IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON SECURITY AND TRUSTED COMPUTING FOR INDUSTRIAL INTERNET OF THINGS

Received April 20, 2018, accepted May 24, 2018, date of publication June 4, 2018, date of current version June 26, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2843336*

# An Integrated Method for Anomaly Detection From Massive System Logs

**ZHAOLI LIU[1], TAO QIN[1,2], (Member, IEEE), XIAOHONG GUAN[1,2], (Fellow, IEEE), HEZHI JIANG[1], AND CHENXU WANG[1]**

[1]Key Laboratory for Intelligent Networks and Network Security of the Ministry of Education, Xi'an Jiaotong University, Xi'an 710049, China
[2]Shenzhen Research School, Xi'an Jiaotong University, Shenzhen 518057, China

Corresponding author: Tao Qin (qin.tao@mail.xjtu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772411, Grant 61672026, Grant 61602370, and Grant U1736205, in part by SZSTI under Project JCYJ20170816100819428, and in part by the Fundamental Research Funds for the Central University. The work of Z. Liu was supported by the China Scholarship Council under Grant 201706280180.

**ABSTRACT** Logs are generated by systems to record the detailed runtime information about system operations, and log analysis plays an important role in anomaly detection at the host or network level. Most existing detection methods require a priori knowledge, which cannot be used to detect the new or unknown anomalies. Moreover, the growing volume of logs poses new challenges to anomaly detection. In this paper, we propose an integrated method using K-prototype clustering and k-NN classification algorithms, which uses a novel clustering-filtering-refinement framework to perform anomaly detection from massive logs. First, we analyze the characteristics of system logs and extract 10 features based on the session information to characterize user behaviors effectively. Second, based on these extracted features, the K-prototype clustering algorithm is applied to partition the data set into different clusters. Then, the obvious normal events which usually present as highly coherent clusters are filtered out, and the others are regarded as anomaly candidates for further analysis. Finally, we design two new distance-based features to measure the local and global anomaly degrees for these anomaly candidates. Based on these two new features, we apply the k-NN classifier to generate accurate detection results. To verify the integrated method, we constructed a log collection and anomaly detection platform in the campus network center of Xi'an Jiaotong University. The experimental results based on the data sets collected from the platform show our method has high detection accuracy and low computational complexity.

**INDEX TERMS** Anomaly detection, clustering-filtering-refinement, K-prototype clustering, k-NN classification, massive logs.

## I. INTRODUCTION

Logs are generated by systems to record the detailed runtime information about system operations, and they are one of the most important data sources for anomaly detection. There are many types of logs, and each of them records different information. By analyzing these logs, we can achieve the goal of abnormal behaviors detection and potential security threats mining [1]. Abnormal behaviors refer to the behaviors that are caused by attacks and can affect the normal operations of computers or the network. Our goal is to discover these abnormal behaviors in a timely manner and reduce their damages and effects to the systems. For traditional standalone systems, administrators can investigate the system logs manually and detect anomalies based on a keyword search or regular string matching. However, such methods that rely heavily on manual inspection have become inadequate for large-scale systems [2]. Anomaly detection based on automated log analysis has attracted much attention from both researchers and commercial providers in the past decade, and many log-based anomaly detection methods are proposed. However, most existing detection methods require a priori knowledge; they use the patterns of known attacks to identify anomalies; thus, these methods cannot address the new or unknown threats. Furthermore, the increasing scale and complexity of modern systems make the volume of logs increase quickly, which requires that the newly developed anomaly detection methods should have high accuracy and low computational complexity. Facing the above challenges, we propose a new anomaly detection method which integrates the K-prototype clustering and k-NN classification algorithms and uses a

novel clustering-filtering-refinement framework to improve the detection accuracy and efficiency.

First, we analyze the characteristics of abnormal behaviors and system log data and use the session information as a basic unit to describe the user behaviors. We extract 10 features based on the session information to build accurate user profiles from two aspects: login activity and session statistics. For login activity, we extract the username, the login host, the login physical location, the time point of session establishment and the session lifetime to describe the login behavior using the 3W approach, which describes Who, Where, and When aspects. For session statistics, the frequent operating command sequences are extracted first to reflect the user's behavior habits; we can also infer the user's role based on the command sequences. The password failed times during a given time period are then extracted to measure the risk degree of the account. The frequency of file operations (such as read, write, create and delete), the frequency of accessing sensitive files, and the frequency of authority promotion are extracted to measure the threat degree of user behaviors. These statistical features are used to describe the user behaviors from the first W (What) aspect: What does the user do during the session? Our method exclusively uses the session information as a data source, which can significantly reduce the computational complexity in characterizing user behaviors [4].

Second, the extracted features contain numerical and categorical attributes, and most of the traditional clustering algorithms, such as DBSCAN and K-means, cannot handle this kind of data. K-prototype [5], [22] is one of the famous methods which works well for data sets of mixed attributes, so we employ the K-prototype clustering algorithm to group the dataset into different clusters. Then, we design a filtering threshold to classify the clusters into normal clusters and anomaly candidates based on their distributions and densities. The clusters that have a large number of events and present as highly coherent clusters are regarded as normal clusters [6]. In addition, the sparse clusters are possibly caused by the anomalous activities, which can be marked as the anomaly candidates for further processing. By filtering out the majority of normal events, we can focus on addressing only the remaining anomaly candidates, which can greatly reduce the computational complexity of the following refinement process and improve the detection accuracy.

Third, we analyze the characteristics of anomaly candidates, then propose two new distance-based features based on cluster centers and nearest neighbors to measure the local and global anomaly degrees for each anomaly candidate. The first feature is used to measure the isolation degree of anomaly candidates, and the higher value denotes a higher anomaly degree in the local perspective. The other feature is used to measure the distance difference of anomaly candidates to normal clusters and anomaly candidate clusters. The higher value denotes a higher anomaly degree in the global perspective. Consequently, a new dataset containing only two features can be obtained to characterize the anomaly candidates, and then we employ a k-NN classifier to identify real anomalies. Since the size of anomaly candidates and the dimensions of features are significantly smaller than the original data, the time consumption of the refinement process would be significantly reduced, and the detection accuracy can also be improved.

Finally, to verify the integrated method proposed in this paper, we construct a log collection and anomaly detection platform in the campus network center of Xi'an Jiaotong University. We introduce Flume [7], HDFS (Hadoop Distributed File System, [8]) and Spark [9] technologies into the platform as parallel data processing and a reliable data storage mechanism that can help to process massive logs more efficiently and more stably. We collect system logs from more than 50 actual network servers and labeled anomaly logs from one target server based on the platform. Experimental results based on the datasets collected from the platform validate that the proposed method has high detection accuracy and low computational complexity.

The rest of this paper is structured as follows. We briefly review the related works in the Section II. Section III describes the workflow of the proposed method, the anomaly detection platform and the features extracted. Section IV presents the integrated methodology using K-prototype clustering and the k-NN classifier. Dataset collections and experimental results are presented in Section V. Then, the conclusion and future work are presented in Section VI.

## II. RELATED WORKS

Recently, the technologies for analyzing log files have extensively been explored in the areas of anomaly detection. Most previous works are signature-based methods. Schultz *et al.* [10] proposed a method for detecting malicious executables with minimum a priori information using a multi-label Naive Bayes classification algorithm, which can achieve very good results in comparison to traditional signature-based methods. Though these kinds of algorithms can be used to detect known anomalies, they generally do not perform well against unknown attacks.

Many anomaly detection methods introduce clustering algorithms to help detect unknown abnormal behaviors or potential attacks. Chen and Li [11] proposed an enhanced DBSCAN algorithm for anomaly detection. This algorithm can extract the normal behavior profile from the audit records and adjust the profile with the progress of the program. Münz *et al.* [12] presented a novel flow-based anomaly detection scheme based on a K-means clustering algorithm, using the corresponding cluster centroids as patterns for efficient distance-based anomaly detection. Hoglund *et al.* [13] and Lichodzijewski [4] constructed a host-based anomaly detection system which applied the self-organizing maps (SOMs) algorithm to judge whether the user behavior is abnormal or not. Syarif *et al.* [14] compared five clustering algorithms to select the best one with the highest detection accuracy. Even though these methods result in higher accuracies for detecting unknown anomalies than

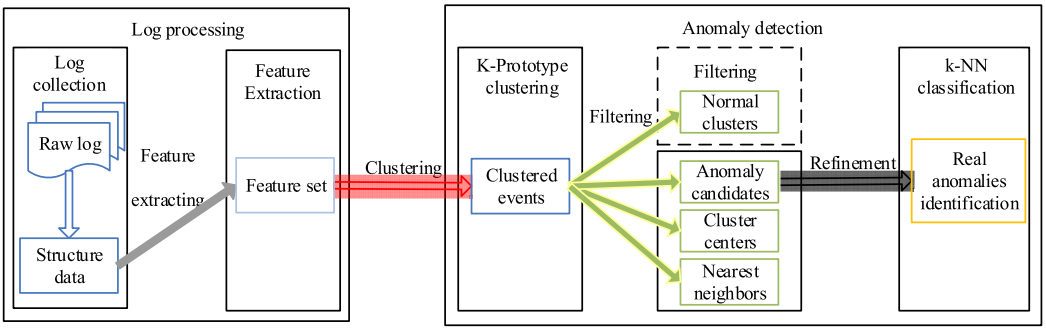


**FIGURE 1.** Workflow of the proposed method.

the signature-based detection methods, they are not feasible enough for practical applications due to the high number of false positives generated.

There are some other works which focus on analyzing the system calls at the program level to achieve the goal of anomaly detection [15]–[17]. Forrest *et al.* [15] and Hofmeyr *et al.* [16] constructed a list of unique short sequences of system call traces as a normal database, and the sequences being tested are compared with the normal database to judge whether they are abnormal or not. Hoang *et al.* [17] build a multilayer model based on hidden Markov models and enumerating methods for detecting anomalous behavior of programs. However, these algorithms have difficulty obtaining accurate information about abnormal jobs and usually have high computational complexity.

To solve the abovementioned challenges, some recent studies focus on analyzing log files and combining different techniques to improve the detection performance. Makanju *et al.* [18] propose a framework integrated with anomaly detection and signature generation based on frequent pattern mining to utilize the advantages of signature-based and anomaly-based approaches. Asif-Iqbal *et al.* [19] concentrate on parsing logs from different sources and then clustering the logs to identify and remove unneeded entries, which greatly help in correlating different logs. Hajamydeen *et al.* [20] use a clustering algorithm twice in different stages to classify the events: 1) to assist the filtering process to identify the abnormal events, and 2) to support the analysis method in detecting anomalies. However, along with the increasing scale and complexity of modern network systems, determining how to design an efficient detection algorithm to analyze massive logs and improve the detection accuracy is still an open question.

Enlightened by the related works, we extract features from the session information instead of the raw log records as a data source, which can greatly reduce the amount of data to be processed. Furthermore, we propose an integrated method that uses the K-prototype clustering and k-NN classification algorithms and use a clustering-filtering-refinement framework to improve the detection accuracy and efficiency.

## III. FRAMEWORK OF THE PROPOSED METHOD

### A. WORKFLOW OF THE PROPOSED METHOD

To overcome the limitation of the existing anomaly detection methods that spend the greatest amount of time building models for the whole dataset, we develop a clustering- filtering-refinement framework to improve the detection accuracy and efficiency. Figure 1 illustrates the overall workflow of the proposed method, which mainly involves five steps described as in the following:

1) Log collection: Multitype logs are collected using the Linux syslog mechanism from remote servers [27]. The collected logs include logs from the servers running in an actual network environment and labeled anomaly logs from the target server.

2) Feature extraction: To capture the 4W (Who, Where, When, and What) characteristics of the user behaviors, we extract 10 features about logging activity and session statistics from the session information of users, which contain both numeric and categorical attributes.

3) Clustering: Based on the features extracted, we employ the K-prototype clustering algorithm to group the log events into different clusters.

4) Filtering: We use a filtering threshold to classify the clusters into normal clusters and anomaly candidate clusters. The normal events are usually similar to each other and are grouped into highly coherent clusters; we regard these clusters containing events that are greater than the threshold as normal clusters, and the rest of the clusters are regarded as anomaly candidate clusters. After filtering out these normal clusters, we can focus on addressing the anomaly candidates.

5) Refinement: We extract two new distance-based features to characterize the local and global anomaly degrees of the anomaly candidates, and then we apply a k-NN classifier to the candidates to identify real anomalies based on the two new features extracted.

### B. LOG COLLECTION AND ANOMALY DETECTION PLATFORM

To bridge the gap between research in academia and practice in industry, we construct a log collection and anomaly detection platform which can process massive logs effectively [26]; the structure of the platform is shown in Figure 2.

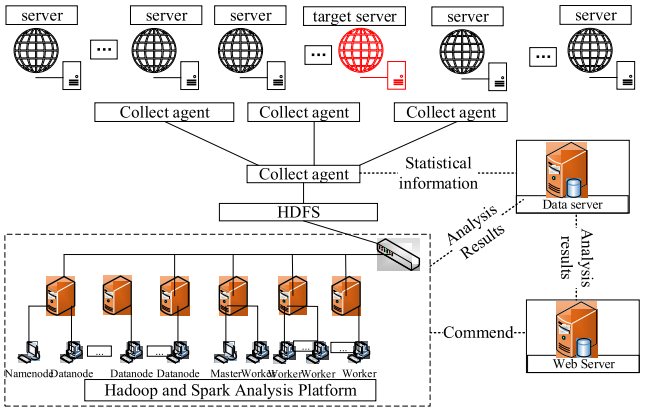


**FIGURE 2.** The structure of the anomaly detection platform.

The platform collects system logs from more than 50 actual running servers and labeled logs from one target server (marked as red in Figure 2) that are located in the campus network center of Xi'an Jiaotong University. The servers are mainly used by colleges or institutions to establish their own websites and provide news and email services for their employees. The target server has the same configurations as the other servers, which will be attacked by security engineers to generate labeled logs. Based on the platform, we collect several kinds of typical logs such as secure log, message log and audit log. We need to configure the system for collecting meaningful audit log and message log, the secure log can be directly collected from the log files. To process the massive logs efficiently and stably, we employ Flume to collect massive logs and store them into HDFS (Hadoop Distributed File System) and introduce Spark technology into the platform for real-time massive log processing. In the platform, the Web management server is responsible for the human-computer interaction, which can receive the instructions from the administrators, send the commands to the anomaly detection platform for appropriate operations, and then visualize the final detection results. The significant benefits of the anomaly detection platform we constructed are its scalability and reliability as well as its fault tolerance.

### C. FEATURE EXTRACTION

To achieve the goal of anomaly detection, we need to extract proper features to characterize the user behaviors. Some algorithms [15]–[17] treat the job's trace sequence as a data source and simply recognize a log sequence as a symbol string. However, these kinds of algorithms can hardly obtain the insight and accurate information about the abnormal jobs. Unlike previous works, we use the session information of the users as a data source to detect potential abnormal behaviors, which can minimize the computational complexity and obtain detailed information about abnormal behaviors.

A session is defined as a specific connection starting when a user connects to the server and ending when a user disconnects from the server after performing some operations. In this paper, we use the session as a basic unit and extract 10 features from the session information based on the characteristics of user behaviors, which are shown in Table 3. The features we extracted mainly contain two kinds of attributes: one is the basic attribute about logging activity; the other is the session statistics about user behaviors during each session. A detailed definition and explanation are given as follows:

1) Features related to logging activity are mainly used to characterize the users' logging behaviors, to measure whether the user login to a server is from a usual host, to determine whether the user login to a server occurs during the usual period of time or in a usual location, etc. These features are used to describe the logging behavior characteristics from the 3W (Who, Where, and When) aspects.

*Feature 1 (The Login Username):* This feature represents the identity of the user who logged in or tried to log in, labeled by the UNIX username. This feature can be extracted from the secure log.

*Feature 2 (The Login Host):* This feature represents the host from which the connection was established, it is usually denoted by the IP address. This feature can be extracted from the secure log.

*Feature 3 (The Login Physical Location):* This feature represents the physical location where the connection was established, typically it means the city where the IP address located. We can obtain the physical address by searching database of the IP addresses and their corresponding physical addresses.

*Feature 4 (The Time Point When the Connection Was Established):* This feature represents the time point that the session was started and it can be extracted from the secure log.

*Feature 5 (The Session Lifetime):* This feature can be obtained based on the time point of the session establishment and termination, this features can be extracted from the secure log.

2) Features related to session statistics are mainly used to characterize the users' operating behaviors and measure whether the users have performed some sensitive operations. These statistical features are used to describe the user behavior characteristics from the 1W (What) aspect: What does the user do during each session.

*Feature 6 (The Frequent Operating Command Sequences):* We can obtain the command sequences that a user performed during each session, and then use the PrefixSpan algorithm provided by Spark to mine the frequent operating command sequences for the specific user from all the command sequences he/she performed. This feature is used to reflect the user's customary behaviors or to infer the user's role, this feature can be extracted from the message log.

*Feature 7 (The Number of Password Failed Times During a Given Time Period:* This feature is used to measure the risk degree of the account; if there are a large number of failed login attempts in a short time window, the corresponding

account may have suffered a crack attack. This feature can be extracted from the secure log.

*Feature 8 (The Frequency of File Operations Such as Read, Write, Create and Delete During Each Session):* This feature is used to characterize the user's behavior related to file operations; if the user tries to delete files frequently, this may indicate a sensitive behavior. This feature can be extracted from the audit log.

*Feature 9 (The Frequency of Accessing Sensitive and Key Files During Each Session):* Sensitive files are very important for normal running of a system; therefore, frequently accessing these files may indicate a potential abnormal behavior. This feature can be extracted from the audit log.

*Feature 10 (The Frequency of Authority Promotion During Each Session):* Most of the users have only limited authority to perform operations on the servers. Frequent authority promotion from a normal user to root access may lead to many sensitive operations and is dangerous for security management. This feature can be extracted from the message log.

From the above descriptions, we can see that features 8, 9 and 10 are used to measure the threat degree of the user behaviors. If the user performs the related operations frequently, we can assign a higher threat degree to the user. As described above, all the features extracted from the session information are used to characterize the user behaviors effectively from the 4W (Who, Where, When, and What) aspects.

## IV. ANOMALY DETECTION METHODS

### A. CLUSTERING: ANOMALY CANDIDATES MINING

Clustering algorithms have been widely applied to various areas to explore the hidden and useful patterns inside massive data. Traditional clustering algorithms use the Euclidean distance to measure the similarity of two data elements and perform clustering [21], which works well for a data set containing purely numeric attributes. However, the features extracted from the massive logs in this paper contain both numeric and categorical attributes; most of the traditional clustering algorithms cannot handle this kind of data effectively. Most studies about solving clustering problems with mixed data address redefining the distance measure and applying it to the existing clustering algorithms. K-prototype [5], [22] is one of the most famous methods, which inherits the ideas of K-means and applies the Euclidean distance to numeric attributes, designing a new distance function to measure the closeness between categorical attributes.

The K-prototype algorithm [5] works well for mixed data with numeric and categorical attributes by defining a new cost function and distance measure based on the co-occurrence of values. We briefly introduce the K-prototype clustering algorithm [5] as follows:

Let us assume $D = (d_1, d_2, \ldots, d_n)$ represents the mixed dataset with $n$ data objects, and $d_i = (A_1, A_2, \ldots, A_m)$ represents an object defined by $m$ attributes, where the first $m_r$ attributes are numeric and the next $m_c$ attributes are categorical and $m_r + m_c = m$.

The cost function of the K-prototype algorithm is specified in Equation 1, which is to be minimized for clustering mixed datasets.

$$\theta = \sum_{i=1}^{n} \varphi\left(d_i, C_j\right) \tag{1}$$

where $\varphi\left(d_i, C_j\right)$ is the distance of a data object $d_i$ from its closest cluster center $C_j$ which is defined as follows:

$$\varphi\left(d_i, C_j\right) = \sum_{t=1}^{m_r} \left(w_t \left(d_{it}^r - C_{jt}^r\right)\right)^2 + \sum_{t=1}^{m_c} \Omega\left(d_{it}^c, C_{jt}^c\right)^2 \tag{2}$$

where $\sum_{t=1}^{m_r} \left(w_t \left(d_{it}^r - C_{jt}^r\right)\right)^2$ denotes the Euclidean distance of object $d_i$ from its closest cluster center $C_j$ for numeric attributes only and $w_t$ denotes the significance of the $t^{th}$ numeric attribute which is to be computed from the dataset. $\sum_{t=1}^{m_c} \Omega\left(d_{it}^c, C_{jt}^c\right)^2$ denotes the distance between the data object $d_i$ from its closest cluster center $C_j$ for categorical attributes only.

The cluster center $C_j$ has a dual representation. For a numeric attribute, the central attribute value is represented by the mean of all the values in the corresponding cluster. For a categorical attribute, the center $C_j$ is represented by a proportional distribution of all its values in the corresponding cluster. $\Omega\left(d_{it}^c, C_{jt}^c\right)$ is computed as a function of the actual value for a categorical attribute and the proportional distribution of all categorical values for that attribute in the cluster. The computation of this function is described below:

Let $A_{i,k}$ denote the $k^{th}$ value for categorical attribute $A_i$, and $p_i$ denote the total number of distinct values for $A_i$.

$$\begin{aligned}\Omega\left(x, C\right) = &\left(N_{i,1,c}/N_c\right) * \delta\left(x, A_{i,1}\right) \\ &+ \left(N_{i,2,c}/N_c\right) * \delta\left(x, A_{i,2}\right) \\ &+ \ldots + \left(N_{i,p_i,c}/N_c\right) * \delta\left(x, A_{i,p_i}\right)\end{aligned} \tag{3}$$

where $N_c$ is the number of data objects in cluster $C$ and $N_{i,k,c}$ denotes the number of elements in cluster $C$ which has the $k^{th}$ attribute value for the $i^{th}$ attribute, assuming that $i^{th}$ attribute has $p_i$ different values. Thus, the cluster center represents the proportional distribution of each categorical value in the cluster.

$\delta\left(x, y\right)$ represents the distance between two distinct values of $x$ and $y$ of any categorical attribute $A_i$, which holds the following properties:

(1) $0 \le \delta\left(x, y\right) \le 1$
(2) $\delta\left(x, y\right) = \delta\left(y, x\right)$
(3) $\delta\left(x, x\right) = 0$

Another important parameter for the K-prototype algorithm is the number of clusters $K$. Since we want to separate different event patterns into different clusters, $K$ should be calculated based on the patterns mined from the logs. An EM (expectation maximization) clustering algorithm [23] has the ability to predict the ideal number of clusters [24] using cross-validation, and the clusters generated by the EM clustering

**TABLE 1. Simple description of the features extracted.**

| # | Classification | Simple Description |
|---|---|---|
| *Feature 1* | Logging activity | The username used in the session |
| *Feature 2* | Logging activity | The login host (IP address) used in the session |
| *Feature 3* | Logging activity | The login physical location |
| *Feature 4* | Logging activity | The time point when the connection was established |
| *Feature 5* | Logging activity | The session lifetime |
| *Feature 6* | Session statistics | The frequent operating command sequences |
| *Feature 7* | Session statistics | The number of password failed times during a time period |
| *Feature 8* | Session statistics | The frequency of file operations such as read, write, create and delete during each session |
| *Feature 9* | Session statistics | The frequency of accessing sensitive and key files during each session |
| *Feature 10* | Session statistics | The frequency of authority promotion during each session |

algorithm reveal that the number of clusters generated is not influenced by the number of features and events in the logs but is influenced by the patterns of the logs. Hence, we employ an EM clustering algorithm to predict the number of clusters $K$, and then we use the K-prototype algorithm to group our dataset into $K$ clusters.

The workflow of the K-prototype clustering algorithm is shown as follows:

**Algorithm 1** The K-Prototype Clustering

Begin

Initialization – Allocate data objects to predetermined $K$ clusters randomly.

For every categorical attribute:

Compute distance $\delta(x, y)$ between two categorical values $x$ and $y$.

For every numeric attribute:

Compute significance of attribute.

Assign data objects to different clusters randomly.

Repeat steps 1-2:

1. Compute cluster centers for $C_1, C_2, \ldots C_k$.
2. Each data object $d_i$ is assigned to its closest cluster center using $\varphi\left(d_i, C_j\right)$.

Until no elements change clusters or a predefined number of iterations are reached.

End.

### B. FILTERING: ANOMALY CANDIDATE SELECTION

The goal of this step is to remove the obvious normal clusters while retaining the anomaly candidates for further processing. According to the experience of network management, most of the time the servers are running in the normal state, so most of the log data collected are generated by normal behaviors, and anomalies should be a minority that has different patterns than others [6]. Hence, normal events usually form highly coherent clusters with an extensive number of events since the normal events are similar to each other while abnormal events are clustered into smaller sparse clusters or isolated points [25]. The filtering process is performed based on the above observations. We claim that the dense clusters with a large number of events are normal clusters which can be filtered out to reduce the computational complexity of the following refinement process.

We propose a filtering threshold to classify the clusters into normal clusters and anomaly candidate clusters. The threshold ($T$) is calculated based on the number of events and the number of clusters, $T = n/K$, where $n$ is the total number of events and $K$ is the number of clusters. The clusters containing events that are greater than the threshold are regarded as normal clusters, and the rest of the clusters containing events that are less than or equal to the threshold are regarded as anomaly candidate clusters. Most of the normal events (usually more than 70%, based on our experiments when analyzing datasets) are removed in the filtering stage. With a small amount of anomaly candidates, the effort and time consumption to analyze it will be considerably smaller. Furthermore, the splitting of the analysis into two separate stages allows the possibility of applying different algorithms to process the dataset, which can generate more accurate anomaly results.

### C. REFINEMENT: ACCURATE ANOMALY DETECTION

In the refinement stage, each anomaly candidate should be examined further to finally determine whether it is an anomaly or not. After the cluster centers and nearest neighbors for every data point in the given dataset are identified, two types of distances can be calculated. The first one is based on the distance from each data point to the cluster centers. The second type is based on the distance from each data point to its nearest neighbors. Based on the above two types of distances, we design two new distance- based features for each anomaly candidate to measure its local and global anomaly degrees, then we can apply a k-NN classifier to the anomaly candidates to identify real anomalies based on the two new features. Let $N_{\mathrm{k}}(d_{\mathrm{i}})$ denote the $k$ nearest neighbors for object $d_{\mathrm{i}}$; we use the feature below to measure the isolation degree of the anomaly candidate. If the data point is unique in the dataset, the value of $F_l$ would be high; the higher value of $F_l$ denotes a higher anomaly degree in the local perspective.

$$F_{\mathrm{l}}(d_{\mathrm{i}}) = \frac{1}{k} \sum_{o \in N_{\mathrm{k}}(d_{\mathrm{i}})} \frac{\varphi_{\mathrm{kd}}(d_{\mathrm{i}})}{\varphi_{\mathrm{kd}}(o)} \tag{4}$$

Where

$$\varphi_{\mathrm{kd}}(o) = \frac{1}{k} \sum_{p \in N_{\mathrm{k}}(o)} \varphi(p, o) \tag{5}$$

$\varphi_{\mathrm{kd}}(o)$ represents the relative distances from $o$ to its $k$ nearest neighbors. $\varphi(p, o)$ denotes the distance between two objects $p$ and $o$.

Let us assume $S_{\mathrm{out}}$ and $S_{\mathrm{left}}$ denote the cluster centers which are filtered out and left for further processing, respectively. We use the feature below to measure the distance difference of an anomaly candidate to normal clusters and to anomaly candidate clusters; the higher value of $F_g$ means the anomaly candidates are far from the normal clusters and denotes a higher anomaly degree in the global perspective.

$$F_{\mathrm{g}}(d_{\mathrm{i}}) = \frac{\frac{1}{size(S_{\mathrm{out}})} \sum_{o \in S_{\mathrm{out}}} \varphi(d_{\mathrm{i}}, o)}{\frac{1}{size(S_{\mathrm{left}})} \sum_{p \in S_{\mathrm{left}}} \varphi(d_{\mathrm{i}}, p)} \tag{6}$$

After we extracted the above two distance-based features for each anomaly candidate, the original dataset which contains $n$ data objects, with each object having 10 features, can be replaced by a new dataset containing $l$ ($l \ll n$) data objects, and each object has only two features. Then, we employ a k-NN classifier to the new dataset to identify real anomalies. However, the k-NN classification algorithm needs to evaluate all the anomaly candidates to calculate $k$ nearest neighbors, which is inefficient. After eliminating most of the normal events and reducing the feature dimensions, the calculation time would be reduced significantly.

## V. EXPERIMENTS

### A. DATASETS

Based on the platform introduced in Section 3.2, we mainly collect two kinds of system logs: one is the system log from the actual servers, and the other is the labeled anomaly log from the target server. The target server is used to generate labeled anomaly logs, which has the same configuration as the other actual servers, except that the access control strategy of the target server is less restricted compared to the other actual servers. All the logs are collected from these servers during the period of January 2016 to June 2016. During the data collection phase, the target server has been attacked by the engineers from our lab, Huawei's security engineers and Dutch hackers. Some typical intrusion behaviors such as illegal log modification and illegal authority promotion are performed on the target server to construct the benchmark data. As the actual servers and target server are running in the same network environment, the labeled logs collected from the target server have the same characteristics as those logs collected from actual server. We mixed them together to construct the datasets for the performance evaluation. Table 2 shows the detailed information of the datasets used in the following experiments. In these datasets, we try to encompass different attack scenarios with different data distributions as much as possible.

**TABLE 2. Datasets.**

| Type | Number of Sessions | Anomalies | Time |
|---|---|---|---|
| SDS 1 | 1227 | 45 | 2016.01.11-2016.06.24 |
| SDS 2 | 2060 | 98 | 2016.01.11-2016.06.24 |
| SDS 3 | 4693 | 181 | 2016.01.11-2016.06.24 |
| SDS 4 | 6390 | 200 | 2016.01.11-2016.06.24 |
| SDS 5 | 8120 | 248 | 2016.01.11-2016.06.24 |
| SDS 6 | 10258 | 289 | 2016.01.11-2016.06.24 |

### B. METRICS FOR PERFORMANCE EVALUATION

We use the rates of precision, recall and false alarm, which are widely used in the related literature to evaluate the performance of the integrated method we proposed, as we already have the ground truth (anomaly or not) for all the datasets. The definitions of precision, recall and false alarm are given as follows, where:

True positives (*TP*), the number of the anomalies that are correctly classified as anomalies;

True negatives (*TN*), the number of the normal events that are correctly classified as normal events;

False positives (*FP*), the number of the normal events that are wrongly classified as anomalies; and

False negatives (*FN*), the number of the anomalies that are wrongly classified as normal events.

$$precision = \frac{TP}{TP + FP} \tag{7}$$

$$recall = \frac{TP}{TP + FN} \tag{8}$$

$$false\ alarm = \frac{FP}{FP + TN} \tag{9}$$

As defined in Equations 7, 8 and 9, the precision rate measures the percentage of the reported anomalies that are correctly identified, the recall rate measures the percentage of the real anomalies which are detected, and the false alarm rate indicates the percentage of the normal events that are wrongly identified as anomalies.

### C. EVALUATION OF THE CLUSTERING AND FILTERING PROCESS

We first evaluate the performance of the clustering and filtering process. As mentioned before, the number of clusters $K$ for K-prototype algorithm was selected by the EM clustering algorithm. In this paper, the number of clusters generated by using the EM clustering algorithm is 7. Table 3 shows the performance evaluation results of the K-prototype algorithm on different datasets, where the second column NNC represents the number of clusters that are filtered out as normal clusters, the third column NEFO means the number of events in the normal clusters which are filtered out, and the fourth column percentage denotes the percentage of the events that are filtered out. While the last column precision means the

**TABLE 3.** Evaluation on K-prototype clustering.

| Type | NNC | NEFO | Percentage | Precision |
|---|---|---|---|---|
| SDS 1 | 2 | 1043 | 85.00% | 100% |
| SDS 2 | 2 | 1466 | 71.17% | 99.18% |
| SDS 3 | 2 | 3691 | 78.65% | 99.59% |
| SDS 4 | 3 | 4487 | 70.22% | 100% |
| SDS 5 | 3 | 6839 | 84.22% | 99.87% |
| SDS 6 | 2 | 7727 | 75.33% | 99.78% |

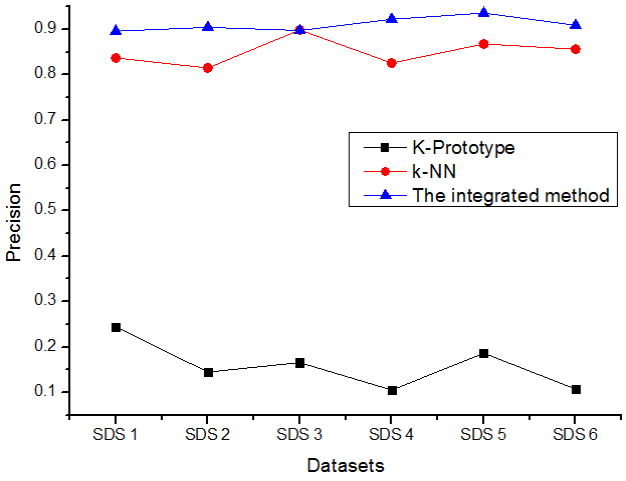


**FIGURE 3.** Precision rate evaluation.

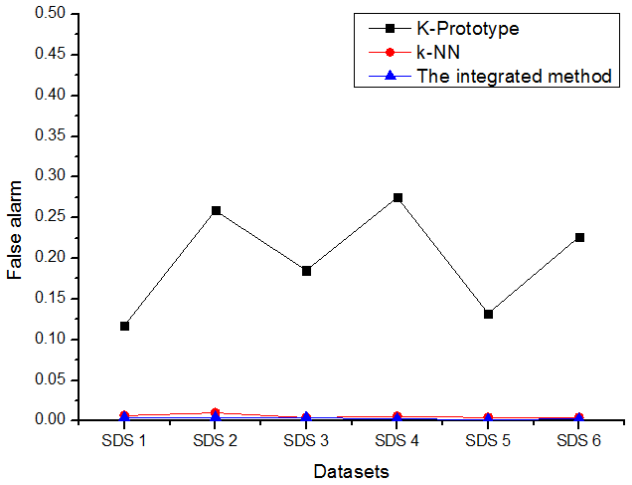


**FIGURE 4.** False alarm rate evaluation.

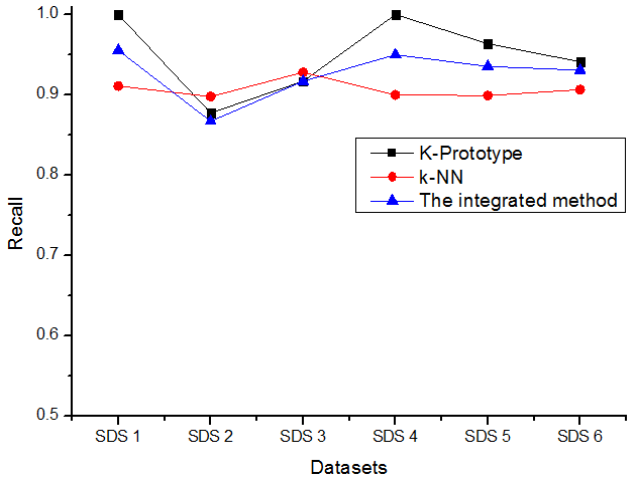


**FIGURE 5.** Recall rate evaluation.

precision of the events that are filtered out, it represents how many events are correctly identified as normal events in these clusters that are filtered out. From the evaluation results, we can observe that usually more than 70% of the normal events are filtered out with a filtering precision up to 99% during the clustering and filtering steps, which greatly reduces the computational complexity of the following refinement step.

Specifically, we analyze the filtering result of SDS 1. There are 1227 total events and 45 anomaly events; we eliminate 1043 normal events that occupy 85% of the total dataset during the filtering process and obtain 184 anomaly candidates. There is no false negative mistake in any of the 1043 normal events that are filtered out, so the precision of the filtering process on this dataset is 100%.

### *D. PERFORMANCE EVALUATION OF THE INTEGRATED METHOD*

The first step of the proposed method is based on the K-prototype algorithm, and because there are some reports in the literature [12], [14] that only applied a clustering algorithm for anomaly detection, we use the K-prototype algorithm alone to compare it with the integrated method we proposed. The baseline K-Prototype algorithm is tested on the datasets without the filtering and refinement process, so all the anomaly candidates are regarded as anomalies here.

The final step of the proposed method is based on the k-NN algorithm; therefore, we also use the k-NN algorithm alone to compare it with the integrated method we proposed. The baseline k-NN algorithm is trained and tested on the original datasets without the clustering and filtering process. In particular, for the $k$ value of the k-NN algorithm, we examine $k = 1, 3, 5, \ldots, 25$ in order to obtain the best performance.

We mainly compare the integrated method we proposed with the baseline K-prototype algorithm and the baseline k-NN algorithm in terms of the performance and the processing time. Figures 3, 4 and 5 present the precision rate, the recall rate and false alarm rate of these methods on different datasets, respectively. Figure 6 presents the processing time of these methods on different datasets.

From Figures 3 and 4, we can see that the integrated method we proposed provides very stable and effective results on different datasets and produces the highest

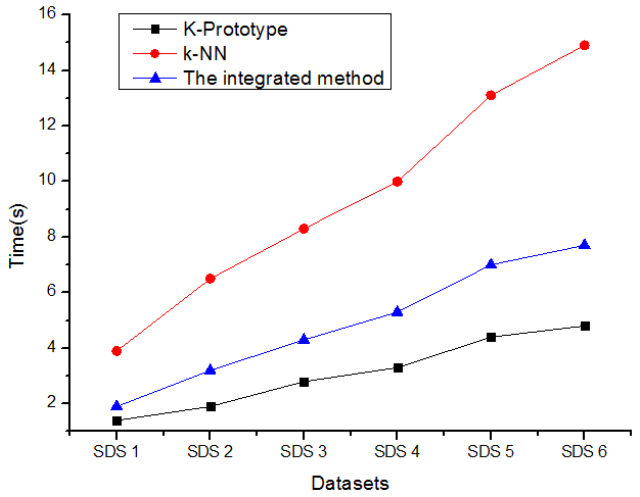


**FIGURE 6.** Computational complexity evaluation.

accuracy and lowest false alarm rate, outperforming the baseline K-prototype and k-NN algorithms. The results of the K-prototype algorithm without the filtering and refinement process are the worst; it generates a high fraction of false alarms on different datasets. This is because the major purpose of the K-prototype clustering in this paper is to filter out the obvious normal events, so there are still many normal events classified as anomaly candidates, which generates a large number of false positives if we only use the K-prototype algorithm. Although the K-prototype algorithm produces the highest recall as Figure 5 shows, it is still not a good choice for anomaly detection because of the high false alarm rate, and the recall rate of the integrated method is close to the K-prototype algorithm. The results of the k-NN algorithm are close to the integrated method in terms of the precision, the recall and false alarm rate, but in most cases the integrated method still performs better than the k-NN algorithm. The final detection accuracy of the integrated method is above 89%, and the false alarm rate is very low, with the experiment results verifying the effectiveness of the integrated method. In addition, they also validated that the two new distance-based features can well capture the characteristics of anomalies.

Evaluation results on the processing time of these methods are shown in Figure 6. We can observe that K-prototype is the most efficient algorithm, whereas the k-NN algorithm is the most inefficient. The integrated method is not as efficient as the K-prototype algorithm, but the processing time of these two methods are close. The integrated method is much faster than the k-NN algorithm, as the integrated method first filters out the majority of normal events and then runs the k-NN classifier on a small set of anomaly candidates; thus, the processing time of the integrated method is reduced significantly. However, the k-NN classifier is trained and tested on the original datasets, and it needs to evaluate all the events in the original datasets, which results in high computational complexity. Additionally, the processing time of the k-NN algorithm increases quickly with the increasing size of data, which is not applicable for practical usage.

The results of our experiment show that the integrated method performs much better than the K-prototype algorithm, and it performs similar to the k-NN algorithm but with a much lower processing cost. The integrated method we proposed takes advantages of different algorithms and generates better anomaly results with lower time complexity, which strikes a good balance of accuracy and computational complexity, which is important for practical applications in today's networks.

## VI. CONCLUSION AND FUTURE WORK

The significant growth volume of logs poses new challenges for anomaly detection; both the detection accuracy and computational complexity need to be considered at the same time. In this paper, we propose an integrated anomaly detection method and use a clustering-filtering-refinement framework to mine anomalies from massive logs with high accuracy and low time complexity.

We extract the features from the session information of users, in contrast to the methods which use system calls as a data source; therefore, our method can characterize the user behaviors more effectively and reduce the computational overheads. Then, we divide the anomaly detection process into three steps: In the clustering step, we employ a K-prototype clustering algorithm to partition the whole dataset into different clusters. In the filtering step, we propose a threshold to identify and filter out the obvious normal clusters. In the refinement step, we assign each anomaly candidate with two new distance-based features and then employ a k-NN classifier to identify the real anomalies. After eliminating most of the normal events, the processing time of the refinement step is reduced significantly. Finally, we construct an anomaly detection platform to process massive logs effectively and evaluate the proposed method based on the datasets collected from the platform. The experiment results validate the effectiveness and efficiency of the integrated method we proposed.

For future work, we will collect more anomaly logs generated by different types of attacks to make the labeled logs more representative and focus on characterizing the anomaly candidates and designing methods for identifying anomalies to achieve better performance.

## REFERENCES

[1] J. Breier and J. Branišová, "Anomaly detection from log files using data mining techniques," in *Information Science and Applications*, vol. 339. Berlin, Germany: Springer, Feb. 2015, pp. 449–457.
[2] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *Proc. 27th Int. Symp. Softw. Rel. Eng.*, Ottawa, ON, Canada, Oct. 2016, pp. 207–218.
[3] C. Abad, J. Taylor, C. Sengul, W. Yurick, Y. Zhou, and K. Rowe, "Log correlation for intrusion detection: A proof of concept," in *Proc. 19th Annu. Comput. Secur. Appl. Conf.*, Las Vegas, NV, USA, Dec. 2003, pp. 255–264.
[4] P. Lichodzijewski, A. N. Zincir-Heywood, and M. I. Heywood, "Host-based intrusion detection using self-organizing maps," in *Proc. Int. Joint Conf. Neural Netw.*, Honolulu, HI, USA, vol. 2, May 2002, pp. 1714–1719.

[5] A. Ahmad and L. Dey, "A k-mean clustering algorithm for mixed numeric and categorical data," *Data Knowl. Eng.*, vol. 63, no. 2, pp. 503–527, Nov. 2007.
[6] X. Yu, L. A. Tang, and J. Han, "Filtering and refinement: A two-stage approach for efficient and effective anomaly detection," in *Proc. 9th IEEE Int. Conf. Data Mining*, Miami, FL, USA, Dec. 2009, pp. 617–626.
[7] *Apache Flume*. Accessed: Aug. 2016. [Online]. Available: https://flume.apache.org/
[8] *Apache Hadoop*. Accessed: Aug. 2016. [Online]. Available: http://hadoop.apache.org/
[9] *Apache Spark*. Accessed: Aug. 2016. [Online]. Available: https://spark.apache.org/
[10] M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 2001, pp. 38–49.
[11] Z. Chen and Y. F. Li, "Anomaly detection based on enhanced DBScan algorithm," *Procedia Eng.*, vol. 15, no. 1, pp. 178–182, Jan. 2011.
[12] G. Münz, S. Li, and G. Carle, "Traffic anomaly detection using k-means clustering," in *Proc. GI/ITG Workshop MMBnet*, 2007, pp. 1–8.
[13] A. J. Hoglund, K. Hatonen, and A. S. Sorvari, "A computer host-based user anomaly detection system using the self-organizing map," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Netw.*, Como, Italy, vol.5, Jul. 2000, pp. 411–416.
[14] I. Syarif, A. Prugel-Bennett, and G. Wills, "Unsupervised clustering approach for network anomaly detection," in *Proc. Int. Conf. Netw. Digit. Technol.*, vol. 293, 2012, pp. 135–145.
[15] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 1996, pp. 120–128.
[16] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *J. Comput. Secur.*, vol. 6, no. 3, pp. 151–180, 1998.
[17] X. D. Hoang, J. Hu, and P. Bertok, "A multi-layer model for anomaly intrusion detection using program sequences of system calls," in *Proc. 11th IEEE Int. Conf. Netw.*, Sydney, NSW, Australia, Oct. 2003, pp. 531–536.
[18] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "Investigating event log analysis with minimum apriori information," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, Ghent, Belgium, May 2013, pp. 962–968.
[19] H. Asif-Iqbal, N. I. Udzir, R. Mahmod, and A. A. A. Ghani, "Filtering events using clustering in heterogeneous security logs," *Inf. Technol. J.*, vol. 10, no. 4, pp. 798–806, 2011.
[20] A. I. Hajamydeen, N. I. Udzir, R. Mahmod, and A. A. A. Ghani, "An unsupervised heterogeneous log-based framework for anomaly detection," *Turkish J. Elect. Eng. Comput. Sci.*, vol. 24, no. 3, pp. 1117–1134, 2016.
[21] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probab.*, vol. 1, 1967, pp. 281–297.
[22] Z. Huang, "Clustering large data sets with mixed numeric and categorical values," in *Proc. 1st Pacific–Asia Conf. Knowl. Discovery Data Mining*, 1997, pp. 21–34.
[23] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc., B (Methodol.)*, vol. 39, no. 1, pp. 1–38, 1977.
[24] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.
[25] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009.
[26] T. Qin, Y. Gao, L. Wei, Z. Liu, and C. Wang, "Potential threats mining methods based on correlation analysis of multi-type logs," *IET Netw.*, to be published, doi: 10.1049/iet-net.2017.0188.
[27] C. Lonvick, *The BSD Syslog Protocol*, document RFC 3164, 2001.

**ZHAOLI LIU** received the B.S. and M.S. degrees in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 2007 and 2010, respectively, where she is currently pursuing the Ph.D. degree in computer science and technology. She is currently a Visiting Ph.D. Student with the Prof. Weibo Gong Group, University of Massachusetts, Amherst, USA, which is funded by the China Scholarship Council. Her research interests lie primarily in network security and online social network analysis.

**TAO QIN** received the B.S. degree in information engineering and the Ph.D. degree in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 2004 and 2010, respectively. He is currently an Associate Professor with the Department of Computer Science and Technology and the MOE KLINNS Lab, Xi'an Jiaotong University, while he is also a Research Fellow with the Shenzhen Research School at Shenzhen. His research focuses on Internet traffic analysis, traffic modeling, anomaly detection, and online social network analysis.

**XIAOHONG GUAN** (F'07) received the B.S. and M.S. degrees in control engineering from Tsinghua University, Beijing, China, in 1982 and 1985, respectively, and the Ph.D. degree in electrical and systems engineering from the University of Connecticut, Storrs, USA, in 1993. He was a Visiting Professor with the Division of Engineering and Applied Science, Harvard University, Cambridge, USA, in 1999. He was elected as an Academician of the Chinese Academy of Sciences in 2017. He is currently a Cheung Kong Professor with the Department of Automation and the Dean of the School of Electronic and Information Engineering. His research focuses on allocation and scheduling of complex networked resources, network security, and sensor networks.

**HEZHI JIANG** received the B.S. degree in software engineering from South Central University for Nationalities, Wuhan, China, in 2014, and the M.S. degree in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 2017. His research focuses on anomaly detection and online social network analysis.

**CHENXU WANG** received the B.S. degree in information engineering and the Ph.D. degree in control science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2009 and 2015, respectively. He was a Post-Doctoral Research Fellow with The Hong Kong Polytechnic University in 2016. He is currently an Assistant Professor with the School of Software Engineering, Xi'an Jiaotong University. His research focuses on online social network analysis and information diffusion.