On Challenges in Evaluating Malware Clustering

Peng Li[1], Limin Liu[2], Debin Gao[3], and Michael K. Reiter[1]

[1] Department of Computer Science, University of North Carolina, Chapel Hill, NC, USA
[2] State Key Lab of Information Security, Graduate School of Chinese Academy of Sciences
[3] School of Information Systems, Singapore Management University, Singapore

Abstract. Malware clustering and classification are important tools that enable analysts to prioritize their malware analysis efforts. The recent emergence of fully automated methods for malware clustering and classification that report high accuracy suggests that this problem may largely be solved. In this paper, we report the results of our attempt to confirm our conjecture that the method of selecting ground-truth data in prior evaluations biases their results toward high accuracy. To examine this conjecture, we apply clustering algorithms from a different domain (plagiarism detection), first to the dataset used in a prior work's evaluation and then to a wholly new malware dataset, to see if clustering algorithms developed without attention to subtleties of malware obfuscation are nevertheless successful. While these studies provide conflicting signals as to the correctness of our conjecture, our investigation of possible reasons uncovers, we believe, a cautionary note regarding the *significance* of highly accurate clustering results, as can be impacted by testing on a dataset with a biased cluster-size distribution.

Keywords: malware clustering and classification, plagiarism detection.

1 Introduction

The dramatic growth of the number of malware variants has motivated methods to classify and group them, enabling analysts to focus on the truly new ones. The need for such classification and pruning of the space of all malware variants is underlined by, e.g., the Bagle/Beagle malware, for which roughly 30,000 distinct variants were observed between January 9 and March 6, 2007 [8]. While initial attempts at malware classification were performed manually, in recent years numerous *automated* methods have been developed to perform malware classification (e.g., [11,6,5,16,9,13,15]). Some of these malware classifiers have claimed very good accuracy in classifying malware, leading perhaps to the conclusion that malware classification is more-or-less solved.

In this paper, we show that this may not be the case, and that evaluating automated malware classifiers poses substantial challenges that we believe require renewed attention from the research community. A central challenge is that with the dearth of a well-defined notion of when two malware instances are the "same" or "different", it is difficult to obtain ground truth to which to compare the results of a proposed classifier. Indeed, even manually encoded rules to classify malware seems not to be enough — a previous study [6] found that a majority of six commercial anti-virus scanners concurred on the classification of 14,212 malware instances in only 2,658 cases. However, in the absence of better alternatives for determining ground truth, such instances and their corresponding classifications are increasingly used to evaluate automated methods of malware clustering. For example, a state-of-the-art malware clustering algorithm due to Bayer et al. [6] achieved excellent results using these 2,658 malware instances as ground truth; i.e., the tool obtained results that largely agreed with the clustering of these 2,658 malware instances by the six anti-virus tools.

The starting point of the present paper is the possibility, we conjectured, that one factor contributing to these strong results might be that these 2,658 instances are simply easy to classify, by any of a variety of techniques. We report on our efforts to examine this possibility, first by repeating the clustering of these instances using algorithms from an ostensibly different domain, namely plagiarism detectors that employ dynamic analysis. Intuitively, since plagiarism detectors are developed without attention to the specifics of malware obfuscation, highly accurate clustering results by these tools might suggest that this method of selecting ground-truth data biases the data toward easy-to-classify instances. We describe the results of this analysis, which indicate that plagiarism detectors have nearly the same success in clustering these malware instances, thus providing tentative support for this conjecture.

To more thoroughly examine this possibility, we then attempted to repeat the evaluation methodology of Bayer et al. on a new set of malware instances. By drawing from a database of malware instances, we assembled a set for which four anti-virus tools consistently labeled each member. We detail this study and report on its results that, much to our surprise, find that neither the Bayer et al. technique nor the plagiarism detectors we employed were particularly accurate in clustering these instances. Due to certain caveats of this evaluation that we will discuss, this evaluation is materially different from that for the previous dataset, causing us to be somewhat tentative in the conclusions we draw from it. Nevertheless, these results temper the confidence with which we caution that the selection of ground-truth data based on the concurrence of multiple anti-virus tools biases the data toward easy-to-classify instances.

But this leaves the intriguing question: Why the different results on the two datasets? We complete our paper with an analysis of a factor that, we believe, contributes to (though does not entirely explain) this discrepancy, and that we believe offers a cautionary note for the evaluation of malware clustering results. This factor is the makeup of the ground-truth dataset, in terms of the distribution of the sizes of the malware families it contains. We observe that the original dataset, on which the algorithms we consider perform well, is dominated by two large families, but the second dataset is more evenly distributed among many families. We show that this factor alone biases the measures used in comparing the malware clustering output to the dataset families, specifically precision and recall, in that it increases the likelihood of good precision and recall numbers occurring by chance. As such, the biased cluster-size distribution in the original dataset erodes the *significance* (c.f., [22, Section 8.5.8]) of the high precision and recall reported by Bayer et al. [6]. This observation, we believe, identifies an important factor for which to control when measuring the effectiveness of a malware clustering technique.

While we focus on a single malware classifier for our analysis [6], we do so because very good accuracy has been reported for this algorithm and because the authors of that technique were very helpful in enabling us to compare with their technique. We hasten to emphasize, moreover, that our comparisons to plagiarism detectors are not intended to suggest that plagiarism detectors are the equal of this technique. For one, we believe the technique of Bayer et al. is far more scalable than any of the plagiarism detectors that we consider here, an important consideration when clustering potentially tens of thousands of malware instances. In addition, the similar accuracy of the technique of Bayer et al. to the plagiarism detectors does not rule out the possibility that the plagiarism detectors are more easily evaded (in the sense of diminishing clustering accuracy); rather, it simply indicates that malware today does not seem to do so. We stress that the issues we identify are not a criticism of the Bayer et al. technique, but rather are issues worth considering for any evaluation of malware clustering and classification.

To summarize, the contributions of this paper are as follows. First, we explore the possibility that existing approaches to obtaining ground-truth data for malware clustering evaluation biases results by isolating those instances that are simple to cluster or classify. In the end, we believe our study is inconclusive on this topic, but that reporting our experiences will nevertheless raise awareness of this possibility and will underline the importance of finding methods to validate the ground-truth data employed in this domain. Second, we highlight the importance of the *significance* of positive clustering results when reporting them. This has implications for the datasets used to evaluate malware clustering algorithms, in that it requires that datasets exhibiting a biased cluster-size distribution not be used as the sole vehicle for evaluating a technique.

2 Classification and Clustering of Malware

To hinder static analysis of binaries, the majority of current malware makes use of obfuscation techniques, notably binary packers. As such, dynamic analysis of such malware is often far more effective than static analysis. Monitoring the behavior of the binary during its execution enables collecting a profile of the operations that the binary performs and offers potentially greater insight into the code itself if obfuscation is removed (e.g., the binary is unpacked) in the course of running it. While this technique has its limitations — e.g., it may be difficult to induce certain behaviors of the malware, some of which may require certain environmental conditions to occur [10,14,19,20] — it nevertheless is more effective than purely static approaches. For this reason, dynamic analysis of malware has received much attention in the research community. Analysis systems such as CWSandbox [25], Anubis [7], BitBlaze [18], Norman [2] and ThreatExpert [1] execute malware samples within an instrumented environment and monitor their behaviors for analysis and development of defense mechanisms.

A common application for dynamic analysis of malware is to group malware instances, so as to more easily identify the emergence of new strains of malware, for example. Such grouping is often performed using machine learning, either by *clustering* (e.g., [6,17,15]) or by *classification* (e.g., [13,5,16,11]), which are unsupervised and supervised techniques, respectively.

Of primary interest in this paper are the methodologies that these works employ to evaluate the results of learning, and specifically the measures of quality for the clustering or classification results. Let M denote a collection of m malware instances to be clustered, or the “test data” in the case of classification. Let $\mathcal{C} = \{C_i\}_{1 \le i \le c}$ and

$\mathcal{D} = \{D_i\}_{1 \le i \le d}$ be two partitions of M, and let $f : \{1 \ldots c\} \to \{1 \ldots d\}$ and $g : \{1 \ldots d\} \to \{1 \ldots c\}$ be functions. Many prior techniques evaluated their results using two measures:

$$\mathsf{prec}(\mathcal{C}, \mathcal{D}) = \frac{1}{m} \sum_{i=1}^{c} |C_i \cap D_{f(i)}|$$

$$\mathsf{recall}(\mathcal{C}, \mathcal{D}) = \frac{1}{m} \sum_{i=1}^{d} |C_{g(i)} \cap D_i|$$

where $\mathcal{C}$ is the set of clusters resulting from the technique being evaluated and $\mathcal{D}$ is the clustering that represents the "right answer".

More specifically, in the case of classification, C_i is all test instances classified as class i, and D_i is all test instances that are "actually" of class i. As such, in the case of classification, $c = d$ and f and g are the identity functions. As a result $\mathsf{prec}(\mathcal{C}, \mathcal{D}) = \mathsf{recall}(\mathcal{C}, \mathcal{D})$, and this measure is often simply referred to as *accuracy*. This is the measure used by Rieck et al. [16] to evaluate their malware classifier, and Lee et al. [13] similarly uses *error rate*, or one minus the accuracy.

In the clustering case, there is no explicit label to define the cluster in $\mathcal{D}$ that corresponds to a specific cluster in $\mathcal{C}$, and so one approach is to define

$$f(i) = \arg\max_{i'} |C_i \cap D_{i'}|$$

$$g(i) = \arg\max_{i'} |C_{i'} \cap D_i|$$

In this case, f and g will not generally be the identity function (or even bijections), and so precision and recall are different. This approach is used by Rieck et al. [17] and Bayer et al. [6] in evaluating their clustering techniques. In this case, when it is desirable to reduce these two measures into one, a common approach (e.g., [17]) is to use the F-measure:

$$\mathsf{F\text{-}measure}(\mathcal{C}, \mathcal{D}) = \frac{2 \cdot \mathsf{prec}(\mathcal{C}, \mathcal{D}) \cdot \mathsf{recall}(\mathcal{C}, \mathcal{D})}{\mathsf{prec}(\mathcal{C}, \mathcal{D}) + \mathsf{recall}(\mathcal{C}, \mathcal{D})}$$

This background is sufficient to highlight the issues on which we focus in the paper:

Production of $\mathcal{D}$: A central question in the measurement of precision and recall is how the reference clustering $\mathcal{D}$ is determined. A common practice is to use an existing anti-virus tool to label the malware instances M (e.g., [16,13,11]), the presumption being that anti-virus tools embody hand-coded rules to label malware instances and so are a good source of "manually verified" ground truth. Unfortunately, existing evidence suggests otherwise, in that it has been shown that anti-virus engines often disagree on their labeling (and clustering) of malware instances [5]. To compensate for this, another practice has been to restrict attention to malware instances M on which multiple anti-virus tools agree (e.g., [6]). Aside from substantially reducing the number of instances, we conjecture that this practice might contribute to more favorable evaluations of malware classifiers, essentially by limiting evaluations to easy-to-cluster instances. To demonstrate this possibility, in Section 3 we consider malware instances selected in this way and show that they can be classified by plagiarism detectors (designed without attention to the subtleties of malware obfuscation) with precision and recall comparable to that offered by a state-of-the-art malware clustering tool.

Distribution of cluster sizes in $\mathcal{C}$ and $\mathcal{D}$: In order to maximize both precision and recall (and hence the F-measure), it is necessary for $\mathcal{C}$ and $\mathcal{D}$ to exhibit similar cluster-size distributions; i.e., if one of them is highly biased (i.e., has few, large clusters) and the other is more evenly distributed, then one of precision or recall will suffer. Even when they exhibit similar cluster-size distributions, however, the degree to which that distribution is biased has an effect on the *significance* (e.g., [22, Section 8.5.8]) that one can ascribe to high values of these measures. Informally, the significance of a given precision or recall is related to the probability that this value could have occurred by random chance; the higher the probability, the less the significance. We will explore the effect of cluster-size distribution on significance, and specifically the impact of cluster-size distribution on the sensitivity of the F-measure to perturbations in the distance matrix from which the clustering $\mathcal{C}$ is derived. We will see that all other factors held constant, good precision and recall when the reference clusters in $\mathcal{D}$ are of similar size is more significant than if the cluster sizes are biased. That is, small perturbations in the distance matrix yielding $\mathcal{C}$ tends to decay precision and recall more than if $\mathcal{D}$ and $\mathcal{C}$ are highly biased.

We will demonstrate this phenomenon using the malware clustering results obtained from the state-of-the-art malware clustering tool due to Bayer et al., which obtains very different results on two malware datasets, one with a highly biased clustering and one with a more even clustering. While this is not the only source of variation in the datasets, and so the different results cannot be attributed solely to differences in cluster size distributions, we believe that the cluster size distribution is a factor that must be taken into account when reporting malware clustering results.

3 A Potential Hazard of Anti-virus Voting

As discussed in Section 2, a common practice to produce the ground-truth reference clustering $\mathcal{D}$ for evaluating malware clustering algorithms is to use existing anti-virus tools to label the malware instances and to restrict attention to malware instances M on which multiple anti-virus tools agree. The starting point of our study is one such ground-truth dataset, here denoted BCHKK-data, that was used by Bayer et al. for evaluating their malware clustering technique [6]. Using this dataset, their algorithm, here denoted BCHKK-algo, yielded a very good precision and recall (of 0.984 and 0.930, respectively). BCHKK-data consists of $2,658$ malware instances, which is a subset of $14,212$ malware instances contributed between October 27, 2007 and January 31, 2008 by a number of security organizations and individuals, spanning a wide range of sources (such as web infections, honeypots, botnet monitoring, and other malware analysis services). Bayer et al. ran six different anti-virus programs on these $14,212$ instances, and a subset of $2,658$ instances on which results from the majority of these anti-virus programs agree were chosen to form BCHKK-data for evaluation of their clustering technique BCHKK-algo. Bayer et al. explained that such a subset was chosen because they are the instances on which ground truth can be obtained (due to agreement by a majority of the anti-virus programs they used).

This seems to be a natural way to pick M for evaluation, as they are the only ones for which the ground-truth clustering (i.e., $\mathcal{D}$) could be obtained with good confidence. However, this also raises the possibility that the instances on which multiple anti-virus tools agree are just the malware instances that are relatively easy to cluster, while the difficult-to-cluster instances are filtered out of M. If this were the case, then this could contribute to the high precision and recall observed for the BCHKK-data dataset, in particular.

Unfortunately, we are unaware of any accepted methodology for testing this possibility directly. So, we instead turn to another class of clustering tools derived without attention to malware clustering, in order to see if they are able to cluster the malware instances in BCHKK-data equally well. Specifically, we apply plagiarism detectors to the BCHKK-data to see if they can obtain good precision and recall.

3.1 Plagiarism Detectors

Plagiarism detection is the process of detecting that portions within a work are not original to the author of that work. One of the most common uses of software plagiarism detection is to detect plagiarism in student submissions in programming classes (e.g., Moss [4], Plaque [24], and YAP [26]). Software plagiarism detection and malware clustering are related to one another in that they both attempt to detect some degree of similarity in software programs among a large number of instances. However, due to the uniqueness of malware samples compared to software programs in general (e.g., in using privileged system resources) and due to the degree of obfuscation typically applied to malware instances, we did not expect plagiarism detectors to produce good results when clustering malware samples.

Here we focus on three plagiarism detectors that monitor dynamic executions of a program. We do not include those applying static analysis techniques as they are obviously not suitable for analyzing (potentially packed) malware instances.

– APISeq: This detector, proposed by Tamada et al. [21], computes the similarity of the sequences of API calls produced by two programs to determine if one is plagiarized from the other. Similarity is measured by using string matching tools such as diff and CCFinder [12].
– SYS3Gram: In this detector, due to Wang et al. [23], short sequences (specifically, triples) of system calls are used as "birthmarks" of programs. Similarity is measured as the Jacaard similarity of the birthmarks of the programs being compared, i.e., as the ratio between the sizes of two sets: (i) the intersection of the birthmarks from the two programs, and (ii) the union of the birthmarks from the two programs.
– API3Gram: We use the same idea as in SYS3Gram and apply it to API calls to obtain this plagiarism detector.

We emphasize that the features on which these algorithms detect plagiarism are distinct from those employed by BCHKK-algo. Generally, the features adopted in BCHKK-algo are the operating system objects accessed by a malware instance, the operations that were performed on the objects, and data flows between accesses to objects. In contrast, the features utilized by the plagiarism detectors we adopted here are system/API call sequences (without specified argument values).

3.2 Results

We implemented these three plagiarism detectors by following the descriptions in the corresponding papers and then applied the detectors to BCHKK-data (instances used by Bayer et al. [6] on which multiple anti-virus tools agree). More specifically, each detection technique produced a distance matrix; we then used single-linkage hierarchical clustering, as is used by BCHKK-algo, to build a clustering $\mathcal{C}$, stopping the hierarchical clustering at the point that maximizes the p-value of a chi-squared test between the distribution of sizes of the clusters in $\mathcal{C}$ and the cluster-size distribution that BCHKK-algo induced on BCHKK-data.[1] We then evaluated the resulting clustering $\mathcal{C}$ by calculating the precision and recall with respect to a reference clustering $\mathcal{D}$ that is one of

- AV: clustering produced by multiple anti-virus tools, i.e., $\mathcal{D}$ in the evaluation clustering ("ground truth") in Bayer et al.'s paper [6];
- BCHKK-algo: clustering produced by the technique of Bayer et al., i.e., $\mathcal{C}$ in the evaluation in Bayer et al.'s paper [6].

To make a fair comparison, the three plagiarism detectors and BCHKK-algo obtain system information (e.g., API call, system call, and system object information) from the same dynamic traces produced by Anubis [7]. Results of the precision and recall are shown in Table 1.

Table 1. Applying plagiarism detectors and malware clustering on BCHKK-data

$\mathcal{C}$	$\mathcal{D}$	prec$(\mathcal{C},\mathcal{D})$	recall$(\mathcal{C},\mathcal{D})$	F-measure$(\mathcal{C},\mathcal{D})$
BCHKK-algo	AV	0.984	0.930	0.956
APISeq		0.965	0.922	0.943
API3Gram		0.978	0.927	0.952
SYS3Gram		0.982	0.938	0.960
APISeq	BCHKK-algo	0.988	0.939	0.963
API3Gram		0.989	0.941	0.964
SYS3Gram		0.988	0.938	0.963

One set of experiments, shown where $\mathcal{D}$ is set to the clustering results of BCHKK-algo in Table 1, compares these plagiarism detectors with BCHKK-algo directly. The high (especially) precisions and recalls show that the clusterings produced by these plagiarism detectors are very similar to that produced by BCHKK-algo. A second set of

[1] More specifically, this chi-squared test was performed between the cluster-size distribution of $\mathcal{C}$ and a parameterized distribution that best fit the cluster-size distribution that BCHKK-algo induced on BCHKK-data. The parameterized distribution was Weibull with shape parameter $k = 0.4492$ and scale parameter $\lambda = 5.1084$ (p-value $= 0.8763$).

experiments, shown where $\mathcal{D}$ is set to AV, compares the precisions and recalls of all four techniques to the "ground truth" clustering of BCHKK-data. It is perhaps surprising that SYS3Gram performed as well as it did, since a system-call-based malware clustering algorithm [13] tested by Bayer et al. performed relatively poorly; the difference may arise because the tested clustering algorithm employs system-call arguments, whereas SYS3Gram does not (and so is immune to their manipulation). That issue aside, we believe that the high precisions and recalls reported in Table 1 provide support for the conjecture that the malware instances in the BCHKK-data dataset are likely relatively simple ones to cluster, since plagiarism detectors, which are designed without attention to the specific challenges presented by malware, also perform very well on them.

4 Replicating Our Analysis on a New Dataset

Emboldened by the results in Section 3, we decided to attempt to replicate the analysis of the previous section on a new dataset. Our goal was to see if another analysis would also support the notion that selecting malware instances for which ground-truth evidence is inferred by "voting" by anti-virus tools yields a ground-truth dataset that all the tools we considered (BCHKK-algo and plagiarism detectors alike) could cluster well.

4.1 The New Dataset and BCHKK-algo Clustering

To obtain another dataset, we randomly chose $5,121$ instances from the collection of malware instances from VX heavens [3]. We selected the number of instances to be roughly twice the $2,568$ instances in BCHKK-data. We submitted this set of instances to Bayer et al., who kindly processed these instances using Anubis and then applied BCHKK-algo to the resulting execution traces and returned to us the corresponding distance matrix. This distance matrix covered $4,234$ of the $5,121$ samples; Anubis had presumably failed to produce meaningful execution traces for the remainder.

In order to apply the plagiarism detectors implemented in Section 3 to this data, we needed to obtain the information that each of those techniques requires, specifically the sequences of system calls and API calls for each malware instance. As mentioned in Section 3, we obtained this information for the BCHKK-data dataset via Anubis; more specifically, it was already available in the BCHKK-data data files that those authors provided to us. After submitting this new dataset to the Anubis web interface, however, we realized that this information is not kept in the Anubis output by default. Given that obtaining it would then require additional imposition on the Anubis operators to customize its output and then re-submitting the dataset to obtain analysis results (a lengthy process), we decided to seek out a method of extracting this information locally. For this purpose, we turned to an alternative tool that we could employ locally to generate API call traces from the malware instances, namely CWSandbox [25]. CWSandbox successfully processed (generated non-empty API call traces) for $4,468$ of the $5,121$ samples, including $3,841$ of the $4,234$ for which we had results for the BCHKK-algo algorithm.

We then scanned each of these $3,841$ instances with four anti-virus programs (Activescan 2.0, Nod32 update 4956, Avira 7.10.06.140 and Kaspersky 6.0.2.690). Analyzing the results from these anti-virus programs, we finally obtained $1,114$ malware instances for which the four anti-virus programs reported the same family for each; we denote these $1,114$ as VXH-data in the remainder of this paper. More specifically, each instance is given a label (e.g, Win32.Acidoor.b, BDS/Acidoor.B) when scanned by an anti-virus program. The family name is the generalized label extracted from the instance label based on the portion that is intended to be human-readable (e.g., the labels listed would be in the "Acidoor" family). We defined a reference clustering $\mathcal{D}$ for this dataset so that two instances are in the same cluster $D \in \mathcal{D}$ if and only if all of the four anti-virus programs concurred that these instances are in the same family.[2] Our method for assembling the reference clustering for VXH-data is similar to that used to obtain the reference clustering of BCHKK-data [6], but is more conservative.[3]

We obtained the BCHKK-algo clustering of VXH-data by applying single linkage hierarchical clustering to the subset of the distance matrix provided by Bayer et al. corresponding to these instances. In this clustering step, we used the same parameters as in the original paper [6]. To ensure a fair comparison with other alternatives, we confirmed that this clustering offered the best F-measure value relative to the reference VXH-data clustering based on the anti-virus classifications, in comparison to stopping the hierarchical clustering at selected points sooner or later.

4.2 Validation on BCHKK-Data

As discussed above, we resorted to a new tool, CWSandbox (vs. Anubis), to extract API call sequences for VXH-data. In order to gain confidence that this change would not greatly influence our results, we first performed a validation test, specifically to see whether our plagiarism detectors would perform comparably on the BCHKK-data dataset when processed using CWSandbox. In the validation test, we submitted BCHKK-data to CWSandbox to obtain execution traces for each instance. Out of the $2,658$ instances in BCHKK-data, CWSandbox successfully produced traces for $2,099$ of them. Comparing the API3Gram and APISeq clusterings on these $2,099$ samples, first with reference clustering AV and then with the clustering produced using BCHKK-algo (which, again, uses Anubis) as reference, yields the results in Table 2. Note that due to the elimination of some instances, the reference clusterings have fewer clusters than before (e.g., AV now has 68 families instead of 84 originally). Also note that SYS3Gram results are missing in Table 2 since CWSandbox does not provide system call information. However, high F-measure values for the other comparisons suggest that our plagiarism detectors still work reasonably well using CWSandbox outputs.

[2] The VX heavens labels for malware instances are the same as Kaspersky's, suggesting this is the anti-virus engine they used to label.

[3] The method by which Bayer et al. selected BCHKK-data and produced a reference clustering for it was only sketched in their paper [6], but their clarifications enabled us to perform a comparable data selection and reference clustering process, starting from the $3,841$ instances from VX heavens successfully processed by both CWSandbox and the BCHKK-algo algorithm (based on Anubis). This process retained a superset of the 1,114 instances in VXH-data and produced a clustering of which every cluster of VXH-data is a subset of a unique cluster.

Table 2. Applying plagiarism detectors and malware clustering on BCHKK-data. API3Gram and APISeq are based on CWSandbox traces.

$\mathcal{C}$	$\mathcal{D}$	prec$(\mathcal{C},\mathcal{D})$	recall$(\mathcal{C},\mathcal{D})$	F-measure$(\mathcal{C},\mathcal{D})$
API3Gram	AV	0.948	0.918	0.933
APISeq		0.958	0.934	0.946
API3Gram	BCHKK-algo	0.921	0.931	0.926
APISeq		0.937	0.939	0.938

4.3 Results on VXH-Data

In Section 4.1 we described how we assembled the VXH-data dataset and applied BCHKK-algo and the anti-virus tools to cluster it. We now compare the results of the four clustering techniques run on VXH-data: AV from the anti-virus tools, API3Gram (based on CWSandbox), APISeq (based on CWSandbox) and BCHKK-algo (based on Anubis). Results are shown in Table 3. These results again show that the plagiarism detectors produce comparable clustering results to BCHKK-algo when AV is the reference, offering generally greater precision, worse recall, and a similar F-measure.

Table 3. Applying plagiarism detectors and malware clustering on VXH-data

$\mathcal{C}$	$\mathcal{D}$	prec$(\mathcal{C},\mathcal{D})$	recall$(\mathcal{C},\mathcal{D})$	F-measure$(\mathcal{C},\mathcal{D})$
BCHKK-algo	AV	0.604	0.659	0.630
API3Gram		0.788	0.502	0.613
APISeq		0.704	0.536	0.609
API3Gram	BCHKK-algo	0.790	0.826	0.808
APISeq		0.770	0.798	0.784

Surprisingly, however, these measures indicate that both BCHKK-algo and our plagiarism detectors perform more poorly on VXH-data than they did on BCHKK-data. On the face of it, the results in Table 3 do not support the conjecture of Section 3, i.e., that determining a reference clustering of malware instances based on the concurrence of anti-virus engines might bias the reference clustering toward easy-to-cluster instances. After all, were this the case, we would think that *some* method (if not all methods) would do well when AV is used as the reference clustering. Instead, it may simply be the case that the plagiarism detectors and malware clustering tools leverage features for clustering that are more prevalent in BCHKK-data than in VXH-data. In that case, one might thus conclude that these features are not sufficiently reliable for use across a broad range of malware.

Of course, the results of this section must be taken as a bit more speculative, owing to the different systems (CWSandbox and Anubis) from which the malware traces were gathered before being consumed by the clustering techniques we consider. It is true that there is substantial variability in the length and composition of the API sequences gathered by the different tools, in some cases. For example, Figure 1 shows the CDFs of the

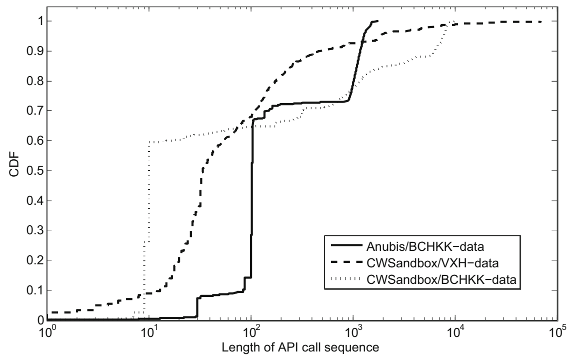

Fig. 1. Lengths of API call sequences extracted from BCHKK-data or VXH-data datasets using CWSandbox or Anubis. Note that x-axis is log-scale.

API call sequence lengths elicited by the different tools. As can be seen in Figure 1, no tool was uniformly better than the other in extracting long API call sequences, though it is apparent that the sequences they induced are very different in length.

Another viewpoint is given in Figure 2, which shows the fraction of malware instances
in each dataset in which certain activities are present. While some noteworthy differences exist, particularly in behaviors related to network activity, it is evident that both tools elicited a range of activities from large portions of the malware datasets. We suspect that some of the differences in frequencies of network activities (particularly "send data" and "receive data") result from the dearth of other malware instances with which to communicate at the time the malware was run in CWSandbox. Again, and despite these differences, our validation tests reported in Table 2 suggest that the sequences induced by each tool are similarly effective in supporting clustering of BCHKK-data.

Activity	Searched Strings	BCHKK-data (Anubis)	BCHKK-data (CWSandbox)	VXH-data (CWSandbox)
create new process	"`CreateProcess`"	100%	87.40%	70.80%
open reg key	"`RegOpenKey`"	100%	95.00%	92.90%
query reg value	"`RegQueryValue`"	100%	94.80%	89.00%
create reg key	"`RegCreateKey`"	98.70%	98.20%	94.20%
set reg value	"`RegSetValue`"	98.30%	97.10%	80.40%
create file	"`CreateFile`"	100%	98.10%	80.60%
send ICMP packet	"`IcmpSendEcho`"	82.10%	82.60%	0.71%
try to connect	"`connect`", "`WSASocket`"	85.10%	89.80%	34.70%
found no host	"`WSAHOST_NOT_FOUND`"	N/A	72.30%	9.06%
send data	"`AFD_SEND`", "`socket_send`"	83.10%	1.50%	14.40%
receive data	"`AFD_RECV`", "`socket_recv`"	83.20%	1.40%	14.90%

Fig. 2. Percentage of malware instances in which listed behavior is observed

The evidence above suggests to us that a different reason for the relatively poor accuracy of BCHKK-algo and our plagiarism detectors on VXH-data is at work. One possible contributing factor is that BCHKK-data samples within the same reference cluster tended to produce API-call sequences of more uniform length than did VXH-data samples in the same reference cluster. For example, the relative standard deviation of the API sequence lengths per cluster in BCHKK-data, averaged over all clusters, is 23.5% and 6.9% for traces produced by Anubis and CWSandbox, respectively, while this number is 130.5% for CWSandbox traces of VXH-data. However, in the following section we focus our attention on another explanation for the poorer clustering performance on VXH-data versus BCHKK-data, and that we believe is more generally instructive.

5 Effects of Cluster-Size Distribution

In seeking to understand the discrepancy between the precision and recall of the BCHKK-algo (and plagiarism-detection) clustering on the BCHKK-data (Section 3) and VXH-data datasets (Section 4), one attribute of these datasets that stood out to us is the distribution of cluster sizes in each. Specifically, the reference clustering for the BCHKK-data is highly biased, in that it contains two large clusters comprising 48.5% and 27% of the malware instances, respectively, and remaining clusters of size at most 6.7%. In contrast, the VXH-data reference dataset is more evenly distributed; the largest cluster in that dataset comprises only 14% of the instances. Figure 3 shows the cluster size distribution of the reference clustering of each dataset; note that the x-axis is log-scale.

The reason that cluster size distribution matters can be seen from an example of clustering 8 points in one of two extreme ways. If when clustering these 8 points, the reference clustering $\mathcal{D}$ comprises two clusters, one of size 7 and one of size 1, then *any* other clustering $\mathcal{C}$ of these 8 points into two clusters of size 7 and 1 is guaranteed to yield $\mathsf{prec}(\mathcal{C}, \mathcal{D})$ and $\mathsf{recall}(\mathcal{C}, \mathcal{D})$ of at least 7/8. If, on the other hand, $\mathcal{D}$ comprises two clusters of size 4 each, then another clustering $\mathcal{C}$ could yield $\mathsf{prec}(\mathcal{C}, \mathcal{D})$ and $\mathsf{recall}(\mathcal{C}, \mathcal{D})$ as low as 4/8, and in fact $\binom{4}{2}\binom{4}{2}/\binom{8}{4} = 36/70$ of such clusterings do so. In this sense,

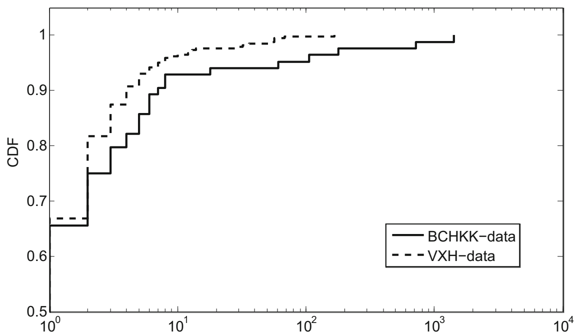

Fig. 3. Reference cluster-size distribution of BCHKK-data and VXH-data. Note that x-axis is log-scale.

it is considerably "harder" to produce a clustering yielding good precision and recall in the latter case, and a good precision and recall in the latter case is thus much more *significant* than in the former.

While providing insight, this combinatorial argument is too simplistic to illustrate the effect that cluster size distribution plays in the BCHKK-algo clustering of the VXH-data and BCHKK-data datasets. A more direct, but still coarse, indication of this effect can be seen by downsampling the large clusters in the BCHKK-data dataset. Specifically, we randomly removed malware instances from the two largest families in the BCHKK-data reference clustering until they were each of size 200. After re-clustering the remaining malware instances using BCHKK-algo with the same parameters, the resulting F-measure averaged over 10 downsampling runs was only 0.815 (versus 0.956 before downsampling).

An alternative and more refined view of the effects of significance to the clustering results of BCHKK-algo for the VXH-data and BCHKK-data datasets can be seen by illustrating the resilience of the clustering results to perturbations in the underlying distance matrix. The heart of the BCHKK-algo clustering technique is the distance measure that it develops, which is tuned to measure the activities of malware. As such, one strategy in examining the potential for bias due to cluster-size distribution is to introduce perturbations into the original BCHKK-algo distance matrices for the VXH-data and BCHKK-data up to some limit, re-cluster the resulting distance matrix into the same cluster-size distribution, and evaluate the rate at which the precision and recall drop. Intuitively, if the precision and recall drop more quickly for the VXH-data than for the BCHKK-data, then this supports the idea that minor errors in the BCHKK-algo distance are more amplified (in terms of the effects on precision and recall) when the clusters are distributed as in the VXH-data than when they are distributed as in the BCHKK-data dataset. By the contrapositive, this will show that a high precision and recall in the VXH-data case is more significant.

In attempting to perform this analysis, however, some difficulties arise.

- The BCHKK-algo distance matrices for the VXH-data and BCHKK-data datasets are different in that the VXH-data matrix results in precision and recall far below that yielded by BCHKK-data. As such, the VXH-data matrix is already "decayed" more from the best possible precision and recall than is that for the BCHKK-data; introducing perturbations in an already decayed distance matrix will do little to demonstrate the sensitivity of a highly accurate distance matrix to perturbations. In order to start from good precision and recall, then, we adopt the *testing* VXH-data matrix and BCHKK-data matrix (i.e., resulting from BCHKK-algo) as the *reference* matrices, i.e., so that we start from precision and recall of 1.0. We then measure the rate of degradation from this ideal as the perturbations are introduced into the distance matrices, compared to these references.
- When re-clustering a perturbed distance matrix, the cluster-size distribution might be altered, in that hierarchical clustering simply might not produce an identical cluster-size distribution as the original from the perturbed distance matrix. For this reason, we fit a parameterized distribution to the reference cluster-size distribution and stop hierarchical clustering at the point that maximizes the p-value of a chi-squared test between the test cluster-size distribution and the fitted reference

distribution. In general, we find that a Weibull distribution with shape parameter $k = 0.7373$ and scale parameter $\lambda = 1.9887$ is a good fit for the reference clustering (i.e., the initial test clustering resulting from BCHKK-algo, as described above) of the VXH-data dataset (p-value of 0.8817), and that the corresponding values for the BCHKK-data are $k = 0.4492$ and $\lambda = 5.1084$ (p-value of 0.8763).

– Given that we have only a distance matrix, a method of perturbing it so that its entries maintain properties of a distance (notably, satisfying the triangle inequality) is necessary. To do this, we map the distance matrix into a d-dimensional space, i.e., creating d-dimensional points to represent the malware instances, separated according to the distances in the matrix. To then perturb the distances, we simply move each point to a random spot in the ball of radius r centered at that point. We can then produce the corresponding distance matrix for these perturbed points, and re-cluster. By increasing r, we then increase the maximum perturbation to the distances.

The results of this analysis are shown in Figure 4. In this figure, the x-axis shows the radius r within which we perturbed each point from its original position in d-dimensional space. The y-axis shows the F-measure that resulted for each radius r, averaged over five runs; standard deviations were negligible. As this figure shows, the cluster-size distributions characteristic of the VXH-data were indeed more sensitive to perturbations in the underlying data than were those characteristic of the BCHKK-data. In addition, in Figure 5 we show the p-values of chi-squared tests comparing the cluster-size distribution of the clustering after perturbation and the fitted (Weibull) reference cluster-size distribution. The fact that these p-values are not significantly decreasing indicates that the cause of degradation in the F-measure was not primarily due to deviation from the intended cluster-size distributions.

We also plot a "Downsized BCHKK-data" line in Figure 4 to control for the discrepancy in the number of malware instances represented in the BCHKK-data and VXH-data datasets. To do this, we randomly removed instances from BCHKK-data (irrespective of the reference clusters in which they fall) until the size of the data set is the same as that of VXH-data, i.e., $1,114$ instances. Using the correspondingly downsized

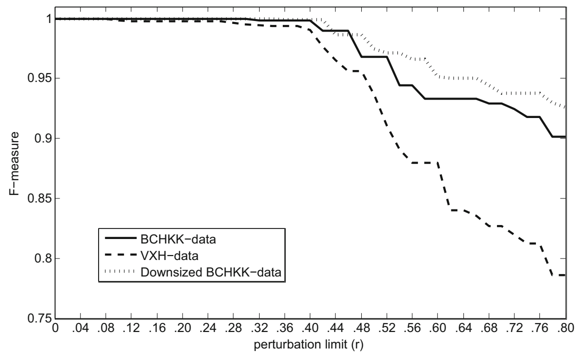

Fig. 4. Tolerance to perturbations

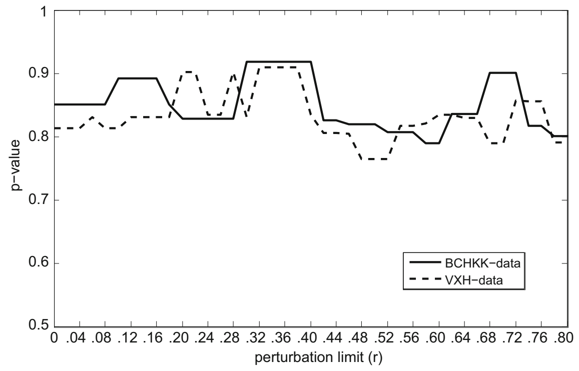

Fig. 5. p-values for perturbation tests

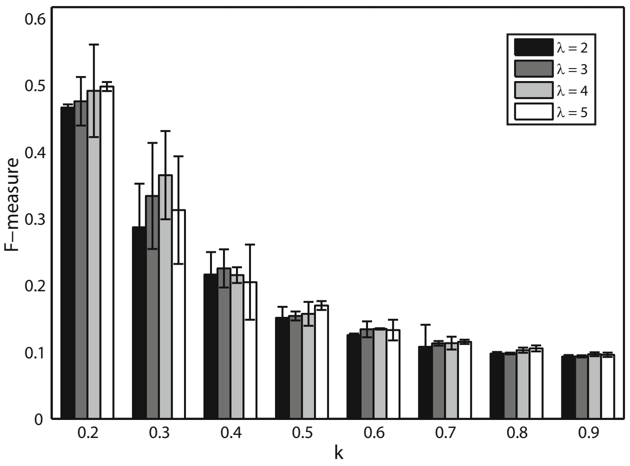

Fig. 6. F-measure of *random* test and reference clusterings with cluster sizes drawn from a Weibull distribution with scale parameter $\lambda \in [2, 5]$ and shape parameter $k \in [0.2, 0.9]$, averaged over 10 trials. Error bars show standard deviation. Note that the best-fit (k, λ) value for the BCHKK-data reference clustering is $(0.4488, 4.8175)$ and for the VXH-data reference clustering is $(0.7803, 2.5151)$.

distance matrix, we applied hierarchical clustering using the same threshold to stop clustering as reported in [6], resulting in a clustering whose cluster-size distribution has corresponding estimated Weibull parameters $k = 0.4307$ and $\lambda = 2.0399$. We took this clustering as the starting point for "Downsized" perturbation test and show the results (averaged over 5 runs) in Figure 4. And as we can see, "Downsized" BCHKK-data is still more immune to perturbation than VXH-data.

To further examine the effects of cluster size distributions on precision and recall, in Figure 6 we plot the average F-measure for reference clusters $\mathcal{D}$ and test clusters $\mathcal{C}$ whose cluster sizes are chosen from a Weibull distribution with the shape parameter

k shown on the x-axis. Once the reference and test cluster sizes are chosen (independently), the reference and test clusters are then populated independently at random (i.e., each point is assigned to a cluster in $\mathcal{D}$ and a cluster in $\mathcal{C}$ independently). As Figure 6 shows, the F-measure that results simply from different values of k provides further insight into the bias that cluster size distribution can introduce.

We do not mean to suggest that the complete discrepancy between the results of the BCHKK-algo clustering technique on the VXH-data and BCHKK-data is due solely to the cluster size distributions underlying the two datasets. However, we do believe that this case and our analysis of it offers sufficient evidence to recommend that evaluation of future clustering techniques be done on datasets with a variety of cluster size distributions. It is also possible that measures of cluster accuracy other than precision and recall better avoid this source of bias. For example, Perdisci et al [15] employed an approach based on the compactness of each cluster and the separation among different clusters, which may be preferable.

6 Conclusion

In this paper we have reported on our investigation of the impact that ground-truth selection might have on the accuracy reported for malware clustering techniques. Our starting point was investigating the possibility that a common method of determining ground truth, namely utilizing the concurrence of multiple anti-virus tools in classifying malware instances, may bias the dataset toward easy-to-cluster instances. Our investigation of this possibility was based on clustering using a different set of tools developed without attention to the subtleties of malware, namely plagiarism detectors. While our application of these tools, first to a dataset used in the evaluation of a state-of-the-art malware clustering technique and second to a whole new malware dataset, arguably leaves our conjecture unresolved, we believe that highlighting this possibility is important to facilitate discussion of this issue in the community.

It has also led us to examine an issue that we believe to be important for future analyses of malware clustering, namely the impact of the ground-truth cluster-size distribution on the significance of results suggesting high accuracy. In particular, we have shown that the highly accurate results reported for a state-of-the-art malware classifier (BCHKK-algo) are tempered by a reduced significance owing to having tested on a dataset with a biased cluster-size distribution. We consequently recommend that future evaluations employ data with a cluster-size distribution that is more even.

We caution the reader from drawing more conclusions from our study than is warranted, however. In particular, despite the similar performance of the BCHKK-algo algorithm and the plagiarism detectors in clustering on the malware datasets we considered, it is not justified to conclude that these algorithms are equally effective for malware clustering. The design of the BCHKK-algo algorithm should make it more difficult to evade, not to mention more scalable. It is evident, however, from our results in Section 3 that either malware today is not designed to exploit differences in the clustering abilities of BCHKK-algo and plagiarism detectors, or else that the ground-truth selection of the test datasets eliminated malware instances that do so.

We recognize that our paper has perhaps introduced more questions than it has definitively answered. Nevertheless, we believe that in addition to the observations above, multiple opportunities for future research can be drawn from our investigation. In particular, we believe our investigation underlines the importance of further research in malware clustering, specifically in better methods for establishing ground truth, in identifying more reliable features for malware clustering, or in both.

Acknowledgements. We are deeply grateful to Ulrich Bayer, Engin Kirda, Paolo Milani Comparetti, and Christopher Kruegel for helpful assistance, for providing access to the BCHKK-data dataset, for applying BCHKK-algo to our VXH-data dataset, for helpful discussions, and for useful comments on drafts of this paper. This work was supported in part by NSF award CT-0756998 and by DRTech Singapore under grant POD0814140.

References

1. Threatexpert3, `http://www.threatexpert.com/`
2. Norman sandbox center (2008), `http://www.norman.com/security_center/security_tools/en`
3. VX Heavens (2010), `http://vx.netlux.org/`
4. Aiken, A.: Moss: a system for detecting software plagiarism, `http://theory.stanford.edu/~aiken/moss/`
5. Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F., Nazario, J.: Automated classification and analysis of internet malware. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 178–197. Springer, Heidelberg (2007)
6. Bayer, U., Comparetti, P.M., Hlauschek, C., Kruegel, C., Kirda, E.: Scalable, behavior-based malware clustering. In: Proceedings of the Network and Distributed System Security Symposium (2009)
7. Bayer, U., Kruegel, C., Kirda, E.: Ttanalyze: A tool for analyzing malware. In: 15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference (2006)
8. Commtouch, Inc. Malware outbreak trend report: Bagle/beagle (March 2007), `http://www.commtouch.com/documents/Bagle-Worm_MOTR.pdf`
9. Gheorghescu, M.: An automated virus classification system. In: Proceedings of the Virus Bulletin Conference, VB (1994)
10. Ha, K.: Keylogger.stawin, `http://www.symantec.com/security_response/writeup.jsp?docid=2004-012915-2315-99`
11. Hu, X., Chiueh, T., Shin, K.G.: Large-scale malware indexing using function-call graphs. In: Proceedings of 16th ACM Conference on Computer and Communications Security (2009)
12. Kamiya, T., Kusumoto, S., Inoue, K.: Ccfinder: A multi-linguistic token-based code clone detection system for large scale source code. IEEE Trans. on Software Engineering, 654–670 (2002)
13. Lee, T., Mody, J.J.: Behavioral classification. In: 15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference (2006)
14. McAfee. W97m/opey.c, `http://vil.nai.com/vil/content/v_10290.htm`
15. Perdisci, R., Lee, W., Feamster, N.: Behavioral clustering of http-based malware and signature generation using malicious network traces. In: USENIX Symposium on Networked Systems Design and Implementation, NSDI 2010 (2010)
16. Rieck, K., Holz, T., Willems, C., Dussel, P., Laskov, P.: Learning and classification of malware behavior. In: Zamboni, D. (ed.) DIMVA 2008. LNCS, vol. 5137, pp. 108–125. Springer, Heidelberg (2008)

17. Rieck, K., Trinius, P., Willems, C., Holz, T.: Automatic analysis of malware behavior using machine learning. Technical Report 18-2009, Berlin Institute of Technology (2009)
18. Song, D., Brumley, D., Yin, H., Caballero, J., Jager, I., Kang, M.G., Liang, Z., Newsome, J., Poosankam, P., Saxena, P.: Bitblaze: A new approach to computer security via binary analysis. In: Proceedings of the 4th International Conference on Information Systems Security (December 2008)
19. Symantec. Spyware.e2give, `http://www.symantec.com/security_response/writeup.jsp?docid=2004-102614-1006-99`
20. Symantec. Xeram.1664, `http://www.symantec.com/security_response/writeup.jsp?docid=2000-121913-2839-99`
21. Tamada, H., Okamoto, K., Nakamura, M., Monden, A., Matsumoto, K.: Dynamic software birthmarks to detect the theft of windows applications. In: International Symposium on Future Software Technology (2004)
22. Tan, P., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Reading (2006)
23. Wang, X., Jhi, Y., Zhu, S., Liu, P.: Detecting software theft via system call based birthmarks. In: Proceedings of 25th Annual Computer Security Applications Conference (2009)
24. Whale, G.: Identification of program similarity in large populations. Computer Journal, Special Issue on Procedural Programming, 140–146 (1990)
25. Willems, C., Holz, T., Freiling, F.: Toward automated dynamic malware analysis using cwsandbox. In: Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P 2007), pp. 32–39 (2007)
26. Wise, M.J.: Detection of similarities in student programs: Yaping may be preferable to plagueing. In: Proceedings of the 23rd SIGCSE Technical Symposium (1992)