3, we have tried many parameters for training and evaluated under different parameter conditions. Finally, the parameters we selected are listed in Table 2–5.

The captured features of the train and test file are shown in Table 6. These data have gone through the preprocessing stage and removed the unobvious features.

4.3. Experimental Result. In Table 7, we use accuracy, precision, recall, and F1-score to compare several machine learning algorithms.

Accuracy is our most common evaluation index, and it is easy to understand, that is, the number of samples divided by all the samples. Generally speaking, the higher the accuracy, the better the classifier.

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{28}$$

Precision is defined as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{29}$$

This represents the proportion of instances that are divided into positive instances that are actually positive instances.

Recall rate is a measure of coverage, which measures that multiple positive cases are divided into positive cases.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{30}$$

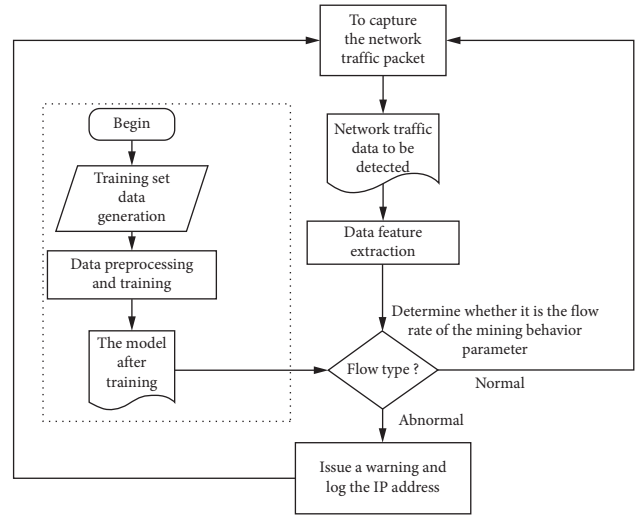

Figure 8: Flow chart of detection system.

Table 1: The experimental environment parameters.

Category	Parameters
CPU	Intel xeno 4210R
RAM	32 GB
GPU	Nvidia RTX3090
Operation system	Ubuntu 18.04 LTS
CUDA version	10.2
Machine learning platform	Pytorch 1.7.0 + scikit-learn 0.23

Table 2: Parameters used in support vector machine.

Parameter name	Parameter
Penalty coefficient of error term	0.3
Kernel	Linear
Probability	False
Max_iter	10000000
Random state	None

Table 3: Parameters used in AdaBoost.

Parameter name	Parameter
Base estimator	Decision tree
Number of base classifier cycles	60
Probability	False
Learning rate	1.2
Random state	None
Algorithm	SAMME.R

Sometimes, there are contradictions between precision and recall indicators, so we need to consider them comprehensively. The most common method is F-measure (also known as F-score):

$$F1 - \text{score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{31}$$

It can be seen from Table 7 that the effects of different machine learning methods on such tasks vary greatly, but the accuracy of these methods is high, but the precision

Table 4: Parameters used in KNN.

Parameter name	Parameter
N_neighbors	3
Weights	Uniform
Algorithm	Auto

Table 5: Training parameters used in CNN.

Parameter name	Parameter
Epoch	5
Iteration	30000
Batch size	128
Learning rate	0.01
Weight decay	1e-8
Momentum	0.9

Table 6: Description of traffic characteristics.

Name	Description
Protocol	Type of the protocol used
Flow duration	Length of connection
Flow Bytes (s)	Number of data bytes
Flow Packets (s)	Number of data packets
Flow IAT	Packets flow interarrival time
Fwd IAT	Forward interarrival time
Bwd IAT	Backward interarrival time
Active time	The amount of time a flow was active before becoming idle
Idle	The amount of time a flow was idle before becoming active

Table 7: Classification performance statistics.

ML algorithm	Accuracy	Precision	Recall	F1-Score
SVM	0.98	0.66	0.66	0.66
AdaBoost	0.99	0.78	0.81	0.79
KNN	0.91	0.64	0.63	0.64
CNN	1.00	0.96	0.86	0.91

performance is quite different. If we evaluate the classification effect according to F1-Score, deep learning is the most potential method to provide relatively correct prediction results.

5. Conclusion

The experimental results show that our detection system can effectively alert malicious mining behavior and protect user privacy. Our detection system can run on any node or host of the network and can quickly learn the characteristics of network traffic generated by different mining behaviors. For the malicious use of server computing resources for cryptocurrency mining, the existing detection methods all have some deficiencies; we used machine learning and data mining technology to analyze network traffic and greatly improve the efficiency of the detection system and effectively protect user privacy.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

At the same time, the authors also need to state that this study was funded by the key Project of the Ministry of Education of China under grant ZD2020070201 and Science and Technology Research Program of Chongqing Municipal Education Commission under grant KJZD-K201904301.

References

[1] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," *Decentralized Business Review*, Article ID 21260, 2008.

[2] J. D. P. Rodriguez and J. Posegga, "Rapid: Resource and Api-Based Detectionagainst In-Browser miners," in *Proceedings of the 34th Annual Computer Security ApplicationsConference*, San Juan, PR, USA, December 2018.

[3] R. Tahir, M. Huzaifa, A. Das et al., "mining on someone else's dime: mitigating covert mining operations in clouds and enterprises," *Research in Attacks, Intrusions, and Defenses*, Springer International Publishing, in *Proceedings of the International Symposium on Research in Attacks,Intrusions, and Defenses (RAID)*, pp. 287–310, September 2017.

[4] S. Eskandari, A. Leoutsarakos, T. Mursch, and J. Clark, "A first look at browser-based cryptojacking," in *Proceedings of the 2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 58–66, IEEE, London, UK, April 2018.

[5] S. Noether, "Ring signature confidential transactions for Monero," *IACR Cryptology*, vol. 1098, 2015.

[6] R. Konoth, "Malicious cryptocurrency miners: status and outlook," 2019, https://arxiv.org/abs/1901.10794.

[7] B. Ainapure, D. Shah, and A. Ananda Rao, "Performance analysis of virtual machine introspection tools in cloud environment," in *Proceedings of the International Conference on Informatics and Analytics*, Pondicherry, India, August 2016.

[8] G. Hong, "How you get shot in the back: a systematical study about cryptojacking in the real world," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, October 2018.

[9] R. K. Konoth, "Minesweeper: an in-depth look into drive-by cryptocurrency mining and its defense," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Toronto, Canada, October 2018.

[10] A. Gangwal, "Detecting covert cryptomining using hpc," in *Proceedings of the International Conference on Cryptology and Network Security*, December 2020.

[11] R. Tahir, "Mining on someone else's dime: mitigating covert mining operations in clouds and enterprises," in *Proceedings of the International Symposium on Research in Attacks, Intrusions, and Defenses*, Springer, Cham, Switzerland, October 2017.

[12] J. A. K. Suykens, J. Vandewalle, and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[13] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121, no. 2, pp. 256–285, 1995.

[14] S. Wold, E. Kim, and G. Paul, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[15] H. Ma, "Dark web traffic detection method based on deep learning," in *Proceedings of the 2021 IEEE 10th Data Driven Control and Learning Systems Conference (DDCLS)*, May 2021.

[16] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, July 2015.

[17] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and Its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[18] R. A. Maxion, "Anomaly detection for diagnosis," in *Proceedings of the Digest of Papers. Fault-Tolerant Computing: 20th International Symposium*, IEEE Computer Society, Tyne, UK, June 1990.

[19] Lashkari and A. Habibi, "Characterization of tor traffic using time based features," *ICISSp*, vol. 3, 2017.

[20] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[21] A. Orebaugh, G. Ramirez, and J. Beale, *Wireshark & Ethereal Network Protocol Analyzer Toolkit*, Elsevier, Amsterdam, Netherlands, 2006.