Hindawi
Security and Communication Networks
Volume 2021, Article ID 6658066, 10 pages
https://doi.org/10.1155/2021/6658066

Research Article

Cache Pollution Detection Method Based on GBDT in Information-Centric Network

Dapeng Man, Yongjia Mu, Jiafei Guo, Wu Yang, Jiguang Lv, and Wei Wang

Information Security Research Center, Harbin Engineering University, Harbin 150001, China

Correspondence should be addressed to Wei Wang; w_wei@hrbeu.edu.cn

Received 25 December 2020; Accepted 7 June 2021; Published 16 June 2021

Academic Editor: M.A. Jabbar

There is a new cache pollution attack in the information-centric network (ICN), which fills the router cache by sending a large number of requests for nonpopular content. This attack will severely reduce the router cache hit rate. Therefore, the detection of cache pollution attacks is also an urgent problem in the current information center network. In the existing research on the problem of cache pollution detection, most of the methods of manually setting the threshold are used for cache pollution detection. The accuracy of the detection result depends on the threshold setting, and the adaptability to different network environments is weak. In order to improve the accuracy of cache pollution detection and adaptability to different network environments, this paper proposes a detection algorithm based on gradient boost decision tree (GBDT), which can obtain cache pollution detection through model learning. *Method.* In feature selection, the algorithm uses two features based on node status and path information as model input, which improves the accuracy of the method. This paper proves the improvement of the detection accuracy of this method through comparative experiments.

1. Introduction

With the popularity of the Internet and Internet of Things technologies and the transformation of IPV4 and IPV6 technology, more and more smart devices can access the Internet, and network traffic is beginning to grow rapidly. The current application range and scale of the Internet have far exceeded the original intention of the design. The information-centric network [1–4] was proposed, hereinafter referred to as ICN. The current information-centric network has protocols such as NDN [5], PSIRP [6], DONA [7], and NetInf [8]. Although these protocols have different forms, the key point is to use content names or IDs to obtain content, and all routers that pass through are supported for caching. Among the many information-centric network implementations, the most mainstream feasible solution is the named data network, hereinafter referred to as NDN. All the solutions discussed in this paper are based on NDN.

Since the original intention of the information-centric network design includes the use of caching to increase network utilization, the cache is an indispensable part of the information-centric network. If there is no cache, the efficiency of the network will be significantly reduced. In IP-based networks, there are various kinds of network attacks, and one of the well-known attacks is the DDoS attack [9, 10]. Unlike the IP network, the principal part of the information-centric network is content rather than IP. The attacker cannot specify a certain packet to send to the target host. Therefore, the information-centric network has inherently capable resistance to such attacks. However, due to the massive use of cache in the information-centric network to improve network efficiency, it naturally brings the cache pollution attack. The attacker can send many nonpopular content requests through the controlled host so that the routers on the path cache the nonpopular content. When the normal user makes a request, the cache hits failed because the node cache cannot find the corresponding content, the router only forwards the request to the content producer for processing, which makes the original intention of the information-centric network design, using the cache to optimize the network message to the maximum extent becomes useless, so that the traffic of the backbone link of the network

is greatly increased, resulting in network congestion and other phenomena.

Although ICN has rethought some of the design concepts of optimization and innovation, in many respects, some central issues have not been completely resolved in the initial ICN network framework. This paper mainly discusses the problem of cache pollution detection. Cache pollution attack is one of the most serious attacks in information-centric networks. Most of the current detection algorithms need to manually set thresholds. These methods have poor adaptabilities to different environments. Therefore, in this paper, a GBDT-based cache pollution detection method is proposed, which does not require a manual setting of thresholds and has high accuracy.

1.1. Related Work. NDN is an information-centric network architecture with high scientific research value and development potential [5, 11], which has attracted great attention from the academic community in recent years. Although the native architecture of NDN tries to provide certain data security by encrypting and signing data packets by network content producers, in the face of various malicious attacks under complex environmental conditions in the actual network, its network nodes still have great potential risks, for example, Distributed Denial-of-Service (DDoS) attacks and cache pollution. Research by Virgilio et al. [12] shows that DDoS attacks can use a large number of forged interests to exhaust the memory of the pending interest table (PIT) in the NDN node. Tan Nguyen et al. calculated the difference in the satisfaction rate of interest packets caused by DDoS to detect and defend DDoS [13], but the effect of this solution is not ideal in the face of network cache pollution attacks.

Cache pollution attacks [14] have been widely studied in IP-based networks. More research studies focus on the cache of web traffic. The current research divides cache pollution attacks into two categories, destroying the content distribution (locality-disruption) characteristics attack and the false-locality feature attack [15]. In the destruction of the content distribution model, the attacker or the controlled host sends many interest packets of nonpopular content to the network. Therefore, the router' CS table in the network caches is occupied by a large amount of nonpopular content, so that the requests of the normal popular contents cannot utilize the CS table of the router, increasing the network delay and achieving the purpose of the attack; in the forged content distribution attack, the attacker's attack model, and the broken content distribution differently, the attack does not destroy the overall distribution feature of the content in the network, but periodically sends a request for non-streaming content, so that the cache is occupied by non-popular contents for a long time, resulting in a decrease in network cache capability.

Although cache pollution attacks have been extensively studied in IP networks, most of the detection methods for cache pollution in IP networks cannot be applied to ICN networks. This is because in ICN networks, the request packet does not contain any information of the requester, so it cannot be the source of the request packet which is traced, and the attack object of the cache pollution in the ICN is also different from that in the IP network. The cache pollution attack in the IP network is mostly directed to the cache server, and the target of the cache pollution attack in the ICN is the routing node in the network. This also shows that, in ICN cache pollution attacks, cache routes are difficult to perceive the existence of attacks [16, 17]. As an information-centric network architecture with network cache, NDN is vulnerable to cache poisoning and pollution attacks.

Mauri et al. consider a case in which an attacker uses NDN's routing and caching system to add a large amount of malicious content to the cache storage of network nodes [18]. Literature [19] analyzed the essential characteristics of content pollution attacks in NDN networks based on some cases. Li et al. proposed a lightweight integrity verification and access control mechanism for network cache pollution attacks [20]. Literature [21] studied local cache pollution attacks and proposed a cache shield, which enhances the robustness of the network by increasing the cost of cache pollution attacks.

Park et al. proposed a matrix random check method [22], which uses the content name. The method is mapped to the matrix. Each position of the matrix represents the number of corresponding requests. Whenever a request arrives, the rank of the matrix is checked. When the rank of the matrix is below a certain threshold, this node is considered to be attacked. Conti et al. proposed a lightweight mechanism [23] for detecting cache pollution attacks, which first defines a random sample set of content, monitors the distribution of current sample sets, and dynamically counts the arrival rates of these requests, once these cache pollutions have occurred when the arrival rate changes and exceeds a certain threshold.

2. Analytical Methods

Caching is one of the reasons why information-centric networks are efficient. This chapter will discuss the cache pollution detection problem in the information-centric network and study how to use the machine learning method to solve the problem that the traditional detection model needs to manually set the threshold. Then, a GBDT-based cache pollution detection model is proposed, which is shown in Figure 1. First, the node status information and path information are collected as features, and then the gradient boosting tree algorithm is combined with the two features to build a cache pollution detection model. The training speed of this model is fast, and the accuracy is high. Finally, we evaluate the model through experiments.

2.1. GBDT Model. The cache pollution detection model is essentially a classifier and is a two-class classifier. One is that the current node is being attacked, and the other is that the current node is not attacked. This chapter uses the GBDT model for cache pollution detection. GBDT is the abbreviation of gradient boost decision tree, which is the gradient lifting tree. This model is practically a gradient promotion model in the decision tree, that is, multiple decision trees are

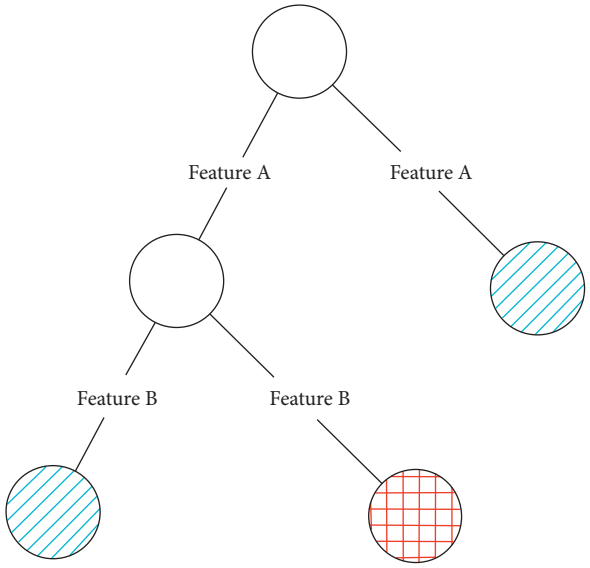

Figure 1: Schematic diagram of the decision tree.

fused according to the gradient promotion method. This section will introduce the model-related content .

(1) The decision tree is divided into a classification tree and a regression tree. The classification tree refers to the decision tree used for classification problems. This chapter uses the GBDT model to classify the cache pollution problem. Therefore, only the classification tree is introduced here. The classification tree is a kind of classification model using the tree structure. Each leaf node of the tree represents the classification result. Each branch of the tree is a decision mode. As shown in Figure 2, according to a set of new features A and B and the trunk of each tree according to the feature selection, the child node of the tree is entered. If the child node is a leaf node, the classification result is obtained. Therefore, for any training set, if there is no data with the same characteristics but different results, the decision tree can obtain 100% accuracy on the training set. This is because the depth of the tree can grow all the time, and the decision algorithm will eventually get the result after judging all the attributes. Although the decision tree can get almost 100% accuracy on the training set, the test set may get very poor results. This is because the decision tree is just remembered all the training data, but did not learn how to judge this classification problem which the data that does not appear in the training set. Thus, it will produce more random results. This phenomenon is also called overfitting problem.

For the decision tree, its training process aims to determine the characteristics of each branch, that is, finding the optimal feature in each node state. The usual method is to use information gain and information gain ratio and Gini index.

The formula for information entropy is defined as follows:

$$H(U) = E[-\log\ p_i] = -\sum_{i=1}^{n} p_i \log\ p_i. \tag{1}$$

To facilitate the operation, the log here takes the base 2 logarithm. From the formula, the value range of $H(U)$ is [0, 1], and the amount of information entropy quantitatively identifies the uncertainty of the information. A larger value indicates a stronger uncertainty, that is, less information is included. For example, a box has red and white balls. If there are no restrictions, you can only assume that the probability distribution is 1/2 and 1/2. Then, the entropy now is 1, which is the maximum value. This indicates that the information of the situation is the least, and if we know that there is only a white ball in this box, the probability distribution becomes 0 and 1. In this event, the information entropy is 0, which means that the amount of information is the largest currently, without any uncertainty.

The ID3 algorithm uses the information gain method, that is, the change of the information entropy before and after the split is used to measure the optimal attribute, and the information gain obtained by dividing the D state by the feature A is defined as equation (2), that is, each iteration selects the largest information gain value to produce subsets of the data.

$$\text{Gain}(D, A) = \text{Entropy}(D) - \text{Entropy}(D, A). \tag{2}$$

However, the ID3 algorithm has a significant limitation. The ID3 algorithm will choose the minimal Entropy(D, A), which will lead to the algorithm tending to select those features with more subclasses and purer features; therefore, the C4.5 algorithm is proposed to solve the ID3' s drawback. The C4.5 algorithm uses the information gain rate as the measure of the best feature. The optimal gain rate Gain_Rate(D, A) is defined as equation (3), that is, the attribute with the maximum normalized information gain is chosen to make the decision.

$$\text{Gain\_Rate}(D, A) = \frac{\text{Entropy}(D)}{\text{Entropy}(D, A)}. \tag{3}$$

The CART tree can be used for both classification and regression problems. The CART tree uses the Gini index or Gini impurity to determine the optimal division point. The Gini index is defined as equation (4), and the Gini index can also represent the uncertainty of the sample set S. Since the cart is a binary decision tree, each partition can only divide the set into two parts, so each partition needs to use the i th attribute value of attribute A, as shown in equation (5). $\text{Gain}_{A,i}(S)$ represents the uncertainty of set s after A_i segmentation. The larger the Gini index, the greater the uncertainty of the sample set S after A_i division, which is like entropy.

$$\mathrm{GINI}(S) = 1 - \sum P_k^2, \tag{4}$$

$$\mathrm{Gain}_{A,i}(S) = \frac{n1}{N}\mathrm{GINI}(S_1) + \frac{n2}{N}\mathrm{GINI}(S_2). \tag{5}$$

(2) The boosting algorithm [24] is an important part of ensemble learning. Boosting can be used to primarily reduce the bias of the model and enhance the weak models. The principle of the boosting algorithm is to use a weak model to fuse the strong model. First, weak classifier α is trained with the initial weights. The data weight of the training set is updated according to the prediction results of the weak classifier α so that the weight of the sample points whose prediction is error was made by the weak classifier α is increased. Therefore, these samples with a high error rate can get higher accuracy in later learning with weak learner β. Then, we use the weighted training set to train weak classifier β, which is repeated until the number of weak classifiers reaches the number T appointed beforehand. Finally, these T weak classifiers are assembled through a set strategy to obtain the final strong classifier.

(3) GBDT is one of the ensembles learning boosting algorithm. GBDT is also an iterative model that uses a forward distribution algorithm, but the weak learner limits the use of the CART regression tree model.

For the cache pollution problem, binary classification is needed. The log loss function can be used, and the loss function is shown as follows:

$$L(y, f(x)) = \log\left(1 + \exp(-yf(x))\right). \quad (y \in (-1, +1)) \tag{6}$$

The negative gradient of the loss function of the ith sample of the tth round is expressed as follows:

$$r_{\mathrm{ti}} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{t-1}(x)}. \tag{7}$$

Here, the loss function of the cache pollution problem is brought in, and the negative gradient error at this time is as follows:

$$r_{ti} = \frac{y_i}{\left(1 + \exp\left(y_i f(x_i)\right)\right)}. \tag{8}$$

Using (x_i, r_{ti}) $(i = 1, 2, \ldots, m)$, we can fit a CART regression tree and get the tth regression tree, and its corresponded leaf node region R_{tj}, $j = 1, 2, \ldots, J$, where J is the number of leaf nodes.

For each sample in the leaf node, the loss function is minimized, and the best output value c_{tj} of the fitting leaf node is as follows.

$$c_{tj} = \underbrace{\arg\min}_{c} \sum_{x_i \in R_{tj}} L\left(y_i, f_{t-1}(x_i) + c\right). \tag{9}$$

For the problems provided in this chapter, the loss function of the cache pollution problem is brought in, and the optimal residual fit value of each leaf node is as shown in equation (10).

$$c_{tj} = \underbrace{\arg\min}_{c} \sum_{x_i \in R_{tj}} \log\left(1 + \exp\left(-y_i\left(f_{t-1}(x_i) + c\right)\right)\right). \tag{10}$$

Since the above formula is more difficult to optimize, we use an approximation instead, as follows:

$$c_{tj} = \frac{\sum_{x_i \in R_{tj}} r_{tj}}{\sum_{x_i \in R_{tj}} \left|r_{tj}\right|\left(1 - \left|r_{tj}\right|\right)}. \tag{11}$$

Thus, the fitting function for each iteration is obtained as follows:

$$h_t(x) = \sum_{j=1}^{J} c_{tj} I, \quad \left(x \in R_{tj}\right). \tag{12}$$

Finally, the resulting strong learner expression is given as follows:

$$f_t(x) = f_{t-1}(x) + \sum_{j=1}^{J} c_{tj} I, \quad \left(x \in R_{tj}\right). \tag{13}$$

2.1.1. Node Status Information. In the NDN-based network and the IP-based network, the cache pollution attack has similarities. In both networks, the attacker attempts to attack a terminal. In the IP-based network, the terminal refers to some servers. In the network of the NDN architecture, this terminal is a certain router, so this kind of attack is an attack that only the attacked node can judge.

In the NDN, the most intuitive reflection of the attack that occurred is the cache hit rate of the normal request, but the intermediate router responsible for forwarding and caching cannot distinguish the difference between the normal request interest packet and the attack interest packet, so the data cannot be directly or indirectly obtained through the router. It is only possible to estimate whether an attack has occurred by some available quantities. The variable quantities available in the NDN router are shown in Table 1.

Since the NDN is designed to follow the "thin waist" principle, it has less available information for routers. Firstly, the cache pollution attack is implemented by sending abundant nonpopular interest packets to the network. Therefore, the correlation amount of the data packet does not have much significance and is not a feature. Secondly, for the attack detection model, some overall quantities have no meaning of the model detection, such as the total number of interest packets and the total cache hit rate, so these quantities are not suitable as model parameters. In addition, some ID-type quantities such as the name of the interest package and the cached interest package name are substantially independent of the cache attack. Thus, such ID-type variables should not be used as the features of the model. Existing research has shown that, for interest packet requests in routers, the Zip-f distribution is normally satisfied, that is, the most frequent requests are only a small part

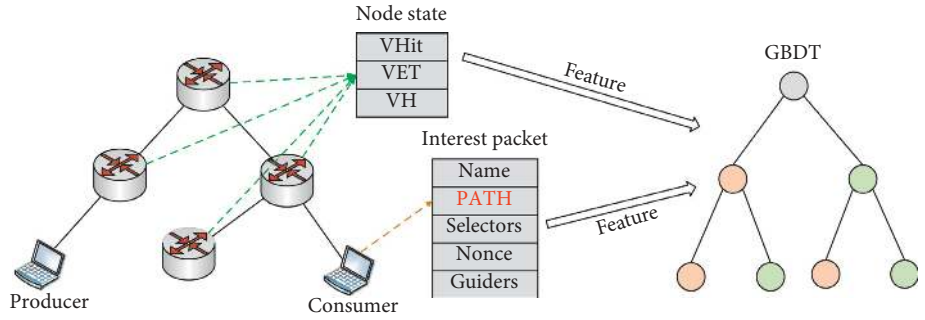

Figure 2: Schematic diagram of GBDT cache contamination detection.

Table 1: NDN router availability.

Available in NDN routers	Description
Interest package content name, IName	Obtained from the content name field of the interest package
The content name of the packet, DName	Obtained from the content name field of the data packet
Cached content name, CName	Get the content name from the CS table
Cache replacement rate, pmiss	Replacement rate by calculation
Cache hit ratio, EH	Cache hit rate by calculation
Number of interest packages, TI	Accumulate the number of interest packages
Number of data packets, TD	Accumulate the number of data packets

of all the data [25]. Therefore, when retrieving features, we should consider extracting features that can reflect the distribution of content. Considering that the number of interest packets per unit time can reflect the distribution of content, the number of interest packets with the largest number of first K requests per unit time is used to form a K-dimension feature to enable the model to learn the current distribution characteristics and then select the K-cache hit rate of corresponding content as the feature.

For the above features, the number of interest packets may greatly depend on the usage of the network. For example, the total number of interest packets in the high-interval and low-peak networks varies greatly, but the difference does not mean whether it is attacked. Therefore, if you directly select the number of interest packages as a feature, the model may be too dependent on the number of packets in the network. Therefore, it is necessary to normalize the number of interest packets, not using the quantity, using the proportion as a feature, the normalization formula is as shown in equation (14), and cnt_k indicates the number of K interest packages with the largest number of units of interest per unit time, total represents the total number of interest packets per unit time, and the characteristics of the final selection node are shown in Table 2. In Table 2, VHit is cache replacement rate under cache replacement policy which can be obtained by the source code of ndnSIM because we use the built-in cache policy, that is, the LRU policy. And, VH is the K cache hit ratios corresponding to interest packages also can be obtained by the source of ndnSIM.

$$\text{VEI} = \frac{\text{cnt}_k}{\text{total}}. \tag{14}$$

2.1.2. Path Information Feature. In the NDN network, in addition to the feature based on the state of the node, path-based information can be extracted as an aid. To save the path information, a path field needs to be added to the interest packet. This section proposes a hash-based path feature extraction algorithm. The algorithm only uses a few assembly instructions in the operation, almost no reduction in the speed of the original router processing packets. In the memory footprint, the algorithm only needs to add an integer variable in the interest package, and memory also hardly affects network bandwidth.

The algorithm needs to select a random integer as the ID of the router when each NDN router starts, and the path field of the consumer sending interest packet is 0, that is, the content consumer does not participate in the maintenance process of the entire path, if the attacker attempts to change this field to forge the path information, the first hop router can also judge the attacker's attack based on the value being nonzero. The algorithm of the router is as follows:

$$\text{PATH}_{i+1} = \text{PATH}_i \text{ xor } \text{ID}_{i+1}. \tag{15}$$

When the NDN router receives an interest packet, the value of PATH is updated by using equation (15), where PATH_{i+1} represents the PATH value in the interest packet forwarded by the $(i+1)$ th router, ID_{i+1} represents the ID of the $(i+1)$ th router, and xor is the exclusive-or operation. This kind of replacement or filling produces only one assembly code per forwarding time, so it hardly affects the delivery rate of interest packets.

The above PATH value can be approximated to represent the path of the interest packet to a certain terminal, and the definition $\text{Unique}(C)$ indicates the number of different PATH values in the interest packet requesting the content C in the current terminal.$\text{Cnt}(c)$ is defined to represent the number of interest packets that request content C in the current terminal. Obviously, in the case of no cache pollution, there is a positive correlation between the number of interest packets $\text{Cnt}(C)$ and $\text{Unique}(C)$. Therefore,

Table 2: Node characteristics of the model.

Selected features	Description
Cache hit ratio VHit	Cache replacement rate under cache replacement policy
Interest package ratio vector VEI	Proportion of K interest packages with the highest proportion of interest packages
Cache hit rate vector	K cache hit ratios corresponding to interest packages

Unique(C) cannot be directly used as a feature, and Unique(C) needs to be normalized to define the diversification ratio CP(C) of content C as in equation (16). The diversification ratio can reflect the richness of the source of certain content C to some extent, and the diversification ratio can be known according to the definition whose range is between 0 and 1. The smaller the value, the more singular the source of the interest packet is, the more likely it is the attack. The feature has a negative correlation with the cache attack; therefore, the feature can increase the accuracy of the model.

$$\mathrm{CP}(C) = \frac{\mathrm{Unique}(C)}{\mathrm{Cnt}(C)}. \tag{16}$$

It can be known from equation (17) that, to calculate the diversification ratio, CP(C) of content C needs to calculate Cnt(C) and Unique(C), both of which are statistical values. Cnt(C) is the number of interest packets per unit time which requires a numeric variable, and Unique(C) is the number of different kinds of PATH. For an NDN network, considering the network traffic, the interest packets will not be stored, so the hash is required to statistic the values above. Hash the PATH, and, in addition, use the bitmap to reduce the memory usage. Using one bit to represent whether the current PATH has appeared or not; if so, set that bit to 1. Calculate the number of bits with the value of 1 in unit time, which can be approximately considered as the number of different kinds of PATH in network.

3. Experimental Results and Analysis

3.1. Experimental Environment. The experimental environment of this paper is shown in Table 3:

Table 3: Experimental configuration.

Configuration name	Configuration parameter
CPU	Intel I5-8265U
RAM	8G
Operating system	Ubuntu 16.04
Python version	Python 3.6
Python toolkit	Anaconda

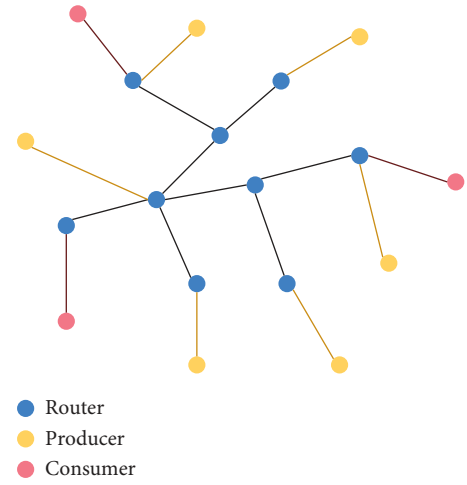

Figure 3: Network topology.

3.2. Experimental Program

3.2.1. ndnSIM Simulation. ndnSIM is an NDN simulation platform developed based on the NS-3 network simulator, which can simulate diversified NDN scenarios [26]. This article changes the source code of the interest package structure in ndnSIM, adds the PATH variable, randomly assigns an ID to each NDN router, and adds related operations to the PATH variable in the routing and forwarding process according to the description above in this chapter. This chapter conducts simulation experiments on known complex topologies. The experimental network topology is shown in Figure 3. In each experiment, the attacker randomly selected the host as the controlled host, and the controlled host sends a great quantity of nonpopular requests.

According to the current researches, most researchers believe that the request in the information-centric network should obey the Zip-f distribution. Therefore, the request in the simulation experiment network should follow the Zip-f distribution, and the normal request distribution's parameter should set $a = 1.2$, and the request rate is 1000/s. In the experiment, the cache policy of the NDN router adopts the LRU policy. The experiment builds the environment through ndnSIM [24], and the statistics of the experimental related data including the number of arrivals of the interest packets are performed by modifying the source code. In order to obtain the training data of GBDT, respectively, simulate the network when there is no attack, and when there is an attack, the attacker sends a large number of nonpopular interest packets to simulate the occurrence of the attack, and statistics when there is an attack and the statistics when there is no attack are recorded and saved separately, and the data are divided into a training set and a test set for multiple experiments. The training set and test set data selection in each experiment are shown in Table 4.

3.2.2. Model Training. This article uses Python's lightGBM library to build the GBDT model. lightGBM is Microsoft's boosting framework, which has faster training efficiency, lower memory usage, and higher accuracy than xgboost [27], and supports parallel learning. In this experiment, the GBDT model is used for training on 10,000 sets of data, and the test

Table 4: Training set and test set data selection.

Dataset category	Positive sample	Negative sample
Training data set	5000	5000
Test data set	1000	1000

is performed on 2,000 sets of data to analyze the accuracy of the model.

When training the model, in order to prevent overfitting of the model, the maximum depth of the decision tree and the maximum number of leaf nodes in the GBDT model should be set, and the regularization parameters should be set. Besides, for the number of iterations, the fast stop strategy is selected, and the training data is divided into two parts: one as the training set and one as the evaluation set (to distinguish it from the test set, here called the evaluation set), used to make a quick stop. The training set and the evaluation set are disjoint sets. In the experiment, their ratio is 4 : 1. The loss function on the evaluation set is calculated in each iteration. When the performance on the evaluation set will not improve anymore (that is, the loss function does not change anymore), stop training and the model loss function uses the Log loss function. When training the GBDT model, some of the parameter settings for using lightGBM are shown in Table 5.

3.2.3. Evaluating Indicator. The validity of the method is measured by the precision, recall rate, accuracy, and F-measure. It defines the following concepts:

Cache pollution is classified as cache pollution: TP (true positive)

Cache uncontaminated is classified as cache uncontaminated: TN (true negative)

Cache uncontaminated is classified as cache pollution: FP (false positive)

Cache pollution is classified as cache uncontaminated: FN (false negative).

Precision (Pre), which is the actual proportion of the sample classified as cache pollution, is calculated as equation (17).

Accuracy (abbreviation: Acc) is the ratio of the correctly classified samples to all samples and measures the overall correctness of the model. The formula is given by equation (18).

The recall rate (recall) indicates how many positive samples are correctly classified, and the formula is calculated as in equation (19).

F-measure is the harmonic mean of the accuracy rate and the recall rate, and the formula is calculated as in equation (20).

Table 5: LightGBM-related parameter settings.

Parameter	Value	Description
Objective	Binary	Two classification problem
Num_leaves	50	Up to 50 leaf nodes per tree
max_depth	8	The maximum depth of the tree is 8
learning_rate	0.01	Learning rate is 0.01
reg_alpha	0.005	$L1$ regular weight is 0.005
reg_lambda	0.0005	$L2$ regular weight 0.0005
Subsample	0.9	Selective feature probability
n_estimators	4000	Maximum number of iterations
early_stopping_rounds	200	Early stopping rounds

$$\text{Pre} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{17}$$

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{18}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{19}$$

$$F1 = \frac{2 \times \text{Pre} \times \text{Recall}}{\text{Pre} + \text{Recall}}. \tag{20}$$

In this paper, the training time of the model is used as the main indicator of model performance evaluation. The relationship between the number of GDBT iterations and time when the experiment counts 10,000 sets of data.

4. Analysis of Results

For the GBDT model proposed in this section, two types of features are used, node status and path information. Since normalization is used, all values are in the range [0, 1], and for the NDN cache pollution attack, the attacker's attack strength will be numerically strong and weak. The characteristics of different attack strengths also change within a certain range. Therefore, the final decision model should be a range model. This property is like the decision tree. GBDT is currently a very good model for improving the decision tree, so the model is adopted. The experiment also proves that the model can achieve good detection results.

Figure 4 shows the relationship between the loss function and the number of iterations when using the parameters described in Section 3 on 10,000 sets of data. It can be seen from the figure that, as the number of iterations increases, the performance of the training set becomes better. However, the performance of the evaluation set is not getting better anymore, and there is a trend of deterioration. If the number of iterations continues to increase, there will be overfitting. In the current model parameters, the loss function of the training set and the evaluation set is better at 736 iterations. At this time, the loss function on the evaluation set is 0.0386, the loss function on the training set is

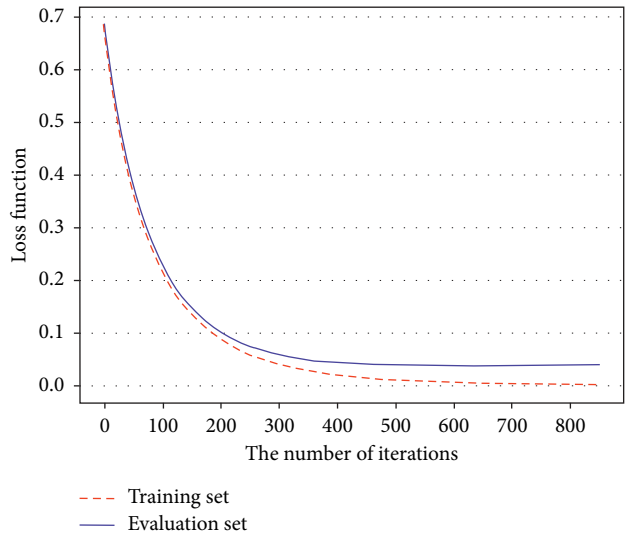

Figure 4: Diagram of the number of iterations and the loss functions.

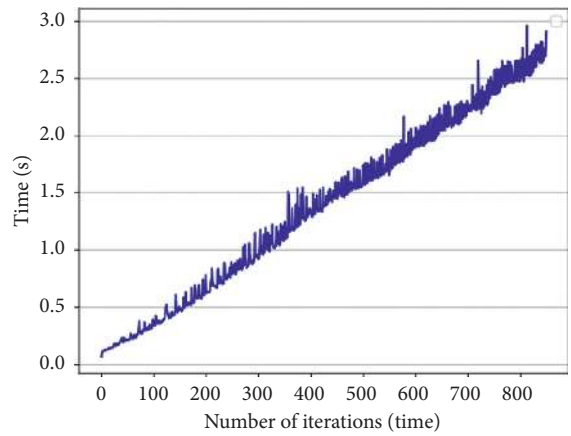

Figure 5: GBDT iteration times and time diagram.

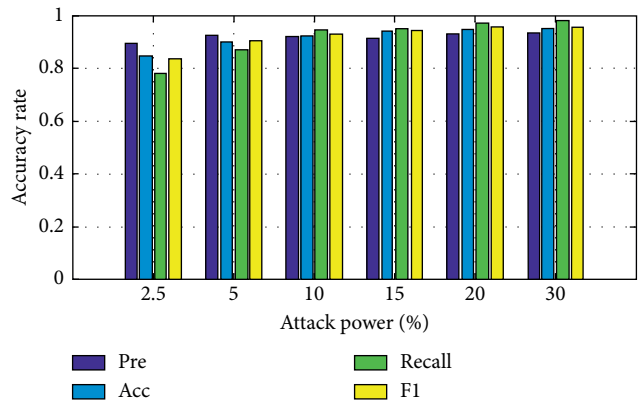

Figure 6: Relationship between attack strength and correctness.

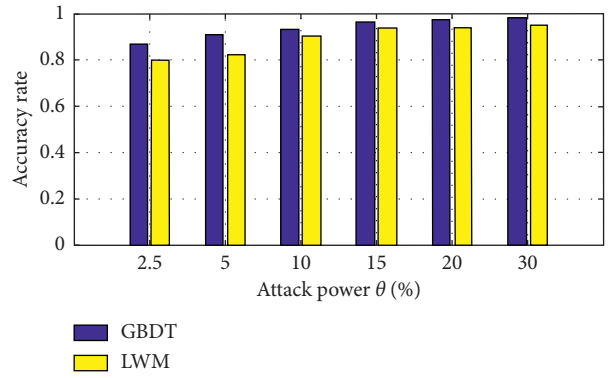

Figure 7: Attack strength and detection accuracy.

0.0029175, and the loss on the test set is 0.015377. Therefore, under the current feature, the model parameters of 736 iterations should be taken.

As can be seen from Figure 5, using the lightGBM for the training of the GBDT model, the training is very fast in the case of 10000 sets of data. When the iteration is about 300 times, the time is still less than 1 second. In the simulation experiment, it is best to iterate. It took only about 2 seconds, which means that the GBDT model training for lightGBM is very fast.

The attack strength θ is defined as the proportion of attack packets in the request packet. The stronger the attack strength is, the greater the impact on the state of the network node is. The accuracy of the model has a certain relationship with the attack strength. Therefore, the simulation is as follows. In the experiment, the relationship between attack intensity and detection accuracy was analyzed.

As shown in Figure 6, when the attack intensity of network cache contamination is lower than 15%, with the continuous increase of the attack intensity, the accuracy rate PRE, accuracy rate Acc, and recall rate of GBDT model detection are all continuously improved. When the attack intensity exceeds 20%, the model can distinguish the attack more clearly, and the above indicators tend to be stable, with the accuracy rate up to 93%, accuracy rate up to 95%, and recall rate up to 97%. This is because with the increase of attack intensity, the cache hit ratio of nodes in the network and the proportion distribution of interest packets will also be affected more and more, which makes it easier for the model to detect attacks.

Figure 7 shows the comparison result of the detection accuracy with the light weight mechanism method proposed in [17]. The traditional LWM method needs to set a threshold, which affects the detection accuracy of the model. The GBDT square model uses the current mainstream machine learning method to learn the judgment standard, so there is no need to set the threshold. It can be seen from the numerical values that the detection accuracy of the GBDT model is higher than that of the LWM method under various attack intensities, and the detection accuracy of the GBDT model can reach more than 85% under the attack intensity of 2.5%, which is better than that of the LWM. The detection accuracy of the method is 5% higher, so it can indicate that the model has a fairly strong cache pollution perception ability.

5. Conclusion

In this paper, a new detection method based on machine learning is proposed for the current problem of cache pollution in the information-centric network. Firstly, the node state and the path information are used as the feature. The node information is obtained through statistics, and the path information is hashed to achieve the diversification rate. In the interest packet delivery process, only one field and one assembly instruction need to be added, so completely it does not affect the forwarding efficiency of normal routes. Then, the two features are combined to use the gradient lifting tree algorithm to build a cache pollution detection model. Compared with the current detection methods that mostly need to set thresholds, the method has higher accuracy and adaptability to different networks. Finally, the model is implemented by ndnSIM and lightGBM, and the advantages of the method in detection accuracy are demonstrated compared with other detection methods.

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

This research was supported by the National Natural Science Foundation of China (grant nos. 61771153, 61831007, and 61971154) and the Fundamental Research Funds of the Central Universities (grant no. 3072020CF0601).

References

[1] M. Amadeo, C. Campolo, J. Quevedo et al., "Information-centric networking for the internet of things: challenges and opportunities," *IEEE Network*, vol. 30, no. 2, pp. 92–100, 2016.

[2] G. Xylomenos, C. N. Ververidis, V. A. Siris et al., "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.

[3] R. Tourani, S. Misra, T. Mick, and G. Panwar, "Security, privacy, and access control in information-centric networking: a survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 566–600, 2018.

[4] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pp. 1–12, Rome, Italy, December 2009.

[6] S. Tarkoma, M. Ain, and K. Visala, "The publish/subscribe internet routing paradigm (PSIRP): designing the future internet architecture," in *Future Internet Assembly*, pp. 102–111, IOS press, Amsterdam, The Netherlands, 2009.

[7] T. Koponen, M. Chawla, B. Chun et al., "A data-oriented (and beyond) network architecture," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 181–192, Kyoto, Japan, August 2007.

[8] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of Information (NetInf) - an information-centric networking architecture," *Computer Communications*, vol. 36, no. 7, pp. 721–735, 2013.

[9] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[10] L. Zhu, X. Tang, M. Shen, X. Du, and M. Guizani, "Privacy-preserving DDoS attack detection using cross-domain traffic in software defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 628–643, 2018.

[11] L. Zhang, A. Afanasyev, J. Burke et al., "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[12] M. Virgilio, G. Marchetto, and R. Sisto, "PIT overload analysis in content centric networks," in *Proceedings of the 3rd ACM SIGCOMM Workshop Information-Centric Network (ICN)*, pp. 67–72, San Jose, CA, USA, August 2013.

[13] T. Nguyen, H.-L. Mai, R. Cogranne et al., "Reliable detection of interest flooding attack in real deployment of named data networking," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2470–2485, 2019.

[14] H. Park, I. Widjaja, and H. Lee, "Detection of cache pollution attacks using randomness checks," in *Proceedings of the 2012 IEEE International Conference on Communications (ICC)*, pp. 1096–1100, Ottawa, ON, Canada, June 2012.

[15] L. Deng, Y. Gao, Y. Chen, and A. Kuzmanovic, "Pollution attacks and defenses for internet caching systems," *Computer Networks*, vol. 52, no. 5, pp. 935–956, 2008.

[16] M. Li, Y. Sun, H. Lu, S. Maharjan, and Z. Tian, "Deep reinforcement learning for partially observable data poisoning attack in crowdsensing systems," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6266–6278, 2020.

[17] L. Yao, Z. Fan, J. Deng, X. Fan, and G. Wu, "Detection and defense of cache pollution attacks using clustering in named data networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 6, pp. 1310–1321, 2020.

[18] G. Mauri, R. Raspadori, M. Gerla, and G. Verticale, "Exploiting information centric networking to build an attacker-controlled content delivery network," in *Proceedings of the 14th IFIP Annual Mediterranean Ad Hoc Networking Workshop*, pp. 1–6, Vilamoura, Portugal, June 2015.

[19] C. Ghali, G. Tsudik, and E. Uzun, "Network-layer trust in named-data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 12–19, 2014.

[20] Q. Li, X. Zhang, Q. Zheng, R. Sandhu, and X. Fu, "LIVE: live: lightweight integrity verification and content access control for named data networking," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 308–320, 2015.

[21] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," in *Proceedings of IEEE INFOCOM*, pp. 2426–2434, Orlando, FL, USA, March 2012.

[22] Z. Yi, S. Jia, G. Pu et al., "Collaborative detection mechanism for low-rate cache pollution attack in named data networking," *Journal of Beijing University of Posts & Telecommunications*, vol. 38, pp. 44–48, 2015.

[23] M. Conti, P. Gasti, and M. Teoli, "A lightweight mechanism for detection of cache pollution attacks in Named Data Networking," *Computer Networks*, vol. 57, no. 16, pp. 3178–3191, 2013.

[24] G. Ridgeway, "The state of boosting," *Computing Science and Statistics*, pp. 172–181, Springer, Berlin, Germany, 1999.

[25] L. Breslau, C. Pei, F. Li, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *Proceedings of the IEEE INFOCOM'99. Conference on Computer Communications. Proceedings Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, vol. 1, pp. 126–134, New York, NY, USA, March 1999.

[26] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the evolution of ndnSIM," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 19–33, 2017.

[27] T. Chen and C. Guestrin, "Xgboost: a scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, San Francisco, CA, USA, August 2016.