Clustering Analysis of Email Malware Campaigns

Ruichao Zhang
School of Engineering and Technology
University of Washington Tacoma
rczhang@uw.edu

Shang Wang
School of Engineering and Technology
University of Washington Tacoma
swang848@uw.edu

Renee Burton
Infoblox
rburton@infoblox.com

Minh Hoang
Infoblox
mihnhoang@infoblox.com

Juhua Hu
School of Engineering and Technology
University of Washington Tacoma
juhuah@uw.edu

Anderson C A Nascimento
School of Engineering and Technology
University of Washington Tacoma
andclay@uw.edu

***Abstract*—The task of malware labeling on real datasets faces huge challenges—ever-changing datasets and lack of ground-truth labels—owing to the rapid growth of malware. Clustering malware on their respective families is a well known tool used for improving the efficiency of the malware labeling process. In this paper, we addressed the challenge of clustering email malware, and carried out a cluster analysis on a real dataset collected from email campaigns over a 13-month period. Our main original contribution is to analyze the usefulness of email's header information for malware clustering (a novel approach proposed by Burton [1]), and compare it with features collected from the malware directly. We compare clustering based on email header's information with traditional features extracted from varied resources provided by VirusTotal [2], including static and dynamic analysis. We show that email header information has an excellent performance.**

***Index Terms*—Malware labeling, email campaigns, clustering analysis, malware feature extraction**

I. Introduction

A. Challenges for malware labeling

Labeling malware samples as variants of known malware families is a very important task for multiple cybersecurity applications, such as identifying new threats, selecting disinfection mechanisms, attribution, and malware lineage. Unfortunately, malware grows in amounts and variants very rapidly. For example, in 2019 Kaspersky identified 24,610,126 unique malware samples [3]. In 2020, such number has grown to 33,412,568 [4]. The rapid growth of malware poses two challenges to the task of malware labeling:

a) Ever-changing datasets: with the new malware models discovered every day, malware in the field is ever-changing. In contrast, reference datasets used for academic purpose such as *Drebin* [5] are often outdated.

b) Lack of ground-truth labels: the rapid growth of malware makes it difficult for human experts to manually label the newly collected malware samples. However, ground-truth labels are essential for various malware labeling approaches, using both heuristic and machine learning techniques.

978-1-7281-5684-2/20/$31.00 ©2021 IEEE

B. Previous works and their shortcomings

There are a variety of malware labeling approaches in both academia and industry.

Anti-virus (AV) checkers are a common tool for performing automatic malware labeling. However, it is not uncommon for labels given by different AV checkers to be inconsistent, due to both different detection techniques and different naming conventions adopted (conventions such as CARO [6] and CME [7] are not widely adopted by the industry [8], [9], [10]). Such inconsistency causes great confusion to researchers. Different labels may be assigned by different AV checkers to samples in the same family, *e.g.*, `troj.win32.firseria` and `trojan.Win_firseria`; similar labels may be assigned to samples from different variants in a malware family, *e.g.*, `X97M/Agent.FQ.gen!Eldorado` and `X97M/Eldorado.Agent.nemdk`. In addition, many AV checkers will add extra information to the labels, such as md5 hashes and variant versions, which makes it more difficult to summarize this information (*e.g.*, `Trojan:Win32/Emotedted.44223fe`).

Previous researches, like AVClass [11] and Euphony [12], work on the basis of AV labels to extract family names, applying self-defined heuristic rules, such as generic token check and suffix token removal. Plurality vote is then used to reach a consensus among different AV checkers. Such methods have a good performance on strongly labeled malware. However, they rely too much on accurate and reliable raw AV labels, which are difficult to produce due to reasons discussed above.

VirusTotal is a tool commonly used by malware researchers [2]. It is an online service that analyzes files and URLs, enabling the detection of viruses, worms, trojans and other kinds of malicious contents using AV engines and website scanners. It has a dataset of more than 2 billion analyzed samples, with their AV labels, metadata, behavioral analyses, etc. However, most previous works did not take full advantage of the sufficient resources made available by VirusTotal. AVClass [11] and Euphony [12] merely used the raw AV labels provided by VirusTotal. Song *et al.* [13] only used the basic metadata of malware. Faridi *et al.* [14] made use

of dynamic analysis directly from the Cuckoo Sandbox [15], while VirusTotal provides results from multiple sandboxes, with a wider coverage of malware samples. Therefore, in our research we take advantage of the varied resources provided by VirusTotal, including metadata, behavioral analysis, static and dynamic features.

C. Email campaigns

A large proportion of malware is spreaded through emails, which are also known as malicious spam, or **malspam**. In 2019, in the 975,491,360 attacks detected by Kaspersky [3], 467,188,119, or 47.89% were malspam; in all email traffic, 56.51% was contributed by malspam [16]. Thus, one will face the challenges described in subsection I-A when labeling malspam.

Malspam is sent through different campaigns. A malspam **campaign** is defined as a set of emails sent by a threat actor, limited in a certain timeframe and to a certain theme. Burton [1] proposed a novel approach for labeling malspam. They use email metadata, such as email subjects, senders' IP addresses, attachment filenames, *etc.*, rather than information in malware itself. Graph-based approaches are applied to cluster the malware samples. Comparing to traditional heuristic and machine learning approaches, this novel approach does not require much human resources (for manually labeling the dataset) or computational resources (for training models), nor does it require the source code or binaries of malware samples (for static and dynamic analysis).

In this paper, we evaluate such a novel approach and compare it to the use of more traditional features for malware clustering.

D. Paper structure

The rest of the paper is organized as follows. Section II introduces some previous works of malware labeling. Section III summarizes our main contributions. Section IV describes our methodology, and discusses the feasibility and limitation of using malware labeling tools' outputs as "labels". Section V describes the dataset we work with, shows its family distribution and points out that the dataset is skewed. We also introduce in this section the features we choose from metadata and behavioral reports provided by VirusTotal. Section VI describes the three methods to construct matrices from features, as well as the clustering algorithm we apply. Section VII gives the definition of the evaluation metrics we use, and explains why purity is the most suitable metric for our context. Section VIII shows the evaluation result and provides a detailed analysis. Finally, in section IX, we draw conclusions for our research, points out the difficulties while working with the real dataset, and suggest methods to improve our work in future.

II. Related work

In this section, we discuss about previous works in the field of malware labeling, especially those related to our work.

Sebastián *et al.* [11] designed AVClass, an automatic labeling tool that takes as input a large amount of AV labels for malware samples, and outputs the most likely family name for each sample, by addressing three key challenges: normalization, removal of generic tokens, and alias detection. The limitation of AVClass is that it is heavily dependent on *a priori* knowledge, both on ground-truth family names and on AV-specific naming conventions. Because of the rapid update of real malware datasets and lack of universally adopted naming conventions, AVClass faces huge problems when working on real datasets.

Hurier *et al.* [12] proposed Euphony, a system which also produces family labels based on AV checker labels, but is independent of *a priori* knowledge on malware families or AV naming schemes. It uses an incremental parsing algorithm based on a few pre-defined heuristic rules and a knowledge base which does not include *a priori* knowledge for family names. Theoretically, Euphony adapts to real datasets better than AVClass does, which is a significant improvement. However, Euphony's output heavily depends on the AV label input; as a consequence, it cannot identify tags if AV checkers only provide overly generic tokens for malware samples.

Most related to our work is that of Faridi *et al.* [14], who studied the malware clustering problem using different algorithms, distance functions and sets of features. This is the first attempt not only for affinity propagation and two forms of spectral clustering, but for two cutoff methods of hierarchical clustering as well. Our process of feature construction is related to theirs, in the sense that we both use malware behavioral reports, TF–IDF matrix and ssdeep distance. Furthermore, They deployed multiple clustering algorithms, including hierarchical, density-based, prototype-based, *etc.* However, they only focused on malware of the *pe* file type, and did not use as much information made available by VirusTotal as we do.

Similarly, Mohaisen *et al.* [17] developed AMAL, a system collecting behavioral features to characterize malware usage of file system, memory and network. AMAL utilized the behavioral features to create representative features, which are used for classification tasks.

Trinius *et al.* [18] also created a representation using behavioral features. The system, known as MIST, is designed for malware analysis using data mining and machine learning techniques.

Ducau *et al.* [19] developed a representation space for malware samples, where samples with similar behaviors appear close to each other by introducing deep learning approaches.

Zhang *et al.* [20] also used deep learning techniques to bridge the information gap between machine learning and signature-based detection. They proposed a new representation learning approach to cluster labels by preserving different kinds of information from multiple sources (including static code analysis, metadata and raw AV labels).

Song *et al.* [13] analyzed *pe* malware in November 2018, with the metadata provided by VirusTotal. The analysis includes trend of the number of malware outbreaks, statistics of

malware size, and distribution of malware types. Their result shows that malware appear in bursts and that distribution of malware are highly skewed.

III. Main Contributions

Our work follows a long line of research where clustering is used as a tool for aiding malware labeling. The main idea is to cluster malware samples in the hope that malware samples belonging to the same family will end up in the same clusters. Picking up the right features, clustering metrics, and evaluation metrics are central points to be addressed in such kind of approach.

To the best of our knowledge, we are the first work that systematically analyze the effectiveness of using email header information as a feature for malware clustering. This technique has recently been proposed in [1] and shows very interesting properties. It is much easier to extract email header information. There is no need for sandboxes, reverse engineering or even having direct access to the malware code itself. We show that email header information is an excellent approach compared to more traditional dynamic and static features. We systematically compare email header features to dynamic and static features presented in VirusTotal and see how well they perform for clustering malware families.

IV. Methodology

As mentioned in section I, the main goals of our research are:

1) to address the two challenges of labeling malware on real datasets;
2) to propose an approach for malware clustering and labeling, utilizing several resources from static and dynamic analysis;
3) to evaluate the performance of the novel approach based on email campaigns, and to compare it with our approach based on malware features.

We propose the following methodology. We work on a dataset with n_1 samples, $S = \{s_1, ..., s_{n_1}\}$, collected from email campaigns. $C_0 = \{c_1^0, ..., c_{n_1}^0\}$ is a set of clusters from the email-based approach, where c_i^0 indicates the cluster of sample s_i ($i \in \{1, ..., n_1\}$). Because the dataset is unlabeled, we use the results of AVClass and Euphony, $L_1 = \{l_1^1, ..., l_{n_1}^1\}$ and $L_2 = \{l_1^2, ..., l_{n_1}^2\}$, as "labels". Metadata reports and behavioral reports, which contains the static and dynamic analysis, are retrieved from VirusTotal for every sample. We choose a set of n_2 features $F = \{f_1, ..., f_{n_2}\}$ from metadata reports and behavioral reports, and from each feature $f \in F$ we construct one or more matrices. Denote the set of n_3 matrices as $M = \{m_1, ..., m_{n_3}\}$, $(n_3 \geqslant n_2)$. Each matrix $m_j \in M$ is fed into a clustering algorithm, which produces a set of clusters $C_j = \{c_1^j, ..., c_{n_1}^j\}$ as its output.

We choose a set of n_4 evaluation metrics $V = \{v_1, ..., v_{n_4}\}$. Denote $E(A, B, v)$ where $\{A, B\} \subseteq \{L_1, L_2, C_0, ..., C_{n_1}\}$, as the evaluation result of A and B, using metric $v \in V$. We compute $E(C_0, L_i, v)$ where $i \in \{1, 2\}$ and $v \in V$ as a measure of performance of the email-based approach; $E(C_j, L_i, v)$ where $j \in \{1, ..., n_3\}$ measures the performance of the corresponding malware feature.

Our methodology is built on the basis of assumption 1:

Assumption 1: For our dataset, AVClass and Euphony results are close enough to the hypothetical ground-truth labels.

But how can we be so sure that such assumption is true for our dataset? We gain our confidence from previous evaluations of AVClass and Euphony on ground-truth data. Moreover, we also noticed that the results of AVClass and Euphony are close enough to each other, on different parts of the dataset, evaluated by multiple metrics, which strengths our belief on assumption 1.

V. Dataset and features

We worked on a dataset collected by Burton [1] over a 13-month period, from April 2019 to April 2020. It was collected from real email campaigns. It consists of two file types: Microsoft Office documents (*doc*) and archives (*archive*). We retrieved metadata reports and behavioral reports from VirusTotal, and chose the sandboxes that produced most behavioral reports for samples for each file type, as is shown in table I.

File type	Samples	Samples processed by specified sandbox	Sandbox chosen
doc	10,554	7,401	Lastline [21]
archive	20,105	4,231	RedDrip [22]

TABLE I: File types of the dataset

Since we do not have ground-truth labels for the dataset, we used AVClass and Euphony results as "labels". The malware family distribution of each file type is shown as tables II, III, IV and V, using AVClass and Euphony results as "labels", respectively. For the same file type, the total number of samples may be different for AVClass and Euphony. This is because the number of unlabeled samples are different for AVClass and Euphony: generally, there are much more unlabeled samples for AVClass. We noticed that the dataset collected from email campaigns is quite skewed, *e.g.*, 83.201% samples are AVClass "emotet" and 60.300% samples are Euphony "emotet".

Family	Count	%
emotet	5542	83.201
sagent	729	10.944
w2000m	131	1.967
obfuse	51	0.766
others	208	3.123

TABLE II:
doc family distribution,
AVClass as "labels"

Family	Count	%
emotet	4461	60.300
sagent	660	8.921
emodldr	463	6.258
efgr	190	2.568
generickdz	181	2.447
eldorado	160	2.163
eiob	84	1.135
efzn	83	1.122
efjn	80	1.081
cve	79	1.068
others	957	12.937

TABLE III:
doc family distribution,
Euphony as "labels"

Family	Count	%
mydoom	2401	60.739
autoit	276	6.982
netsky	200	5.059
mudrop	104	2.631
agensla	88	2.226
bagle	62	1.568
agenttesla	60	1.518
genkryptik	53	1.341
noon	48	1.214
others	661	16.722

TABLE IV:
archive family distribution, AVClass as "labels"

Family	Count	%
cuyllc	1273	30.152
mudrop	764	18.096
eldorado	226	5.353
agensla	224	5.306
netsky	200	4.737
dsipmr	152	3.600
mytob	146	3.458
acna	99	2.345
generickdz	64	1.516
bagle	62	1.468
ponystealer	48	1.137
agenttesla	46	1.090
others	918	21.742

TABLE V:
archive family distribution, Euphony as "labels"

We retrieved metadata reports and behavioral reports for every sample from VirusTotal, and chose features from both reports.

A. Metadata reports

Metadata reports are fundamental analysis reports provided by VirusTotal, which provide the most basic information about the threat, such as submission date, AV checker results, original filename, file size, file sha256 hash, *etc.* We use the following features in metadata reports.

a) tags: tags is a high-level summary provided by VirusTotal. It describes the malware sample from multiple aspects, such as file type (*e.g.*, "xlsx", "rar"), behaviors (*e.g.*, "copy-file", "download"), and other characteristics (*e.g.*, "assembly", "macros").

b) ssdeep: ssdeep [23] is a program for context triggered piecewise hashes (CTPH), which is a fuzzy hashing algorithm, whereby similar files will have similar ssdeep hashes [24]. The edit distance of two ssdeep hashes can reflect the edit distance of the two original files. Thus, *ssdeep* can be used as a measure of file dissimilarity.

B. Behavioral reports

Behavioral reports provides dynamic analysis of malware samples. VirusTotal collects the outputs of multiple sandboxes for a malware sample, which are made available in a single behavioral report. However, unlike metadata reports, not every sample has a behavioral report; even for those with a behavioral report, different samples may be analyzed by different sets of sandboxes. Therefore, we had to choose the sandboxes that analyze the most samples for each file type.

For each file type, we chose the behavioral features that occur in most samples, as is shown in table VI.

Feature	*doc*	*archive*
files_attribute_changed	✓	
files_dropped		✓
files_opened		✓
modules_loaded	✓	✓
mutexes_opened	✓	
mutexes_created	✓	
processes_created	✓	
processes_tree	✓	✓
registry_keys_deleted	✓	
registry_keys_opened		✓
registry_keys_set	✓	
Total	8	5

TABLE VI: Behavioral features for *doc* and *archive* samples

VI. Preprocessing and clustering

Once the features are selected, we enter the stage of preprocessing, where one or more matrices are constructed for each feature. Each matrix is then fed into the HDBSCAN clustering algorithm.

A. One-hot matrix

One-hot encoding is one of the simplest encodings for categorical data in the field of machine learning. If the data has m categories, the one-hot vector $\boldsymbol{h}$ of a sample can be defined as $\boldsymbol{h} = (h_1, ..., h_m)$, where for $1 \leqslant j \leqslant m$

$$h_j = \begin{cases} 1, & \text{if sample is in the } j\text{-th category} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

In a one-hot vector, there is exactly one bit of 1. However, in our context, we use one-hot for feature *tags*, which is multi-valued. Therefore, we define the *tags* one-hot vector $\boldsymbol{t}$ of a sample as $\boldsymbol{t} = (t_1, ..., t_m)$, where for $1 \leqslant j \leqslant m$

$$t_j = \begin{cases} 1, & \text{if sample has the } j\text{-th tag} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Furthermore, we construct the *tags* one-hot matrix $\boldsymbol{A_1}$ as

$$\boldsymbol{A_1} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix} \tag{3}$$

where each row represents a sample and each column represents a tag. Sample i has tag j in its *tags* iff $a_{i,j} = 1$.

B. TF–IDF matrix

In natural language processing, TF–IDF (term frequency–inverse document frequency) is a measure of how important a token is to a document in a corpus. The TF–IDF value $tfidf(t, d, D)$ of a token t to a document d in a corpus D is defined as

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{4}$$

where term frequency $tf(t, d)$ is

$$tf(t, d) = \frac{\text{count of } t \text{ in } d}{\text{number of tokens in } d} \tag{5}$$

and inverse document frequency $idf(t, D)$ is

$$idf(t) = \log \frac{\text{number of documents in } D}{1 + \text{number of docs in } D \text{ where } t \text{ occurs}} \tag{6}$$

The intuition behind TF–IDF is as follows. $tf(t,d)$ measures how important t is to d: the higher $tf(t,d)$ is, the more times t occurs in d, therefore the more important t is to d. $idf(t,D)$ measures how informative t is in D: the higher $idf(t,D)$ is, the *fewer* documents t occurs, therefore the more likely t is a specific word and is related to the topics of these documents. As the product of $tf(t,d)$ and $idf(t,D)$, $tfidf(t,d,D)$ takes into consideration both the importance of t to d and the informativeness of t in D.

In our dataset, we use TF–IDF for *tags* and every behavioral features. A document is a sample in this context. For *tags*, each unique tag is a token. For behavioral features, we do some preprocessing before tokenization, including conversion to lowercase, removal of punctuation and stitching tokens that make sense together from *a priori* knowledge, splitting filenames from the path to it, *etc.* We also stitch together all texts from all behavioral features and do tokenization on the resulting larger texts. Then we construct the TF–IDF matrix $\boldsymbol{A_2}$ as

$$\boldsymbol{A_2} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix} \tag{7}$$

where each row represents a sample and each column represents a token. The value of $a_{i,j}$ equals to the TF–IDF value of the j-th token to the i-th document in the corpus.

C. ssdeep distance matrix

As is described in subsection V-A, *ssdeep* is a measure for dissimilarity of the original files by comparing the *ssdeep* hashes. For example, if a single byte of a file is modified, the *ssdeep* hashes of the original file and the modified file are considered highly similar. The *ssdeep* is useful for searching for similar files. For instance, two malware samples generated from the same malware family which inserts configuration statically into a stub sample, may be easy to identify as having a high similarity. Thus we compute the *ssdeep* hash of feature strings to compare the similarity of different malware samples.

The Python package `ssdeep` [23] provides a function to get the similarity score of two *ssdeep* hashes, which ranges from 0 (no similarity or negligible similarity) to 100 (very similar, if not an exact match). We can transform the similarity score s to dissimilarity score (*i.e.*, distance) s' by

$$s' = \frac{100}{s+1} - \frac{100}{101} \tag{8}$$

Then we construct the *ssdeep* distance matrix $\boldsymbol{A_3}$ as

$$\boldsymbol{A_3} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \tag{9}$$

where each row and each column represent a sample. The value of $a_{i,j}$ equals to the distance of the i-th sample and j-th sample.

In addition, we constructed *ssdeep* distance matrix for each of the behavioral features. For a sample, a given behavioral feature usually has multiple strings. We sort these string alphabetically (so that edit distance makes sense) and concatenate these strings. Then we compute the pairwise distance of these strings and construct an *ssdeep* distance matrix as described above.

D. HDBSCAN

Now that we have constructed the matrices from the features, we feed them into a clustering algorithm, for which we choose HDBSCAN.

HDBSCAN [25] is an extension to DBSCAN [26], the density-based clustering algorithm. Not only can it cluster data of varying shape, like DBSCAN does, it can also cluster data of varying density.

HDBSCAN defines the concept of mutual reachability distance, which is based on density. The original space is transformed to a mutual reachability distance-based space, where a weighted graph is built. HDBSCAN builds the minimal spanning tree for the graph, via Prim's algorithm. The minimal spanning tree is then converted to a single linkage tree, and ultimately to a condensed tree, which represents a hierarchy of clusters. HDBSCAN selects the most stable clusters as its output.

VII. Evaluation

To evaluate the performance of our malware features and the email-based approach, we adopted multiple evaluation metrics for clustering. Purity measures the homogeneity of a set of clusters, while ARI, AMI and FMI measure the similarity between two sets of clusters. Purity is apparently weaker than the other three metrics, but fits the nature of our research. We also use some auxiliary metrics.

A. Purity

Purity is a metric for the homogeneity of the labels in each clusters. Given a set of m clusters $C = \{c_1, ..., c_m\}$ and a set of n labels $L = \{l_1, ..., l_n\}$, we define

$$p_i = \frac{\max\limits_{1 \leqslant j \leqslant n} |c_i \cap f_j|}{|c_i|} \tag{10}$$

for $i \in \{1, ..., m\}$. Purity $\mathrm{purity}(C, L)$ is defined as

$$\mathrm{purity}(C, L) = \frac{\sum\limits_{i=1}^{m} p_i |c_i|}{\sum\limits_{i=1}^{m} |c_i|} \tag{11}$$

Purity reflects the extent of clusters having few (*i.e.*, homogeneous) labels. However, even when purity is high, the clusters and labels are not necessarily similar enough, for one label can be distributed into multiple clusters. It is possible that for the same pair of clusters C and labels L, their other metrics such as ARI could be drastically lower, while the purity is relatively high. This does not mean purity is not good enough

as an evaluation metric. On the contrary, purity best fits the nature of our research, for two reasons:

1) The main goal of our approach is to make sure in each cluster there are homogeneous labels, so that each cluster can be given a proper name; in later processes, those clusters with the same name can be merged together.
2) More importantly, in the email-based approach clusters are derived from email campaigns. An email campaign is limited in a certain timeframe and to a certain threat actor and theme. However, a malware family can surely occur in multiple timeframes, be related to multiple thems and sent by multiple threat actors. Therefore families are naturally distributed in multiple clusters in this approach, for which the only suitable metric is purity.

B. ARI

Given a set of ground-truth labels and a clustering, the Rand index RI is [27]:

$$\mathrm{RI} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{12}$$

and the adjusted Rand index ARI is defined by:

$$\mathrm{ARI} = \frac{\mathrm{RI} - E[\mathrm{RI}]}{\max(\mathrm{RI}) - E[\mathrm{RI}]} \tag{13}$$

C. AMI

Given two clusterings U and V, the mutual information MI between U and V is defined as [28]:

$$\mathrm{MI}(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} P(i,j) \log\left(\frac{P(i,j)}{P(i)P'(j)}\right) \tag{14}$$

Adjusted mutual information AMI is defined by:

$$\mathrm{AMI} = \frac{\mathrm{MI} - E[\mathrm{MI}]}{\mathrm{mean}(H(U), H(V)) - E[\mathrm{MI}]} \tag{15}$$

where $H(U)$ and $H(V)$ are entropy of U and V.

D. FMI

Given a set of ground-truth labels and a clustering, the Fowlkes-Mallows index FMI is defined as:

$$\mathrm{FMI} = \frac{\mathrm{TP}}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})}} \tag{16}$$

E. Auxiliary metrics

In addition, we report three auxiliary metrics, which do not evaluate the features' performance, but help us understand their qualities.

- *dimension*: the dimension of a feature matrix. Only applicable to one-hot matrices and TF–IDF matrices. If the dimension is very high, the feature matrix may be sparse and there may be an amount of noise.
- *n_clusters*: number of clusters returned by clustering algorithms. If this metric is too high, the dataset is split into too many small clusters; as a result, purity may be falsely high.
- *n_noise*: number of samples for which clustering algorithms fail to assign a cluster. Since our evaluation excludes the noise points, if this metric is too high, purity, ARI, AMI and FMI may be falsely high.

VIII. Results and discussions

Tables VII, VIII, IX and X show the evaluation results for *doc* and *archive*, using AVClass and Euphony as "labels", respectively.

Approach	dimension	n_clusters	n_noise	Purity	ARI	AMI	FMI
avclass	-	46	740	1.0000	1.0000	1.0000	1.0000
euphony	-	159	3	**0.9622**	**0.4437**	**0.4554**	**0.7649**
email	-	1764	0	**0.9466**	0.0210	0.1586	0.1970
tags_onehot	38	94	59	**0.9110**	0.0964	0.2570	**0.4301**
tags_tfidf	38	96	41	**0.9102**	0.0970	0.2569	**0.4299**
ssdeep	-	389	148	**0.9626**	0.0039	0.1494	0.0847
modules_loaded_tfidf	1634	544	230	**0.9270**	0.0068	0.1252	0.1462
processes_created_tfidf	691	618	237	**0.9367**	0.0277	0.1307	0.3090
processes_tree_tfidf	756	901	495	**0.9361**	0.0261	0.1125	0.3008
registry_keys_set_tfidf	5204	792	4370	**0.9032**	0.0128	0.1017	0.1324
files_attribute_changed_tfidf	15909	245	460	**0.8972**	0.0450	0.1270	0.3359
mutexes_created_tfidf	1047	241	181	**0.9111**	0.0669	0.1877	0.4055
registry_keys_deleted_tfidf	3191	1613	1302	**0.9082**	0.0010	0.0378	0.0509
mutexes_opened_tfidf	149	68	22	0.8884	**0.3694**	**0.3378**	**0.7884**
all_features_tfidf	23066	149	182	**0.9008**	**0.3733**	**0.3025**	**0.8144**
modules_loaded_ssdeep	-	463	357	**0.9221**	0.0072	0.1244	0.1483
processes_created_ssdeep	-	513	325	**0.9301**	0.0261	0.1272	0.2984
processes_tree_ssdeep	-	684	597	**0.9214**	0.0284	0.1112	0.3050
registry_keys_set_ssdeep	-	435	985	0.8848	0.0127	0.0660	0.1989
files_attribute_changed_ssdeep	-	234	513	0.8793	0.0371	0.1112	0.3117
mutexes_created_ssdeep	-	189	148	**0.9082**	0.0686	0.1934	0.4053
registry_keys_deleted_ssdeep	-	448	803	0.8897	0.0146	0.0886	0.2083
mutexes_opened_ssdeep	-	61	41	0.8887	**0.3673**	**0.3369**	**0.7888**
baseline	-	-	-	0.8320	-	-	-

TABLE VII: Evaluation results for *doc*, AVClass as "labels"

Approach	dimension	n_clusters	n_noise	Purity	ARI	AMI	FMI
avclass	-	46	740	**0.7587**	**0.4437**	**0.4554**	**0.7649**
euphony	-	159	3	1.0000	1.0000	1.0000	1.0000
email	-	1764	0	**0.8043**	0.0448	**0.2832**	0.1869
tags_onehot	38	94	59	**0.7046**	0.1520	**0.3772**	0.3717
tags_tfidf	38	96	41	**0.7037**	0.1528	**0.3776**	0.3715
ssdeep	-	389	148	**0.8400**	0.0110	**0.3058**	0.0924
modules_loaded_tfidf	1634	544	230	**0.7591**	0.0101	0.2165	0.1143
processes_created_tfidf	691	618	237	**0.7967**	0.0351	0.2375	0.2363
processes_tree_tfidf	756	901	495	**0.7955**	0.0393	0.2045	0.2354
registry_keys_set_tfidf	5204	792	4370	**0.7480**	0.0257	0.1468	0.1299
files_attribute_changed_tfidf	15909	245	460	**0.6816**	0.0600	0.1492	0.2636
mutexes_created_tfidf	1047	241	181	**0.6898**	0.0600	0.2128	0.2994
registry_keys_deleted_tfidf	3191	1613	1302	**0.7228**	0.0028	0.0724	0.0483
mutexes_opened_tfidf	149	68	22	0.6655	0.2108	**0.2675**	**0.5896**
all_features_tfidf	23066	149	182	0.6779	0.1800	0.2389	**0.6007**
modules_loaded_ssdeep	-	463	357	**0.7458**	0.0103	0.2111	0.1152
processes_created_ssdeep	-	513	325	**0.7821**	0.0367	0.2324	0.2320
processes_tree_ssdeep	-	684	597	**0.7606**	0.0429	0.1935	0.2395
registry_keys_set_ssdeep	-	435	985	0.6600	0.0062	0.0812	0.1352
files_attribute_changed_ssdeep	-	234	513	0.6559	0.0482	0.1233	0.2393
mutexes_created_ssdeep	-	189	148	**0.6840**	0.0624	0.2155	0.2995
registry_keys_deleted_ssdeep	-	448	803	0.6780	0.0162	0.1286	0.1535
mutexes_opened_ssdeep	-	61	41	0.6648	0.2080	**0.2640**	**0.5899**
baseline	-	-	-	0.6030	-	-	-

TABLE VIII: Evaluation results for *doc*, Euphony as "labels"

Approach	dimension	n_clusters	n_noise	Purity	ARI	AMI	FMI
avclass	-	121	278	1.0000	1.0000	1.0000	1.0000
euphony	-	297	9	**0.8720**	**0.3946**	**0.6063**	**0.5818**
email	-	1061	0	**0.8583**	**0.7433**	**0.5338**	**0.8577**
tags_onehot	11	11	7	0.6595	**0.3243**	0.2331	**0.6735**
tags_tfidf	11	13	1	0.6597	**0.3236**	0.2325	**0.6729**
ssdeep	-	274	554	**0.8134**	0.0300	0.2530	0.1592
processes_tree_tfidf	3076	415	470	0.6234	0.0047	-0.0013	0.1587
registry_keys_opened_tfidf	12626	699	577	0.6359	0.0005	-0.0021	0.0687
modules_loaded_tfidf	2997	691	838	0.6327	0.0003	0.0030	0.0411
files_opened_tfidf	11754	475	622	0.6196	-0.0046	-0.0061	0.1583
files_dropped_tfidf	9775	400	586	0.6211	0.0042	0.0016	0.1802
all_features_tfidf	27988	538	1379	0.6391	-0.0000	-0.0025	0.0489
processes_tree_ssdeep	-	295	539	0.6140	0.0033	0.0021	0.1414
registry_keys_opened_ssdeep	-	437	502	0.6191	0.0009	0.0011	0.0753
modules_loaded_ssdeep	-	470	673	0.6223	0.0002	0.0021	0.0590
files_opened_ssdeep	-	495	475	0.6170	0.0007	0.0019	0.0998
files_dropped_ssdeep	-	240	309	0.6093	0.0041	0.0047	0.1765
baseline	-	-	-	0.6074	-	-	-

TABLE IX: Evaluation results for *archive*, AVClass as "labels"

Approach	dimension	n_clusters	n_noise	Purity	ARI	AMI	FMI
avclass	-	121	278	**0.5760**	**0.3946**	**0.6063**	**0.5818**
euphony	-	297	9	1.0000	1.0000	1.0000	1.0000
email	-	1061	0	**0.5784**	**0.3130**	**0.4265**	**0.5290**
tags_onehot	11	11	7	0.3841	0.1205	0.2039	**0.4099**
tags_tfidf	11	13	1	0.3843	0.1208	0.2052	**0.4096**
ssdeep	-	274	554	**0.7869**	0.1370	**0.4392**	0.2600
processes_tree_tfidf	3076	415	470	0.3532	0.0046	0.0016	0.0956
registry_keys_opened_tfidf	12626	699	577	0.3977	0.0010	0.0022	0.0423
modules_loaded_tfidf	2997	691	838	0.4015	0.0005	0.0048	0.0253
files_opened_tfidf	11754	475	622	0.3567	-0.0050	-0.0039	0.0937
files_dropped_tfidf	9775	400	586	0.3498	0.0033	0.0012	0.1086
all_features_tfidf	27988	538	1379	0.4013	0.0005	0.0004	0.0306
processes_tree_ssdeep	-	295	539	0.3412	0.0046	0.0048	0.0870
registry_keys_opened_ssdeep	-	437	502	0.3605	0.0013	0.0040	0.0462
modules_loaded_ssdeep	-	470	673	0.3639	0.0006	0.0032	0.0360
files_opened_ssdeep	-	495	475	0.3632	-0.0009	-0.0002	0.0576
files_dropped_ssdeep	-	240	309	0.3249	0.0037	0.0047	0.1067
baseline	-	-	-	0.3015	-	-	-

TABLE X: Evaluation results for *archive*, Euphony as "labels"

Since our dataset is very skewed, purity could be falsely high because most data points happen to fall in the dominant family by its prediction. Therefore we introduced a hypothetical set where every label is the dominant label according to AVClass/Euphony. The purity of this hypothetical set and AVClass/Euphony equals to the ratio of the dominant family in AVClass/Euphony, and we use it as a baseline for purity. For both file types, evaluated by both AVClass and Euphony, every approach beats the baseline in the case of purity.

We highlight the best results with **boldface** fonts in the tables. For purity, if the purity of an approach is higher than the average of AVClass/Euphony and the baseline, we annotate it with boldface. For ARI, AMI and FMI, if the metric is higher than the average of the maximum and minimum, we annotate it with boldface. One can notice that the results for the two file types are quite different; within the same file type, the results for AVClass and Euphony as "labels" are quite similar (which also coordinates with assumption 1).

For the *doc* file type, most of our features extracted from static and dynamic analysis produce excellent results, in the case of purity. Some features, like *email*, *tags*, *ssdeep*, *mutexes_opened_tfidf*, *all_features_ssdeep* and *mutexes_opened_ssdeep*, also produces good results in ARI, AMI and FMI. *email*'s purity is among the highest, which confirms the novel approach can produce highly homogeneous clusters; on the other hand, *email* is not good evaluated by other metrics, but this is acceptable due to the reason explained in subsection VII-A. It is also notable that some of *all_features_ssdeep*'s metrics are better than any of the behavioral features—some even higher than Euphony.

As for the *archive* file type, the results are quite different. Our malware analysis-based approaches do not excel in behavioral features. However, *email* is quite high for all metrics, some of which is even much higher than that of Euphony.

There are also similarities between the two file types. For example, in our malware features, *tags* and *ssdeep* are the most stable, with some metric noticeably high for both file types. If we take *n_clusters* and *n_noise* into consideration, *tags* is the best feature among all—which is not so surprising, since it is a high-level summary derived from multiple analysis. Also, the results of *tags_onehot* and *tags_tfidf* are always so close, showing that representing *tags* with one-hot or TF–IDF does not have a significant effect, probabaly because each token occurs in each document for at most once. What's more, there proves to be little difference in performance, transforming behavioral features to whether TF–IDF feature matrices or *ssdeep* distance matrices.

As validation of assumption 1, we check AVClass and Euphony's purity, comparing to one another. For *doc*, we have $E(L_2, L_1, \text{purity}) = 0.9622$ and $E(L_1, L_2, \text{purity}) = 0.7587$; for *archive*, we have $E(L_2, L_1, \text{purity}) = 0.8720$ and $E(L_1, L_2, \text{purity}) = 0.5760$. This proves that AVClass and Euphony are fairly homogeneous for our dataset (the lowest purity, 0.5760 means in an AVClass cluster, there are $\frac{1}{0.5760} = 1.736$ Euphony labels on average). That strengths our belief in assumption 1.

IX. Conclusions and future

Our evaluation shows that the novel approach based on email campaigns is as good as—if not better—traditional approaches based on features retrieved from static and dynamic analysis. Because of its nature, the email-based approach splits one family into multiple clusters, resulting in low ARI, AMI and FMI for certain file types. However, this is acceptable in our context, for families distributed in multiple clusters can be merged again in later processes, as long as the clusters are homogeneous.

As for our own approach, which is based on features from malware analysis and evaluated on the same real dataset, there is not a universal conclusion for both file types and all features. This is perhaps based on different mechanisms of *doc* and *archive*, or the insufficiency of data available from VirusTotal, which is the best we can begin with when working with a real dataset. However, we do discover that features such as *tags* and *ssdeep* will produce excellent results, no matter for which file type.

We met with some difficulties when working with the real dataset. To begin with, as just mentioned, analysis reports may not be available from VirusTotal, as the real dataset is updated rapidly. Second, since we lack ground-truth labels, we have to used AVClass and Euphony results as "labels". Based on assumption 1, such a methodology seems quite promising, but can never be proved. Also, the real dataset is quite skewed, which makes the baseline for purity very high and may affect how we evaluate the performance of our approach.

In future, we will improve our work from the following aspects:

- Use our approach on a more balanced dataset with ground-truth label (not necessarily a real dataset).
- Use cluster ensemble techniques (as we have shown that merging multiple behavioral features may produce better results than any of those does separately).
- Make use of AV checker labels provided by VirusTotal, as AVClass and Euphony do.
- Try more sophisticated embedding techniques, rather than those we currently use.

References

[1] Renee Burton. "Distilling Malicious Campaigns in Spam". https : / / info . infoblox . com / resources - whitepapers - tools - of - the - trade - distilling - malicious - campaigns-in-spam.

[2] *VirusTotal*. https://www.virustotal.com/.

[3] *Kaspersky Security Bulletin 2019. Statistics*. https : / / securelist . com / kaspersky - security - bulletin - 2019 - statistics/95475/.

[4] *Kaspersky Security Bulletin 2020. Statistics*. https : / / securelist . com / kaspersky - security - bulletin - 2020 - statistics/99804/.

[5] Daniel Arp et al. "Drebin: Effective and explainable detection of android malware in your pocket." In: *Ndss*. Vol. 14. 2014, pp. 23–26.

[6] AS Fridrik Skulason and Vesselin Bontchev. "A new virus naming convention". In: *CARO meeting*. 1991.

[7] D Beck and J Connolly. "The common malware enumeration initiative". In: *Proceedings of the Virus Bulletin Conference*. 2006.

[8] Vesselin Bontchev. "Current status of the caro malware naming scheme". In: *Virus Bulletin (VB2005), Dublin, Ireland* (2005).

[9] Tom Kelchner. "The (in) consistent naming of malcode". In: *Computer Fraud & Security* 2010.2 (2010), pp. 5–7.

[10] *Malware Naming Hell: Taming the mess of AV detection names*. https://www.gdatasoftware.com/blog/2019/08/35146-taming-the-mess-of-av-detection-names.

[11] Marcos Sebastián et al. "Avclass: A tool for massive malware labeling". In: *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer. 2016, pp. 230–253.

[12] Médéric Hurier et al. "Euphony: Harmonious unification of cacophonous anti-virus vendor labels for Android malware". In: *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE. 2017, pp. 425–435.

[13] Linhai Song et al. "Learning from big malwares". In: *Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems*. 2016, pp. 1–8.

[14] Houtan Faridi, Srivathsan Srinivasagopalan, and Rakesh Verma. "Performance Evaluation of Features and Clustering Algorithms for Malware". In: *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2018, pp. 13–22.

[15] *Cuckoo Sandbox*. https://cuckoosandbox.org/.

[16] *Spam and phishing in 2019*. https : / / securelist . com / spam-report-2019/96527/.

[17] Aziz Mohaisen, Omar Alrawi, and Manar Mohaisen. "Amal: High-fidelity, behavior-based automated malware analysis and classification". In: *computers & security* 52 (2015), pp. 251–266.

[18] Philipp Trinius et al. *A malware instruction set for behavior-based analysis*. Tech. rep. University of Mannheim, Institute of Computer Science, 2009.

[19] Felipe N Ducau et al. "Automatic Malware Description via Attribute Tagging and Similarity Embedding". In: *arXiv preprint arXiv:1905.06262* (2019).

[20] Yanxin Zhang et al. "Familial clustering For weakly-labeled Android malware using hybrid representation learning". In: *IEEE Transactions on Information Forensics and Security* 15 (2019), pp. 3401–3414.

[21] *Detect and respond to cyber attacks, cyber threats*. https://www.lastline.com/.

[22] *File deep analysis platform*. https://sandbox.ti.qianxin.com/sandbox/page.

[23] Helmut Grohne Jesse Kornblum and Tsukasa OI. *ssdeep – Fuzzy hashing program*. https://ssdeep-project.github.io/ssdeep/index.html.

[24] Jesse Kornblum. "Identifying almost identical files using context triggered piecewise hashing". In: *Digital investigation* 3 (2006), pp. 91–97.

[25] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. "Density-based clustering based on hierarchical density estimates". In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2013, pp. 160–172.

[26] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[27] William M Rand. "Objective criteria for the evaluation of clustering methods". In: *Journal of the American Statistical association* 66.336 (1971), pp. 846–850.

[28] Nguyen Xuan Vinh, Julien Epps, and James Bailey. "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance". In: *The Journal of Machine Learning Research* 11 (2010), pp. 2837–2854.