Automatic Malware Categorization Using Cluster Ensemble

Yanfang Ye
Dept. of Computer Science
Xiamen University
Xiamen, 361005, P.R.China
yeyanfang@yahoo.com.cn

Tao Li
School of Computer Science
Florida International University
Miami, FL, 33199, USA
taoli@cs.fiu.edu

Yong Chen
Internet Security R&D Center
Kingsoft Corporation
Zhuhai, 519015, P.R.China
chenyong@kingsoft.com

Qingshan Jiang
Software School
Xiamen University
Xiamen, 361005, P.R.China
qjiang@xmu.edu.cn

ABSTRACT

Malware categorization is an important problem in malware analysis and has attracted a lot of attention of computer security researchers and anti-malware industry recently. Today's malware samples are created at a rate of millions per day with the development of malware writing techniques. There is thus an urgent need of effective methods for automatic malware categorization. Over the last few years, many clustering techniques have been employed for automatic malware categorization. However, such techniques have isolated successes with limited effectiveness and efficiency, and few have been applied in real anti-malware industry.

In this paper, resting on the analysis of instruction frequency and function-based instruction sequences, we develop an Automatic Malware Categorization System (AMCS) for automatically grouping malware samples into families that share some common characteristics using a cluster ensemble by aggregating the clustering solutions generated by different base clustering algorithms. We propose a principled cluster ensemble framework for combining individual clustering solutions based on the consensus partition. The domain knowledge in the form of sample-level constraints can be naturally incorporated in the ensemble framework. In addition, to account for the characteristics of feature representations, we propose a hybrid hierarchical clustering algorithm which combines the merits of hierarchical clustering and k-medoids algorithms and a weighted subspace K-medoids algorithm to generate base clusterings. The categorization results of our AMCS system can be used to generate signatures for malware families that are useful for malware detection. The case studies on large and real daily malware collection from Kingsoft Anti-Virus Lab demonstrate the effectiveness and efficiency of our AMCS system.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; D.4.6 [**Operating System**]: Security and Protection - Invasive software

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'10, July 25–28, 2010, Washington, DC, USA.
Copyright 2010 ACM 978-1-4503-0055-1/10/07 ...$10.00.

General Terms

Algorithms, Experimentation, Security

Keywords

malware categorization, cluster ensemble, signature

1. INTRODUCTION

1.1 Malware Categorization

Due to its damage to computer security, malware (such as virus, worms, Trojan Horses, spyware, backdoors, and rootkits) has caught the attention of computer security researchers for decades. Currently, the most significant line of defense against malware is Anti-Virus (AV) software products which mainly use signature-based method to recognize threats. Given a collection of malware samples, these AV venders first categorize the samples into families so that samples in the same family share some common traits, and generate the common string(s) to detect variants of a family of malware samples.

For many years, malware categorization have been primarily done by human analysts, where memorization, looking up description libraries, and searching sample collections are typically required. The manual process is time-consuming and labor-intensive. Today's malware samples are created at a rate of millions per day with the development of malware writing techniques. For example, the number of new malware samples collected by the Anti-virus Lab of Kingsoft is usually larger than $10,000$ per day. There is thus an urgent need of effective methods for automatic malware categorization.

Over the last few years, many research efforts have been conducted on developing automatic malware categorization systems [4, 12, 10, 15, 18, 24]. In these systems, the detection process is generally divided into two steps: feature extraction and categorization. In the first step, various features such as Application Programming Interface (API) calls and instruction sequences are extracted to capture the characteristics of the file samples. These features can be extracted via static analysis and/or dynamic analysis. In the second step, intelligent techniques are used to automatically categorize the file samples into different classes based on computational analysis of the feature representations. These intelligent malware detection systems are varied in their use of feature representations and categorization methods. They have isolated successes in clustering and/or classifying particular sets of malware samples, but they have limitations on the effectiveness and efficiency and few have

IPR2023-01160
KASPERSKY EX1024 PAGE 1

been applied in real anti-malware industry. For example, clustering techniques can be naturally used to automatically discover malware relationships [4, 15, 18]. However, clustering is an inherently difficult problem due to the lack of supervision information. Different clustering algorithms and even multiple trials of the same algorithm may produce different results due to random initializations and stochastic learning methods [23].

1.2 Contributions of The Paper

In this paper, resting on the analysis of instruction frequency and function-based instruction sequences of the Windows Portable Executable (PE) files, we develop AMCS for automatically grouping malware samples into families that share some common characteristics using a cluster ensemble by aggregating the clustering solutions generated by different base clustering algorithms.

To overcome the instability of clustering results and improve clustering performance, our AMCS system use a cluster ensemble to aggregate the clustering solutions generated by different algorithms. We develop new base clustering algorithms to account for the different characteristics of feature representations and propose a novel cluster ensemble framework for combining individual clustering solutions. We show that the domain knowledge in the form of sample-level constraints can be naturally incorporated in the ensemble framework. To the best of our knowledge, this is the first work of applying such cluster ensemble methods for malware categorization. In short, our AMCS system has the following major traits:

- *Well-Chosen Feature Representations:* Instruction frequency and function-based instruction sequences are used as malware feature representations. These instruction-level features well represent variants of malware families and can be efficiently extracted. In addition, these features can be naturally used to generate signatures for malware detection.

- *Carefully-Designed Base Clusterings:* The choice of base clustering algorithms is largely dependent on the underlying feature distributions. To deal with the irregular and skewed distributions of instruction frequency features, we propose a hybrid hierarchical clustering algorithm which combines the the merits of hierarchical clustering and k-medoids algorithms. To identify the hidden structures in the subspace of function-based instruction sequences, we use a weighted subspace K-medoids algorithm to generate base clusterings.

- *A Principled Cluster Ensemble Scheme:* Our AMCS system uses a cluster ensemble scheme to combine the clustering solutions of different algorithms. Our cluster ensemble scheme is a principled approach based on the consensus partition and is able to utilize the domain knowledge in the form of sample-level constraints.

- *Human-in-the-Loop:* In many cases, the domain knowledge and expertise of virus analysts can greatly help improve the categorization results. Our AMCS system offers a mechanism to incorporate the domain knowledge in the form of sample-level constraints (such as, some file samples are variants of a single malware; or some file samples belong to different malware types).

- *Natural Application for Signature Generation:* The categorization results generated by our AMCS system can be naturally used to generate signatures for each malware family. These signatures are very useful for malware detection.

All these traits make our AMCS system a practical solution for automatic malware categorization. The case studies on large and real daily malware collection from Kingsoft Anti-Virus Lab demonstrate the effectiveness and efficiency of our AMCS system. As a result, our AMCS has already been incorporated into Kingsoft's Anti-Virus software products.

1.3 Organization of The Paper

The rest of this paper is organized as follows. Section 2 presents the overview of our AMCS system and Section 3 discusses the related work. Section 4 describes the feature extraction and representation; Section 5 introduces the base clustering methods we proposed to account for different characteristics of feature representations; Section 6 presents the cluster ensemble framework used in our AMCS system. In Section 7, using the daily data collection obtained from Kingsoft Anti-virus Lab, we systematically evaluate the effects and efficiency of our AMCS system in comparison with other proposed classification/clustering methods, as well as some of the popular Anti-Virus software such as Kaspersky and NOD32. Section 8 presents the details of system development and operation. Finally, Section 9 concludes our discussion.

2. SYSTEM ARCHITECTURE

Figure 1 shows the system architecture AMCS and we briefly describe each component below.

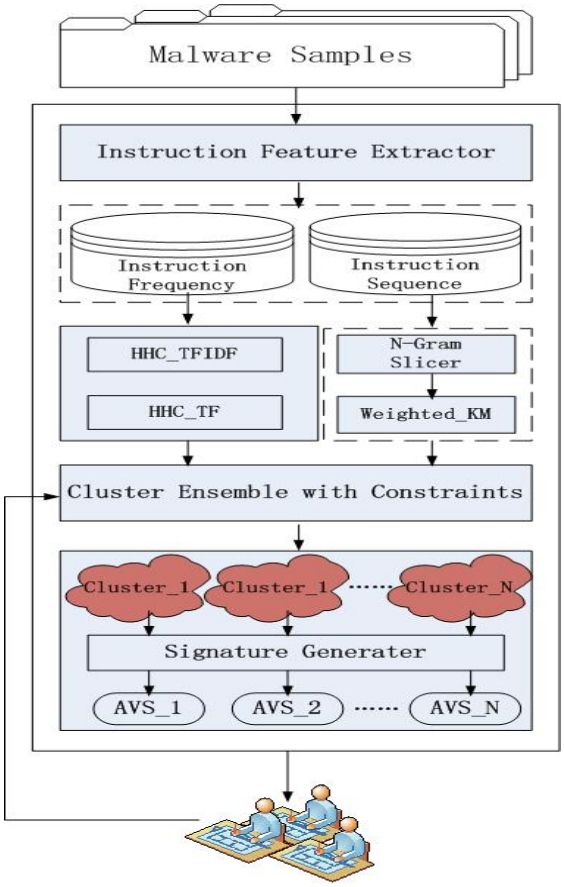

Figure 1: The system architecture of AMCS.

1. *Feature Extractor*: AMCS first uses the feature extractor to extract the function-based instructions from the collected PE malware samples, converts the instructions to a group of 32-bit global IDs as the features of the data collection, and stores these features in the signature database. A sample signature database is shown in Figure 2, in which there are 4 fields: record ID, PE file name, function-based instructions, Instruction IDs. These integer vectors are then transformed to instruction frequencies and function-based instruction sequences features and stored in the database respectively. The transaction data can also be easily converted to relational data if necessary.

Resultset 1

id	filename	instructionseq	invectorofinstruction
1	Adware.Clicker.vb__dl4f57351ef68da183...	sub_004005e8,push,call,sub_004005d3e,retn,sub_004106b4,dcb...	5,6,21,439,434,10,10,10,10,10,10,10,43,10,9,43,488,9...
2	Adware.Doddolook.ic__a1a6af0f8622e458...	sub_100010b7,push,mov,sub,push,push,push,and,push,pop,xor,le...	5,2,7,5,5,5,31,5,22,9,8,425,426,427,2,2,5,22,9,8,425,42...
3	AdWare.Win32.Nodel.kp__037afc07313...	sub_10001000,mov,dec,jnz,mov,mov,mov,retn;sub_10001020,pus...	2,27,12,2,2,2,21,5,5,2,5,2,20,4,2,20,4,5,5,5,6,10,20,4,2,...
10	AdWare.Win32.Zwangi.a__053badc5edd...	sub_10001010,push,mov,mov,add,test,mov,jnz,xor,pop,retn;sub_1...	5,2,2,10,20,2,12,9,22,21,2,2,10,20,434,2,5,2,2,10,20,43...
31	Adware.WSearch.a__ac79431e17508d9...	sub_10001000,ret;sub_10001001,mov,movsx,push,push,push,call...	424,2,24,5,5,5,6,10,424,2,2,424,2,2,9,11,46,2,21,5,2,5,...
32	Adware.WSearch.a__bc1e933c9431dd8...	sub_10001000,mov,movsx,push,push,push,call,add,ret;sub_1000...	2,24,5,5,5,6,10,424,2,2,424,2,2,9,11,46,2,21,5,2,5,5,5,5...
33	Adware.WSearch.a__f6b0cecc483304b0...	sub_10001000,ret;sub_10001001,mov,movsx,push,push,push,call...	424,2,24,5,5,5,6,10,424,2,2,424,2,2,9,11,46,2,21,5,2,5,...
34	Backdoor.DVC.trojan__b0ca17564c9952...	sub_00401000,push,nop,nop,mov,push,push,push,mov,push,nop,...	5,23,23,2,5,5,5,2,5,2,35,5,23,9,9,23,23,22,50,2,50,11,431...
35	Backdoor.Agent.ml__365a041413a0168...	sub_004004b0,mov,mov,lea,push,push,push,push,push,push,pus...	2,2,8,5,5,5,5,5,5,5,5,5,6,20,12,2,5,2,3,490,49,27,5,2,5,5...
36	Backdoor.Agent.ml__e30362daf5b2940e6...	sub_10001000,push,push,push,xor,push,mov,push,mov,push,push,...	5,5,5,9,5,2,5,2,5,5,5,2,5,5,5,5,5,2,6,2,2,2,2,2,2,8,2,2,5...
37	Backdoor.Hugezi.ec__0ad0798bab0320...	sub_10001000,push,mov,xor,mov,mov,mov,lea,push,mov,call,mov...	5,2,9,2,2,2,8,5,2,6,2,22,424,5,2,2,2,20,4,5,5,5,6,10,5,6,...
38	Backdoor.Hugezi.ec__0c14a56cdb806b2f...	sub_10001000,push,mov,xor,mov,mov,mov,lea,push,mov,call,mov...	5,2,9,2,2,2,8,5,2,6,2,22,424,5,2,2,2,20,4,5,5,5,6,10,5,6,...
39	Backdoor.Hugezi.ec__1ae6f942d32737...	sub_00401580,push,mov,push,push,push,mov,push,mov,sub,pus...	5,2,5,5,5,2,5,2,7,5,5,5,9,2,2,2,2,2,2,2,2,2,2,2,8,5,8,5,5,8...
40	Backdoor.Hugezi.ec__5afa6631190320d8...	sub_00404190,push,push,call,mov,push,push,call,test,jz,mov,mov,...	5,5,6,2,5,5,6,20,4,2,2,5,2,5,5,6,5,6,2,22,424,5,5,6,2,5,5,...
43	Backdoor.Hugezi.ec__c9da9e1355a366...	sub_10001190,add,and,add,and,add,and,add,add,pusha,and,add...	10,31,10,31,10,31,10,10,75,31,10,10,43,31,10,10,5,5,5,...
45	Backdoor.Hugezi.ec__d26632a9d9ef157...	sub_10001000,push,push,mov,push,mov,sub,push,mov,mov,lea,c...	5,5,2,5,2,7,5,2,2,8,6,8,2,6,8,2,6,8,2,6,8,2,5,5,6,5,5,5,5,6...
46	Backdoor.Hugezi.ec__e61e03a4fa72626...	sub_00401700,push,mov,push,push,push,mov,push,mov,sub,pus...	5,2,5,5,5,2,5,2,7,5,5,5,9,2,2,2,2,2,2,2,2,2,2,2,8,5,8,5,5,8...
47	Backdoor.Hupigon.AAFC__424c744129d...	sub_004b6285,cmpxd,cmp;sub_004b9406,cmp;sub_004c2000,su...	344,11,11,7,23,75,6,1,22,43,5,424,22,2,10,7,11,12,2,11...
49	Backdoor.Hupigon.AAFC__9515de4b5c02...	sub_0041eb00,bdcxd,adc,test,cmp,and,xchg,add;sub_004b6285,c...	451,58,20,11,31,55,10,344,11,11,7,23,75,6,1,22,43,5,4...
50	Backdoor.Hupigon.AAFC__bea008bd93d...	sub_004b6285,cmpxd,cmp;sub_004c2000,nop;sub_004c2001,pu...	344,11,23,75,6,1,22,43,5,424,22,2,10,7,11,12,2,11,2,12...
51	Backdoor.Hupigon.AAFC__d4b59e47753...	sub_004b6285,cmpxd,cmp;sub_004b9406,cmp;sub_004c2000,su...	344,11,11,7,23,75,6,1,22,43,5,424,22,2,10,7,11,12,2,11...
52	Backdoor.Hupigon.AYPE__e9d458929a6...	sub_005ee2e3,push,mov,pusha,inc,dec,popa,nop,pushf,pusha,ca...	5,2,75,43,27,76,23,180,75,6,22,10,8,11,2,5,23,23,23,23...
53	Backdoor.KillAV.a__a7d3511cd8ce6c36...	sub_00401000,call,push,call;sub_0040100b,push,mov,add,jmp,im...	6,5,6,5,2,10,1,16,483,5,2,5,5,5,6,2,8,2,27,11,12,2,41,21...
55	Backdoor.PcClient.ae__68eca4e757f00a...	sub_10001000,push,mov,push,push,push,push,mov,push,xor,xor,j...	5,2,5,5,5,5,2,5,9,9,4,7,7,23,23,23,2,31,2,9,2,11,431,2,3...
56	Backdoor.PcClient.aq__e471a36a72a62...	sub_00401000,sub,push,mov,cmp,push,jnz,pop,mov,pop,add,ret;s...	7,5,2,11,5,438,22,2,22,10,424,5,2,5,2,5,5,5,6,10,20,12,...
57	Backdoor.PcClient.cr__91b37126129e9e...	sub_10001000,push,mov,push,push,push,push,push,mov,mov,nop,nop,...	5,2,5,5,5,5,2,2,23,23,23,23,23,75,76,9,5,5,5,5,5,6,2,5,5,5,...

Figure 2: A sample signature database after data transformation.

2. *Base clustering algorithms*: Base clustering solutions are generated by applying different clustering algorithms based on different feature representations. The hybrid hierarchical clustering algorithm is applied on the instruction frequency vectors with the tf-idf and tf weighting schemes [2] which are widely used for document representation in IR (information retrieval). The weighted subspace K-medoids partitioned algorithm is applied on the function-based instruction sequences.

3. *Cluster ensemble with constraints*: Cluster ensemble is used to combine different base clusterings. The cluster ensemble is also able to utilize the domain knowledge in the form of sample-level constraints.

4. *Malware family signature generator*: According to the categorization results generated by the cluster ensemble, the "signature" of the malware family will be automatically generated for detecting malware variants.

5. *Human-in-loop*: Our system provides a user-friendly mechanism for incorporating the expert knowledge and expertise of human experts. Virus analysts can look at the partitions and manually generate sample-level constraints. These constraints can be used to improve the categorization performance.

3. RELATED WORK

3.1 Malware Categorization

Various classification approaches including association classifiers, support vector machines, and Naive Bayes have been applied in malware detection [21, 22, 28]. These classification methods require a large number of training samples to build the classification models. In recent years, there are several initiatives in automatic malware categorization using clustering techniques [12]. Bayer et al. [18] used locality sensitive hashing and hierarchal clustering to efficiently group large datasets of malware samples into clusters. Lee et al. [15] adopted k-medoids clustering approach to categorize the malware samples. Several efforts have also been reported on computing the similarities between different malware samples using *Edit Distance* (ED) measure [10] or statistical tests [24]. These techniques have isolated successes in clustering particular sets of malware samples, but they have limitations which leave a large room for improvement and none of them has been applied in real anti-malware industry. In addition, no work has been reported on ensemble different clusterings for malware categorization.

3.2 Clustering Ensemble

Clustering ensemble refers to the process of obtaining a single (consensus) and better-performing clustering solution from a number of different (input) clusterings for a particular dataset [23]. Many approaches have been developed to solve ensemble clustering problems over the last few years [1, 7, 11, 19, 25, 16]. However, most of these methods are designed for combining partitional clustering methods and few have been reported for combining both partitional and hierarchical clustering methods. In addition, they do not take advantage the domain-related constraints. In our work, we use a cluster ensemble to aggregate the clustering solutions generated by both hierarchical and partitional clustering methods. Our ensemble framework is also able to incorporate the domain knowledge in the form of sample-level constraints.

4. FEATURE REPRESENTATION

There are mainly two ways for feature extraction in malware analysis: static extraction and dynamic extraction. Dynamic feature extraction can well present the behaviors of malware files and especially perform well in analyzing packed malware [15, 18]. However, it has limited coverage: only executable files can be executed or simulated. Actually, from the daily data collection of Kingsoft anti-malware lab, more than 60 percent of malware samples are *Dynamic Link Library* (DLL) files which can not be dynamically analyzed. In addition, dynamic feature extraction is time-consuming. Therefore, in our work, we choose static feature extraction methods for malware representation. If a PE file is previously encrypt or compressed by a third party binary compress tool such as UPX and ASPack Shell or embedded a homemade packer, it needs to be decrypt or decompressed first.

We use the dissembler K32Dasm developed by Kingsoft Anti-Virus Lab to dissemble the PE code and output the file of decrypt or unpacked format as the input for feature extraction. In this paper, we use the instructions as the basis for malware representation. Two aspects of the instructions are extracted for our malware categorizers: one is the frequency of each instruction obtained from the disassembled malware sample; the other is the function-based instruction sequence. The extraction and transformation processes are shown in Figure 3.

Comparing with other static features [24], such as construction phylogeny tree, control flow graph, Windows API calls or arbitrary binaries, the instruction frequencies and function-based instruction

IPR2023-01160
KASPERSKY EX1024 PAGE 3

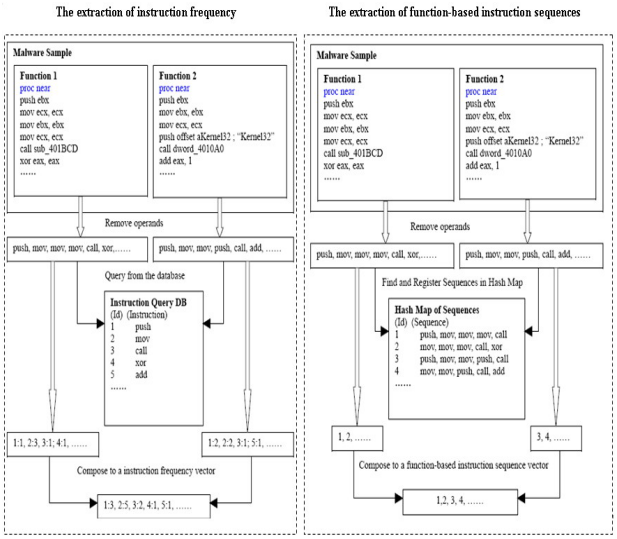

Figure 3: The feature extraction and transformation processes of AMCS.

sequences for malware representation have the following advantages:

1. Great ability for representing variants of a malware family: It has been observed in practice that malware samples in the same family or derived from the same source code share similar shapes of instruction frequency patterns or a large number of basic blocks which can be constructed using function-based instruction sequences. Figure 4 illustrates that the shapes of instruction frequency patterns are similar for the same malware family and they are different for different malware families. Figure 5 gives an example that the trojan family stealing QQ game's passwords shares a large number of basic blocks which can be constructed by function-based instruction sequences.

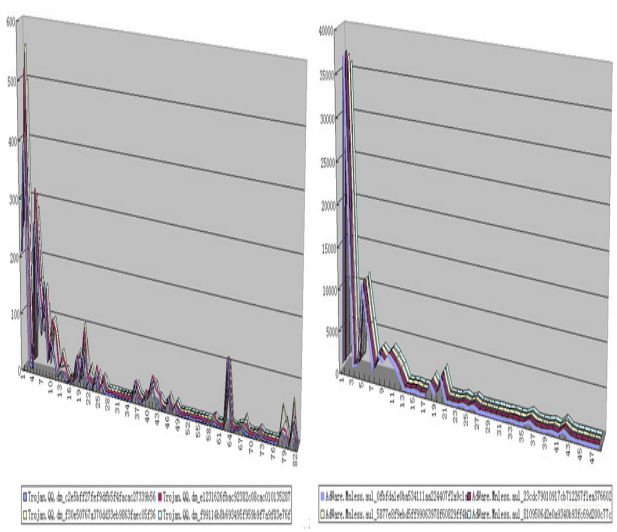

Figure 4: Shapes of instruction frequency patterns are shared by same malware family and differ between different families.

2. Easy for generating signatures for a malware family to detect its variants: Compared with other complex structural features, like construction phylogeny tree or control flow graph, the instruction sequences which are frequently appeared within a malware family but rarely appeared in other families can be used as the signature for AV products to detect malware variants.

3. High coverage rate of malware samples: Both the two instruction features can be extracted from most of the malware samples, while the Window API calls for around half of the malware files can be effectively extracted from import tables.

4. Semantic Implications: Compared with binary strings, instruction features have meaningful semantic characteristics. Segments of instruction sequences can well reflect the functionality of program code pieces.

5. High efficiency for feature extraction: In this paper, in order to improve the system performance, we develop the feature extractor to construct the instruction features of malware samples instead of using a third party disassembler. The instruction feature extractor developed by us can extract more than 100 malware samples per second, which is time saving for the whole malware categorization process.

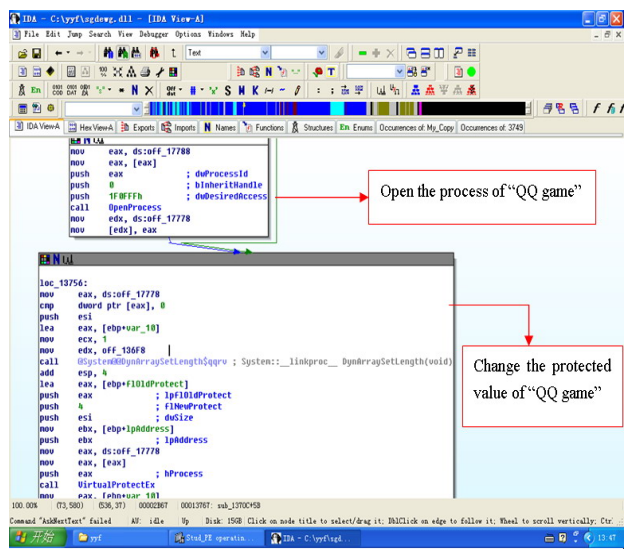

Figure 5: The function-based instruction sequences shared by the "Trojan.QQ.dm" family.

5. BASE CLUSTERINGS

Instruction frequency and function-based instruction sequences are different yet complimentary feature representations for malware analysis. For example, different malware families may have similar shapes of instruction frequency patterns due to the large number of common library codes, but they typically have different function-based sequences. In our work, both of these features are used for generating base clusterings.

In our application, a cluster is a collection of malicious files that share some common traits between them and are "dissimilar" to the malware samples belonging to other clusters. Hierarchical and

partitioning clustering are two common type of clustering methods and each of them has its own traits [27]. Hierarchical clustering method can deal with irregular data set more robustly, while partitioning clustering like k-medoids is efficient and can produce tighter clusters especially if the clusters are of globular shape.

The choice of clustering algorithms is largely dependent on the underlying feature distributions. Figure 6 shows the distribution of instruction frequency on a sample dataset with 1,434 malware samples with 1,222 dimensions. The instruction features with tf-idf scheme have been been extracted and Principal Component Analysis is performed to select the first two and three important dimensions for visualization. As shown in Figure 6, the distribution of malware samples is typically skewed, irregular and of densities. Therefore, in our work, a hybrid hierarchical clustering algorithm which combines the the merits of hierarchical clustering and k-medoids algorithms for malware clustering is proposed to generate base clusterings on instruction frequency features.

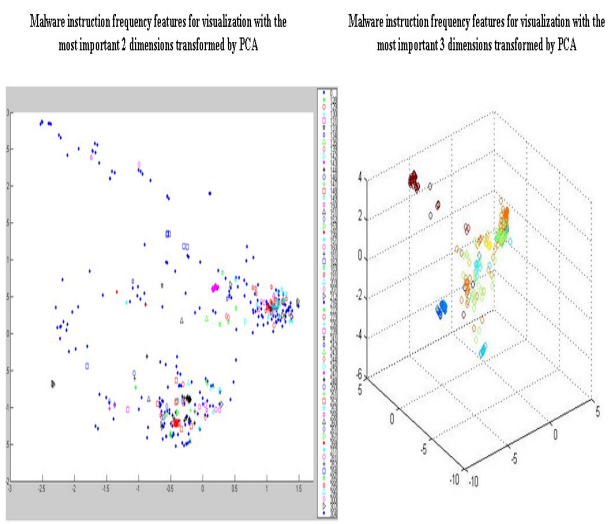

Figure 6: Malware distributions after PCA transformation

On the other hand, for function-based instruction sequences, the correlations among the features are often specific to data locality, in the sense that some malware samples are correlated with a given set of sequences and others are correlated with respect to different sequences. Therefore, effective methods for malware clustering on instruction sequences should explore the associations between the features and clusters. In our work, we use a weighted subspace K-medoids algorithm to generate base clusterings on instruction sequences.

5.1 Hybrid Hierarchical Clustering (HHC)

We propose to use a *Hybrid Hierarchical Clustering* (HHC) which combines the merits of hierarchical clustering and k-medoids algorithms to general base clusterings. HHC utilizes the agglomerative hierarchical clustering algorithm as the frame, starting with N singleton clusters, successively merges the two nearest clusters until only one cluster remains. Different from traditional agglomerative hierarchical clustering algorithms, at each iteration of the merging process, HHC adopts k-medoids algorithm to generate a partition. HHC also computes a cluster validity index at each iteration and generates the best number of clusters by comparing these indice. The outline of HHC is described in Algorithm 1.

We use *Fukuyama-Sugeno index(FS)* [9] as the cluster validity index. FS evaluates the partition by exploiting the compactness within each cluster and the distances between the cluster representatives. It is defined as

Input: The data set D
Output: The best K and data clusters

Set each sample as a singleton cluster;
for $K \leftarrow N-1$ **to** 1 **do**
  Merge two clusters with closest medoids;
  Generate the new medoids of the merged clusters;
  Run K-medoids to obtain a partition;
  Calculate the validity index;
  Compare and keep the best K and corresponding clusters until now ;
end
Return the best K and corresponding clusters.

Algorithm 1: The algorithm description of HHC.

$$FS = \sum_{i=1}^{N}\sum_{j=1}^{nc} u_{ij}^{m}(\|x_i - v_j\|_A^2 - \|v_j - v\|_A^2),$$

where v_j is the medoid [14] of cluster C_j, v is the medoid of the whole data collection , and A is an 1×1 positive definite, symmetric matrix. It is clear that for compact and well-separated clusters, we expect small values for FS.

5.2 Weighted Subspace K-Medoids (WKM)

Existing work on malware clustering fails to explore the associations between the features and malware clusters. The instruction sequence representation extracted from file samples for malware detection is usually of high dimensions and it has been shown that in a high dimensional space the distance between every pair of points is almost the same for a wide variety of data distributions and distance functions [5]. The simplest approach to address this problem is to first use feature selection and dimension reduction techniques to select a set of important features and then perform clustering process [6, 20]. However, the correlations among the instruction sequences are often specific to data locality, in the sense that some malware samples are correlated with a given set of sequences and others are correlated with respect to different sequences. In our work, we propose a weighted subspace K-medoids (WKM) algorithm and use it to generate base clusterings on instruction sequences.

WKM dynamically assigns a weight to every feature for each malware family, which makes the clusters hiding in the subspaces and the common features of the same family can be easily generated [13]. Intuitively, the importance of a feature to a cluster can be estimated by 1) how consistent its values for the samples within the cluster, and 2) how well its values distinguish between samples that are in different clusters. If a feature has a small variation within a cluster and large variations between the cluster and other clusters, then the feature can be viewed as an important feature for the cluster. Formally, denote the feature weight for cluster i as $W_i = (w_{i1}, \cdots, w_{id})$ where w_{ij} denotes the weight of the j-th feature for cluster i and can be updated as follows:

$$w_{ij} = \begin{cases} \dfrac{\sum_{l=1}^{d} D_{il} - D_{ij} + E_{ij}}{(d-1)\sum_{l=1}^{d} D_{il} + \sum_{l=1}^{d} E_{il}}, & \text{if } \sum_{l=0}^{d} D_{il} > 0 \\ \dfrac{1}{d} & \text{Otherwise} \end{cases} \quad (1)$$

where

$$D_{ij} = \sum_{x_t \in C_i} w_{ij}(x_{tj} - m_{ij})^2, E_{ij} = \sum_{x_t \notin C_i} w_{ij}(x_{tj} - m_{ij})^2,$$

C_i is the i-th cluster, and m_{ij} is the j-th feature of the medoid for C_i. Note that $\sum_{l=1}^{d} w_{il} = 1$. Using the feature weight vector, we can compute the weighted distance between data points. The weighted distance is then used for computing the medoids and for assigning points into clusters. The algorithm procedure for WKM is described in Algorithm 2.

Input: N points in d-dimensional space, number of clusters k
Output: k clusters and the corresponding weight vector

Randomly choose k cluster medoids;
set initial weights to be $\frac{1}{d}$;
repeat
  Assign each points to the nearest cluster;
  Update the cluster medoids;
  Update the weight vector using Eq.(1);
  Calculate the validity index;
until *the weight vectors and the medoids do not change*;

Algorithm 2: The algorithm description of WKM.

6. CLUSTER ENSEMBLE

6.1 Introduction

Clustering algorithms are valuable tools for malware categorization. However, clustering is an inherently difficult problem due to the lack of supervision information. Different clustering algorithms and even multiple trials of the same algorithm may produce different results due to random initializations and stochastic learning methods [23, 25]. In our work, we use a cluster ensemble to aggregate the clustering solutions generated by different both hierarchical and partitional clustering algorithms. We also show that the domain knowledge in the form of sample-level constraints can be naturally incorporated in the cluster ensemble. To the best of our knowledge, this is the first work of applying such cluster ensemble methods for malware analysis.

6.2 Formulation

Formally let $X = \{x_1, x_2, \cdots, x_n\}$ be a set of n malware samples. Suppose we are given a set of T clusterings (or partitioning) $\mathcal{P} = \{P^1, P^2, \cdots, P^T\}$ of the data points in X. Each partition $P^t (t = 1, \cdots, T)$ consists of a set of clusters $C^t = \{C_1^t, C_2^t, \cdots, C_K^t\}$ where K is the number of clusters for partition P^t and $X = \bigcup_{\ell=1}^{K} C_\ell^t$. Note that the number of clusters K could be different for different clusterings.

We define the *connectivity* matrix $M(P^t)$ for the partition P^t as

$$M_{ij}(P^t) = \begin{cases} 1 & \text{if } x_i \text{ and } x_j \text{ belong to the same cluster in } C^t \\ 0 & \text{Otherwise} \end{cases} \tag{2}$$

Using the connectivity matrix, the distance between two partitions P^a, P^b can be defined as follows [11, 17]:

$$\begin{aligned} d(P^a, P^b) &= \sum_{i,j=1}^{n} d_{ij}(P^a, P^b) \\ &= \sum_{i,j=1}^{n} |M_{ij}(P^a) - M_{ij}(P^b)| \\ &= \sum_{i,j=1}^{n} [M_{ij}(P^a) - M_{ij}(P^b)]^2 \end{aligned}$$

Note that $|M_{ij}(P^a) - M_{ij}(P^b)| = 0$ or 1.

A general way for cluster ensemble is to find a *consensus* partition $\mathcal{P}^*$ which is the closest to all the given partitions:

$$\begin{aligned} \min_{P^*} J &= \frac{1}{T} \sum_{t=1}^{T} d(P^t, P^*) \\ &= \frac{1}{T} \sum_{t=1}^{T} \sum_{i,j=1}^{n} [M_{ij}(P^t) - M_{ij}(P^*)]^2. \end{aligned} \tag{3}$$

Since J is convex in $M(P^*)$, by setting $\nabla_{M(P^*)} J = 0$, we can easily show that the partition P^* that minimizes Eq.(3) is the consensus (average) association: the ij-th entry of its connectivity matrix is

$$\widetilde{M}_{ij} = \frac{1}{T} \sum_{t=1}^{T} M_{ij}(P^t). \tag{4}$$

PROPOSITION 6.1. *The partition P^* that minimizes Eq.(3) is the consensus (average) association $\widetilde{M}_{ij}$.*

In our application, we construct 6 base categorizers using the algorithms described in Section 5: 1) Two clusterings obtained by applying HHC on the instruction frequency vectors with tf-idf and tf weighting schemes (denoted by HHC_TFIDF and HHC_TF); and 2) Four clustering by applying WKM on the function-based instruction sequences with four different number of clusters: two numbers are those used in HHC_TFIDF and HHC_TF and two numbers are specified by human experts. According to the experience of our malware analysts in Kingsoft Anti-malware Lab, specifying the number of the clusters to be the twentieth or thirtieth of the number of malware features is a practical choice.

Based on Proposition 6.1, we could derive the final clustering from the consensus association $\widetilde{M}_{ij}$. The ij-th entry of $\widetilde{M}_{ij}$ represents the number of times that sample i and j have co-occurred in a cluster. We could then use the following simple strategy to generate the final clustering: 1) For each sample pair, (i, j), such that $\widetilde{M}_{ij}$ is greater than a given threshold (in our application, the threshold is $0.5 \times 6 = 3$), than assign the samples to the same cluster. If the samples were previously assigned to two different clusters, then merge these clusters into one. 2) For each remaining sample not included in any cluster, form a single element cluster. Note that we do not need to specify the number of clusters.

6.3 Incorporating sample-level constraints

We also show that the domain knowledge in the form of sample-level constraints can be naturally incorporated in the cluster ensemble. In this scenario, in addition to t partitions, we are also given two sets of pairwise constraints (1) Must-link constraints.

$$A = \{(x_{i1}, x_{j1}), \cdots, (x_{ia}, x_{ja})\}, a = |A|,$$

where each pair of points are considered similar and should be clustered into the same cluster. (2) Cannot-link constraints.

$$B = \{(x_{p1}, x_{q1}), \cdots, (x_{pb}, x_{pb})\}, b = |B|,$$

where each pair of points are considered dissimilar and they cannot be clustered into the same clusters. Such constraints have been widely used in *semi-supervised clustering* [3], however, few research efforts have been reported on incorporating constraints for cluster ensemble [26].

To incorporate the constraints in A and B into cluster ensemble, we need to solve the following problem:

$$\begin{aligned} \min_{P^*} J &= \frac{1}{T}\sum_{t=1}^{T}\sum_{i,j=1}^{n}[M_{ij}(P^t) - M_{ij}(P^*)]^2 \quad (5) \\ s.t. \quad & M_{ij}(P^*) = 1, \text{ if } (x_i, x_j) \in A \\ & M_{ij}(P^*) = 0, \text{ if } (x_i, x_j) \in B \end{aligned}$$

Eq.(5) is a convex optimization problem with linear constraints. Let $C = A \bigcup B$ be the set of all constraints, then $c = |C| = |A| + |B|$. We can represent C as $C = \{(x_{i1}, x_{j1}, b_1), \cdots, (x_{ic}, x_{jc}, b_c)\}$ where $b_s = 1$ if $(x_{is}, x_{js}) \in A$ and $b_s = 0$ if $(x_{is}, x_{js}) \in B, s = 1 \ldots c$. Similar to [26], we can then rewrite Eq.(5) as

$$\begin{aligned} \min_{P^*} J &= \frac{1}{T}\sum_{t=1}^{T}\sum_{i,j=1}^{n}[M_{ij}(P^t) - M_{ij}(P^*)]^2 \quad (6) \\ s.t. \quad & (\mathbf{e}_{i_s})M(P^*)\mathbf{e}_{j_s} = b_s, \ s = 1, 2, \cdots, c \end{aligned}$$

where $\mathbf{e}_{i_s} \in \mathbb{R}^{n\times 1}$ is a indicator vector with only the i_s-th element being one and all other elements being zero. Now we introduce a set of Lagrangian multipliers $\{\alpha_i\}_{i=1}^{c}$ and construct the Lagrangian for problem (6) as

$$\mathcal{L} = J + \sum\nolimits_s \alpha_s \left((\mathbf{e}_{i_s})^T M(P^*)\mathbf{e}_{j_s} - b_s\right) \quad (7)$$

Note that $(\mathbf{e}_{i_s})^T M(P^*)\mathbf{e}_{j_s} = M_{i_s j_s}(P^*)$. Hence we can show that the solution to problem (6) is:

$$M_{i_s j_s}(P^*) = \begin{cases} \frac{1}{T}\sum_{t=1}^{T} M_{ij}(P^t) & \text{if } (i_s, j_s) \text{ is not in } C \\ b_s & \text{Otherwise} \end{cases} \quad (8)$$

In other words, the solutions for regular elements in $\widetilde{M}_{ij}$ do not change. And for constrained elements, according to Eq.(8), we need to set the corresponding entries of the consensus association $\widetilde{M}_{ij}$ to be the exact values based on their constraints [26].

7. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we conduct four sets of experimental studies using our data collection obtained from the Kingsoft's Anit-Virus Lab to evaluate the categorization methods we proposed in this paper: (1) In the first set of experiments, on the basis of instruction frequency features with tf-idf and tf weighting schemes, we compare our proposed Hybrid Hierarchical Clustering(HHC) algorithm with individual hierarchical and K-medoids clustering methods. (2) In the second part of experiments, resting on the analysis of function-based instruction sequences, we evaluate our proposed weighted subspace K-medoids clustering algorithm by comparing it with individual K-medoieds clustering method. (3) In the third experiment set, we compare and evaluate the malware categorization results of AMCS system. (4) In the last set of experiments, we compare our AMCS system with some of the popular anti-malware software products such as Norton AntiVirus, Bitdefender, MaAfee VirusScan and Kaspersky Anti-Virus. All the experimental studies are conducted under the environment of Windows XP operating system plus Intel P4 1.83 GHz CPU and 2 GB of RAM.

Day	Num	D	F	Alg	$Macro$	$Micro$
1	3546	1226	88	KM_TFIDF	0.6925	0.7376
1	3546	1226	88	HC_TFIDF	0.7501	0.7134
1	3546	1226	88	HHC_TFIDF	0.8218	0.8015
1	3546	1226	88	KM_TF	0.6279	0.6802
1	3546	1226	88	HC_TF	0.7228	0.7237
1	3546	1226	88	HHC_TF	0.8162	0.8128
2	3005	1178	102	KM_TFIDF	0.7033	0.7661
2	3005	1178	102	HC_TFIDF	0.787	0.7921
2	3005	1178	102	HHC_TFIDF	0.8263	0.8101
2	3005	1178	102	KM_TF	0.5687	0.602
2	3005	1178	102	HC_TF	0.6605	0.6845
2	3005	1178	102	HHC_TF	0.7655	0.7468
3	5162	2375	324	KM_TFIDF	0.5942	0.5905
3	5162	2375	324	HC_TFIDF	0.6365	0.6591
3	5162	2375	324	HHC_TFIDF	0.722	0.7335
3	5162	2375	324	KM_TF	0.6398	0.6126
3	5162	2375	324	HC_TF	0.7436	0.7228
3	5162	2375	324	HHC_TF	0.7895	0.7957

Table 1: Based on instruction frequency, the categorization results of different categorizers on the real daily new malware collection from Jan 11th, 2010 to Jan 13th, 2010. Remark: "Num"-the total number of the malware samples, "D"-Dimensions of the data set, "F"-the real malware families, "Macro"-Macro-F1 measure and "Micro"-Micro-F1 measure.

7.1 Comparisons of Malware Clustering Methods Based on Instruction Frequency

In this set of experiments, we evaluate the effectiveness of malware categorization results of different categorizers: hierarchical clustering used in [12, 18], K-medoids used in [15] and the hybrid hierarchical clustering algorithm proposed in Section 5. In this paper, we measure the categorization performance of different algorithms using Macro-F1 and Micro-F1 measures,which emphasize the performance of the system on rare and common categories respectively [8].

In this section, we use the daily new malware sample collection obtained from Kingsoft Anti-Virus Lab from every 9:00am to 12:00am from Jan 11th, 2010 to Jan 13th, 2010. The experimental results as shown in Table 1 demonstrate that, with the instruction frequency feature vectors, the Hybrid Hierarchical Clustering(HHC) algorithm used in our AMCS system outperforms Hierarchical Clustering(HC) and K-Medoids(KM) clustering methods. Here, the number of clusters is determined as follows: (1) K-Medoids(KM): use the real malware family number as the specified K; (2) Hierarchical Clustering(HC) and Hybrid Hierarchical Clustering(HHC): use *Fukuyama-Sugeno index(FS)* [9] as the cluster validity index and choose the cluster number that leads to the smallest *FS* value.

7.2 Comparisons of Malware Clustering Methods Based on Instruction Sequences

In this section, we use the data set described in Section 7.1. The results as shown in Table 2 demonstrate that, using function-based instruction sequences, the Weighted subspace K-Medoids(WKM) algorithm used in our AMCS system outperforms K-Medoids(KM)

clustering method. For both K-Medoids(KM) clustering and Weighted subspace K-Medoids(WKM) algorithms, we use the real malware family number as the specified K.

Day	Num	D	F	Alg	*Macro*	*Micro*
1	3546	7208	88	KM	0.6135	0.6747
1	3546	7208	88	WKM	0.8196	0.8559
2	3005	6923	102	KM	0.6882	0.6421
2	3005	6923	102	WKM	0.8071	0.8015
3	5162	11054	324	KM	0.6279	0.6252
3	5162	11054	324	WKM	0.7874	0.8147

Table 2: Based on function-based instruction sequences, the categorization results of different categorizers on the real daily new malware collection from Jan 11th, 2010 to Jan 13th, 2010.

7.3 Evaluation of Cluster Ensemble with Constraints

In this set of experiments, we evaluate the effectiveness of malware categorization results of our proposed cluster ensemble, especially with sample-level constraints. Using the data set described in Section 7.1, we construct the cluster ensemble using 6 base clusterings: HHC_TFIDF and HHC_TF as described in Section 6.2, and four WKM categorizers with on the function-based instruction sequences with four different K's. From Table 3, we observe that the malware categorization results of the cluster ensemble outperform each individual algorithm. Here, we only list the best categorization result on each data collection generated by the four base WKM categorizers for comparison.

Day	Num	F	Alg	*Macro*	*Micro*
1	3546	88	HHC_TFIDF	0.8218	0.8015
1	3546	88	HHC_TF	0.8162	0.8128
1	3546	88	WKM	0.8196	0.8559
1	3546	88	NCE	0.9017	0.9137
1	3546	88	CE	0.9302	0.9437
2	3005	102	HHC_TFIDF	0.8263	0.8101
2	3005	102	HHC_TF	0.7655	0.7468
2	3005	102	WKM	0.8071	0.8015
2	3005	102	NCE	0.8989	0.8669
2	3005	102	CE	0.9245	0.9113
3	5162	324	HHC_TFIDF	0.722	0.7335
3	5162	324	HHC_TF	0.7895	0.7957
3	5162	324	WKM	0.7874	0.8147
3	5162	324	NCE	0.8605	0.8896
3	5162	324	CE	0.9183	0.9181

Table 3: Evaluation of the malware categorization results of clustering ensemble. Remark: "NCE"-cluster ensemble without constraints, "CE"-cluster ensemble with constraints.

It should be pointed out that in many cases, categorizing a malware sample to a certain family is still the prerogative of virus analysts. For example, as shown in Figure 7, though some of the malware files complied by Delphi compiler or E-language compiler which uses Chinese for program development share similar shape of instruction frequency patterns and lots of basic blocks of function-based instruction sequences and thus may be categorized to a same family, according to their intents and behaviors, they should be divided into different families. On the contrary, there are some metamorphic malware samples, like "Trojan.Swizzors", they may differ from static feature representations, but they are in the

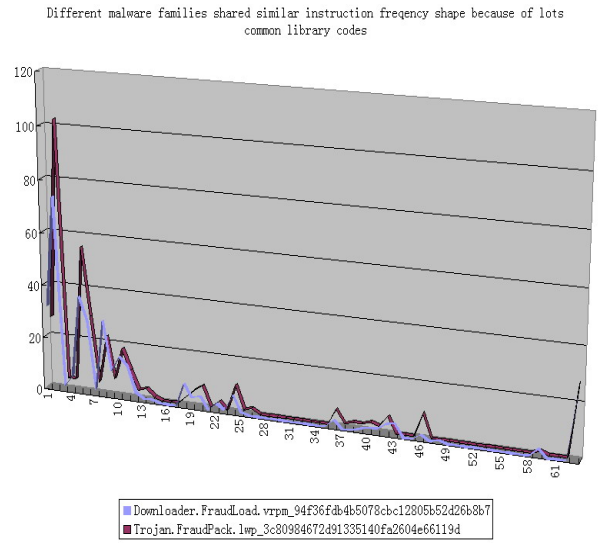

Figure 7: Example of sample-level inequivalence constraints.

same family. In such cases, if we can add some sample-level constraints, the categorization results will be improved. Our malware categorization system AMCS provides a user-friendly mechanism for incorporating the expert knowledge and expertise of human experts. The cluster ensemble scheme of our malware categorization system not only combines the clustering results of individual categorizers, but also incorporates the sample-level constraints provided by the human analysts. According to the expertise of the malware analysts, AMCS totally gets 1,385 pairs of must-link constraints and 1,078 cannot-link constraints.

To further demonstrate the advantage of incorporating sample-level constraints, we use the real daily new malware collection for two weeks (from Jan 11th to Jan 24th) to compare the categorization results of cluster ensemble without constraints and with constraints. Experimental results in Figure 8 clearly shows that ensemble with constraints outperforms the one without constraints.

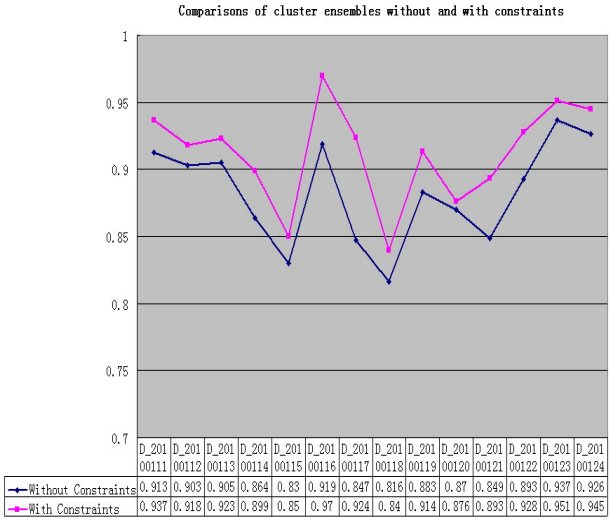

Figure 8: Comparisons of malware categorization results of cluster ensembles without and with constraints

7.4 Comparisons with Different AV Venders

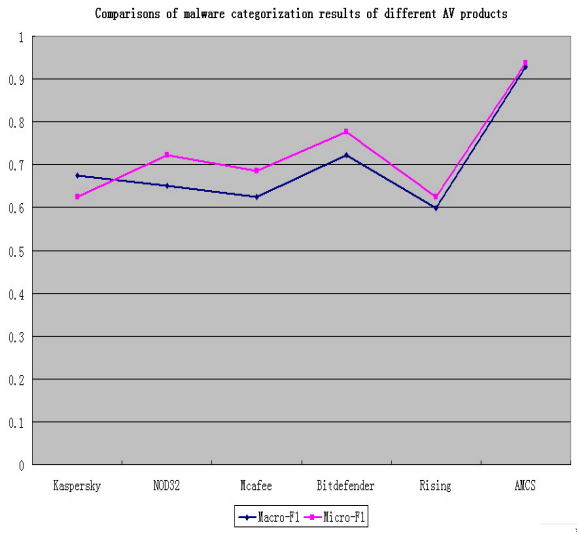

Figure 9: The comparison of malware categorization results of different AV software on the whole data collection of 42,180 malware samples.

In this section, we apply AMCS in real applications to evaluate its malware categorization effectiveness and efficiency of the daily data collection. We use the whole data collection for two weeks (from Jan 11th, 2010 to Jan 24th, 2010) which consists of 42,180 malware samples with 2,713 families to compare the malware categorization effectiveness of AMCS with some of the popular AV products, like Kaspersky(Kasp), NOD32, Mcafee, Bitdefender(BD) and Rising. For comparison purpose, we use all of the Anti-Virus scanners' newest versions of the base of signature on the same day(Jan 24th, 2010). Table 4 and Figure 9 show that the malware categorization effectiveness of our AMCS outperforms other popular AV products.

AV.	Detected	Families	$MacroF1$	$MicroF1$
Kasp	35,433	1,998	0.6736	0.6246
Nod32	33,634	1,598	0.6498	0.7233
Mcafee	35,500	1,859	0.6253	0.6856
BD	39,723	1,916	0.7215	0.7761
Rising	35,712	1,885	0.5983	0.6257
AMCS	41,623	2,271	0.9274	0.9369

Table 4: The categorization results of different AV software on the whole data collection of 16,123 malware samples.

For robust evaluation, we track the malware categorization results of our AMCS and AV software products above, based on 30 consecutive days of new malware sample collection with a total number of 103,712. The real daily experiments demonstrate that the average of Macro-F1 and average of Micro-F1 of AMCS are higher than 0.90, while none of those five popular AV software of them are higher than 0.80. In addition, we also evaluate the efficiency of our AMCS system: (1) Categorizing 3,546 malware samples by our AMCS system including feature extraction needs 3 minutes; (2)The whole process of 42,180 malware samples needs 15 minutes. Besides good performance and high efficiency on malware categorization results, our AMCS system can also automatically generate signatures for malware families to detect malware variants.

The signature with id "72237142":

```
[ScriptScan]
    mov Reg00, 0x00
    mov Reg01, 0x01
    mov Reg02, 0x02
    mov Reg03, 0x03
    mov Reg04, 0x04
    Call PE.GetImageBase Reg05
    ReturnIfFalse

    SetMatchBuffer NormalEntryBufferReg, NormalEntryBufferSizeReg
    mov SearchPosReg, 0xB9
    mov SearchLenReg, 0x20
    mov Reg06, 0x35FF
    SearchWORD Reg06
    ......
```

Figure 10: The signature with id "72237142".

Our AMCS system can automatically generate the function-based instruction sequences which frequently appeared within a malware family but rarely presented in other malware families by using weighted subspace k-medoids clustering algorithm. Together with their related operands, they can be the signatures of their belonging malware families for variants detection. Figure 11 illustrates that the signature with id "72237142" as shown in Figure 10 can detect 28,935 file samples. All of these traits make it possible for real anti-malware industry application.

8. SYSTEM DEVELOPMENT AND OPERATION

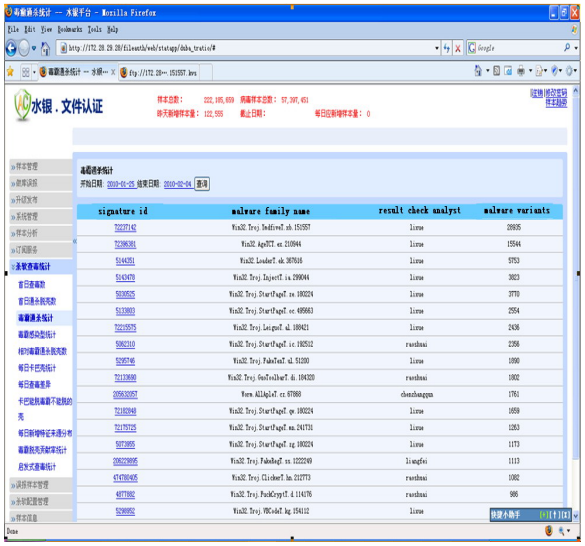

Figure 11: The rank list of detection ability of malware family signatures.

Kingsoft has spent over $500K in the development of the AMCS

system and about $100K on the hardware equipment. The system is monitored 24/7 via scripts that verify functionality and availability and is managed in a revision control system. Over 30 virus analysts at Kingsoft's Anti-Virus lab are utilizing the system on the daily basis. In practice, a virus analyst has to spend at least 10 hours to manually analyze 100 malware samples for categorization. Using the AMCS system, the categorization of about 30000 malware samples (including feature extraction and categorization) can be performed within 20 minutes. The high efficiency of our AMCS system can greatly save human labors and reduce the staff cost. We are currently performing a comprehensive investigation on the signature extraction for the malware samples in a cluster based on our AMCS system, in order to construct a more streamlined signature library for better malware detection on the client anti-malware products. This would benefit over 10 million Internet users of Kingsoft's client anti-malware products.

9. CONCLUSION

In this paper, we develop an automatic malware categorization system (AMCS) for categorizing malware samples into families that share some common traits by an ensemble of different clustering solutions generated by different clustering methods. Empirical studies on large and real daily data sets from Kingsoft Anti-Virus Lab illustrate that our AMCS system outperform other malware categorization methods as well as some of the popular AV products. The system has been incorporated into the Kingsoft's Anti-Virus products.

Acknowledgments

The work of T. Li is partially supported by the US National Science Foundation under grants IIS-0546280 and DMS-0915110. The work of Y. Ye and Q. Jiang is partially supported by the National Science Foundation of China under grant #10771176 and Guangdong Province Foundation under grant #2008A090300017. The authors would also like to thank the members in the Anti-Virus Lab at Kingsoft Corporation for their helpful discussions and suggestions.

10. REFERENCES

[1] Javad Azimi and Xiaoli Fern. Adaptive cluster ensemble selection. In *Proceedings of IJCAI*, pages 992–997, 2009.

[2] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[3] S. Basu, I. Davidson, and K. L. Wagstaff, editors. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. CRC Press, 2008.

[4] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. Scalable, behavior-based malware clustering. *NDSS'09 Security Symposium*, 2009.

[5] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is nearest neighbor meaningful? In *Proceedings of 7th International Conference on Database Theory(ICDT'99)*, pages 217–235. Springer, 1999.

[6] Chris Ding and Tao Li. Adaptive dimension reduction using discriminant analysis and k-means clustering. In *ICML*, pages 521–528, 2007.

[7] Xiaoli Zhang Fern and Carla E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *ICML*, page 36, 2004.

[8] F.Sebastiani. Text categorization. *ACM Computing Surveys*, 34:1–47, 2002.

[9] Y. Fukuyama and M. Sugeno. A new method of choosing the number of clusters for the fuzzy c-means method. *Proc. 5th Fuzzy Syst. Sym*, pages 247–250, 1989.

[10] Marius Gheorghescu. An automated virus classification system. *Virus Bulletin Conference*, 2005.

[11] Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. Clustering aggregation. In *ICDE*, pages 341–352, 2005.

[12] Ibai Gurrutxaga, Olatz Arbelaitz, Jesus Ma Perez, Javier Muguerza, Jose I. Martin, and Inigo Perona. Evaluation of malware clustering based on its dynamic behaviour. *Seventh Australasian Data Mining Conference*, 2008.

[13] Liping Jing, Michael K. Ng, and Joshua Zhexue Huang. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Trans. on Knowl. and Data Eng.*, 19(8):1026–1041, 2007.

[14] Kaufman L and Rousseeuw PJ. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, 1990.

[15] Tony Lee and Jigar J.Mody. Behavioral classification. *EICAR 2006*, May 2006.

[16] Tao Li and Chris Ding. Weighted Consensus Clustering. In *SIAM Data Mining*, 2008.

[17] Tao Li, Chris Ding, and Michael I. Jordan. Solving consensus and semi-supervised clustering problems usi ng nonnegative matrix factorization. In *ICDM*, 2007.

[18] M.Bailey, J.Oberheide, J.Andersen, Z. M.Mao, F.Jahanian, and J.Nazario. Automated classification and analysis of internet malware. *RAID*, 4637:178–197, 2007.

[19] Stefano Monti, Pablo Tamayo, Jill Mesirov, and Todd Gloub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning Journal*, 52(1-2):91–118, 2003.

[20] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations*, 6:90–105, 2004.

[21] Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Dussel, and Pavel Laskov. Learning and classification of malware behavior. In *Fifth Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA'08)*, pages 108–125, 2008.

[22] M. Schultz, E. Eskin, and E. Zadok. Data mining methods for detection of new malicious executables. In *Proccedings of 2001 IEEE Symposium on Security and Privacy*, pages 38–49, 2001.

[23] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *JMLR*, 3:583–617, March 2003.

[24] R. Tian, L.M. Batten, and S.C. Versteeg. Function length as a tool for malware classification. *3rd International Conference on Malicious and Unwanted Software (MALWARE)*, 2008.

[25] Alexander P. Topchy, Anil K. Jain, and William F. Punch. Clustering ensembles: Models of consensus and weak partitions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(12):1866–1881, 2005.

[26] Fei Wang, Xin Wang, and Tao Li. Generalized cluster aggregation. In *Proceedings of IJCAI*, pages 1279–1284, 2009.

[27] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE transactions on neural networks*, 16, May 2005.

[28] Yanfang Ye, Dingding Wang, Tao Li, and Dongyi Ye. IMDS: Intelligent malware detection system. In *SIGKDD*, 2007.

IPR2023-01160
KASPERSKY EX1024 PAGE 10