Received December 8, 2019, accepted December 20, 2019, date of publication December 25, 2019, date of current version January 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2962198

Semi-Supervised Malware Clustering Based on the Weight of Bytecode and API

YONG FANG, WENJIE ZHANG, BEIBEI LI, FAN JING, AND LEI ZHANG
College of Cybersecurity, Sichuan University, Chengdu 610065, China

Corresponding author: Lei Zhang (zhanglei2018@scu.edu.cn)

This work was supported by the National Key Technology Research and Development Program of China under Grant 2017YFB0802900.

ABSTRACT With the rapid advances of anti-virus and anti-tracking technologies, three aspects in malware clustering need to be improved for effective clustering, i.e., the robustness of features, the accuracy of similarity measurements, and the effectiveness of clustering algorithms. In this paper, we propose a novel malware family clustering approach based on dynamic and static features with their weights. In this approach, we employ a new similarity measurement method based on EMD to improve the accuracy of feature similarities. In addition, to reduce convergence time and improve clustering purity, we design a novel semi-supervised clustering algorithm, termed as S-DBSCAN by involving supervision information into the original algorithm known as Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The experimental results demonstrate that the proposed approach can correctly and accurately distinguish the samples among various families and achieve outperformed purity with 98.7%.

INDEX TERMS EMD, hybrid features, semi-supervised clustering, weight.

I. INTRODUCTION

An increasingly important problem of malware analysis is the large number of new malware samples. This number has exponentially increased throughout the years. According to the 2019 Symantec Network Security Threat Report [1], there were over 624 million new malware variants produced in 2019, an average of 673,000 each day. New malware variants are emerging at a rate that far exceeds the capability of manual analysis. An efficient way to cluster new variants is to cluster malware samples automatically and assign them labels according to their similarity. Then, malware researchers can pay more attention to new unknown malware instances.

Three main aspects need to be improved in malware clustering research. These three aspects are feature selection, similarity measurements, and clustering algorithms. For feature selection, some research performed malware clustering based on static analysis for fast malware clustering [2]–[9]. Static analysis is simple and fast, but it can be easily evaded if malware is packed or obfuscated. Hence, more researchers focused on the clustering approaches based on dynamic analysis [10]–[13]. These approaches are less vulnerable to mutation schemes, such as run-time packers or binary obfuscation. However, approaches based on dynamic features also suffer from several limitations. First, these approaches only cover part of the behaviors of the samples. Second, malware often includes triggers and performs malicious behaviors when certain conditions are met. Third, some malware has the functions of anti-sandbox and anti-virtual machines, which evades the dynamic analysis. For the similarity measurement, most similarity measurements are based on traditional methods, such as Euclidean distance [4], [14], [15], edit distance [16], cosine similarity [17]. However, these similarity measurements are not accurate as before and cannot handle similarities between features in different dimensions. Therefore, it is necessary to try new similarity measurement methods, with excellent performance in other fields for clustering. For the clustering algorithm, there are many clustering algorithms applied for malware detection, such as hierarchical clustering algorithms [10], [11], density-based clustering algorithms [5], [18], and prototype-based clustering algorithms [4], [19]. However, these hierarchical clustering algorithms are sensitive to noise and outliers, and they cannot handle clusters of different sizes and convex shaped clusters, either. These situations are common in malware clustering. These Prototype-based clustering algorithms need to specify the number of clusters artificially that is unknown in practical malware clustering conditions. These Density-based clustering algorithms do not apply to the case of uneven sample density and a large difference in clustering spacing.

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamed Elhoseny.

In this paper, we propose a semi-supervised clustering approach based on the hybrid features to further increase the purity and efficiency of clustering. The main contributions of this paper are as follows:

Feature selection: We propose a novel feature expression method that includes hybrid features and their weights. Combining the advantages of dynamic and static features with their weights can obtain more complete behaviors of a sample by complementing the drawbacks of each single category of feature.

Similarity measurement: We propose a new similarity measurement based on EMD for malware similarity calculation. This approach employs dynamic programming to calculate the minimal distance of the features, which is more accurate than traditional methods. Besides, the similarity between different dimensional features can be calculated by the proposed similarity measurement.

Clustering algorithm: We propose a new semi-supervised clustering algorithm based on the unsupervised DBSCAN algorithm by involving a small amount of supervised information. The proposed clustering algorithm significantly improves the efficiency of the clustering process.

The proposed approach is improved in three aspects. In feature selection, different from the traditional single static features or dynamic features [20], [21], we combine static features with dynamic features and involve weight to further enhance the robustness. In similarity measurement, we propose a new method for the similarity calculation of malware features which can calculate the similarity of features of different dimensions. Compared with the traditional similarity measurements such as Euclidean distance [15], the proposed method obtains more accurate results. In clustering algorithm, we propose a new semi-supervised clustering algorithm based on DBSCAN algorithm, which outperforms the DBSCAN algorithm's poor effect on sparse datasets and reduces the convergence time. Improvements in these three areas are reflected in the final clustering results, achieving purity of 0.987.

The organization of the remaining paper is as follows: Section II discusses the related work of our proposed approach. Section III presents the architecture of the proposed clustering process and describes the overview and details of our method. Section IV shows the experimental results and Section V gives our conclusion and outlines the future research directions.

II. RELATED WORK AND BACKGROUND

A. MALWARE CLUSTERING APPROACHES BASED ON STATIC ANALYSIS

The specific signature of each malware is used to perform static analysis, such as disassembling Opcode sequences [4]–[7], control flow graphs [8], function call graphs [2], PE headers [9] and API calls [22]. These features are extracted by static analysis tools, such as Interactive Disassembler Professional (IDA Pro [23]), PE Explorer [24]. The advantages of static analysis are high code coverage rate and the relatively high speed in extracting features. There is a lot of work based on static methods for malware clustering. In [6], C. Wang et al. proposed a method to extract Opcode sequence features by comparing the characteristics of different malware families, which mitigates the inaccuracy and instability caused by comparing malware with normal software. They used a fast density clustering algorithm to conduct malware family clustering. Awad and Sayre *et al.* [8] proposed a malware family clustering algorithm based on control flow graphs. The control flow graphs are represented by the strings and the edit distance is used to measure the similarity of the strings. The QT clustering algorithm is used for clustering. N. Singh et al. [25] proposed a clustering approach, called ByteFreq that can cluster numerous samples using the byte frequency. The Byte frequency is represented as time series and SAX (Symbolic Aggregation approXimation) is used to convert the time series in symbolic representation. Ahmadi *et al.* [20] extracted multi-dimensional features from the hexadecimal bytecode and disassembly opcode based on Microsoft Kaggle dataset [26] and then fused the features to classify malware.

The methods above extract features from different levels of PE file, such as Bytecode, disassembly sequence, opcode, and call graph. These methods perform well in different scenarios. However, these methods based on static analysis are easily evaded by encryption, polymorphism and code obfuscation. While the methods based on dynamic analysis are less vulnerable by these mutation schemes. Therefore, static analysis and dynamic analysis are combined in the proposed semi-supervised framework. Moreover, Bytecode is selected as a static feature, and weight information is involved as supplementary because Bytecode reflects the characteristics of malware from a lower level.

B. MALWARE CLUSTERING APPROACHES BASED ON DYNAMIC ANALYSIS

Dynamic analysis has the advantage of generating accurate analysis results regardless of the level of anti-static analysis techniques used in the malware, such as code obfuscation, encryption. Thus, dynamic features have been applied to malware clustering in some existing work [10], [11], [21], [27], [28]. Perdisci *et al.* [11] presented a new scalable system for network-level behavioral clustering of HTTP-based malware that aims to efficiently group newly collected malware samples into malware family clusters. Cheng *et al.* [28] proposed a novel malware detection approach based on the family graph. They generated the dependency graph based on the dependency relationship of API calls. At last, they constructed the family dependency graph via clustering the graphs of a known malware family. In [27], a dynamic behavior analysis system is proposed based on three levels of API calls. The sequences of API calls are dealt with as text-like terms and terms are constructed with N-grams for the clustering process. Shamsi *et al.* [10] proposed a hierarchical clustering algorithm based on the API call sequences. By comparing the accuracy of Optimal Matching (OM), the Longest Common Subsequence (LCS), and Long Common Prefix (LCP) [29], [30] for calculating the distance of sequences, the LCP is chosen as the sequence comparison algorithm for hierarchical clustering. A similar approach has also been proposed in [21]. The difference is that they combined the hierarchical algorithm with the density-based algorithm for fast clustering.

A critical limitation of dynamic analysis is that only a single execution path is examined. Another shortcoming is that dynamic analysis is less effective to malware which performs malicious behavior when certain conditions are met. However, static analysis can obtain more complete sample behaviors. In this paper, static features are combined to compensate for the deficiency of dynamic analysis. Moreover, the above methods based on API calls only consider whether the API appears, while the weight of the API, which is important to malware family clustering, is not considered. Hence, we supplement each API feature with its weight. The weight reflects the different importance of distinct families, thus these API features with weights can better represent the behaviors of malware.

C. FEATURE SELECTION AND SIMILARITY MEASUREMENT

The original feature vectors are usually of high dimensions with some useless features, increasing in time cost and decreasing in accuracy [31]–[35]. Therefore, it is necessary to perform certain dimensionality reduction on features. In [31], the performance of machine learning algorithms with and without feature selection by Information Gain is evaluated. The results show that the machine learning algorithms with feature selection perform better. An efficient bi-gram extraction technique was proposed in [32], then an enhanced feature selection based on a genetic algorithm is used. In text document clustering field, many feature selection methods have already been proposed to optimize the feature selection [33]–[35]. In deep learning, there are similar methods [28], [36], such as using attention mechanism to weight features to obtain better results. Therefore, feature selection is also an important part of malware analysis. In malware classification and clustering research, Information Gain is a relatively common method to reduce the dimension of features.

After feature selection, the next step is to calculate the similarity for the features. There are many traditional similarity measurements, such as Euclidean distance [4], [14], [15], and edit distance [16]. In recent deep learning studies, some new methods have emerged, such as Siamese Network or its variants [37]–[39]. The Siamese Network feeds the two inputs into the two neural networks. After mapping the inputs into the new space respectively, the representations of the inputs are formed in the new space. The similarity of two inputs is evaluated through the calculation of Loss. EMD [40] is also a similarity metric for two distributions. It is a dynamic programming distance which can be used to calculate the minimum cost (distance) for transforming one distribution into another distribution. Therefore, this paper regards the features as a distribution and then uses EMD to calculate similarity for different distributions in this paper. To the best of our knowledge, there is no related work that EMD is used for malware detection.

D. MACHINE LEARNING IN CLUSTERING

Machine learning algorithms are widely used in clustering [6], [8], [14], [15], [41]–[47]. Shen *et al.* [41] proposed a real-time image superpixel segmentation method by using the density-based spatial clustering of application with noise (DBSCAN) algorithm. Wang *et al.* [6] proposed a density-based fast clustering algorithm that automatically identified variants of malware from known families. F. A. Shamsi et al. [10] proposed an unsupervised learning (clustering) framework to complement the supervised learning approach. Y. N. Zhang et al. [14] proposed a novel scalable malware analysis framework to leverage the complementary nature of different features and algorithms for further improving the analysis result. Pitolli *et al.* [15] used the BIRCH clustering algorithm for clustering with the dynamic and static features extracted by the Cuckoo Sandbox [48] and then evaluated the ground truths of the sample labels by two different approaches.

These current clustering algorithms applied in malware clustering are mostly supervised learning algorithms, such as hierarchical clustering algorithms and density clustering algorithms. Different from the above algorithms, this paper adopts a semi-supervised clustering algorithm for malware family clustering. We consider the hierarchical clustering algorithms are sensitive to noise and outliers. Moreover, these algorithms can't handle different-sized clusters and convex shaped clusters which are common in malware clustering. Thus we design our clustering algorithm based on density-based clustering algorithm (DBSCAN). By involving supervision into the origin DBSCAN algorithm and Optimizing the process of data query, the proposed clustering algorithm obtains better clustering results and reduces the time cost.

III. THE PROPOSED APPROACH

The clustering approach proposed in this paper comprises four parts, including preparation, feature extraction, similarity measurement, and S-DBSCAN algorithm. The preparation includes the collection of datasets and the construction of the analysis environment. Then, the process of the feature extraction comprises two parts including the static feature extraction and dynamic feature extraction. The static features consist of the Bytecodes and their weights. The dynamic features comprise API calls and their weights. The similarity measurement metric chosen is EMD, which calculates the distance of feature distributions. Based on the average EMD of static feature vectors and dynamic feature vectors, the similarity matrix is obtained by calculating the EMD of the hybrid features with different ratios of static features and dynamic features. Finally, based on our similarity matrix and a little supervision information, we perform semi-supervised clustering by S-DBSCAN algorithm for 1124 samples from

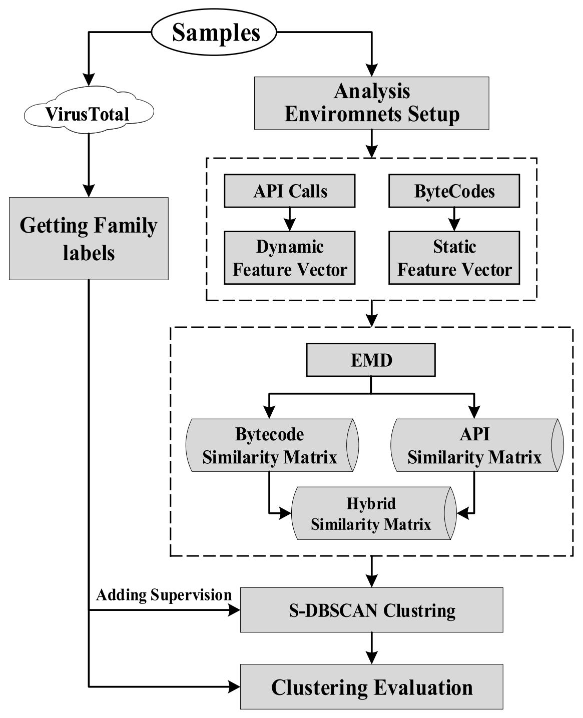


FIGURE 1. **Malware family clustering framework.**

9 families. The process of our clustering approach is shown in Fig. 1.

A. PREPARATION

Many studies [49], [50] perform malware analysis based on the datasets collected from VirusShare. The dataset used in this paper follows these previous work. Since Cuckoo Sandbox supports only parts of formats, the samples with unsupported formats are removed. For evaluating the effect of our approach, the ground truths of the whole samples are required. Then AVClass [51] is leveraged to generate a family label for each sample and some families that contain only a few samples are removed. The detailed dataset information is described in section IV. The analysis environment is based on Cuckoo Sandbox. Cuckoo Sandbox is an open-source malware behavior analysis system. To capture the complete behaviors as far as possible, the frequently detected properties for the existing Cuckoo Sandbox are modified.

B. FEATURE EXTRACTION

The detailed feature extraction process is shown in Fig. 2. This process comprises API feature extraction and Bytecode feature extraction. During feature extraction, we not only obtain these origin features, but also consider their weights. The feature signature and the weight histogram are constructed with these features and their weights. Then the features are pruned according to the Information Gain by removing those useless features. After that, we calculate the corresponding EMD matrix for each signature and form the final hybrid EMD matrix according to different proportions. The details are in the following sections.

Algorithm 1 Bytecode Extraction Algorithm

```
Input:
    A binary sample A;
Output:
    Bytecode signature $P$=($p_1$,$w_1$),($p_2$,$w_2$)...($p_n$,$w_n$)
1:  Reading the binary file in hexadecimal format $A \Rightarrow Bytecode$;
2:  for each $i \in [0, 255]$ do
3:      $num_i = 0$
4:      for $j = 0; j < len(Bytecode); j++$ do
5:          if $Bytecode[j] = i$ then
6:              $num_i = num_i + 1$
7:          end if
8:          $w_i = num_i / len(Bytecode)$
9:          $p_i = i$
10:         $tuple_i = (p_i, w_i)$
11:         $P.append(tuple_i)$
12:     end for
13: end for
14: return $P$;
```

1) BYTECODE FEATURE EXTRACTION

The binary Bytecode features are static features extracted from a binary file, which contain abundant information of the sample. Binary Bytecodes appear differently in each family. Thus, it is an excellent choice for grouping samples to different family clusters with Bytecode. The variants of malware sample generally change only some parameters, such as the C&C server address, some instructions added or deleted, instruction confusion and rearrangement. Such masquerading, confusing techniques or changes in the settings tend to change a little in the Bytecode. Nataraj also proved that in [52]. Thus, Bytecodes and their weights are chosen as features to represent the binary samples. The corresponding feature extraction algorithm is summarized in Algorithm 1.

The input of the algorithm is a binary file, the output of the algorithm is its Bytecode feature signature. The value of Bytecode ranges from 0 to 255 and its weight ranges from 0 to 1. A Bytecode array is generated by reading the binary file in the hexadecimal format (Line 1). We iterate over each Bytecode of the array and count the number of occurrence of each Bytecode (Line 2 to Line 13). Then the weight of this Bytecode is obtained through dividing the number by the total number of Bytecodes. Finally, Bytecode's value and weight make up a two-tuple group (p_i, w_i). Each feature is represented by a two-tuple group (p_i, w_i), p_i represents a Bytecode's value, w_i represents its weight. Finally, a feature signature is generated to represent the whole file $P = \{(p_1, w_1), (p_2, w_2) \ldots (p_n, w_n)\}$.

2) API FEATURE EXTRACTION

API is the interface between the application and the system. The malware implements diverse functions and operations by calling different API calls. As shown in Fig. 3, if a sample performs file operations, the APIs related to file operations will

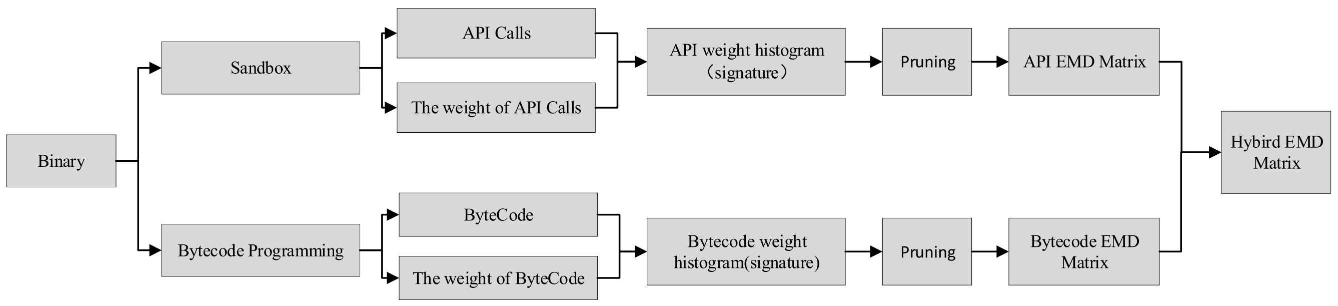


FIGURE 2. Feature extraction process.

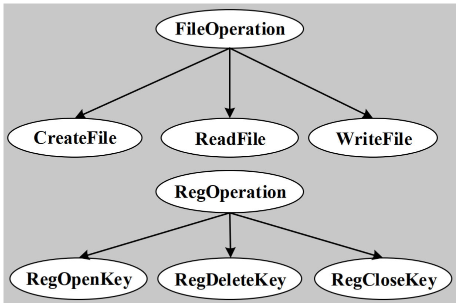


FIGURE 3. Behavior corresponding to the API.

be called, such as CreateFile, ReadFile, WriteFile, etc. Then if it performs registry operations, the APIs related to registry operations will be called, such as RegOpenKey, RegCreateKey, RegDeleteKey, etc. Thus, many malware researchers do their study based on API calls [10], [21], [27].

Based on the considerations above, API calls are used to represent the dynamic behaviors of malware. Although the API calls are able to obtain through static analysis [53], static analysis is vulnerable to the effects of many factors, such as Packing, and encryption. These factors will lead that the complete API calls cannot be obtained. Therefore, dynamic analysis is used to obtain the API calls of the samples. The extraction algorithm is described in Algorithm 2.

The input of Algorithm 2 is the sample dataset, the output of the algorithm is the feature signature of all samples.

All the API sequences of all samples are captured by executing them in an improved Cuckoo Sandbox environment, then all the API sequences $S = \{s_1, s_2, s_3 \ldots s_n\}$ appeared are numbered for further calculation. We describe our method of obtaining the API sequences for each sample and then calculate the total number of all API sequences that have appeared (Line 1 to Line 2). Each API of the dataset is iterated and the number of occurrence of each API is counted (Line 3 to Line 18). The weight of API is calculated through dividing the number by the total number of all the API appeared and a two-tuple group(x_i, w_i) is formed by the API and its weight. Finally, we construct a feature signature $X = \{(x_1, w_1), (x_2, w_2) \ldots (x_n, w_n)\}$ made of tuples(x_i, w_i) for each sample and then a feature signature sets $P = (X_1, X_2, X_3 \ldots X_n)$ of all the samples are produced.

Algorithm 2 API Extraction Algorithm

```
Input:
    Dataset D;
Output:
    The feature vector of all samples
    $P = (X_1, X_2, X_3 \ldots, X_n)$, and $X_i = (x_1, w_1), (x_2, w_2), (x_3, w_3), \ldots, (x_n, w_n)$
1:  Running all the samples in Cuckoo and obtain the API
    sequence for each sample $S_i$
2:  Calculating the total number T of all API sequences that
    have appeared
3:  for $m = 0; m < len(D); m++$ do
4:      $S_m = (s_1, s_2, \ldots s_n))$
5:      for $i = 0; i < T; i++$ do
6:          $num_i = 0$
7:          for $j = 0; j < len(S_m); j++$ do
8:              if $S_m[j] = s_i$ then
9:                  $num_i = num_i + 1$
10:             end if
11:         end for
12:         $w_i = num_i / len(S_m)$
13:         $x_i = s_i$
14:         $tuple_i = (x_i, w_i)$
15:         $X_m.append(tuple_i)$
16:     end for
17:     $P.append(X_m)$
18: end for
19: return $P$;
```

3) FEATURE SELECTION AND FUSION

In this paper, we employ the Information Gain algorithm for feature selection. After feature selection, to better combine dynamic with static features and balance the influence of both features, dynamic features and static features are not mixed directly. We calculate the average EMD of dynamic features and static features based on statistical analysis and then calculate the respective proportions based on the average EMD values. The ratio of different features in hybrid features is calculated in Eqs. (1).

$$ratio(x) = \frac{\sum_{i=1}^{n} EMD(y_i)/n}{\sum_{j=1}^{m} EMD(x_j)/m + \sum_{i=1}^{n} EMD(y_i)/n} \tag{1}$$

EMD is a method for similarity calculation. The detailed introduction is in the following section. In the Eqs. (1), m and n are the numbers of the static and dynamic features, x_i and y_i represent static feature and dynamic feature, respectively.

C. SIMILARITY MEASUREMENT

EMD is the same as the Euclidean distance. It is also a definition of a distance metric. EMD is defined as follows.

There are two signatures P and Q, which are composed of M and N signatures, respectively. The detailed expressions are shown in Eqs. (2) and (3):

$$P = (p_1, w_{p1}), (p_2, w_{p2}) \ldots (p_M, w_{pM}). \quad (2)$$

$$Q = (q_1, w_{q1}), (q_2, w_{q2}) \ldots (q_N, w_{qN}). \quad (3)$$

p_i is a feature of a sample, such as the bin value of a histogram. w_{pi} is the weight of p_i, such as the height of the bin of a histogram. q_j is another feature of the sample, w_{qj} is the weight of q_j. $|d_ij|$ is the distance matrix of M*N, each term represents the distance between p_i and q_j. Then we wish to find a flow matrix $|f_{ij}|$, each term is the amount of flow from p_i to q_j. Finally, the minimal cost function of two signatures is as Eqs. (4):

$$WORK(P, W, F) = \sum_{i=1}^{M} \sum_{j=1}^{N} d_{ij} f_{ij}. \quad (4)$$

This flow is the solution of linear programming and EMD is the renormalized expression of the cost function, which is divided by the sum of f_{ij}. The expression of EMD is as Eqs. (5).

$$EMD(P, Q) = \sum_{i=1}^{M} \sum_{j=1}^{N} d_{ij} f_{ij} / \sum_{i=1}^{M} \sum_{j=1}^{N} f_{ij}. \quad (5)$$

For malware detection, the samples from the same family always exhibit similarity which is characterized by Bytecodes and API calls at a lower level. The difference between the importance of the same family and distinct family is expressed as the differentiation in weight. The EMD is the distance between the two distributions, which is calculated by considering the feature and its weight. As a result, it is very suitable for the malware family clustering. As shown in Fig. 4, A, B, C, D, and E represent the features of the samples. It is a statistical rule of a certain feature of the samples of normal and family A and family B. For different families and normal samples, the unique behavior of the families will lead to a particularly high weight of the corresponding features, which will cause a very high EMD between different families.

D. S-DBSCAN ALGORITHM

Now, we introduce the proposed semi-supervised clustering algorithms (S-DBSACN) in this section. Prototype-based clustering algorithms need to specify the number of clusters artificially, which is unknown in practical conditions. Hierarchical-based clustering algorithms are not suitable for

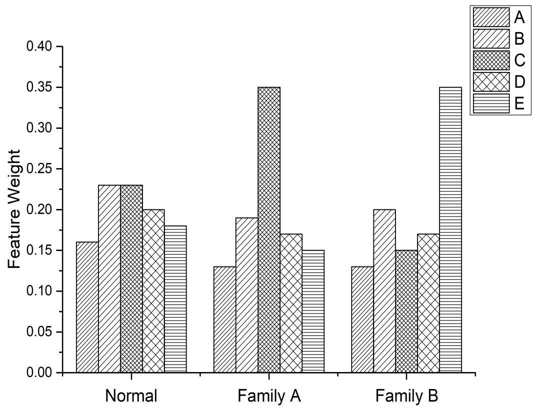


FIGURE 4. Similarity comparison.

large-scale clustering for their high computational complexity. Density-based clustering algorithms need not specify the numbers of clusters and are not sensitive to noise, which is suitable for native data clusters. Thus, we choose the density-based clustering algorithm for malware clustering analysis. DBSCAN algorithm is a typical density-based clustering algorithm, which is widely used in the field of malware clustering [18], [21]. However, the original DBSCAN [54] algorithm applied in malware clustering still needs to be improved in some aspects. On the one hand, the sparse classes will be divided into multiple classes when the sparseness of data is different because of the fixed parameters. These clusters, which are denser and closer, will be merged into one. As a result, the clustering effect is not acceptable. On the other hand, when the dataset is large, the time cost of clustering would be high. In this paper, the original density-based algorithm DBSCAN is improved. First, KD-Tree [55] is used for splitting k-dimensional data space for reducing the time cost. Then, according to the density-reachable principle, some initial clusters are formed. After that, the initial clusters belonging to the same families are merged according to the family labels. Finally, the final clusters are merged again based on the nearest principle in a bottom-up manner.

*Definition 1 (**The nearest cluster of a cluster**):* For cluster A and B, x_i is the core point or only point of the cluster A. If there exists a core point $x_j \in B$ and x_i is in the neighborhood of x_j, at the same time, their similarity is the largest, then we regard that the nearest cluster of cluster A is existing, and it is cluster B.

Now, we introduce the S-DBSCAN algorithm in detail. The specific pseudo code of the algorithm is as Algorithm 3.

The algorithm comprises three parts. The first part comprises splitting the dataset through the KD-Tree and obtaining the initial clusters according to the density-reachable concept. In the second part, the initial clusters are merged into the corresponding known malware family clusters according to these samples with family labels. In the third part, the remaining clusters are merged into known family clusters or grouped into new unknown malware family clusters.

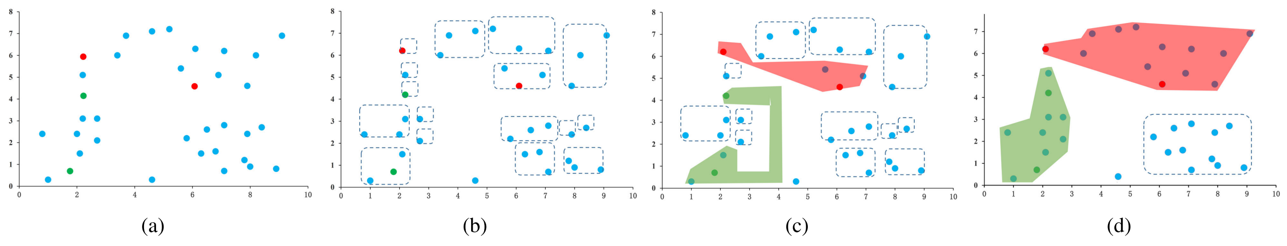


FIGURE 5. S-DBSCAN clustering algorithm schematic diagram.

The inputs of the algorithm include the dataset, labeled samples S, Eps and Minpts. The Eps is the neighborhood radius, and the Minpts is the density threshold. The output of the algorithm is the family clusters C.

Algorithm 3 first initializes some temporary variables T, K and C. T is the core points set, k is the number of clusters and C is the clusters set. The KD-Tree is used for splitting k-dimensional data space (Line 1). Algorithm 3 calculates all the core points based on the definition of the concept of the core object that if the number of sample points in a given object's neighborhood is over or equal to MinPts, the object is regarded as the core object (Line 2 to Line 6). The detailed process that groups the samples into the initial clusters according to the principle of density-reachable is introduced (Line 7 to Line 25). As shown in Fig. 5(a), Red and green points represent the different samples with different family classes. As shown in Fig. 5(b), an initial cluster set *C* is obtained.

The initial clusters are merged into the corresponding known malware family clusters according to these samples with family labels (Line 26 to Line 35). For each sample with the family label, if the sample has not been merged into a known family cluster, the initial clusters with the sample are merged into the known malware family clusters. As shown in Fig. 5(c), clusters with the red and green points represent the clusters of known family. For the remaining initial clusters, we search the nearest clusters of each initial cluster iteratively, and then merge the initial clusters into the nearest cluster according to the nearest principle (Line 36 to Line 40). Until the number of clusters does not change, the iterative process finishes. As shown in Fig. 5(d), the remaining initial clusters are merged into known family clusters or grouped into new unknown malware family clusters.

After the clustering process is completed, the evaluation of clustering results is also very important. In this paper, we use the Adjusted Rand Index (ARI) and Purity as the evaluation indicator, the detailed description is in Section IV.

IV. EXPERIMENTAL ANALYSIS

A. EXPERIMENT ENVIRONMENT

The configurations of all the experiments in this paper are shown in Table 1. Cuckoo Sandbox is an open-source software for automatic analysis of suspicious files. This software makes use of custom components that monitor the behaviors of malicious processes while running in an isolated environment. Considering that some samples have the function to escape virtual machines and disguise as normal software, the physical machine is chosen as the guest to run the samples. To make sure that the physical machine can return to a clean state after a sample has been analyzed, a re-imaging platform called Clonezilla [56] is used to accomplish this work. To simulate a normal network environment, InetSim is used to provide a network environment for the sample. The experimental environment is shown in Fig. 6.

TABLE 1. Host and guest environment configuration.

Category	Host Environment	Guest Environment
Processor	E3-1231 v3	i5-3210M
Memory	16GB	8GB
Hard Disk	Samsung 960	SanDisk SSD 250GB
Operation System	Ubuntu18.04	Windows 7
Software	Cuckoo Sandbox	Python2.7

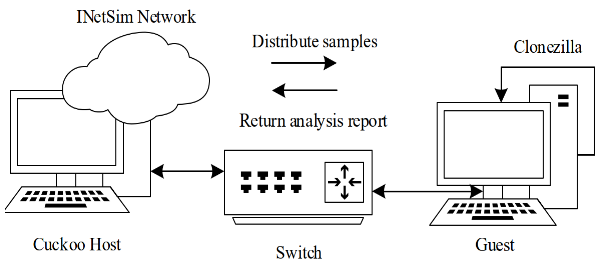


FIGURE 6. Experiment environment.

B. EXPERIMENT DATA

Since Bytecode is selected as feature in this paper, there are no specific requirements for the platform and programming languages of the samples. However, the API calls are extracted by Cuckoo Sandbox and the operating system of the guest machine is Windows. Thus, the specific file formats are chosen, such as PE files, MS-Office files, ZIP archives, and PDF files. To evaluate the clustering results of our work and other work, the ground truth of the families of the samples is needed.

To create a reference dataset with the ground truth of the families, the following approach is taken: First, we obtain a set of 10,600 malware samples that are downloaded from VirusShare [57] which provides malware samples, malware event libraries, analysis and virus samples. Second, each

Algorithm 3 Semi-Supervised Clustering Algorithm S-DBSACN

```
Input:
    Dataset $D = x_1, x_2 \dots x_n$,labeled samples $S = \{s_1, s_2, \dots s_m\}$,$Eps$,$Minpts$
Output:
    Family clusters $C = C_1, C_2 \dots C_k$
1:  Initializing $T = \emptyset, k = 0, C, P = BuildKDTree(D)$
2:  for $i = 1;i < n;i++$ do
3:      if $N_{eps}(x_i) > Minpts$ then
4:          $T = T \cup x_i$
5:      end if
6:  end for
7:  for $j = 0;j < len(T);j++$ do
8:      $P_old = P$
9:      $T_c = \emptyset$
10:     $T_c = T_c \cup x_j$
11:     while $len(T_c)$>0 do
12:         $q \in T_c$
13:         if $N_{eps}(q) > Minpts$ then
14:             $Q = \{x|dist(q, x) < Eps\}$
15:             $S = P \cap Q$
16:             $T_c = T_c \cup S$
17:             $P = P - S$
18:             $T_c = T_c - x_j$
19:         end if
20:     end while
21:     $k = k + 1$
22:     $C_k = P_old - P$
23:     $T = T - (C_k \cap T)$
24:     $C = C \cup C_k$
25: end for
26: for $m = 0;m < k;m++$ do
27:     $Fam_m = \emptyset$
28:     for $i = 0;i < len(S);i++$ do
29:         $x \in S$
30:         if $x \notin Fam_m$ then
31:             $L = \{C_b|x \in C_B, C_B \in C, B = 1, 2, 3 \dots k\}$
32:             $Fam_m = Fam_m \cup L$
33:         end if
34:     end for
35: end for
36: while len(c) not range do
37:     for $i = 0; i < len(c); i++$ do
38:         $C_j = C_i$, which is nearest.
39:         $C_j = C_i \cup C_j$
40:     end for
41: end while
42: return $C$;
```

sample is scanned by 67 different anti-virus programs in VirusTotal [58]. After downloading the reports of these samples from VirusTotal, the AVClass tool is used to generate family labels for samples. We select only those samples for which the majority of the anti-virus programs reported the same family and remove the families with very few samples. This results in a total of 1124 samples from 9 families for experiments. The number of samples of each family is inconsistent for simulating the uneven distribution in real conditions. The detailed information of samples is shown in Table 2.

TABLE 2. Data information.

Malware Family	Quantity	Average Size
Airinstaller	109	2236
Mydoom	129	42
Sofedownloader	112	2856
Capredeam	86	195
Onlinegames	208	79
Sytro	112	132
Fosniw	88	159
Ramnit	150	307
Trymedia	130	320

C. EVALUATION INDICATOR

In this paper, the Adjusted Rand Index (ARI) and Purity are selected as evaluation indicators to evaluate the clustering results.

ARI measures the agreements between the clustering and the partition by class labels. It is defined as the number of pairs of objects which are both placed in the same cluster and the same class, or both in different clusters and different classes, divided by the total number of objects. ARI is set between [-1,1], and the higher the ARI, the more the resemblance between the clustering results and the labels. The detailed definition of ARI is as follows.

K is the clustering result, C is the actual classes, a is the pairs of the same class in both C and K, b is the pairs of the different classes in both C and k. Then Rand Index is defined as Equation 5.

$$RI = \frac{a+b}{C_2^n} \tag{6}$$

The C_2^n represents the total pairs that can be formed with the dataset. The range of RI is from 0 to 1. The larger the value, the more consistent with the real situation. However, for the random conditions, it is not guaranteed that RI is close to 0. The ARI is presented as Equation 6 for guaranteeing that the indicator will be close to 0 in random clustering results. The range of ARI is from -1 to 1. The larger the value, the more consistent with the real situation.

$$ARI = \frac{RI - E(RI)}{max(RI) - E(RI)} \tag{7}$$

Purity is a simple cluster evaluation method, which only needs to calculate the proportion of the correct number of clusters in the total number. Purity reflects the extent to which clusters contain only members of the same class.

$$Purity(W, C) = \frac{1}{N}\sum_{i=1}^{k} max_j|\omega_i \cap c_j| \tag{8}$$

TABLE 3. Comparison of average EMD for API and bytecode.

	Bytecode_EMD		API_EMD	
Family	Same	Different	Same	Different
Airinstaller	0.147	24.68	0.392	24.604
Sytro	0.31	27.446	0.045	14.123
Trymedia	0.112	20.987	3.654	17.08
Capredeam	0.817	21.307	0.192	25.704
Soft32downloader	0.304	20.6	0.192	30.814
Ramit	0.246	64.724	0.633	14.298
Onlinegames	1.762	32.511	0.563	19.747
Mydoom	1.006	21.729	3.323	18.75
Fosniw	1.724	23.211	0.463	16.713

TABLE 4. Comparison of average EMD and Euclidean distance.

Family	API_EMD		API_Euclidean	
	Same	Different	Same	Different
Airinstaller	0.392	24.604	0.202	2.389
Sytro	0.045	14.123	0.076	2.601
Trymedia	3.654	17.08	0.026	1.81
Capredeam	0.192	25.704	0.251	1.588
Soft32downloader	0.192	30.814	1.232	4.368
Ramit	0.633	14.298	0.1	2.264
Onlinegames	0.563	19.747	0.568	1.873
Mydoom	3.323	18.75	0.432	2.863
Fosniw	0.463	16.713	0.113	1.941

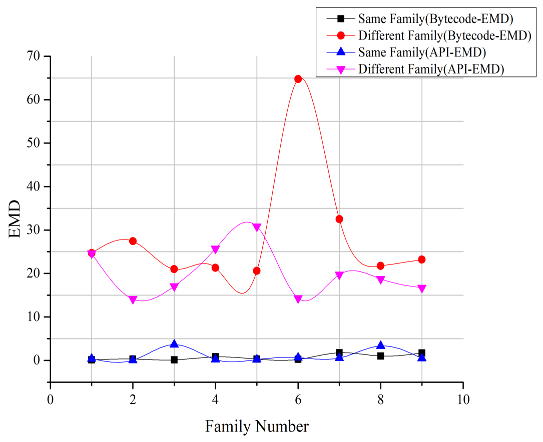


FIGURE 7. Comparison of different features and similarity measurements.

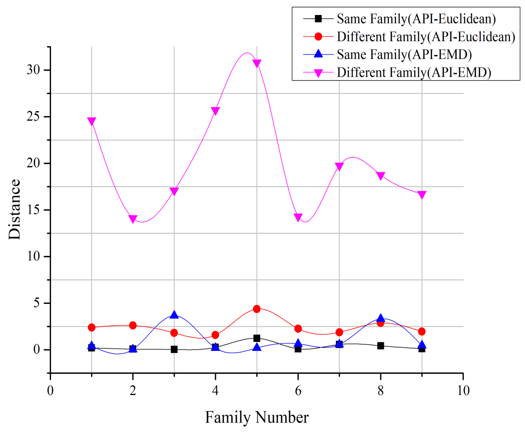


FIGURE 8. Comparison of different features and similarity measurements.

In Equation 8, $\Omega = \{\omega_i, \omega_2 \ldots \omega_k\}$ is the cluster sets. ω_k represents the kth cluster set. $C = \{c_1, c_2 \ldots c_j\}$ is the sample sets and c_j represents the jth sample. N represents the total number of the samples.

D. RESULT ANALYSIS

1) FEATURE SIMILARITY ANALYSIS

First, to show the effectiveness of our proposed similarity measurement method, we perform an average similarity analysis of the same family and different families. As shown in Table 3 and Fig. 7, regardless of the use of the API or Bytecode, the average EMD of the same family is close to 0. This indicates the large similarity of the same family. The average EMD of different families is almost over 10, which shows that there are significant differences between different families. As a result, EMD can be used to distinguish the samples of the same family and different families. For some families, when the Bytecode is chosen as the feature, the difference between the average EMD of different families and the same family is more significant.

Now, we compare the difference between EMD and Euclidean distance in measuring feature similarity. As shown in Table 4 and Fig. 8, API is chosen as a feature to evaluate the effect between EMD and Euclidean distance. It can be seen from Fig. 8 that when Euclidean distance is chosen as a similarity measurement of features, there is little difference between the same family and different families. When EMD is selected as a similarity measurement, samples between the same family and different families can be distinguished obviously. Thus, in this paper, we use EMD as the similarity measurement function of sample features.

2) INVOLVING SUPERVISION INFORMATION

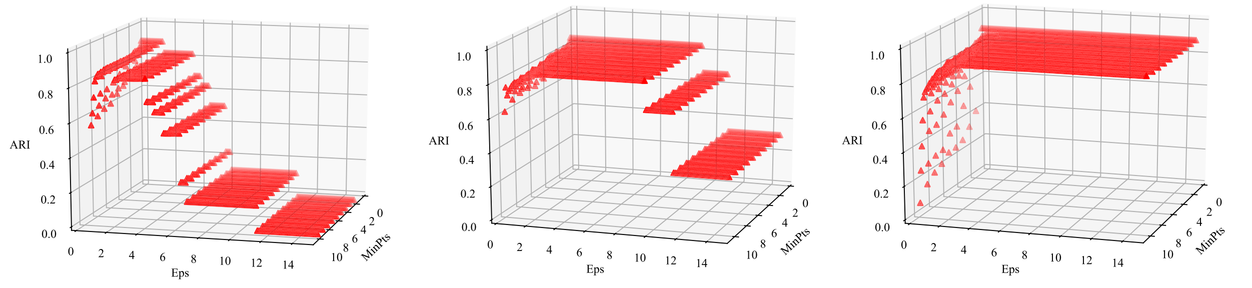
To assess the impact of involving different proportions of the samples with labels to the clustering results, we select 21 different percentages of samples with family class labels to perform clustering. According to different clustering results, a series of ARI values are calculated. The abscissa represents the percentage of the samples with labels as a constraint condition, and the ordinate represents the value of the ARI obtained from the result. After increasing the number of samples with supervised information by about 25%, the clustering result reaches the best state. The experimental results are shown in Fig. 10.

3) DETERMINING EPS AND MINPTS

In this section, the effect of the proposed algorithm under different parameters and the advantages of using hybrid features are shown.

The clustering results vary with the change of Minpts and Eps. To find the best parameters of S-DBSCAN algorithm, the value of Minpts ranges from 1 to 10 whose interval is 1. The value of Eps ranges from 0.1 to 15.0 whose interval is 0.1.

When the feature is Bytecode, its result can be seen from Fig. 9(a). When Eps is between 1.0 and 3.0, the value of the



(a) Determining Eps and Minpts with Bytecode. (b) Determining Eps and Minpts with API. (c) Determining Eps and Minpts with Hybrid features.

FIGURE 9. Determining Eps and Minpts.

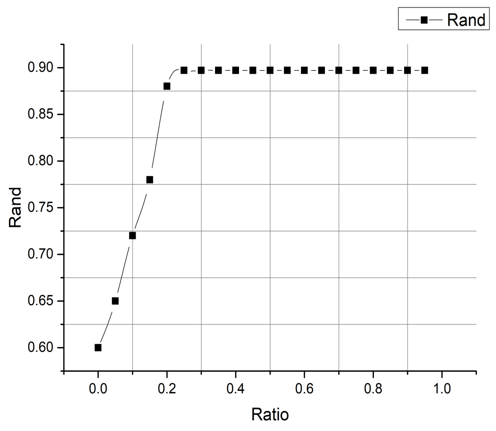


FIGURE 10. Involving different ratio of supervision.

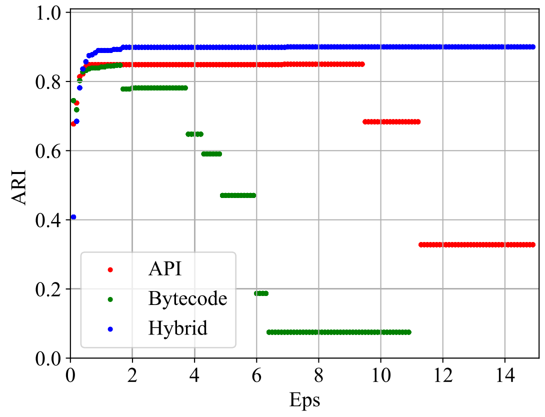


FIGURE 11. Determining Comparison result of different features.

ARI is even higher. When the value of Minpts is 2 and the value of Eps is 1.5, the clustering result is the best and the ARI reaches the highest at 0.831. When Eps is over 2.0, ARI declines with the increase of Eps.

When the feature is API, the results are given in Fig. 9(b). When the value of Eps is between 0.5 and 2.0, the value of the ARI is higher. When Minpts is 2 and Eps is 1.3. The clustering result is the best at this time and the ARI reaches the highest at 0.854. When Eps is over 6.0, ARI declines with the increase of Eps. This shows that when choosing API as a feature, Eps has a wider range for the best performance, which is more conducive to the clustering process.

To better combine dynamic with static features and balance the influence of both features, we select 200 samples randomly and then extract the dynamic and static features separately. The average of EMD is calculated for different features. The ratio of different features in hybrid features is calculated with equation 1. Then we perform our clustering experiment based on hybrid features. The result is shown in Fig. 9(c). When Eps is selected as 0.5 to 2, the ARI obtains a larger value. When the Eps is 0.9, and the Minpts is 2, the ARI reaches a maximum of 0.897. By combining the different features, Eps has the widest range for distinguishing the samples with different families, which is more suitable for practical malware family clustering. This demonstrates that choosing the hybrid features for clustering precedes the single feature.

In summary, when the value of Minpts is 2, the proposed method performs better. As shown in Fig. 11, although selecting API or Bytecode as a feature, the best clustering effect can be achieved is the same, but selection and range of parameters are completely different. When choosing API as the feature, the selection range of Eps is wider. This indicates that the effect of dynamic analysis is slightly better and static analysis for this dataset. However, selecting hybrid features not only optimizes the clustering effect, but also has a wider selection range of Eps, which can greatly reduce the time for parameter selection. Therefore, selecting hybrid features for malware clustering can achieve better results and is more suitable for large-scale cluster analysis.

4) COMPARATIVE EXPERIMENT

To reduce the dimension of the feature without affecting the clustering effect, the influence of information gain on the clustering effect is evaluated. Information Gain is used for feature selection to remove the useless features for clustering. Then the comparison of the results with ARI is shown in Fig. 12.

As shown in Fig. 12, the green line represents the original clustering result with Bytecode features and the red line

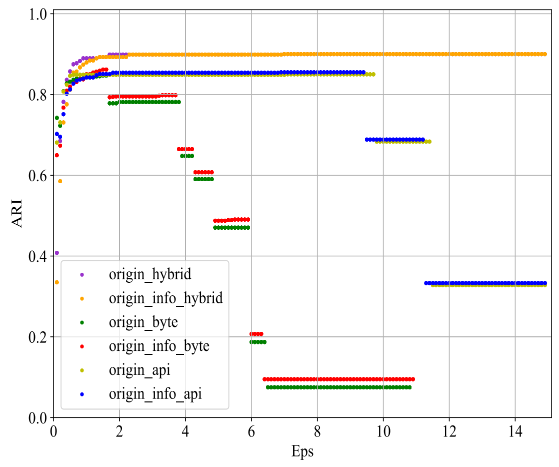


FIGURE 12. Comparison of API features selected by information gain with original features.

TABLE 5. Purity comparison of different algorithms.

Algorithm	ByteCode features	API features	Hybrid features
EM	0.954	0.889	0.962
K-means	0.899	0.841	0.901
Canopy	0.709	0.879	0.891
FarthestFirst	0.891	0.913	0.932
Hierarchical	0.659	0.531	0.663
S-DBSCAN	0.973	0.943	0.987

represents the clustering results with the Bytecode features selected by information gain. When Eps ranges from 0 to 2, the original results are superior to the results after feature selection. When Eps continues to increase, the results are roughly the same.

The blue line represents the original clustering results with API features and the yellow line represents the clustering results with the API features selected by information gain. When Eps ranges from 0 to 2, initially, the results after feature selection are slightly lower than the original results and later the results after feature selection are gradually better than the original results. This shows that after feature selection, the clustering effects are slightly improved.

The darkorchid line represents the original clustering results with hybrid features and the orange line represents the clustering results with the hybrid features selected by information gain. When hybrid features are used to perform clustering, the original results and the results after feature selection are approximately the same. This shows that the feature selection method of information gain is effective and can be used to reduce feature dimensions.

In the case of obtaining the best ARI values, we evaluate the clustering results of all the families with different features which are ByteCode, API and hybrid features. To visually display the comparative results of clustering with other clustering algorithms, purity is chosen as the evaluation indicator and select Bytecode, API and hybrid features respectively as features for clustering. The clustering results are shown in Fig. 13 and Table 5.

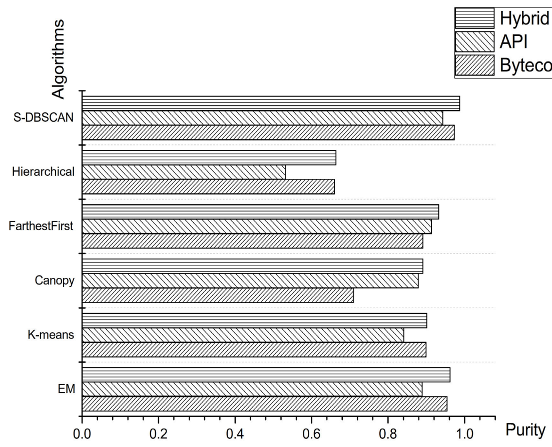


FIGURE 13. Comparing different clustering algorithms with purity.

As shown in Fig. 13 and Table 5, compared with other clustering algorithms, such as Hierarchical, K-means, and Expectation Maximization (EM) algorithms, no matter which feature is used, the clustering result of the S-DBSCAN algorithm is better than other algorithms. And the value of purity is higher than single features when hybrid features are used. As a kind of density-based clustering algorithm, S-DBSCAN need not specify the number of clusters and is suitable for clustering native data with arbitrary shapes with evolutionary characteristics. As a result, S-DBSCAN is more suitable for family clustering. Supervision is introduced into the clustering process, which greatly alleviates the problem of too little or too many clustering clusters caused by the degree of data sparsity. Thus, our clustering algorithm S-DBSCAN outperforms other algorithms in malware family clustering.

E. TIME CONSUMPTION ANALYSIS

At this point, we have shown that our proposed algorithm provides better purity. The time consumption of our proposed algorithm is discussed in this section. As mentioned in section III, the algorithm proposed in this paper is improved based on the DBSCAN algorithm. In this paper, KD-Tree is used to divide the experimental data. These steps of data partitioning and regional queries are combined to reduce the number of accesses to the dataset and reduce the impact of the I/O reading and writing process on the efficiency of the algorithm. The original DBSACN time complexity is $O(N^2)$, which can be reduced to O(Nlog(N)) by speeding up the calculation through the KD-Tree. The experiment results are as for Fig. 14. The algorithmic benefit of our approach becomes more obvious with the increasing of the number of samples.

We perform the clustering experiments ten times and calculate the time consumption of the proposed algorithm and the other clustering algorithms. The experimental results are shown in Table 6. As can be seen from the data in Table 6, the time efficiency of our algorithm exceeds that of most algorithms, such as K-means and Hierarchical

TABLE 6. Time comparison of different algorithms.

Family	First(s)	Second (s)	Third(s)	Fourth(s)	Fifth (s)	Sixth(s)	Seventh(s)	Eighth (s)	Ninth(s)	Tenth(s)
EM	344.930	352.270	332.580	354.479	345.230	352.365	350.061	343.406	335.980	344.615
K-means	0.260	0.257	0.263	0.263	0.256	0.258	0.256	0.259	0.250	0.264
Canopy	0.230	0.225	0.232	0.234	0.221	0.233	0.238	0.233	0.231	0.221
FarthestFirst	0.150	0.157	0.149	0.151	0.142	0.155	0.158	0.148	0.142	0.154
Hierarchical	49.660	50.110	47.98	49.176	49.794	50.160	48.914	50.095	50.194	48.886
S-DBSCAN	0.180	0.179	0.173	0.181	0.180	0.183	0.189	0.175	0.184	0.185

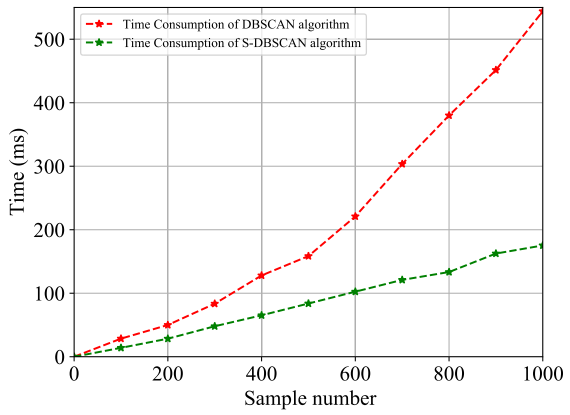


FIGURE 14. Time comparison of DBSCAN and S-DBSCAN.

algorithms, which is only slightly higher than FarthestFirst. But after comparing the purity of the clustering results, the time cost is completely acceptable. And the application scenario of this algorithm is to label numerous unlabeled samples in the anti-virus company's sample library. When the new samples are captured, the new samples can be labeled by performing the family similarity comparison. Since no strict requirements for real-time performance, there is not much consideration for time consumption in this paper.

V. CONCLUSION

In this paper, we propose a semi-supervised density clustering approach to malware family analysis. Our main contributions are in three-folds. First, in feature extraction, we not only use the hybrid features, composed of dynamic and static features but also involve weight information to the features. On the one hand, it is effective and robust to combine the advantages of dynamic analysis with the advantages of static analysis and complement each other. On the other hand, the weight involved expresses the importance of distinct families and help to distinguish samples from different families. In similarity measurement, we leverage a new method called EMD to solve the dynamic programming problems. The similarity comparison experiments show that the EMD performs better than the Euclidean distance in terms of measuring the similarity between feature vectors. Third, in clustering algorithm, we consider the possibility of a semi-supervised clustering algorithm applied in malware clustering. Based on the original DBSACN algorithm, we involve a little supervision to form the S-DBSCAN algorithm. Experiments show that our approach can significantly improve the purity of clustering and reduce the time cost. We will consider other platforms with different file structures and try to explore the defense of the attacks in the future.

REFERENCES

[1] (Apr. 2019). *Symantec Network Security Threat Report*. [Online]. Available:https://resource.elq.symantec.com/e/f2

[2] O. Kostakis, "Classy: Fast clustering streams of call-graphs," *Data Mining Knowl. Discovery*, vol. 28, nos. 5–6, pp. 1554–1585, Sep. 2014.

[3] L. Kellogg, B. Ruttenberg, A. O'connor, M. Howard, and A. Pfeffer, "Hierarchical management of large-scale malware data," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Oct. 2014, pp. 666–674.

[4] E. Gandotra, S. Singla, D. Bansal, and S. Sofat, "Clustering Morphed Malware using opcode sequence pattern matching," *Recent Patents Eng.*, vol. 12, no. 1, pp. 30–36, Mar. 2018.

[5] Z. J. Niu, Z. Qin, J. X. Zhang, and H. Yin, "Malware variants detection using density based spatial clustering with global opcode matrix," in *Proc. Int. Conf. Secur., Privacy Anonymity Comput., Commun. Storage*. Cham, Switzerland: Springer, 2017, pp. 757–766.

[6] C. Wang, Z. Qin, J. Zhang, and H. Yin, "A malware variants detection methodology with an opcode based feature method and a fast density based clustering algorithm," in *Proc. 12th Int. Conf. Natural Comput., Fuzzy Syst. Knowl. Discovery (ICNC-FSKD)*, Aug. 2016, pp. 481–487.

[7] X. Hu, K. G. Shin, S. Bhatkar, and K. Griffin, "MutantX-S: Scalable malware clustering based on static features," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, San Jose, CA, USA, 2013, pp. 187–198.

[8] R. A. Awad and K. D. Sayre, "Automatic clustering of malware variants," in *Proc. IEEE Conf. Intell. Secur. Inform. (ISI)*, Sep. 2016, pp. 298–303.

[9] E. Raff, J. Sylvester, and C. Nicholas, "Learning the PE header, malware detection with minimal domain knowledge," in *Proc. 10th ACM Workshop Artif. Intell. Secur. (AISec)*, 2017, pp. 121–132.

[10] F. A. Shamsi, W. L. Woon, and Z. Aung, "Discovering similarities in malware behaviors by clustering of API call sequences," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2018, pp. 122–133.

[11] R. Perdisci, D. Ariu, and G. Giacinto, "Scalable fine-grained behavioral clustering of HTTP-based malware," *Comput. Netw.*, vol. 57, no. 2, pp. 487–500, Feb. 2013.

[12] K. Berlin, D. Slater, and J. Saxe, "Malicious behavior detection using windows audit logs," in *Proc. 8th ACM Workshop Artif. Intell. Secur. (AISec)*, 2015, pp. 35–44.

[13] X. Y. Deng and J. Mirkovic, "Malware analysis through high-level behavior," in *Proc. 11th USENIX Workshop Cyber Secur. Experimentation Test (CSET)*. Baltimore, MD, USA: USENIX Association, 2018.

[14] Y. Zhang, C. Rong, Q. Huang, Y. Wu, Z. Yang, and J. Jiang, "Based on multi-features and clustering ensemble method for automatic malware categorization," in *Proc. IEEE Trustcom/BigDataSE/ICESS*, Aug. 2017, pp. 73–82.

[15] G. Pitolli, L. Aniello, G. Laurenza, L. Querzoni, and R. Baldoni, "Malware family identification with BIRCH clustering," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2017, pp. 1–6.

[16] S. Dolev, M. Ghanayim, A. Binun, S. Frenkel, and Y. S. Sun, "Relationship of Jaccard and edit distance in malware clustering and online identification (extended abstract)," in *Proc. IEEE 16th Int. Symp. Netw. Comput. Appl. (NCA)*, Oct. 2017, pp. 1–5.

[17] S. Samtani, K. Chinn, C. Larson, and H. Chen, "AZSecure hacker assets portal: Cyber threat intelligence and malware analysis," in *Proc. IEEE Conf. Intell. Secur. Inform. (ISI)*, Sep. 2016, pp. 19–24.

[18] S. W. Wang, B. S. Wang, T. Yong, and B. Yu, "Malware clustering based on SNN density using system calls," in *Proc. Int. Conf. Cloud Comput. Secur.* Cham, Switzerland: Springer, 2015, pp. 181–191.

[19] S. K. Sahay and A. Sharma, "Grouping the executables to detect malwares with high accuracy," *Procedia Comput. Sci.*, vol. 78, pp. 667–674, Apr. 2016.

[20] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in *Proc. 6th ACM Conf. Data Appl. Secur. Privacy (CODASPY)*, 2016, pp. 183–194.

[21] J. Abdullah and N. Chanderan, "Hierarchical density-based clustering of malware behaviour," *J. Telecommun., Electron. Comput. Eng. (JTEC)*, vol. 9, nos. 2–10, pp. 159–164, 2017.

[22] A. Sami, B. Yadegari, H. Rahimi, N. Peiravian, S. Hashemi, and A. Hamze, "Malware detection based on mining API calls," in *Proc. ACM Symp. Appl. Comput.*, 2010, pp. 1020–1025.

[23] *IDA Pro*. Accessed: Apr. 2019. [Online]. Available: http://www.heaventools.com/overview.htm

[24] *PE Explorer*. Accessed: Apr. 2019. [Online]. Available: http://www.heaventools.com/overview.htm

[25] N. Singh and S. S. Khurmi, "ByteFreq: Malware clustering using byte frequency," in *Proc. 5th Int. Conf. Rel., INFOCOM Technol. Optim. (Trends and Future Directions) (ICRITO)*, Sep. 2016, pp. 333–337.

[26] *Microsoft Malware Classification Challenge (Big 2015)*. Accessed: May 2019. [Online]. Available: https://kaggle.com/c/malware-classification

[27] G. J. Széles and A. Coleşa "Malware clustering based on called API during runtime," in *Proc. Int. Workshop Inf. Oper. Technol. Secur. Syst.* Cham, Switzerland: Springer, 2018, pp. 110–121.

[28] B. Cheng, Q. Tong, J. Wang, and W. Tian, "Malware clustering using family dependency graph," *IEEE Access*, vol. 7, pp. 72267–72272, 2019.

[29] C. H. Elzinga, "Sequence analysis: Metric representations of categorical time series," Dept. Social Sci. Res. Methods, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands, 2007.

[30] A. Gabadinho, G. Ritschard, N. S. Müller, and M. Studer, "Analyzing and visualizing state sequences inRwithTraMineR," *J. Stat. Soft.*, vol. 40, no. 4, pp. 1–37, Sep. 2015.

[31] K. O. Babaagba and S. O. Adesanya, "A study on the effect of feature selection on malware analysis using machine learning," in *Proc. 8th Int. Conf. Educ. Inf. Technol. (ICEIT)*, 2019, pp. 51–55.

[32] H. Parvin, B. Minaei, H. Karshenas, and A. Beigi, "A new N-gram feature extraction-selection method for malicious code," in *Proc. Int. Conf. Adapt. Natural Comput. Algorithms*. Berlin, Germany: Springer, 2011, pp. 98–107.

[33] L. M. Q. Abualigah, *Feature Selection Enhanced Krill Herd Algorithm for Text Document Clustering*. Berlin, Germany: Springer, 2019.

[34] L. M. Abualigah and A. T. Khader, "Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering," *J. Supercomput.*, vol. 73, no. 11, pp. 4773–4795, Nov. 2017.

[35] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "A new feature selection method to improve the document clustering using particle swarm optimization algorithm," *J. Comput. Sci.*, vol. 25, pp. 456–466, Mar. 2018.

[36] W. Wang and J. Shen, "Deep visual attention prediction," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2368–2378, May 2018.

[37] W. Wang, J. Shen, and H. Ling, "A deep network solution for attention and aesthetics aware photo cropping," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 7, pp. 1531–1544, Jul. 2019.

[38] X. Dong, J. Shen, D. Wu, K. Guo, X. Jin, and F. Porikli, "Quadruplet network with one–shot learning for fast visual object tracking," *IEEE Trans. Image Process.*, vol. 28, no. 7, pp. 3516–3527, Jul. 2019.

[39] X. P. Dong and J. B. Shen, "Triplet loss in Siamese network for object tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 459–474.

[40] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vis.*, vol. 40, no. 2, pp. 99–121, Nov. 2000.

[41] J. Shen, X. Hao, Z. Liang, Y. Liu, W. Wang, and L. Shao, "Real–time superpixel segmentation by DBSCAN clustering algorithm," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5933–5942, Dec. 2016.

[42] H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, "Malware detection using machine learning and deep learning," in *Proc. Int. Conf. Big Data Anal.* Cham, Switzerland: Springer, 2018, pp. 402–411.

[43] INFOSEC. (Mar. 2017). *Machine Learning for Malware Detection*. [Online]. Available: https://resources.infosecinstitute.com/machine-learning-malware-detection/

[44] Z. X. Xu, S. Ray, P. Subramanyan, and S. Malik, "Malware detection using machine learning based analysis of virtual memory access patterns," in *Proc. Conf. Design, Autom. Test Eur.*, 2017, pp. 169–174.

[45] I. Santos, J. Nieves, and P. G. Bringas, "Semi-supervised learning for unknown malware detection," in *Proc. Int. Symp. Distrib. Comput. Artif. Intell.* Berlin, Germany: Springer, 2011, pp. 415–422.

[46] L. Chen, M. Zhang, C.-Y. Yang, and R. Sahita, "Semi-supervised classification for dynamic Android malware detection," 2017, *arXiv:1704.05948*. [Online]. Available: https://arxiv.org/abs/1704.05948

[47] K. Zhang, C. Li, Y. Wang, X. B. Zhu, and H. P. Wang "Collaborative support vector machine for malware detection," *Procedia Comput. Sci.*, vol. 108, pp. 1682–1691, Jan. 2017.

[48] *Cuckoo Sandbox*. Accessed: Jun. 2019. [Online]. Available: https://cuckoosandbox.org/

[49] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas, "Malware detection by eating a whole exe," in *Proc. Workshops 32nd AAAI Conf. Artif. Intell.*, 2018.

[50] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics–based online malware detection: Towards efficient real–time protection against malware," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 289–302, Feb. 2016.

[51] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "AVclass: A tool for massive malware labeling," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*. Cham, Switzerland: Springer, 2016, pp. 230–253.

[52] L. Nataraj, S. Karthikeyan, and B. S. Manjunath, "SATTVA: SpArsiTy inspired classificaTion of malware VAriants," in *Proc. 3rd ACM Workshop Inf. Hiding Multimedia Secur.*, 2015, pp. 135–140.

[53] M. K. Shankarapani, S. Ramamoorthy, R. S. Movva, and S. Mukkamala, "Malware detection using assembly and API call sequences," *J. Comput. Virol.*, vol. 7, no. 2, pp. 107–119, May 2011.

[54] M. Ester, H. Kriegel, J. Sander, and X. W. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, vol. 96, 1996, pp. 226–231.

[55] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[56] *Clonezilla*. Accessed: Jun. 2019. [Online]. Available: https://clonezilla.org/

[57] *VirusShare*. Accessed: Apr. 2019. [Online]. Available: https://virusshare.com/

[58] *VirusTotal*. Accessed: Jun. 2019. [Online]. Available: https://www.virustotal.com/gui/home/url

YONG FANG received the Ph.D. degree from Sichuan University, Chengdu, China, in 2010. He is currently a Professor with the College of Cybersecurity, Sichuan University. His research interests include information security, the IoT security, and novel feature learning and their applications on networks and malicious code.

WENJIE ZHANG received the B.Eng. degree from the College of Electronics and Information Engineering, Sichuan University, Chengdu, China, in 2017, where he is currently pursuing the master's degree with the College of Cybersecurity.

His current research interests include vulnerability mining, malicious code analysis, and windows security.

BEIBEI LI received the B.E. degree (awarded outstanding graduate) in communication engineering from the Beijing University of Posts and Telecommunications, China, in 2014, and the Ph.D. degree (awarded full research scholarship) from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2019. He was a Visiting Researcher with the Faculty of Computer Science, University of New Brunswick, Canada, from March to August 2018, and the Networked Sensing and Control (NESC) Group, College of Control Science and Engineering, Zhejiang University, China, from February to April 2019. He joined the College of Cybersecurity, Sichuan University, China, in April 2019, where he has been an Associate Professor. His research studies have been published in the IEEE Transactions on Information Forensics and Security, the IEEE Transactions on Industrial Informatics, *ACM Transactions on Cyber-Physical Systems*, and the IEEE Internet of Things Journal, *Information Sciences*. His research studies have been published the IEEE GLOBECOM and the IEEE ICC. His research interests include cyber-physical system security, with a focus on intrusion detection techniques, applied cryptography, and big data privacy in smart grids and industrial control systems. He has served as a TPC Member of several international conferences, including the IEEE GLOBECOM, the IEEE ICNC, and WCSP.

FAN JING received the M.S. degree in computer science from Chongqing University, Chongqing, China, in 2010. He is currently an Assistant Researcher with the College of Cybersecurity, Sichuan University. His research interests include malware detection, artificial intelligence, and network security.

LEI ZHANG received the Ph.D. degree from Sichuan University, Chengdu, China, in 2015. He is currently an Assistant Researcher with the College of Cybersecurity, Sichuan University. His research interests include information security and machine learning.