# VAMO: Towards a Fully Automated Malware Clustering Validity Analysis

Roberto Perdisci
Dept. of Computer Science
University of Georgia
Athens, GA 30602
perdisci@cs.uga.edu

ManChon U
Dept. of Computer Science
University of Georgia
Athens, GA 30602
manchonu@cs.uga.edu

## ABSTRACT

Malware clustering is commonly applied by malware analysts to cope with the increasingly growing number of distinct malware variants collected every day from the Internet. While malware clustering systems can be useful for a variety of applications, assessing the quality of their results is intrinsically hard. In fact, clustering can be viewed as an *unsupervised learning* process over a dataset for which the complete ground truth is usually not available. Previous studies propose to evaluate malware clustering results by leveraging the labels assigned to the malware samples by multiple anti-virus scanners (AVs). However, the methods proposed thus far require a (semi-)manual adjustment and mapping between labels generated by different AVs, and are limited to selecting a *reference sub-set* of samples for which an agreement regarding their labels can be reached across a majority of AVs. This approach may bias the reference set towards "easy to cluster" malware samples, thus potentially resulting in an overoptimistic estimate of the accuracy of the malware clustering results.

In this paper we propose VAMO, a system that provides a *fully automated* quantitative analysis of the validity of malware clustering results. Unlike previous work, VAMO *does not seek a majority voting-based consensus* across different AV labels, and does not discard the malware samples for which such a consensus cannot be reached. Rather, VAMO explicitly deals with the inconsistencies typical of multiple AV labels to build a more representative reference set, compared to majority voting-based approaches. Furthermore, VAMO avoids the need of a (semi-)manual mapping between AV labels from different scanners that was required in previous work. Through an extensive evaluation in a controlled setting and a real-world application, we show that VAMO outperforms majority voting-based approaches, and provides a better way for malware analysts to automatically assess the quality of their malware clustering results.

## 1. INTRODUCTION

Due to the extensive use of *packing* and other code obfuscation techniques [6], the number of new malware samples collected by anti-virus[1] (AV) vendors has grown enormously in recent years, reaching tens or even hundreds of thousand of new malware samples collected per day (e.g., in 2010 Symantec collected 286 million distinct malware variants [19]). To cope with this increasingly growing number of malware samples and boost the scalability and effectiveness of current malware analysis infrastructures, a number of *malware clustering* and automatic malware categorization systems have been recently proposed [1, 2, 3, 8, 11, 15, 18].

The main objective of malware clustering systems is to group malware samples into *families*, whereby samples that are similar to each other can be considered as variants of the same malware family. Intuitively, malware clustering results can be useful in several ways. For example, new malware samples that are clustered with known malware variants of a given family $f$ may be also categorized as belonging to $f$. In turn, these newly discovered variants may be used to derive more generic malware detection signatures that have a better chance to match *future variants* of the same family [15]. In addition, malware clustering results may make it easier to identify new, previously unknown malware families [2], or may be used to perform malware triage [11], thus allowing malware experts to select only a small number of variants of a given malware family for manual analysis.

To take full advantage of the above mentioned benefits, malware clustering systems clearly need to be accurate. Unfortunately, it is very challenging to quantitatively assess the accuracy of malware clustering results, because of the lack of reliable ground truth. A common approach to validating the quality of malware clustering results is to compare them to a *reference clustering* obtained by leveraging family labels assigned to the samples by multiple AV scanners [2, 11]. To compensate for inconsistencies in the AV labels, both [2, 18] and [11] use a majority voting approach to select the samples for which an agreement regarding their *AV family* label can be reached. Therefore, a cluster in the reference clustering will include all samples belonging to the same AV family. However, while this approach may appear as a natural choice in absence of complete ground truth, Li et al. [12] have suggested that it may result in an overoptimistic estimate of the malware clustering accuracy. In particular, limiting the reference clustering to samples for which a majority voting-based consensus on the family label can be reached, and discarding the remaining ones, may reduce the reference clusters to only include "easy to cluster" malware samples (i.e., clear-cut cases of malware samples that are very similar to each other) [12], thus potentially causing the accuracy of the malware clustering results to be largely overestimated. In fact, the

[1]While "anti-malware" is probably a more appropriate term, we use "anti-virus" because that is the way in which many vendors of malware scanners and defense solutions still advertise their products.

experiments reported in [2] state that among 14,212 malware samples, a majority voting-based consensus could be reached only for 2,658 cases. That is, more than 80% of the samples in the clustering results had to be excluded from the cluster validity analysis.

In this paper we propose VAMO[2], a system that enables an automatic quantitative analysis of the validity of malware clustering results. Like previous work, VAMO leverages the labels assigned to malware samples by multiple AV scanners to construct a reference clustering. However, unlike previous work, VAMO *does not seek a majority voting-based consensus*, and does not discard the samples for which such a consensus cannot be reached. Rather, VAMO explicitly deals with (and aims to mitigate the effect of) the inconsistencies typical of the AV labels to build a more representative reference clustering. Furthermore, VAMO avoids the need of a (semi) manual mapping between AV labels from different scanners that was required in previous work (notice that while some efforts exist to standardize the "language" used to assign the AV labels (e.g., `http://maec.mitre.org`), so far they have not been successful). Also, we would like to emphasize that while AV labels suffer from some limitations, as we discuss more in detail in Section 7, they are used as a reference by many researchers because it is hard to obtain a more accurate ground truth for datasets containing tens of thousands of malware samples.

VAMO leverages historic malware archives and the related multiple AV labels to *learn an AV Label Graph* (see Figure 1). An *AV Label Graph* (see Section 5.1) is defined as an undirected weighted graph, which aims to: (1) automatically learn the mapping between malware family names assigned by different AVs, thus avoiding the need to manually build or adjust such mappings; (2) identify cases in which one (or more) AV scanners tend to inconsistently use several family names to label samples that belong to the same family according to other competitors' scanners; (3) learn the *level of similarity* between AV labels assigned by different AV scanners, by looking at the number of times that certain malware family labels are jointly assigned to the same samples. While the concept of *AV Label Graph* was first introduced in [15], here we refine its definition and use it in the context of our novel VAMO system. Also, it is worth noting that the *AV Label Graph* is only one component of the entire VAMO system.

Learning the *AV Label Graph* enables us to measure the similarity between malware samples in a dataset based purely on their AV labels (see Section 5.1 for details). As shown in Figure 1, given a malware dataset $\mathbb{M}$ and the related multiple AV labels assigned to its malware samples, we can (a) apply a third-party malware clustering algorithm (e.g., [2, 11, 15, 18]) on $\mathbb{M}$ to partition it in a number of malware clusters, (b) use VAMO to build a reference clustering for $\mathbb{M}$ using similarities among its samples measured according to their AV labels, and (c) compute the *level of agreement* between VAMO's reference clustering and the third-party malware clustering results, thus quantitatively assessing their quality.

In summary, this paper makes the following contributions:

- We propose a novel system, called VAMO, that enables a *fully automated* malware clustering validity analysis.
- We perform an extensive evaluation of how different types of AV label inconsistencies may negatively impact a validity analysis performed via majority voting-based approaches, and show the advantages that VAMO brings over previous work.
- We perform experiments with real-world malware archives, and demonstrate how VAMO can be applied in practice to assess the quality of malware clustering results over large malware datasets.

[2]Validity Analysis of Malware-clustering Outputs.

## 2. RELATED WORK

**Cluster Validity Analysis** Besides the clustering validity indexes reported in Halkidi et al.'s survey [7], which we summarize in Section 3.2, a number of alternative validity indexes have been proposed. In [17], Rendon et al. present a comparison of internal and external clustering validity indexes, while Meilă [14] and Pfitzner et al. [16] introduce a number of new metrics to compare two different clusterings. In [5], Fowlkes and Mallows introduce a measure of similarity between two hierarchical clusterings obtained by cutting the two dendrograms at heights $h_1$ and $h_2$, respectively, which yield the same number of clusters $k$. Than, for each value of $k$, the number of matching entries from the two different clusterings are counted to obtain a measure of comparison. Our approach to cluster validity analysis (Section 5) is inspired by [5]. However, our method does not focus on comparing different hierarchical clusterings. Rather, VAMO leverages hierarchical clustering to generate a *reference clustering dendrogram*, and compares third-party clustering results to this dendrogram by finding the cut height $h$ that yields the maximum agreement between the third-party results and VAMO's reference clustering.

**Malware Clustering** Bailey et al. [1] presented one of the first studies on behavior-based malware clustering. Furthermore, in [1] the authors presented a quantitative analysis of the inconsistency in the labels assigned by different AVs. Bayer et al. [2] introduced a much more scalable way to perform behavior-based malware clustering. In addition, they proposed to validate their clustering results by comparing them against a clustering obtained using a majority voting-based approach over multiple malware family labels assigned to the samples by six different AVs [2]. A similar validation approach was used in [18]. In [8], Hu et al. perform malware clustering using static analysis, instead of behavior-based features, by leveraging function-call graphs, while [11] introduces a system called BitShred that aims to improve scalability in malware clustering systems.

In [12], Li et al. discuss a number of challenges related to the evaluation of results generated by malware clustering systems. In particular, by using plagiarism detection algorithms to measure the similarity between malware samples, they show that a factor contributing to the strong results reported in [2] might be that the 2,658 validation instances selected via majority voting on multiple AVs are simply easy to classify. However, no complete solution is offered on how to perform a better malware clustering validity analysis. Our work is a step forward towards such a solution.

While most malware clustering systems are based on system-level behavior or static-analysis-based features, [15] proposed a malware clustering system that focuses on the network behavior of malware and introduced the concept of AV Label Graph, which we refine and use in this paper in the context of VAMO. It is worth noting that the use of AV Label Graphs in [15] is significantly different from this paper. Previous work did not present a comprehensive malware clustering validity analysis system, and the cohesion and separation validity indexes used in [15] were mainly *internal* validity indexes that required a significant amount of interpretation through manual analysis. On the other hand, VAMO introduces a comprehensive, fully automated malware clustering validity analysis process that can more readily be used to select the parameters of a malware clustering system, or to compare results obtained using different clustering algorithms.

| | AV1 | AV2 | AV3 | AV4 |
|---|---|---|---|---|
| Detected samples | 590,341 | 825,766 | 702,124 | 1,030,354 |
| Detection rate (%) | 53.3% | 74.5% | 63.4% | 93.0% |
| Distinct AV labels | 20,217 | 15,138 | 2,208 | 175,333 |
| Distinct family labels | 3,330 | 4,729 | 1,710 | 3,520 |
| Distinct first variants | 20,217 | 13,851 | 2,199 | 51,732 |

**Table 1: AV labels for a dataset of 1,108,289 distinct malware samples.**

## 3. BACKGROUND

In this Section, we first provide quantitative information regarding the inconsistency typical of multiple AV labels. Then, we discuss the background concepts that we will use to perform automated clustering validity analysis.

## 3.1 Measuring Inconsistency in AV Labels

In this Section, we aim to quantify the "inconsistency" typical of multiple AV labels that has been *qualitatively* discussed in previous work [2, 15, 18], and analyzed more in details in [1, 13]. Our main goal is to suggest that (semi-) manually creating a mapping between malware family labels and correct the inconsistent (or erroneous) labels, which was required in previous work to perform malware cluster validity analysis (e.g., in [2]), is in fact a fairly difficult task. In addition, we show that in a large number of cases no majority voting-based consensus can be reached. Our results confirm previous findings [1] by using a more recent and much larger malware dataset.

To this end, we performed a number of measurements over a large dataset of AV labels assigned by four different major AV vendors (namely, Symantec, McAffee, Avira, and Trend Micro) to a set of 1,108,289 distinct malware samples[3]. These malware samples were collected from different sources over the course of one entire year, from 2011-01-01 to 2011-12-31 (it is worth noting that we only consider malware samples that were detected as such by at least one out of the four AV scanners). The AVs used to scan the samples were updated daily, and each malware sample was scanned with each AV once a day for 30 days[4], starting from the day in which the sample was collected. In the following, we will refer to the four AV scanners, in no particular order, as AV1, AV2, AV3, and AV4. We intentionally mask the specific AV vendor names, when reporting the results, to avoid controversy (the results we report may be seen as damaging to one or more vendors, due to their low detection rate). After all, we do not intend to establish what vendor performs the best over our malware dataset. Rather, we focus on the inconsistencies in the malware labels, both within a given AV vendor as well as across vendors.

### 3.1.1 Overview

Table 1 summarizes our AV label dataset. As we can see, the detection rate, number of distinct (complete) labels, and the number of distinct malware family labels varies greatly across the different AV scanners. For example, AV3 assigned a label to 702,124 (63.4%) malware samples, but the number of distinct labels was only 2,208. This means that, in average, the same label was assigned to 317 different samples. This behavior is very different from the other AVs, and in particular from AV4 for which in average the same label was assigned to (approximately) six samples. In addition, among the 1,108,289 distinct malware samples, only 420,920 (38%) were labeled (i.e., detected) by more than two different AVs. This suggests that because a majority voting approach would require three out of four AVs to agree on the labels (two out of four would only represent a tie), in our example scenario no majority voting-based consensus can be reached on the correct malware family label for at least 38% of the samples. This problem is exacerbated by the fact that even in the cases in which three or more labels are available, the AVs may not agree on the family those samples belong to, as we discuss in Section 3.1.2

### 3.1.2 Family Labels

We now focus on malware family names, rather than considering full AV labels. We will consider the malware cluster Clust. 1 shown below as an example, to explain how we derive the malware family names. This malware cluster was obtained using [15]. In Clust. 1, each row represents a malware sample (indexed by the last four bytes of its MD5 sum), and reports the labels assigned to the sample by three different AVs, namely McAfee (M), Avira (A), and Trend Micro (T).

**Clust. 1** Malware cluster with inconsistent AV labels.

```
b1b6da81    M=W32/Virut.gen     A=TR/Drop.VB.DU.1    T=PE_VIRUT.XO-1
ec34ca31    M=W32/Virut.gen     A=TR/Drop.VB.DU.1    T=PE_VIRUT.XO-1
c2276216    M=W32/Virut.gen     A=W32/Virut.E.dam    T=PE_VIRUT.NS-4
089ae4f5    M=W32/Virut.gen     A=W32/Virut.AX       T=PE_VIRUT.D-1
8ba552c9    M=W32/Virut.gen     A=TR/Drop.VB.DU.1    T=PE_VIRUT.XO-1
8cb0ab6c    M=W32/Virut.gen     A=WORM/Korgo.U       T=PE_VIRUT.D-4
b0b75f70    M=W32/Virut.gen     A=W32/Virut.X        T=PE_VIRUT.XO-1
a306b4e7    M=W32/Virut.gen     A=W32/Virut.Gen      T=PE_VIRUT.D-1
337a2cf4    M=W32/Virut.gen     A=W32/Virut.Gen      T=PE_VIRUT.D-1
62d18c7e    M=W32/Virut.gen     A=W32/Virut.Gen      T=PE_VIRUT.D-1
8dbca633    M=W32/Virut.gen     A=TR/Drop.VB.DU.1    T=PE_VIRUT.XO-1
ac433383    M=W32/Virut.n       A=W32/Virut.Gen      T=PE_VIRUX.A-3
cae61d9e    M=W32/Virut.gen     A=W32/Virut.X        T=PE_VIRUT.XO-2
7cc795f1    M=W32/Virut.gen     A=W32/Virut.Gen      T=PE_VIRUT.D-1
8de5214b    M=W32/Virut.gen.a   A=W32/Virut.AM       T=PE_VIRUT.XY
4d26cb0a    M=W32/Virut.gen     A=W32/Virut.Gen      T=PE_VIRUT.D-1
9fb75631    M=W32/Virut.n       A=W32/Virut.Gen      T=PE_VIRUX.A-3
229004b9    M=W32/Virut.gen     A=W32/Virut.X        T=PE_VIRUT.XO-1
28a85d8a    M=W32/Virut.gen     A=TR/Drop.VB.DU.1    T=PE_VIRUT.XO-1
663c5f6c    M=W32/Virut.d       A=W32/Virut.Z        T=PE_VIRUT.GEN-2
de6f1e00    M=                  A=W32/Virut.Gen      T=PE_VIRUT.D-4
1ff43bca    M=                  A=W32/Virut.X        T=PE_VIRUT.XO-4
ea580f6d    M=W32/Virut.n       A=W32/Virut.Gen      T=PE_VIRUX.A-3
a844eeff    M=W32/Virut.gen     A=TR/Drop.VB.DU.1    T=PE_VIRUT.XO-1
4f8613fd    M=W32/Virut.gen     A=TR/Drop.VB.DU.1    T=PE_VIRUT.XO-1
```

To derive the malware family name, we split each label into substrings divided by the '.' symbol, and we extract the first substring. For example, W32/Virut.gen becomes W32/Virut (Symantec uses a slightly different notation, compared to the other AV vendors. To extract the family label from Symantec's labels, we consider the first two substrings obtained by splitting the labels by the '.' symbol. For example, W32.Sality.AE would become W32.Sality).

As we can see from Clust. 1, in this case both McAfee and Trend Micro are very consistent, because they label the vast majority of the samples as belonging to the *Virut* malware family, with the exception of two samples that were missed by McAfee and three samples that are labeled as PE_VIRUX (rather than PE_VIRUT) by Trend Micro. On the other hand, Avira is much less consistent, because it assigned three different family names to the samples (i.e., TR/Drop, W32/Virut, WORM/Korgo).

Table 1 reports the total number, per AV, of distinct family names obtained from all labels in our datasets. Also, Table 1 reports the total number, per AV, of distinct "first variant" labels, i.e., labels obtained by combining the first two label substrings (the first three, in case of Symantec). Again, there is a relatively large difference between the numbers obtained from different AVs.

To measure the number of common family names per sample

[3]This dataset was kindly provided by a well-known security company.

[4]If an AV scanner AVi detected a sample $m$ and assigned it a label on day $d < 30$, the data collector would stop scanning $m$ with AVi for the remaining days, but continued scanning the sample with the other AVs until they also assigned a label or $d > 30$.

across different AVs, we further normalized the family names, for example by cutting the label prefix (e.g., `W32/`, `PE_`, etc.) and reducing all labels to lower case. For example, the first sample in Clust. 1 would be labeled as {`virut`, `drop`, `virut`}. This was done to maximize the number of common family names we could find for a given sample across different AVs. Even after this normalization, we could find a common family label across at least three out of four AVs for only 2.4% of the samples, and a common label across at least two out of four AVs for only 5.6% of the samples. Performing a manual mapping between the labels to mitigate the effect of different "terminology" used by different AVs may improve on these results. However, even after such manual mapping a majority voting-based consensus between the AVs cannot be reached for the vast majority of the samples. This findings are consistent with the experiments conducted in [2], in which a majority voting-based consensus could be reached only for less than 20% of the samples. Therefore, a reference clustering generated via majority voting may miss to represent a large portion of the malware dataset, causing a potential overestimate of the clustering quality, as also suggested in [12].

## 3.2 Validity Indexes

Clustering can be viewed as an *unsupervised learning* process over a dataset for which the complete ground truth is usually not available. Therefore, unlike in supervised learning settings, analyzing the *validity* of the clustering results is intrinsically hard. The assessment of the quality of clustering results often involves the use of subjective criteria of optimality [10], which are typically application specific, and commonly involves extensive manual analysis by domain experts. To aid the clustering validation process, a number of methods and quality indexes have been proposed [7, 9]. Halkidi et al. [7] provide a survey of cluster validity analysis techniques, which aim to evaluate the clustering results to *find the partitioning that best fits the underlying data*.

Three main *cluster validity* approaches are described [7]: (1) *external criteria* evaluate the clustering results by comparing them to a pre-specified structure, or *reference clustering*; (2) *internal criteria* rely solely on quantities derived from the data vectors in the clustered dataset (e.g., using a proximity matrix, and computing quantities such as inter- and intra-cluster distances); (3) *relative criteria* compare clustering results obtained using the same clustering algorithm with different parameter settings, to identify the best parameter configuration.

External validation criteria are particular attractive, because they offer a quantitative way to measure the *level of agreement* between the obtained clustering results and a reference clustering that is considered to be the ground truth [7, 16]. However, the main problem is exactly how to construct the reference clustering in the first place. This is one of the problems we address in this paper: building a reference clustering that can be used for validating the results of malware clustering systems.

Assuming a reference clustering is available, different external validity indexes can be used for measuring the quality of the clustering results. We briefly describe some of them below. Let $\mathcal{M}$ be our dataset, $\mathbf{Rc} = \{Rc_1, .., Rc_s\}$ be the set of *s reference clusters*, and $\mathbf{C} = \{C_1, .., C_n\}$ be our clustering results over $\mathcal{M}$. Given a pair of data samples $(m_1, m_2)$, with $m_1, m_2 \in \mathcal{M}$, we can compute the following quantities:

- $a$ is the number of pairs $(m_1, m_2)$ for which if both samples belong to the same reference cluster $Rc_i$, they also belong to the same cluster $C_j$.

- $b$ is the number of pairs $(m_1, m_2)$ for which both samples belong to the same reference cluster $Rc_i$, but are assigned to two different clusters $C_k$ and $C_h$.

- $c$ is the number of pairs $(m_1, m_2)$ for which both samples belong to the same cluster $C_i$, but are assigned to two different reference clusters $Rc_k$ and $Rc_h$.

- $d$ is the number of pairs $(m_1, m_2)$ for which if the samples belong to two different reference clusters $Rc_i$ and $Rc_j$, they also belong to different clusters $C_l$ and $C_m$.

Based on the above definitions, we can compute the following external cluster validity indexes [7]:

- **Rand Statistic**. $RS = \frac{a+d}{a+b+c+d} = \frac{a+d}{|\mathcal{M}|}$

- **Jaccard Coefficient**. $JC = \frac{a}{a+b+c}$

- **Folkes and Mallows Index**. $FM = \frac{a}{\sqrt{(a+b)(a+c)}}$

For all three indexes above, which take values in $[0, 1]$, higher values indicate a closer similarity between the clustering $\mathbf{C}$ and the reference clustering $\mathbf{Rc}$.

The authors of [2, 18], proposed to use different indexes, based on *precision* and *recall*, to measure the level of agreement between behavior-based malware clustering results $\mathbf{C}$ and a (semi-)manually generated reference clustering $\mathbf{Rc}$ derived by using majority voting over multiple AV labels. In this setting, precision and recall, and the related $F1$ index, are defined as follows:

- **Precision**. $Prec = 1/n \cdot \sum_{j=1}^{n} \max_{k=1,..,s}(|C_j \cap Rc_k|)$

- **Recall**. $Rec = 1/s \cdot \sum_{k=1}^{s} \max_{j=1,..,n}(|C_j \cap Rc_k|)$

- **F1 Index**. $F1 = 2\frac{Prec \cdot Rrec}{Prec + Rrec}$

In the remainder of the paper, we will often refer to the external validity indexes defined above.

## 4. SYSTEM OVERVIEW

Figure 1 provides a high-level overview of VAMO. We assume that a third-party has employed a malware clustering system, for example one of the systems proposed in [2, 11, 15], to partition a malware dataset $\mathbf{M}$ into a set of clusters $\mathbf{C} = \{C_1, C_2, .., C_x\}$, with $\bigcup_{i=1}^{x} C_i = \mathbf{M}$. VAMO's objective is to validate the quality of $\mathbf{C}$ (i.e., the malware clustering results). We now provide a description of VAMO's components shown in Figure 1.

**AV Label Dataset** Given a large *historic archive dataset* of malware samples $\mathcal{A}$ (which is different from $\mathbf{M}$), we first collect the set of family labels assigned by multiple AV scanners to each of the malware samples $m_k \in \mathcal{A}$. The resulting AV labels dataset can be represented as a set of tuples $\mathcal{L} = \{(l_{k,1}, l_{k,2}, .., l_{k,\nu})\}_{k=1..n}$, where $l_{k,i}$ is the malware family label assigned by the $i$-th of $\nu$ AV scanners to malware sample $m_k$, with $k = 1, .., n$, and $n = |\mathcal{A}|$. If an AV scanner misses to detect a malware sample, the related label in the set $\mathcal{L}$ will be assigned a *unique* placeholder "unknown" family label. It is worth noting that the malware dataset $\mathcal{A}$ need not contain actual executable malware samples. In fact, $\mathcal{A}$ may simply contain a list of hashes (e.g., `md5` or `sha1`) computed by a third party (e.g., the owner of a large malware dataset who cannot share the malware itself) over known malware samples. In this case, the label dataset $\mathcal{L}$ may be obtained by querying a service such as `virustotal.com` to obtain, for each hash, the related malware family labels from multiple AV scanners.

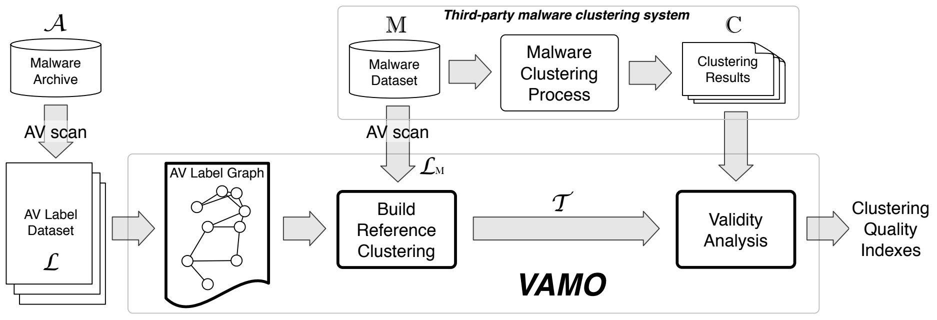


**Figure 1: VAMO System Overview**

**AV Label Graph** VAMO uses the label dataset $\mathcal{L}$ to *learn* an *AV Label Graph* (defined formally in Section 5.). Basically, a node in the graph represents a malware family name attributed by a certain AV scanner (or AV, for short) to one or more malware samples in $\mathcal{A}$. For example, assuming the $i$-th AV assigned the label `family_x` to at least one malware sample, the *AV Label Graph* will contain a node called `AVi_family_x`. Two nodes, say `AVi_family_x` and `AVj_family_y`, will be connected by an undirected edge if there exists at least one malware sample $m_k \in \mathcal{A}$ that has been assigned label `family_x` by the $i$-th AV, and `family_y` by the $j$-th AV, respectively. Each edge is assigned a weight that depends on the number of times that the connected nodes (i.e., the connected labels) were assigned to a same malware sample. Notice that if the $i$-th AV missed to detect a given malware sample, the related missing label will be replace by a label such as `AVi_unknown_U`, where `U` is a unique identifier.

**Reference Clustering** Similarly to what we did with $\mathcal{A}$, given the malware dataset $\mathbf{M}$ (i.e., the input to the third-party clustering system), we first collect the set of labels assigned by $\nu$ different AVs to each of the malware samples $m_k \in \mathbf{M}$, thus obtaining a dataset $\mathcal{L}_M$ consisting of a tuple (or vector) of family labels $L_k = (l_{k,1}, l_{k,2}, .., l_{k,\nu})$ per each sample $m_k$. At this point, we leverage the previously learned *AV Label Graph* to measure the *dissimilarity* (or distance) between samples in $\mathbf{M}$ according to their malware family labels. Specifically, we measure the distance between two malware samples $m_i, m_j \in \mathbf{M}$ by measuring the distance between their respective label vectors $L_i$ and $L_j$ in the graph. We give a formal definition of label-based distance between malware samples in Section 5.1. At a high level, we compute the distance between two samples $m_i, m_j$ by computing the median among the shortest paths in the *AV Label Graph* between all paris of labels $l_k, l_h$, with $l_k \in L_i$ and $l_h \in L_j$. This allows us to compute an $r \times r$ distance matrix $\mathbf{D}$, where $r = |\mathbf{M}|$ and element $\mathbf{D}[i, j]$ is the distance between samples $m_i, m_j$. The final reference clustering is obtained by applying average-linkage hierarchical clustering [9, 10] on the distance matrix $\mathbf{D}$. The result is not an actual partitioning of the malware dataset $\mathbf{M}$. Rather, the reference clustering is represented by a *dendrogram* [9], i.e., a tree-like data structure that expresses the "relationship" between malware samples. Cutting this dendrogram at any particular height would produce a partitioning of $\mathbf{M}$ according to the AV label-based distances (see Section 5 for details).

**Validity Analysis** Let $\mathcal{T}$ be the reference clustering dendrogram output by the previous step. The *Validity Analysis* module takes in input $\mathcal{T}$ and the set of malware clusters $\mathbf{C}$ output by the third-party malware clustering system. At this point, VAMO applies the *external* validity indexes introduced in Section 3 to compute the maximum level of agreement between $\mathbf{C}$ and all possible reference clusterings obtained by cutting $\mathcal{T}$ at different heights. For example, we can compute the maximum Jaccard coefficient $\hat{J}$ between all possible reference clusterings and $\mathbf{C}$. The higher $\hat{J}$, the stronger the agreement between $\mathbf{C}$ and the AV label-based reference clustering.

Effectively, VAMO compares the third-party clustering results $\mathbf{C}$ to a reference clustering obtained by partitioning the dataset $\mathbf{M}$ according to the relationships among multiple AV labels learned from the archive malware dataset $\mathcal{A}$. It is worth noting that this process has some similarities with the majority voting-based approach used in previous work. In fact, the effect of the majority voting approach is to group a subset of the malware in $\mathbf{M}$ according to the labels assigned by multiple AVs to the samples in the very same $\mathbf{M}$ dataset. VAMO is different because (a) it automatically learns the relationships among malware family labels assigned by different AVs, and does not require any manual (or semi-manual) mapping between them; (b) it introduces a measure of label-based distance between malware samples that is not limited to the cases in which a majority voting-based consensus can be achieved; (c) it enables the computation of well known external validity indexes over the entirety of malware clustering results, rather than focusing only on "easy-to-cluster" subset of the malware dataset. In Section 6.1 we empirically show that building a reference clustering based on the *AV Label Graph* and applying the validity analysis process outlined above outperforms the majority voting-based cluster validation approach proposed in previous work.

## 5. VALIDITY ANALYSIS

In this Section, we provide more details on how VAMO builds the reference clustering by leveraging multiple AV labels, and how the clustering validity indexes are computed to compare third-party malware clustering results to VAMO's reference clustering.

### 5.1 Building a Reference Clustering

As mentioned in Section 4, the first step to obtaining the reference clustering is to build an *AV Labels Graph*. This graph expresses the "relationships" between different AV labels, and *automatically learns* the likelihood that different labels from different AVs will be assigned to the same malware sample, based on historic observations.

Assume $\mathbf{M}$ is the malware dataset used as input to a third-party (e.g., behavior-based) malware clustering system, as shown in Fig-

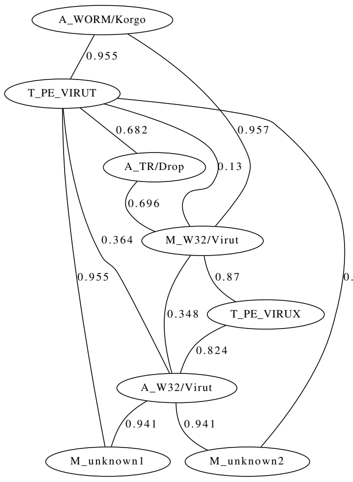


**Figure 2: AV Label Graph for Clust.1 (see Section 3.1)**

ure 1. Also, let $\mathcal{A}$ be a large historic malware archive containing, for example, malware samples collected during the past several months, and that $\mathbf{M} \subset \mathcal{A}$ (i.e., $\mathcal{A}$ contains all the "current" samples collected in $\mathbf{M}$, plus a large set of malware samples collected in the past). We define an *AV Label Graph* learned from $\mathcal{A}$ as follows.

DEFINITION 1. - ***AV Label Graph***. *An AV Label Graph is an undirected weighted graph. Given an archive of $n$ malware samples $\mathcal{A} = \{m_i\}_{i=1..n}$, let $\mathcal{L} = \{L_1 = (l_1, .., l_\nu)_1, .., L_n = (l_1, .., l_\nu)_n\}$ be a set of label vectors, where a label vector $L_h = (l_1, .., l_\nu)_h$ is an ordered set of malware family labels assigned by $\nu$ different AV scanners to malware $m_h \in \mathcal{A}$. The AV Label Graph $\mathcal{G} = \{V_k, E_{k_1,k_2}\}_{k=1..l}$ is constructed by adding a node $V_k$ for each distinct label $l_k \in \mathcal{L}$. Two nodes $V_{k_1}$ and $V_{k_2}$ are connected by a weighted edge $E_{k_1,k_2}$ if the labels $l_{k_1}$ and $l_{k_2}$ related to the two nodes appear at least once in the same label vector $L_h \in \mathcal{L}$ (that is, if they are both assigned to a malware sample $m_h$). Each edge $E_{k_1,k_2}$ is assigned a weight $w = 1 - \frac{m}{\max(n_1,n_2)}$, where $n_1$ is the number of label vectors $L_h \in \mathcal{L}$ that contain $l_{k_1}$, $n_2$ is the number of vectors that contain $l_{k_2}$, and $m$ is equal to the number of vectors containing both $l_{k_1}$ and $l_{k_2}$.*

For example, assume $\mathcal{A}$ contains all (and only) the samples shown in Clust. 1 (shown in Section 3). In this case, the related *AV Label Graph* is shown in Figure 2. Notice that in reality $\mathcal{A}$ will typically contain thousands of samples, and that the graph in Figure 2 is reported simply to provide an example of how the *AV Label Graph* is computed. Also, notice that the missing labels were replaced with *unique* "unknown" identifiers.

Once the *AV Label Graph* is computed, we build a reference clustering *dendrogram* as follows (notice that a dendrogram is a tree-like data structure generated by hierarchical clustering [9]). Given any two samples $m_i, m_j \in \mathbf{M}$ we first "map" each sample onto the graph, and then compute the distance $d_{i,j}$ between $m_i, m_j$ on the graph, thus obtaining a distance matrix $\mathbf{D}$ in which $\mathbf{D}[i,j] = d_{i,j}$. A more formal definition of graph-based distance between malware samples is given below.

DEFINITION 2. - ***Graph-based Distance***. *Let $m_i \in \mathbf{M}$ be a malware sample, and $L_i = (l_1, .., l_\nu)_i$ be its label vector. By definition, each label $l_{h,i} \in L_i$ corresponds to a node $V_{h,i}$ in the AV Label Graph, with $h = 1, .., \nu$. Therefore, sample $m_i$ can be mapped to a list $\mathbf{V}_i = (V_{1,i}, .., V_{\nu,i})$ of $\nu$ nodes in the graph. Now, let $\mathbf{V}_i$ and $\mathbf{V}_j$ be the lists of nodes related to $m_i$ and $m_j$, respectively. To compute the distance $d_{i,j}$ between $m_i, m_j$, we first compute the length of the shortest path $p_k$ among a pair of nodes ($V_{k,i}$,$V_{k,j}$), for each $k = 1, .., \nu$. Then, we compute $d_{i,j}$ as the median among all $p_k$, with $k = 1, .., \nu$.*

After computing the distance matrix $\mathbf{D}$, we apply average-linkage hierarchical clustering, which outputs a *dedrogram* $\mathcal{T}$ that expresses the "relationship" between the malware samples in $\mathbf{M}$ according to their AV labels. Section 5.2 explains in details how the reference clustering dendrogram $\mathcal{T}$ can be used to validate third-party clustering results.

## 5.2 Computing the Validity Indexes

As mentioned above, by cutting the reference clustering dendrogram $\mathcal{T}$ at a given height $h$, we obtain an actual partitioning of the dataset $\mathbf{M}$ into a set of reference clusters $\mathbf{Rc} = \{Rc_1, .., Rc_w\}$. Then, the *level of agreement* between $\mathbf{Rc}$ and the third-party clustering results $\mathbf{C} = \{C_1, C_2, .., C_x\}$ can be computed using the external validity indexes introduced in Section 3.2. Naturally, different values of $h$ will produce a different set of reference clusters, and therefore the values of these validity indexes will also differ. Therefore, to decide where exactly to cut the dendrogram $\mathcal{T}$ we proceed as follows. Let $\mathbf{Rc}(h)$ be the set of reference clusters obtained by cutting $\mathcal{T}$ at height $h$. Also, assume $I(\mathbf{Rc}(h), \mathbf{C})$ is an external validity index computed over the clusterings $\mathbf{Rc}(h)$ and $\mathbf{C}$ (e.g., $I(\cdot)$ could be equal to the Jaccard index, or one of the other indexes outlined in Section 3.2). We then cut $\mathcal{T}$ at height $h^* = \text{argmax}_h\{I(\mathbf{Rc}(h), \mathbf{C})\}$, so that $h^*$ is the cut at which the level of agreement between $\mathbf{C}$ and the VAMO's reference clustering is maximum.

In summary, we perform hierarchical clustering of the malware samples in $\mathbf{M}$ according to similarities in their AV labels by leveraging the previously learned *AV Label Graph*, and then we find the set of reference clusters $\mathbf{Rc}(h^*)$ that *best explains* (or *agrees* with) the third-party clustering results $\mathbf{C}$. This is useful because given two different third-party results $\mathbf{C_1}$ and $\mathbf{C_2}$ (e.g., given by the same behavior-based malware clustering systems configured with different parameter values, or given by different malware clustering systems), VAMO allows us to establish which of them has the *highest level of agreement with the underlying multiple AV labels*.

## 6. EVALUATION

## 6.1 VAMO v.s. Majority Voting

In this Section, we present a set of experiments performed in a controlled setting. Our objective is to show that, when faced with noisy AV labels, VAMO outperforms majority voting-based approaches. Namely, in the vast majority of cases VAMO produces an AV label-based reference clustering that better explains (or agrees with) the *true* malware clusters. To this end, we use the following high-level approach. We simulate a controlled dataset of malware samples for which we know exactly what samples should belong to what malware cluster, and first assume that all samples are perfectly (i.e., correctly) labeled by multiple AVs. Then, we gradually introduce more and more noise into the AV labels, thus simulating the inconsistent labeling typical of real-world AVs (see Section 3.1). For each noise increase, we apply both VAMO and a majority voting-based approach to obtain an AV label-based reference clustering, and the obtained results show that VAMO's reference clustering yields validity indexes that offer a higher level of agreement with the *true* malware clusters, compared to using majority voting.

### 6.1.1 Controlled Datasets

We create a synthetic dataset to simulate a scenario in which we have a historic archive $\mathcal{A}$ consisting of 3,000 distinct malware samples and the related dataset $\mathcal{L}$ of labels assigned by three different AV scanners to each of these 3,000 samples. Furthermore, we create a dataset $\mathbf{M}$ containing 300 distinct samples, with $\mathbf{M} \subset \mathcal{A}$ (i.e., $\mathbf{M}$ is a proper subset of $\mathcal{A}$). Therefore, the label dataset $\mathcal{L}_M$ containing the AV labels for the malware samples in $\mathbf{M}$ can be directly obtained from $\mathcal{L}$ (since $\mathbf{M} \subset \mathcal{A}$, then $\mathcal{L}_M \subset \mathcal{L}$). It is worth noting that we named these datasets following the same terminology that we used in Section 4 and in Figure 1.

At first, we assume to have perfect knowledge (i.e., perfect *ground truth*) regarding the malware family each sample belongs to. Specifically, we construct the datasets so that the samples in $\mathcal{A}$ (and the related malware labels in $\mathcal{L}$) belong to 15 different malware families, with 200 samples per family, and that each of the three AVs consistently assigns the correct malware family name to the samples in $\mathcal{A}$, and therefore also to the samples in $\mathbf{M}$. In practice, to obtain $\mathbf{M}$ we simply randomly (uniformly) select 300 samples from $\mathcal{A}$. Also, since we know exactly what malware belong to what family, we can precisely partition the dataset $\mathbf{M}$ into a set of 15 malware clusters $\mathbf{C} = \{C_1, C_2, ..., C_s\}$, with $s = 15$.

It is worth noting that in this idealized scenario we also assume the AVs use the very same family names for the malware family labels. In other words, we assume the AVs all agree on using the same terminology or notation. This means that no manual mapping between family names assigned by different AVs is needed, and a majority voting-based approach can be applied directly. This typically does not hold in practice, in which case we would need to obtain the name mapping before being able to apply majority voting. On the other hand, VAMO is agnostic to differences in the terminology that the AVs use to assign malware family names, because VAMO will automatically learn the relationships between different malware family names through the *AV Label Graph* construction, as discussed in Section 4 and Section 5.

### 6.1.2 Simulating Inconsistency in the AV Labels

To simulate inconsistency in the AV labels, we proceed as follows. We start from the label dataset $\mathcal{L}$ described above, and we progressively inject more and more noise into the labels. Specifically we inject the following two types of noise:

- **Label Flips** Given a malware $m_k \in \mathcal{A}$, and its label vector $L_k = (l_{1,k}, l_{2,k}, l_{3,k}) \in \mathcal{L}$, with probability $p'_f$ we replace label $l_{\nu,k}$ with a different label $l'_{\nu,k}$ chosen among the 14 other possible malware family labels, where the probability $p'_f$ is a preset "probability of flip".

- **Missing Labels** Similarly, given a malware $m_k \in \mathcal{A}$, and its label vector $L_k = (l_{1,k}, l_{2,k}, l_{3,k}) \in \mathcal{L}$, with probability $p'_m$ we drop label $l_{\nu,k}$ to simulate the case in which the $\nu$-th AV missed to detect $m_k$, where the probability $p'_m$ is a preset "probability of missed detection".

These two types of noise can affect, with different preselected probabilities, either one, two, or three AVs. To better explain this, let $\mathsf{n} = [p_f, p_m; p_1, p_2, p_3]$ be a "noise vector" whose elements express the following probabilities: $p_f$ is the overall probability that a malware sample $m$ will be affected by a label flip, while $p_m$ is the overall probability that a sample will be affected by a missing label; on the other hand, $p_x$ (with $x$ =1,2, or 3) represents the probability that the noise (through label flips and/or missing labels) will affect exactly $x$ out of the three AVs, for a given malware sample. Notice that $p_1 + p_2 + p_3 = 1$, and $p_f + p_m \leq 1$. Namely, with probability $1 - (p_f + p_m)$ a sample will not be affected by any noise (i.e., the sample remains perfectly labeled).

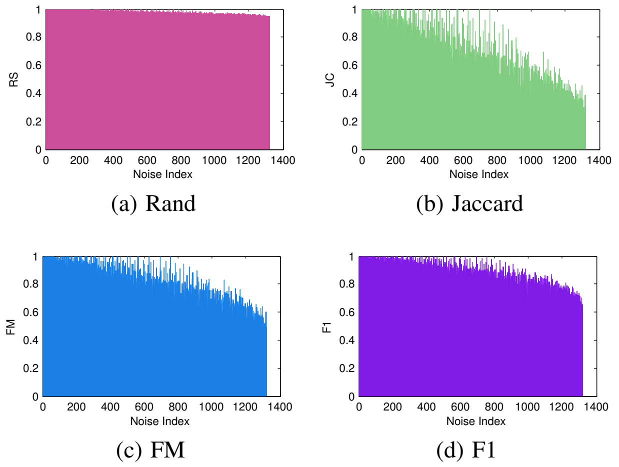


**Figure 3: VAMO, absolute values of cluster validity indexes.**

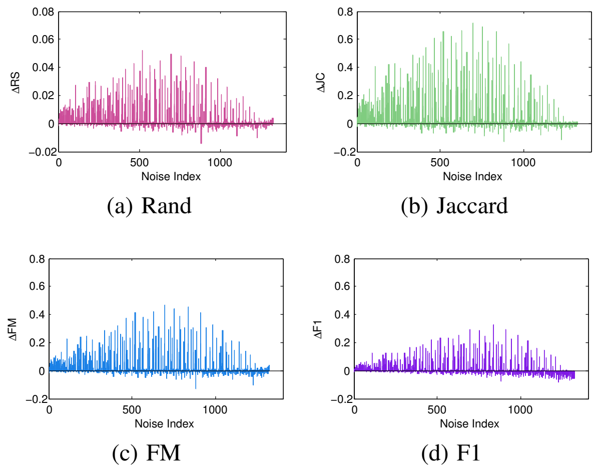


**Figure 4: VAMO vs. Majority Voting (index "deltas").**

### 6.1.3 Building a Reference Clustering via Majority Voting

Intuitively, the majority voting-based approach to construct a reference clustering works as follows. Given a malware sample $m_i \in \mathbf{M}$, and the label vector $L_i = (l_{1,i}, l_{2,i}, l_{3,i}) \in \mathcal{L}_M$ containing the malware family labels assigned to $m_i$ by the three AVs, $m_i$ is assigned to malware cluster $R_j$ if the majority of labels in $L_i$ indicate that $m_i$ belongs to family $f_j$. If no majority-based consensus can be reached (i.e., the majority of AVs disagree on the family name attributed to $m_i$), then the sample $m_i$ is assigned to a *singleton cluster*, namely a cluster that contains only $m_i$. Following this approach, we can partition the dataset $\mathbf{M}$ into a set of majority voting-based reference clusters $\mathbf{Rc}^{MV} = \{Rc_1^{MV}, R_2^{MV}, ..., R_q^{MV}\}$. Then, given $\mathbf{Rc}^{MV}$ and the *ground truth* clusters $\mathbf{C} = \{C_1, ..., C_s\}$ (which are derived before injecting the noise into the AV labels), we can compute the four external validity indexes described in Section 3.2.

### 6.1.4 Computing the Validity Indexes

Let $\mathsf{n}$ be a particular noise vector, with a given combination of values for the probabilities $p_f$, $p_m$, $p_1$, $p_2$, and $p_3$. Applying the noise injection approach described above results in a noisy label dataset $\mathcal{L}(\mathsf{n})$. In turn, if from $\mathcal{L}(\mathsf{n})$ we only consider the labels related to the malware samples in $\mathbf{M}$, we can obtain a (noisy) label dataset $\mathcal{L}_M(\mathsf{n})$ (notice that because $\mathbf{M} \subset \mathcal{A}$, then $\mathcal{L}_M(\mathsf{n}) \subset \mathcal{L}(\mathsf{n})$).

Given $\mathcal{L}(\mathsf{n})$ and $\mathcal{L}_M(\mathsf{n})$, we apply VAMO to compute four validity indexes (see Figure 1), thus essentially measuring the *level of agreement* (see Section 4) between the reference clustering derived from the *AV Label Graph* learned from $\mathcal{L}(\mathsf{n})$, and the *ground truth* clusters $\mathbf{C} = \{C_1, C_2, ..., C_s\}$ in which $\mathbf{M}$ was originally partitioned (i.e., before any noise was applied). Let $RS^{VAMO}(\mathsf{n})$ be the resulting Rand statistic, $JC^{VAMO}(\mathsf{n})$ be the Jaccard coefficient, $FM^{VAMO}(\mathsf{n})$ be the Folkes-Mallows index, and $F1^{VAMO}(\mathsf{n})$ be the F1 index that combines precision and recall (see Section 3.2).

We similarly compute these four external cluster validity indexes by first applying the majority voting-based approach described in Section 6.1.3 over $\mathbf{M}(\mathsf{n})$ to obtain a reference clustering $\mathbf{Rc}^{MV}(\mathsf{n})$, and then comparing this reference clustering to $\mathbf{C}$. Let $RS^{MV}(\mathsf{n})$ be the resulting Rand statistic, $JC^{MV}(\mathsf{n})$ be the Jaccard coefficient, $FM^{MV}(\mathsf{n})$ be the Folkes-Mallows index, and $F1^{MV}(\mathsf{n})$ be the F1 index. Now, for each value of $\mathsf{n}$ we compute the difference between the validity indexes obtained using VAMO and the ones based on the majority voting approach. For example, we compute $\Delta RS(\mathsf{n}) = RS^{VAMO}(\mathsf{n}) - RS^{MV}(\mathsf{n})$, and in a similar way we also compute $\Delta JC(\mathsf{n})$, $\Delta FM(\mathsf{n})$, and $\Delta F1(\mathsf{n})$.

### 6.1.5 Results

Figure 3 reports the absolute values of the cluster validty indexes obtained using VAMO, while Figure 4 plots the difference between the four external validity indexes produced by the comparison between VAMO's results and the majority voting approach, as explained above. In Figure 3, the $\mathtt{y}$ axis reports the absolute value of the indexes, while in Figure 4 it reports the "deltas". In both cases, the $\mathtt{x}$ axis is simply the index of the experiment round, with the noise increasing per each experiment [5]. Specifically, we use 1,320 different noise configurations (i.e., different values of the elements of the noise vector $\mathsf{n}$), with the only constraint that $p_f + p_m \leq 0.5$, i.e., at most 50% of the malware samples will be affected by some noise in their AV labels. It is also worth noting that the $\mathtt{y}$ axis for $\Delta RS$ varies in $[-0.02, 0.08]$, while all other "deltas" graphs have values on the $\mathtt{y}$ axis in $[-0.2, 0.8]$.

Figure 4(a) shows that the difference between $RS^{VAMO}$ and $RS^{MV}$ are relatively small, and $\Delta RS$ varies between $-0.02$ and 0.06. However, the three remaining validity indexes (Figure 4(b) through Figure 4(d)) clearly show that VAMO's reference clustering *agrees more closely* with the underlying true clustering $\mathbf{C}$, compared to the majority voting-based reference clustering. In fact, in all four indexes the "deltas" are positive for the vast majority of the noise combinations, meaning that the quality indexes obtained by VAMO show a better agreement with the true clustering, compared to the quality indexes obtained via majority voting.

To better analyze the effect of the noisy AV labels, Figure 5 through Figure 8 present the validity index "deltas" considering all noise vectors $\mathsf{n}$ for which: (a) at least two AVs are affected by noise, i.e., $p_1 = 0$; (b) the only type of noise affecting the labels is the "label flips", i.e., $p_m = 0$ (no missing labels, which means that all the AVs assign a malware family label to all samples); (c) the only type of noise is "missing labels", i.e., $p_f = 0$ (no label flips). As we can see, whenever the label noise (or inconsistencies) affects a majority of AVs (case (a)), or when any AV misses to detect some malware samples (case (c)), VAMO clearly outperforms the majority voting-based approach, because VAMO's reference clustering more closely agrees with the true malware clusters. While the "label flips" (i.e., case (b), which simulates the scenario in which AVs assign the incorrect malware family name) have a more negative effect on VAMO because they more heavily affect the edges (and their weights) learned through the *AV Label Graph*, VAMO performs comparably to majority voting, as shown by the very small negative "deltas".

[5]The experiment rounds are ordered according to a summary *noise level* computed as $nl = (0.6p_f + 0.4p_m) \cdot (0.1p_1 + 0.3p_2 + 0.6p_3)$.

| *l* | *clusters* | **Rand** | **Jaccard** | **Folkes-Mallows** | **F1** |
|---|---|---|---|---|---|
| 0.10 | 674 | 0.8767 | 0.2086 | 0.4494 | 0.7100 |
| 0.20 | 451 | 0.9172 | 0.5438 | 0.7308 | 0.7918 |
| 0.30 | 313 | 0.9205 | 0.5777 | 0.7482 | 0.7948 |
| **0.31** | **301** | **0.9792** | **0.8924** | **0.9434** | *0.8436* |
| 0.32 | 291 | 0.9790 | 0.8916 | 0.9430 | 0.8431 |
| 0.33 | 288 | 0.9759 | 0.8782 | 0.9357 | **0.8501** |
| 0.34 | 286 | 0.9759 | 0.8782 | 0.9357 | 0.8496 |
| 0.35 | 280 | 0.9758 | 0.8775 | 0.9353 | 0.8479 |
| 0.36 | 274 | 0.9757 | 0.8772 | 0.9352 | 0.8467 |
| 0.37 | 261 | 0.9721 | 0.8614 | 0.9265 | 0.8433 |
| 0.38 | 255 | 0.9721 | 0.8613 | 0.9265 | 0.8424 |
| 0.39 | 248 | 0.9722 | 0.8623 | 0.9270 | 0.8421 |
| 0.40 | 241 | 0.9721 | 0.8617 | 0.9268 | 0.8401 |
| 0.50 | 187 | 0.9585 | 0.8081 | 0.8971 | 0.7937 |
| 0.60 | 142 | 0.9260 | 0.7070 | 0.8366 | 0.7489 |
| 0.70 | 113 | 0.8527 | 0.5614 | 0.7354 | 0.7260 |
| 0.80 | 85 | 0.7789 | 0.4659 | 0.6656 | 0.7124 |

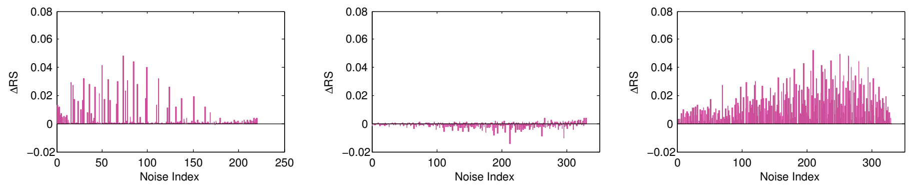
**Table 2: Application of VAMO to behavior-based malware clustering results.**

## 6.2 Real-World Application

In this Section, we discuss how VAMO can be applied in practice to assess the quality of the results produced by malware clustering systems. Specifically, we apply VAMO to the results that the behavior-based malware clustering system presented in [2] produced over a real-world malware dataset $\mathbf{M}$ containing 2,026 distinct malware samples collected in February 2009. To obtained the behavior-based clustering we proceeded as follows. We provided all malware samples in $\mathbf{M}$ to the authors of [2], who kindly agreed to analyze them and provide us a distance matrix $\mathbf{D}$ containing the pair-wise distances between the samples computed based on their system-level behavioral features. Given, $\mathbf{D}$, we applied *precise* average-linkage hierarchical clustering (this step is slightly different from [2], in which the authors applied an *approximate* hierarchical clustering algorithm), and obtained a dendrogram, which we will refer to as $\mathcal{Y}$ in the following. As usual, the dendrogram $\mathcal{Y}$ can be cut at a given height to obtain a partitioning of dataset $\mathbf{M}$ into a number of malware clusters (see discussion below).

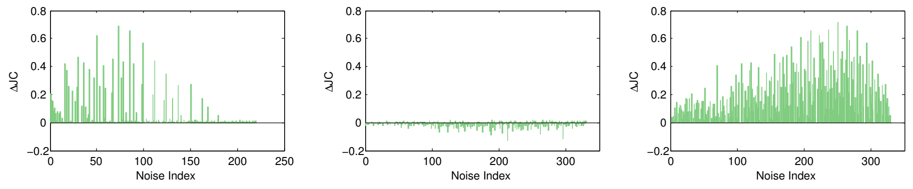
To generate VAMO's *AV Label Graph*, we used a dataset $\mathcal{A}$ consisting of 998,104 real-world distinct malware samples collected between August 2008 and August 2009. All of these 998,104 samples were scanned using four different popular AVs, in a way analogous to the malware dataset we discussed in Section 3.1, to obtain the label dataset $\mathcal{L}$. Each sample in this dataset was assigned at least one AV label. Also, $\mathcal{L}$ contained the labels for most of the 2,026 samples in $\mathbf{M}$. Specifically, $\mathcal{L}$ included at least one label for 1,985 samples in $\mathbf{M}$, while the remaining 41 samples were not represented in $\mathcal{L}$, and therefore remained *unlabeled*.

Taking the labeled dataset $\mathcal{A}$ and the labels for the samples in dataset $\mathbf{M}$ (including the placeholder "unknown" labels for the 41 samples that remained undetected) as input, we applied VAMO to produce a reference clustering, following the procedure outlined in



(a) F & M (> 1 AV) (b) Only Flips (c) Only Missing

**Figure 5: VAMO vs. Maj. Voting: Rand Statistic.**



(a) F & M (> 1 AV) (b) Only Flips (c) Only Missing

**Figure 6: VAMO vs. Maj. Voting: Jaccard Coefficient.**

Section 4 and Section 5. Then, given the dendrogram $\mathcal{Y}$ obtained from the third-party malware clustering system [2], we cut $\mathcal{Y}$ at different heights $l_1, l_2, .., l_n$, thus obtaining a sequence of different clusterings $\mathbf{C}(l_1), \mathbf{C}(l_2), .., \mathbf{C}(l_n)$. For each of these clustering results, we used VAMO to compute a set of validity indexes (see Section 5). Table 2 summarizes our results. The first column in Table 2 reports the value of the hight $l$ at which $\mathcal{Y}$ is cut, while the second column reports the related number of clusters that was obtained from $\mathbf{M}$. For example, by cutting $\mathcal{Y}$ at height $l = 0.5$, $\mathbf{M}$ is partitioned into 187 clusters. The remaining columns represent the values of five different external cluster validity indexes (Section 3.2) measured by comparing the obtained malware clusters to VAMO's reference clustering, as explained in Section 5. We varied $l \in [0, 1]$ at steps equal to 0.01 (in practice, we excluded the extreme values $l = 0$ and $l > 0.8$, because they result either into one malware per cluster or into artificially large clusters, respectively). In the interest of space, because the maximum value of the validity indexes is located between $l = 0.3$ and $l = 0.4$, we report the results at steps of 0.01 only within that range.

As we can see from Table 2, the best value of the cut $l$ is equal to 0.31, because that is the cut hight at which three out of four external validity indexes express the fact that there is maximum agreement between the behavior-based clusters and the AV labels generated by four different AV scanners. Put another way, VAMO's results indicate that the AV labels provide the best explanation of the underlying malware dataset $\mathbf{M}$ when $\mathbf{M}$ is partitioned into 301 clusters by cutting $\mathcal{Y}$ at $l = 0.31$.

It is worth noting that the F1 index is the only external validity index that is not maximum at $l = 0.31$. However, the value of 0.8436 obtained at $l = 0.31$ is quite close to the maximum value of 0.8502 reached at $l = 0.33$. This result suggests that to find the best configuration parameters for the third-party malware clustering system, it may be better to consider multiple validity indexes, rather than focusing only on analyzing precision and recall (and the related F1 index), as proposed in previous work [2, 11].

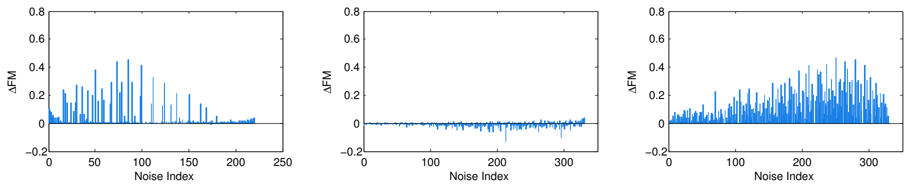
## 7. DISCUSSION

Using AV labels to build a reference clustering has some potential limitations, even though the label inconsistencies can be mitigated using VAMO. First, we need to take into account that the features used by the AVs to characterize malware samples and assign them to a given malware family may be different from the features used by a third-party malware clustering system to measure the similarity among samples. For example, AV vendors often base their malware categorization process on features extracted from reverse engineering the malware binaries. On the other hand, behavior-based malware clustering systems leverage features related to the malware's system [2] or network activities [15], for example. Naturally, different features may highlight different types of similarities in the samples. Therefore, while the AV labels clearly represent a valuable point of reference, especially in absence of a more perfect ground truth, the comparison between behavior-based malware clustering results and AV family labels should be taken with a grain of salt. A similar argument is made in [4], in which the authors outline the potential pitfalls of using labeled datasets meant for training and testing of supervised learning algorithms for evaluating the effectiveness of (unsupervised) clustering algorithms. Nonetheless, AV label-based cluster validity analysis, especially when fully automated such as in VAMO, is certainly a valuable tool that can assist malware analysts in the analysis of their malaware clustering results.

Another factor to consider is the fact that AV labels *evolve* in time. That is, a malware sample $m$ assigned by an AV to family $f_i$ at time $t_0$, may be "renamed" by the same AV as belonging to a different family $f_j$ at a future time $t_1 > t_0$. This is due to the fact that AV signatures are sometimes refined by the AV vendors to reduce possible false positives and more specifically characterize the malware samples (e.g., by assigning a sample previously labeled as "generic" to a more specific malware family). To take this into account, the historic archive of malware labels used by VAMO should be kept updated. This may be done by either periodically rescanning the malware dataset, or by querying online services such as `virustotal.com`.

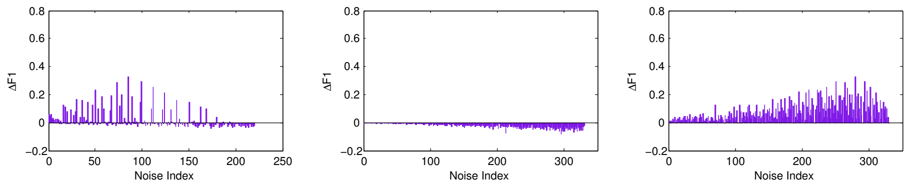
## 8. CONCLUSION

In this paper, we presented a novel system, called VAMO, that provides a fully automated assessment of the quality of malware clustering results. Previous studies propose to evaluate malware clustering results by leveraging the labels assigned to the malware samples by multiple AVs. However, they require a manual mapping



(a) F & M (> 1 AV) (b) Only Flips (c) Only Missing

**Figure 7: VAMO vs. Maj. Voting: Folkes-Mallows.**



(a) F & M (> 1 AV) (b) Only Flips (c) Only Missing

**Figure 8: VAMO vs. Maj. Voting: F1 Index.**

between labels assigned by different AV vendors, and are limited to selecting a *reference sub-set* of samples for which an agreement regarding their labels can be reached across a majority of AVs.

Unlike previous work, VAMO does not require a manual mapping between malware family labels output by different AV scanners. Furthermore, VAMO does not discard malware samples for which a majority voting-based consensus cannot be reached. Instead, VAMO explicitly deals with the inconsistencies typical of multiple AV labels to build a more representative reference set. Our evaluation, which includes extensive experiments in a controlled setting and a real-world application, show that VAMO performs better then majority voting-based approaches, and provides a way for malware analysts to automatically assess the quality of their malware clustering results.

## Acknowledgments

This material is based in part upon work supported by the National Science Foundation under Grant No. CNS-1149051. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## References

[1] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario. Automated classification and analysis of internet malware. In *Recent Advances in Intrusion Detection*, 2007.

[2] U. Bayer, P. Milani Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, behavior-based malware clustering. In *Network and Distributed System Security Symposium*, 2009.

[3] M. Christodorescu, S. Jha, and C. Kruegel. Mining specifications of malicious behavior. In *ACM SIGSOFT symposium on the foundations of software engineering*, ESEC-FSE '07, 2007.

[4] I. Färber, S. Günnemann, H. Kriegel, P. Kröger, E. Müller, E. Schubert, T. Seidl, and A. Zimek. On using class-labels in evaluation of clusterings. In *MultiClust: 1st International Workshop on Discovering, Summarizing and Using Multiple Clusterings Held in Conjunction with KDD*, 2010.

[5] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

[6] F. Guo, P. Ferrie, and T. Chiueh. A study of the packer problem and its solutions. In *Recent Advances in Intrusion Detection*, 2008.

[7] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *J. Intell. Inf. Syst.*, 17(2-3):107–145, 2001.

[8] X. Hu, T.-c. Chiueh, and K. G. Shin. Large-scale malware indexing using function-call graphs. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, 2009.

[9] A. K. Jain and R. C. Dubes. *Algorithms for clustering data.* Prentice-Hall, Inc., 1988.

[10] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[11] J. Jang, D. Brumley, and S. Venkataraman. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, 2011.

[12] P. Li, L. Liu, D. Gao, and M. K. Reiter. On challenges in evaluating malware clustering. In *Proceedings of the 13th international conference on Recent advances in intrusion detection*, RAID'10, 2010.

[13] F. Maggi, A. Bellini, G. Salvaneschi, and S. Zanero. Finding non-trivial malware naming inconsistencies. In *International Conference on Information Systems Security*, ICISS'11, 2011.

[14] M. Meilă. Comparing clusterings—an information based distance. *J. Multivar. Anal.*, 98(5):873–895, May 2007.

[15] R. Perdisci, W. Lee, and N. Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'10, 2010.

[16] D. Pfitzner, R. Leibbrandt, and D. Powers. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowl. Inf. Syst.*, 19(3):361–394, May 2009.

[17] E. RendŮn, I. Abundez, A. Arizmendi, and E. M. Quiroz. Internal versus external cluster validation indexes. *universitypressorguk*, 5(1), 2011.

[18] K. Rieck, P. Trinius, C. Willems, and T. Holz. Automatic analysis of malware behavior using machine learning. *J. Comput. Secur.*, 19(4):639–668, Dec. 2011.

[19] Symantec. Symantec internet security threat report, trends for 2010, April 2011.