On the uniqueness of AntiVirus labels: How many labels do we need to fingerprint an AV?

Marcus Botacin[1*]

[1*]Computer Science and Engineering (CSE) Department, Texas A& University (TAMU), PETR Building, College Station, 77843, Texas, USA.

Corresponding author(s). E-mail(s): botacin@tamu.edu;

Abstract

The biggest drawback of AntiViruses (AVs) experiments is label heterogeneity–each AV labels the same samples very distinctly. Whereas AV labeling issues have been well studied from the sample point of view, they have not been widely studied from the AV perspective, i.e., to what extent label diversity allows AV identification. Thus, we question: (1) *How unique among all the AVs are the labels produced by the same given AV?* (2) *Can we fingerprint AVs based on their assigned labels?* and (3) *How many labels are required to fingerprint an AV?* In this work, we answer these questions via experiments with a dataset of 720000 AV-assigned labels for Windows malware spread over 15 years (2006-2020). We discovered that: (1) AVs can be fingerprinted by their assigned labels with 100% accuracy in many cases; (2) AVs can be fingerprinted with a confidence score of 99% using only 1% of the dataset; (3) AV fingerprinting rates vary over time, as the label changes caused by the AV updates have a key effect on AV recognition, causing some AV models to lose their ability to recognize their AV generated labels over time; and (4) Android AVs can be fingerprinted the same way as Windows AVs, but that Linux labels are harder to be grouped. We expect our work might shed light on the label heterogeneity problem, incentivize further developments to mitigate it, and provide future works with data to support their design decisions.

Keywords: antivirus, labeling, classification, concept drift

1 Introduction

Antiviruses (AVs) are key security solutions, being responsible for protecting millions of users every day. AVs are also key for security research. Many works rely on AV scanners such as Virustotal [1] for their developments. Virustotal and most AVs usually provide detection rates and labels for malware samples. Whereas useful for many research tasks, such as training Machine Learning (ML) models [2], the reliance on AV labels presents many challenges [3].

A major drawback of current AV solutions is the label heterogeneity problem, i.e., the fact that different AVs label the same sample differently, even though there is a naming convention that should be followed [4]. Having multiple different labels for the same threat significantly complicates incident response procedures. Label heterogeneity has been studied in many previous works [5–7], but no previous work studied the inverse phenomenon, i.e., if the label heterogeneity allows fingerprinting their assigning AVs.

If the AVs agreed on assigning similar labels to the same samples, it might be infeasible to differentiate the AVs that generated a given label for a sample. But, if the AV labels are so different, it might be possible to trace back the originating AV from a given assigned label. We here test this hypothesis. The closest work to ours grouped AV labels for family identification [8] and not for AV fingerprinting, such that the test of this hypothesis is an open problem.

To test this hypothesis, we consider a representative dataset of 720K Windows malware labels spread among 72 AVs and over 15 years (2006-2020) of label collections from VirusTotal. We present the ML modeling framework required to encode the labels and to train ML models that allow separating the labels assigned from one AV to another. We characterize the fingerprinting ability regarding its **(1)** viability; **(2)** scalability; and **(3)** longitudinal aspects.

In sum, we discovered that: **(1)** Fingerprinting AVs from their assigned labels is possible, and it might be performed with 100% accuracy for small problem instances (12 AVs); **(2)** AVs can be fingerprinted with 99% confidence with less than 100 queries (lower than 1% of the whole dataset); and **(3)** AV labels change over time causing AV models to lose their ability to recognize prior AV labels.

In sum, our contributions are: **(1)** Proposing classifying AV labels as a fingerprinting mechanism to identify which AVs generated them; **(2)** Presenting the methodology and ML modeling required to test the AV labels fingerprinting hypothesis; and **(3)** Comprehensively characterizing the dynamics of the AV label fingerprinting process regarding viability, scalability, and temporal validity.

2 Motivation

Why would one fingerprint AVs? Suppose an attacker targeting a highly valuable asset (not a low-hanging fruit type of operation). The attacker can offline create multiple Adversarial Examples (AEs) and keep them in a pool [9]. The target system is protected by an unknown detection solution (an AV) and the attacker expects some AE in the pool to transfer [10] to the real world (i.e., bypass the AV). The attacker can randomly sample one AE from the pool and try to infect the system by chance. If it fails, the network host (node) originating the attack is blocked by the defense system [11]. Since many AEs are likely to fail, the attacker uses a botnet to launch individual infection attempts (one AE per node), as in Figure 1.

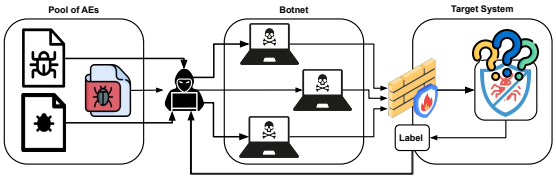

Fig. 1 Attacker probing system's defenses.

While a naive attacker would exhaustively try all AES, a smarter attacker knows that the botnet would have to be as large as the number of AEs (they might be even millions). Larger botnets are hard to build and maintain, and easy to spot. Also, the more failed attempts, the more information attackers provide about their TTPs [12]. This is the reason why attacker want to minimize the number of adversarial queries [11]. A more targeted attack would require attackers to know which AV the target system uses. This way, the attacker could perform an informed selection of AEs from the pool by testing them offline against a substitute model for the target AV [13].

Now, imagine that the attacker has access to the AV labels produced by each failed infection attempt. This might be because **(1)** the system provides it, **(2)** or the attacker can eavesdrop on communications, or **(3)** the has access to the log console due to prior lateral movements. If the attacker could map the labels back to the AV that originated them, then the attacker could perform the targeted attack. If AVs produced homogeneous labels, this task would not be possible, but if the AV labels are heterogeneous, the attackers can identify the AV. An attacker could identify the AV by submitting their detected samples to a pool of AVs (e.g., VirusTotal), but this would once again reveal much information about the attacker. Instead, the attacker could have offline models for the AV labels. It is reasonable to suppose substitute models for AV labels once substitute models for the AV detectors are already initially supposed by most AE generators [13, 14]. The viability of

the attack depends on the feasibility of fingerprinting the AVs by their labels. Characterizing this feasibility is this work's goal.

3 Methodology

3.1 AV Modeling.

We created a uniform AV label representation by tokenizing and encoding them. The tokenization step splits the labels into their separators (e.g., commas, semi-colons, dots, and so on), thus allowing us to obtain the malware information directly. Leaving the separators would make classification less precise because similar words separated differently would be classified differently based on the AV label structure and not on the information provided by the AV about the samples. Since each AV might produce labels with a different number of tokens, we normalized all labels about the longest token vector in our dataset, padding the blanks. Each position of the tokenized and normalized vector is encoded via 1-hot encoding, chosen due to its simplicity. We encoded even the empty tokens since an AV having distinct label lengths might be important information for attributing AV labels.

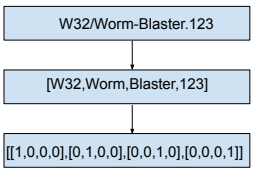

Fig. 2 Label 1. One-hot encoding.

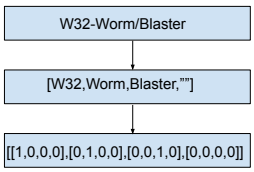

Fig. 3 Label 2. One-hot encoding.

Figure 2 exemplifies the label-handling process. The initial label is tokenized into 4 words. Each word is then encoded in its own vector. The final vector is inputted into the ML model. Figure 3 shows a similar label to the one from Figure 2 but with different separators and without the "version" information (123). It is also tokenized into 4 words, with the missing information leading to an empty entry. All words are encoded, which leads to an array in which the first three positions are similar to the previous one.

After the label representation normalization, we trained multi-class ML models to represent the AVs. In this paper, all mentioned models are based on the RandomForest (RF) classifier, since it is the classifier that usually presents the best results in most related works. When a 1-class classifier is applied (self-recognition experiment), we trained a 2-class classifier where one class is supplied with a single AV label and the other class is trained based on the remaining labels.

3.2 Datasets

To characterize AV label fingerprinting in multiple dimensions we rely on three different datasets. The most significant part of this work is built on top of a dataset of labels from Windows malware samples, since most malware and AVs have been developed for this OS. We also present the first results of AV fingerprinting on Android and Linux, aiming to present potential points of convergence or divergence between them.

Windows Dataset. Our dataset construction goals were to be: (i) Broad, i.e., to cover the maximum number of AVs possible; (ii) Longitudinal, i.e., that the information was spread over the years to allow temporal analyses; and (iii) Coherent, i.e., that the labels refer to the same samples and that they were generated by the same set of AVs, thus allowing comparisons. To meet these criteria, we discard the hypothesis of crawling random samples and consider 3rd-party labels. Instead, we opted to rely on our own collection of malware, which allowed us to have full control of when labels were generated.

We started dataset construction with a collection of 21K malware samples collected from infected machines over multiple years. This dataset was characterized in previous research [15] and provides us with more than 1M different labels queried on Virustotal over time. For the tests, we discarded the Virustotal AVs whose labels are based on Machine Learning (ML) scores (e.g. `label=malware:confidence:90%`), since these labels are not informative of malware families. We identified 72 different AVs that labeled a subset of 667 of our samples. Thus, our test dataset is composed of 720000 labels (10000 labels for each one of the 72 AVs). These labels are spread over 15 years (from 2006 to 2020), thus 48000 labels per year are considered; 667 labels per AV each year. These samples belong to diverse high-level families (e.g., Trojan, Backdoor, and so

on): 9 according to the AV that assigns fewer family labels; 14 on the average of all AVs; and 20 for the one that most assign different family labels.

Android Dataset. We constructed the Android dataset following the same strategy adopted for the Windows one. Unfortunately, we did not have as many APKs in our collection as we have Windows malware, so we had to relax some constraints. We opted to reduce the longitudinal aspect to increase the number of considered samples/labels (broadness) and to be able to present more coherent results. Thus, we could keep evaluating a set of 72 different AVs, but now with 100 labels each one diversely spread over multiple years. In total, we considered 7200 APK labels for the same 100 samples.

Linux Dataset. We applied for the Linux dataset with the same rationale as for the Android one. From our initial set of 5K collected samples, we were able to select 100 samples labeled by the same 72 AVs. In total, we considered 7200 labels diversely spread over multiple years.

Temporal Consistency. We collected all AV labels a significant time after they were first seen in the VT service (¿30 days), which was enough time for the labels to stabilize and not present transient divergencies [3]. Also, we retrieved all data directly from the VT service and not from third parties, thus minimizing the risks of malicious label-flipping attempts that could disrupt our classifiers [16, 17].

4 Evaluation

We following present our experiment results according to the obtained findings.

4.1 Is AV label fingerprint possible?

The Identification Rate (IR) depends on the number of AVs. We first investigated if AV labels could be mapped back (attributed) to their assigning AVs. For such, from the multi-class confusion matrix resulting from the grouping process, we collected the accuracy scores, i.e., the relative frequency in which the labels for each AV were correctly attributed in the respective assigning AV. We hypothesize that label attribution might be possible, but the problem might become harder as more AVs are considered. Thus, in our experiment, we varied the number of considered AVs in each run. Since not all AVs could be considered at once in small test scales, we randomly selected N AVs for each run. We ran the tests multiple times until reaching the 95% confidence score. The results are reported in terms of the average score of the multiple runs (see Section 6).

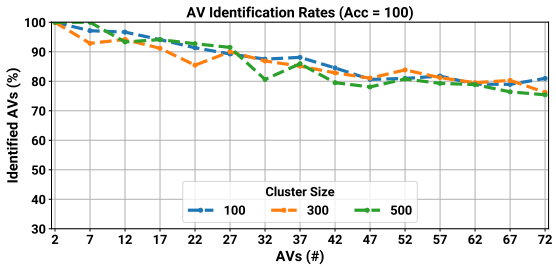

Fig. 4 AV classification accuracy (small dataset). Rate of AVs identified with a 100% accuracy for a distinct number of AVs. The more AVs are considered, the harder the classification problem.

Figure 4 shows the rate of AVs presenting a 100% accuracy score (Y-axis) for an increased number of AVs in the test set (X-axis). The graph shows curves for 3 different numbers of samples per test set: 100, 300, and 500. It means that when 2 AVs and 100 samples are considered, the training is performed with 200 samples of the dataset and tested with another 200 samples. Similarly, when 100 samples and 72 AVs are considered, 7200 samples are used in the training and another 7200 samples are used for the prediction step.

The results show that, for 2 AVs, the classifier achieves a 100% accuracy in all runs, thus showing that the labels are easily separable. The rate of perfect identification decreases as more AVs are added to the test since there is an increased chance that some labels might belong to more than one class (i.e., distinct AVs producing the same label, e.g., Trojan). Despite the observed decrease, the perfect identification rate is still above 90% on average in total for scenarios with up to 27 AVs, which is still very practical for many scenarios, where one is expected to disambiguate between only a few AVs. When all AVs available in our dataset were considered (N=72), the perfect score rate decreased to 80%, i.e., 80% of all AVs have completely separable labels. No differences were observed between the three set sizes (100, 300,

500), thus the results hold for experiments in this order of magnitude.

The Identification Rate (IR) depends on the AV label diversity. Once we demonstrated that fingerprinting AVs was feasible (small-scale test), we investigated to which extent the results hold. Thus, we repeated the previous experiments with different amounts of samples (set sizes).

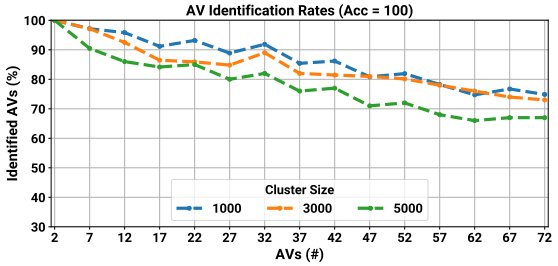

Fig. 5 AV classification accuracy (medium dataset). Rate of AVs identified with a 100% accuracy for distinct AV set sizes. The more AVs are considered, the harder the classification problem.

Figure 5 shows curves for 3 different numbers of samples per AV set: 1000, 3000, and 5000. For 2 samples, the perfect separation score is once again achieved during all runs with different AVs, thus showing that disambiguating between two AVs is easy (computationally speaking). The curves decrease a bit more than before but in a similar proportion to the previous experiment. Now, with 27 AVs, the perfect identification rate is at least 80% for all group sizes. In the most challenging scenario, N=72, 70% of all groups are completely separable.

Figure 6 shows curves for 3 different numbers of samples per group: 10000, 30000, and 50000. Once again, 2 AVs are clearly separable. However, unlike the previous experiments, the decrease is immediate and abrupt as more AVs are added to the experiment. In the worst-case scenario, the perfect identification rate is around 50%, i.e., only half of the AV labels are completely separable. This difference from previous experiment results indicates that this amount of data takes the problem to another difficulty level.

The Identification (IR) depends on the required accuracy. A solution to make the AV fingerprint task feasible for very large instances is

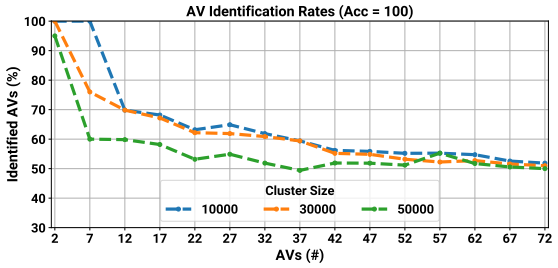

Fig. 6 AV classification accuracy (large dataset). Rate of AVs identified with a 100% accuracy for distinct AV set sizes. The more AVs are considered, the harder the classification problem.

to lax some constraints, thus reducing the problem's complexity. Whereas completely separating the AV labels (100% accuracy) is the ideal case, in some cases a lower accuracy might be enough–i.e., it might be enough if the majority of the samples attributed to a given AV are correct. To evaluate to which extent this task is feasible, we repeated previous experiments with lower thresholds.

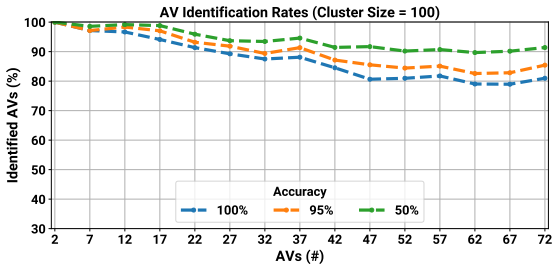

Fig. 7 AV classification accuracy (small dataset). Rate of AVs identified with the targeted accuracy for distinct AV set sizes. The tighter the threshold, the harder the classification problem.

Figure 7 shows the rate of groups (AVs) achieving the target accuracy score (Y-axis) for an increased number of AVs (X-axis). We show the scenario with 100 samples per class as representative of the small dataset problem class. The graph shows curves for 3 different target accuracies: 100%, 95%, and 50%. It means that 100%, 95%, or 50% of the samples assigned to a given group really belong to that group.

Lowering the threshold 5% (from 100% to 95%) increases the average number of groups targeting the desired accuracy in 7% on average.

We consider this a reasonable trade-off since the majority of the labels assigned to each group are still correct. We observe no significant effect for other accuracy values. In fact, lowering the threshold to 50% (impractical for actual scenarios) does not make the problem much easier. In this case, many AVs (10%) still cannot meet the target criteria. This phenomenon is explained by the presence of labels that are common among multiple AVs.

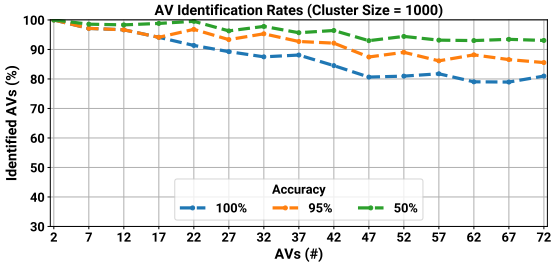

Fig. 8 AV classification accuracy (medium dataset). Rate of AVs identified with the targeted accuracy for distinct AV set sizes. The tighter the threshold, the harder the classification problem.

Figure 8 shows the rate of groups (AVs) achieving the target accuracy score (Y-axis) for an increased number of AV groups (X-axis). We show the scenario with 1000 samples per class as representative of the medium dataset problem class. We observed results similar to the previous experiment. Lowering the threshold by 5% resulted in an average gain of 9% in the number of correctly classified groups. Training with more samples helped mitigate the ambiguous labels (labels common to more than one AV), but lowering the threshold to 50% is still impractical for 5% of all AVs.

Figure 9 shows the rate of groups (AVs) achieving the target accuracy score (Y-axis) for an increased number of AV groups (X-axis). We show the scenario with 10000 samples per class as representative of the large dataset problem class. The overall scores are significantly lower in comparison to previous experiment results, which reinforces that the problem is really harder at this scale. Lowering the target accuracy requirement by 5% increases the number of AVs reaching the target by 13.5% on average. Increasing the training size did not help classify more samples this time. It shows that the complexity of the problem dominates the results.

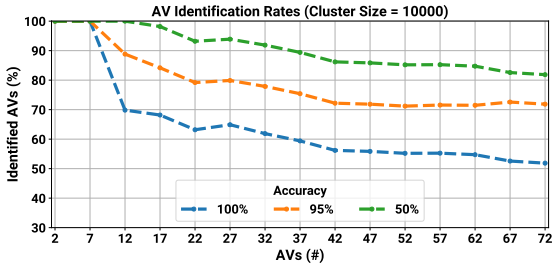

Fig. 9 AV classification accuracy (large dataset). Rate of AVs identified with the targeted accuracy for distinct AV set sizes. The tighter the threshold, the harder the classification problem.

4.2 How fast is it to fingerprint an AV from its labels?

Although we previously showed experiments considering all AV labels at once, in practice, one will likely not assign all labels to a group before making a conclusion. Instead, the most practical scenario from one having a trained grouping model is to test the minimum number of labels possible to discover which AV generated them. This is usually the case when one has a dataset of labels assigned by the same AV and whose task is to discover which AV generated them. One might simply randomly select a label from this set, test it using the trained grouping model, and assume the same result holds for the other labels in the dataset. If one wants a higher confidence in the result, one might repeat this same step for a few other labels. We put this scenario to test in the experiments reported in this section. Our goal is to discover the minimum number of queries one must make to a trained grouping model to discover the AV which generated them. All tests were performed with the trained models that achieved 100% accuracy in the training step (previous section experiments).

The number of queries to fingerprint an AV depends on the targeted confidence level. When one queries a classifier, it outputs a label and a confidence score on the given prediction. Thus, if one queries a single label, one has only a single confidence value to decide if the prediction is correct. If the confidence is low or below a target threshold, one might want to query additional labels to get additional confidence values and decide if the combined statistical confidence meets the target criteria. In the following, we present the

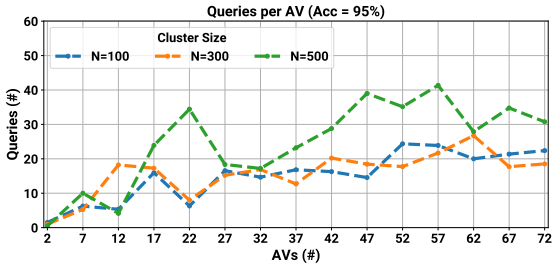

Fig. 10 Average number of queries to fingerprint an AV with 95% confidence (small dataset). On average, a smaller number of queries is enough to fingerprint the AVs even in instances with an increased number of classes.

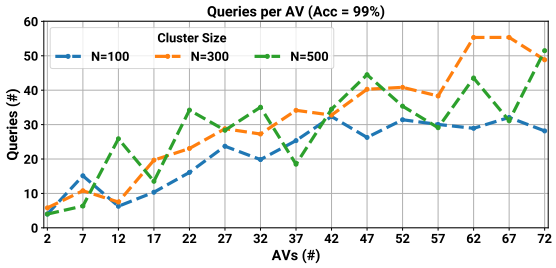

Fig. 11 Average number of queries to fingerprint an AV with 99% confidence (small dataset). On average, a smaller number of queries is enough to fingerprint the AVs even in instances with an increased number of classes.

results of the experiments performed to identify the minimum number of queries required to meet distinct target confidence scores.

Figure 10 shows the number of queries (Y-axis) required to reach the 95% confidence score for the AVs in AV sets of multiple sizes (X-axis). We considered AV sets with 100, 300, and 500 samples each. We observe that as a general rule, the required number of queries is low (in comparison to the number of totally considered samples); Less than 40 queries are enough to identify an AV in all tested scenarios. Although the grouping complexity exponentially increases with AV set size, we notice that the number of required queries for this experiment does not grow in the same proportion, thus showing that identifying which AV labeled a set of samples is an easier task than grouping the samples from scratch.

Figure 11 shows the number of queries (Y-axis) required to reach the 99% confidence score for the AVs in AV sets of multiple sizes (X-axis).

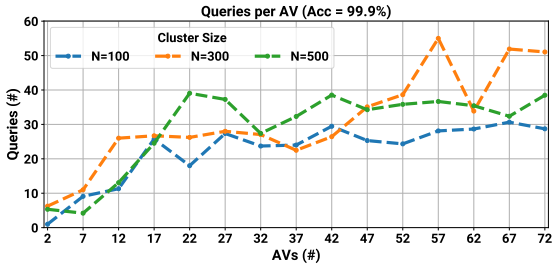

Fig. 12 Average number of queries to fingerprint an AV with 99.9% confidence (small dataset). On average, a smaller number of queries is enough to fingerprint the AVs even in instances with an increased number of classes.

We notice that although increasing only 4% of the target confidence score, it implies a more complicated problem. Now, to cover all cases, one needs a bit less than 60 queries in the worst scenario, a 50% increase in comparison to the previous scenario. Despite that, this number can still be considered moderate in comparison to the total number of considered labels and the growth is still not exponential.

Figure 12 shows the number of queries (Y-axis) required to reach the 99.9% confidence score for the AVs in sets of multiple sizes (X-axis). There is no significant difference from the previous scenario, which shows that we had already achieved a 99.9% confidence level in the previous queries.

The number of queries to fingerprint an AV depends on the number of labels in the test set. Given the impact of group size observed in the previous experiments, we also investigated the effect of the AV set size on the number of queries required to identify the AV that labeled a given dataset.

Figure 13 shows the number of queries (Y-axis) required to reach the 95% confidence score for the AVs in sets of multiple sizes (X-axis). We considered groups with 1000, 3000, and 5000 samples each. In comparison to the previous scenario, whereas the number of samples per group was multiplied by 10, the number of queries is only multiplied by 1.5 on average, thus showing scalability. Most cases can be covered with less than 60 queries. This number once again is low in comparison to the total number of samples (now one order of magnitude lower). Even though the number of samples in each group was multiplied by 10,

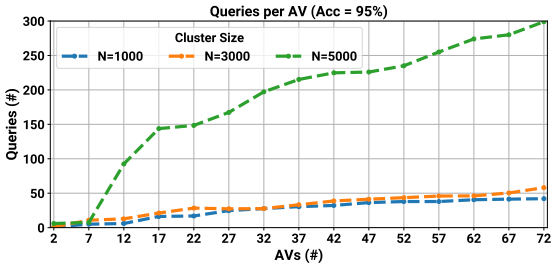

Fig. 13 Average number of queries to fingerprint an AV with 95% confidence (medium dataset). On average, a smaller number of queries is enough to fingerprint the AVs with instances of up to 3 thousand labels. The number of queries becomes exponential for instance with more than 5 thousand labels.

the number of required queries is still in the same magnitude order as in previous experiments. The exceptions are the sets with 5000 labels, whose growth becomes exponential, as in the previous attribution experiments, thus being less practical to address.

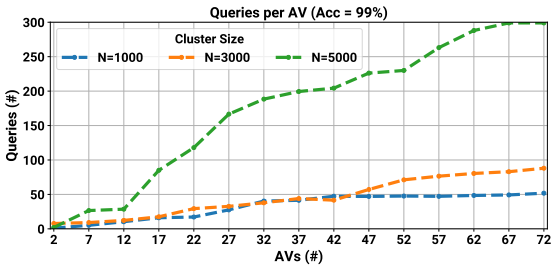

Fig. 14 Average number of queries to fingerprint an AV with 99% confidence (medium dataset). On average, a smaller number of queries is enough to fingerprint the AVs with instances of up to 3 thousand labels. The number of queries becomes exponential for instance with more than 5 thousand labels.

Figure 14 shows the number of queries (Y-axis) required to reach the 99% confidence score for the AVs in AV sets of multiple sizes (X-axis). As for the analogous experiment with a smaller dataset, whereas the target confidence score increased only 4%, the average number of queries increased by 50%. Now, to cover most cases, an average number of 90 queries is required in the worst case. Once again, the exceptions are the sets sized 5000. These sets showed a *stepping* growing characteristic, highlighting that the task gets more complicated in steps. It shows different characteristics in the intervals [2-12], [13-22], [23-32], [33-37], [38-57], and [58-72].

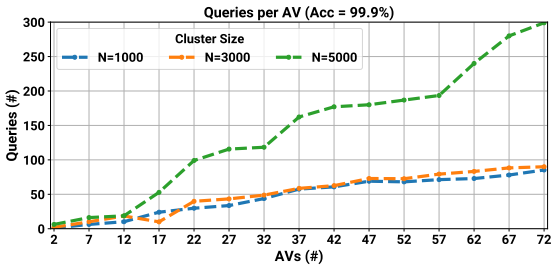

Fig. 15 Average number of queries to fingerprint an AV with 99.9% confidence (medium dataset). On average, a smaller number of queries is enough to fingerprint the AVs with instances of up to 3 thousand labels. The number of queries becomes exponential for instance with more than 5 thousand labels.

Figure 15 shows the number of queries (Y-axis) required to reach the 99.9% confidence score for the AVs in sets of multiple sizes (X-axis). As in the analogous experiment, the required number of queries has not grown significantly from the 99% confidence level, thus showing that a high confidence level had already been achieved. Once again, the sets sized 5000 become exponential with a stepping growth characteristic, thus being impractical. We did not test sets sized by a greater magnitude order (e..g, 10000), as we had already reached the exponential increase level.

4.3 How does AV label fingerprinting change over time?

In addition to demonstrating fingerprinting viability, it is also essential to establish the limits for such possibilities. Thus, we characterize the fingerprinting process according to multiple factors that might influence its viability. Given previous works [3, 18] have shown that AVs are subject to the effect of time, we investigate if it also impacts the fingerprinting viability. To do so, we repeated the previous attribution experiments now breaking down the dataset by year. For each given year, all available AVs were considered.

Labels change over time and it makes AV fingerprinting moderately easier over time. We first attributed all labels for each year.

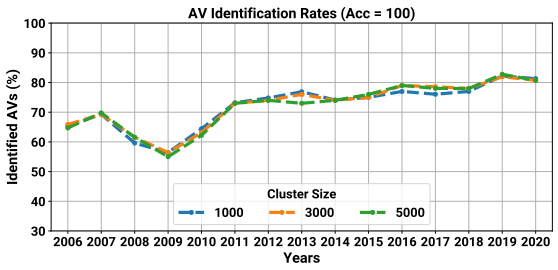

Fig. 16 Attribution accuracy over the years. Label evolution change attribution results. Attribution abilities are moderately getting better over time.

Figure 16 shows the rate of AV sets presenting a 100% accuracy score (Y-axis) over the years (X-axis). The graph shows curves for 3 different numbers of samples per set: 1000, 3000, and 5000. We considered this order of magnitude because it is large enough to be representative but it is still at the border of exponential growth (as seen in previous experiments) to still be tractable.

We notice from the figure that the AV labels indeed change over time and that it affects the attribution, as the number of AVs correctly attributing all labels changed over time. No significant difference is observed for the different problem instance sizes, such that the effect of time dominates the observed effects. We notice that the label differentiation between the AVs is reasonably stable over time since on average 70% of the AVs can be completely differentiated via their generated labels. However, the effect of label change over time cannot be neglected, as a significant variation is observed in the [2007-2009] period. We cannot claim this variation is only statistical as the same samples generated the considered labels and no random AV sampling was performed in this experiment, thus only a change in the labels might have caused a change in the classification accuracy. A moderate trend of accuracy growth is observed over time. At the end of the period, a larger fraction of AVs was correctly classifying 100% of the labels. This implies that the AVs are becoming "easier" to fingerprint over time. We explain it by noticing that the labels become more specific over time, including more sample information, such as family and variant identification. This extra information facilitates associating labels to their generated AVs.

Label variations over the years imply that a distinct number of queries is required to fingerprint AVs with confidence depending on the year. Once the effect of label changes over time has been demonstrated in the controlled environment, we consider the case of one trying to identify which AV generated each set of labels with respect to a given confidence level. For this test, we removed the constraint on the number of samples per set and repeated the sampling experiment based on sets composed of all labels assigned by the AVs in each given year.

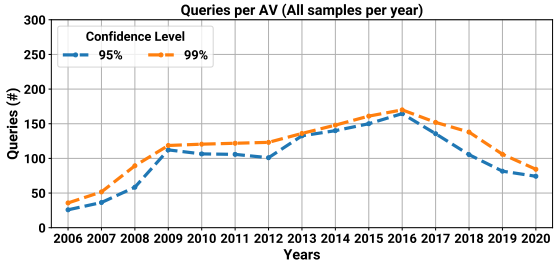

Fig. 17 Number of queries per year (Average AV). Sampling with confidence guarantees becomes easier in the latest years after an initial step involving more complicated labels.

Figure 17 shows the average number of queries (Y-axis) required to reach the 95% and 99% confidence scores for the AVs in the sets corresponding to the multiple years (X-axis). We selected the target confidence values based on the results demonstrated for the previous experiments. The figure shows an exponential increase in the required number of queries to fingerprint an AV in the first observation years, which shows that the AV labels become harder to distinguish in the period. The exponential growth was expected because removing the group size constraint takes the problems to the border of exponential growth, as shown by previous experiment results. We notice a period of moderate stability during the [2009-2017] period. The linear growth is still inside the statistical error margin. There has been a decrease in the number of required queries to fingerprint an AV in recent years, which is compatible with previous findings that AV labels have become easier to distinguish in recent years. As expected, the number of queries to reach the 99% confidence level

is greater than for the 95% level, but the difference is small, thus showing that a small number of queries is already informative. Regardless of the individual variations over the years, the AVs could be fingerprinted in all years with less than 200 queries, which is a small proportion compared to the thousands of labels in each set, thus showing fingerprinting viability.

The average result is different from individual AV results. It is important to highlight that the results presented in all previous experiments are an average for all tested AVs. A way to interpret these results is as if they were the results for an AV solution that is the average of all tested AVs–supposing it is possible to create an average AV solution. Naturally, not all AVs behave the same way, thus the number of queries varies per AV. Whereas our average results already show the feasibility of AV fingerprinting in the average case, we discovered that the average values were pushed up by the AVs which required a greater number of queries. Therefore, it is important to highlight the existence of AVs that can be fingerprinted with a significantly smaller number of queries.

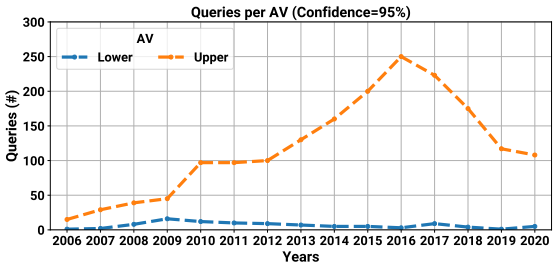

Fig. 18 Number of queries per year (Individual AVs). There is a significant difference between the AV that required more and the one that required fewer queries to be fingerprinted with confidence.

Figure 18 shows the number of queries (Y-axis) over the years (X-axis) required to reach the 95% confidence score in the label fingerprinting task for the AVs that required the biggest and smallest number of queries on average over the years. We chose the 95% values as the target based on the previous experiment's results. The number of queries required by the selected AVs can be considered as lower and upper boundaries for the previously presented average values. We notice that whereas one of the AVs presents a significant variation over time, requiring hundreds of queries to be fingerprinted with confidence, the other AV is very constant in the way it assigns labels, with a few queries being enough to fingerprint it. These results imply: (i) in the first case, the AV produces very similar labels, thus one needs multiple queries to identify a small variation that is unique to the AV; and (ii) in the second case, the AV produces very unique labels, such that a few queries are enough to find characteristics unique to that AV. It highlights once again the feasibility of the fingerprinting process, as less than 50 queries allow the identification of thousand-sized sets of labels with confidence guarantees.

AVs lose their ability to recognize their own labels over time. The previously shown effect of label changes over time on AV label fingerprinting procedures makes us wonder if the AVs still recognize their own labels after a long time has passed. To test that, we developed an experiment based on incremental learning/incremental windowing. We initially trained a one-class classifier model with the labels assigned by an AV in an initial year and then tested its ability to recognize the labels assigned by the same AV in future years. We proceeded by teaching the model data from another year (2 initial years in the training set) and testing it against the labels assigned in future years. We repeated the process by increasing the training data year by year.

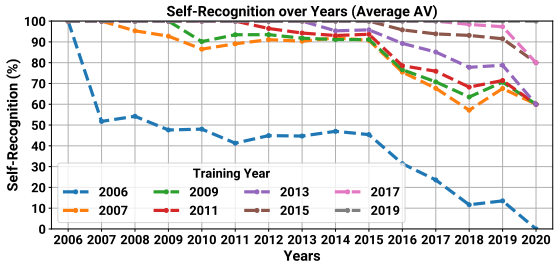

Fig. 19 AVs self-recognition over time. The labels an AV sees in future years are very different from the labels AVs see in their initial years, thus making models lose their self-recognition ability.

Figure 19 shows the average self-recognition rate (Y-axis), i.e., the fraction of the own labels correctly recognized, over the years (X-axis).

We notice that when training the models with 2006's data, the self-recognition rate immediately dropped by half in the subsequent year, thus showing that labels for this next year were very different from the labels from the previous year, such that the AVs did not effectively recognize part of their own labels as generated by them. The label diversity remained moderately stable during the [2007-2015] period, as the same training set was able to keep a self-recognition rate around the same value (50%). The labels significantly changed again after that, thus we can observe a new rate drop. In the last year (2020, 14 years after the initial training), the labels produced by the AVs are so different that the original models do not recognize almost any of the new labels as generated by them. Adding more data to the training set increases the self-recognition rate in all scenarios. For instance, adding the labels from 2007 to the training set helped increase the self-recognition rate for all consecutive years, although we can still observe a moderate decrease in the final years. The curves and thus the inferred label dynamics are overall the same as the training size is increased: the more data (years) in the training set, the more the AVs are able to identify their future labels. Even though, the self-identification rate decreases are still observed, which shows that new labels are still and periodically created by the AV companies.

Recognizing among others and recognizing itself are different problems. It is important to highlight that the presented results are the average values for the results of all AVs. Our goal with that is to shed light on the effect on the whole AV ecosystem. Individual AVs might present particular variations. More specifically, in our tests, whereas 80% of all AVs presented significant variation, 20% of all AVs were able to sustain a self-recognition rate superior to 90% of the whole period. Interesting to note that one of these AVs is the one whose number of queries was considered an upper boundary for the average in the experiment shown in Figure 18. It shows that this AV has always been diverse, such that the label diverse is only spread over the years and did not emerge at a specific moment. It also shows the difference between the tasks of comparing the labels against (i) other AV labels (a challenging task for recognizing this AV); (ii) against its own

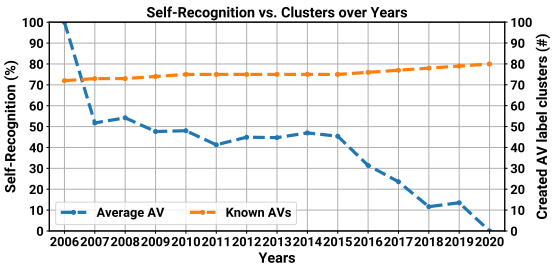

Fig. 20 Self-Recognition vs. Known AVs. When the labels change to the point of not being recognized by the original model, they are perceived as coming from a different AV than the ones from the training set.

labels (easy for the model). Finally, it is important to highlight that as mentioned in Section 3, our experiments did not include AVs completely based on ML. This is important to guarantee that our results are not biased. Pure ML-based AVs do not provide a family label, but only a binary label (e.g., confidence=100), so identifying their labels is easy for the classifiers and no change is observed over time.

The time creates "new" AVs. When AV labels evolve and stop being recognized as coming from the original AV, *what are they recognized as?* If they are confused with labels from the other AVs, this would show that the AV companies are working towards making the labels more similar. As a consequence, this would also imply that the results of our initial experiments would be biased by this temporal incoherence. We tested it by repeating the previous temporal experiments but now with an Out-Of-Distribution (OOD) detection strategy. We start by initially training the multi-class classifier with all known AVs (72) and by only allowing a sample to be assigned to a class if it has a confidence greater than 50% on the decision. Otherwise, it is assigned to a new cluster. As a result, if the incoming samples are confused with samples from other AVs, the number of clusters seen by the classifier would remain stable. In turn, if these samples are not confidently assigned to any cluster, the total number of clusters will grow.

Figure 20 illustrates this phenomenon happening with the "Average AV" from Figure 19. As the self-recognition drops, the number of observed samples grows, indicating that the new samples are assigned to new clusters. Over the years, more new clusters are observed, indicating that the new

labels are not assigned to the existing newly-created clusters, but to newer and newer clusters every year. We can interpret this result as the model understanding the labels of each year as coming from a completely different ("new") AV. This result ensures that our previous experiments are not temporally biased, because although the AV labels change over time, they are disjoint from the existing AVs, such that the complete label recognition model for an AV can be seen as a collection of different concepts that this AV has presented over the years.

The merging of AV engines is reflected in the self-identification rate over the years. The merging of AV engines is a common event in the industry; one product might have been acquired by another company and integrated into its product. A side effect of these merges is that both solutions start to exhibit the same labels. It has been observed in previous works [19, 20] and is also present in our experiments when observing the self-identification rate of some AVs.

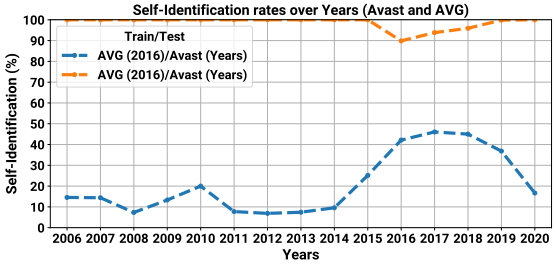

Fig. 21 Avast model predicting AVG labels. The labels became more similar around 2016.

Figure 21 shows the self-identification rate (Y-axis) over the years (X-axis) for two special scenarios: when the models for the AVG AV are trained with its own labels in 2006 and 2016 but tested against the labels produced by Avast in the same years. We chose to represent these two years for didacticism, but the effect is observed for all years. Considering the 2006's model, we notice that the self-cognition rate is initially low (10%) as expected since this is the rate of Avast labels that AVG was considering as created by itself, which is normally interpreted as a classification error. We notice a significant increase (from 10% to 50%) in the self-recognition rate around 2016, the year in which the Avast and AVG engines were merged. Whereas the self-recognition rate is moderate (50%), as the model was trained 10 years before, it is clear that a change in the Avast labels happened and that they are more similar to the AVG ones now. When we train AVG models with the labels up to 2016, we notice that, for that year to the future, the model recognizes all Avast labels as created by itself, thus showing that the engines were integrated and they are producing the same labels. The results are similar if we perform the test in the other direction: training Avast models and testing AVG samples.

4.4 AVs in other OSes

We previously presented a comprehensive characterization of the AV label dynamics for the Windows environment, the OS most targeted by malware writers, and also the most traditional AV ecosystem. However, we know that malware dynamics is OS-dependent. Thus, a characterization of the AV fingerprinting process is only complete when we understand how the environment affects the label dynamics. To evaluate that, we considered two other scenarios: (i) Android, the most popular mobile OS, with an emerging number of malware threats and AVs; and (ii) Linux, a traditional but not-so-popular OS. This combination allows us to explore the two dimensions of label dynamics: OS/AV consolidation and popularity.

APK labels are similar to Windows ones. To investigate the dynamic of the APK labels, we took a comparative approach. i.e., we compared the APK results with the exhaustive investigation of the Windows label dynamics to identify if the APK dynamic is compatible with it. More specifically, we took the AV Identification Rate (IR) as a proxy for all metrics.

Figure 22 shows the rate of groups (AVs) achieving the target accuracy score (Y-axis) for an increased number of AVs (X-axis). The graph shows curves for 2 different target accuracies: 100% and 95%. It means that 100% and 95% of the samples attributed to a given label set were really generated by that AV. The behavior of the curve is very similar to the one for Windows samples presented in Figure 7. In both cases, the IR is sustained at 100% until 12 AVs, when it starts to

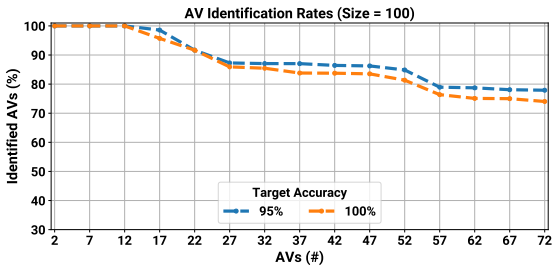

Fig. 22 AV Identification Rate for APK samples. The curve characteristic is similar to the Windows labels.

drop. The final IR is around [80-85] in the Windows case and [75-80] in the APK case. Previous work already suggested that much of the malware dynamics is similar between Windows and Android [21, 22]. Now, we claim that the AV label dynamics are very similar between the two environments. It shows that there is a label dynamics association with the variety of malware samples in a popular platform, despite the internal OS and AV implementations [19] differences.

ELF/Linux labels are harder to classify than Windows ones. To investigate the dynamic of the ELF labels, we also took a comparative approach with Windows. Once again, we took the AV Identification Rate (IR) as a proxy for all metrics.

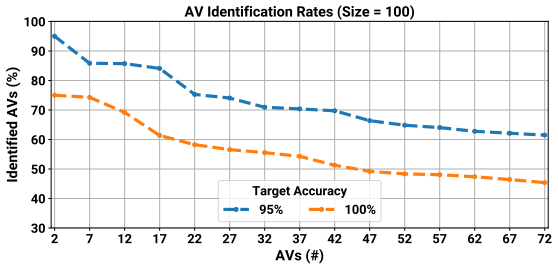

Fig. 23 AV Identification Rate for ELF/Linux samples. The curve characteristic differentiates from the Windows labels, being harder to classify.

Figure 23 shows the rate of AV sets achieving the target accuracy score (Y-axis) for an increased number of AVs (X-axis). The graph shows curves for 2 different target accuracies: 100% and 95%. The behavior of the curve is very different from, the one for Windows samples presented in Figure 7. The ELF/Linux labels are not completely separable even between only two different AVs, even if we lower the threshold, which shows that the Linux AVs are assigning more similar labels to the same samples than the Windows ones. This fact is reflected in the final IR, which is much lower (50% and 70%) than for Windows and Android. We hypothesize that the obtained results can be explained by the different natures of ELF/Linux malware. Previous works [23, 24] have already pointed out the differences between the malware dynamics in Windows and Linux. A major difference is the significantly larger amount of exploits in the ELF/Linux environment, such that if more AVs are all assigning exploit labels to the samples, it is harder to differentiate which AV produced that label.

4.5 What does explain the results?

We proceeded with our investigation to understand what causes AV labels to be easily separable. To that, we delve into the composition of the labels. Our empirical experience demonstrates that the same tokens appear in different positions according to the AV (e.g., `Gen.Trojan.Heur` and `Gen.Variant.Trojan`). We hypothesize that not all components (tokens) in an AV label contribute equally to the fingerprint process. To evaluate that, we repeated the previous attribution experiments to each individual label component.

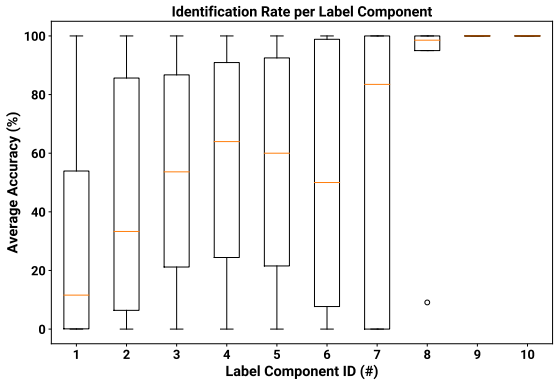

Fig. 24 Recognition rate by AV label component. The more specific, final labels are more discriminant than the initial, more generic ones.

Figure 24 shows the distribution of the accuracy rates of the attribution process for all AVs according to the individual components of their labels, considering only the cases when the components are available. We observe that the attribution rates are diverse over the AVs, ranging from very low to very high, which corroborates previous findings on the heterogeneity of AV behaviors. The graph shows that despite the huge variance, the average accuracy rate grows with the final label components. This happens because the final label components tend to be more specific, carrying information from that specific AV product, thus making it easy to differentiate AV labels based on them. For instance, the label for one of the tested AVs follows the format `a-variant-of-Win32-Bundled-Toolbar-Ask-G-potentially-unsafe`, which makes it unique.

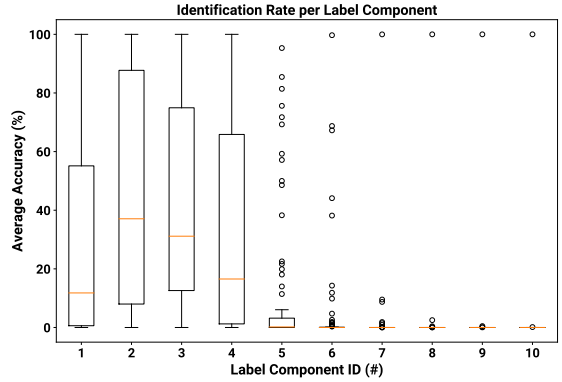

Fig. 25 Recognition rate by AV label component. The initial components provide an overall greater discrimination rate than the final label components.

The specificity of label composition, however, does not explain the whole attribution process. If it did, the labels from all AVs should be 100% separable only by using their specific components, which is not the case. It happens because not all AV labels have the same number of specific components. Long labels as the previously presented ones are in fact rare. Therefore, discriminant analyses must also consider the relative frequency in which these labels appear.

Figure 25 shows the distribution of the accuracy rates of the attribution process for all AVs according to the individual components of their labels, now considering all cases, being components available or not. The high variance is still present in the results, but now the accuracy does not grow with the label components in the overall case. In fact, the attribution capabilities of these final components tend to be zero. It happens because although these components are very discriminative when present, their relative presence is low.

The first 4 components are the ones that most affect the overall attribution process, with a peak caused by the second component. This result leads to two hypotheses. First, if the presence of components affects the AV identification, then the peak should be at the first component, which is present in all AVs, by definition. It does not happen exactly because many AVs present similar first components (e.g., many labels start with `HEUR`), thus complicating the attribution. A second hypothesis is the inverse: the valley should be at the first position if all labels were similar. It does not happen because some AVs are clearly recognized even using a single label, as they use very specific labels (e..g, `BehavesLike.Win32.Autorun.th`). Therefore, the maximum point of attribution is shifted to the second component, which is present in most of the labels and does not present as many common tokens as the initial one.

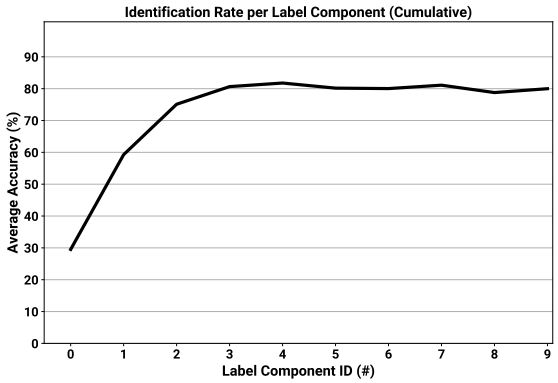

Fig. 26 Recognition rate by AV label component. Initially, the more components are considered, the greater the recognition rate. The growth stops after 4 components.

A final aspect to consider in the explanation is that the label components are not considered

individually, but associated. Therefore, new correlations might appear between the label components. We tested that by repeating the previously presented experiments in an incremental form–i.e., considering each time one additional label component in the experiment. Figure 26 shows how the accuracy of the attribution process varies with the increase in the number of considered label components. We notice that the label recognition grows with the first 4 label components, and presents a marginal increase with the remaining ones. It is possible to state that the accuracy rate achieves stability at the 80% value (remind that the recognition problem is more complex at this scale, as shown in previous experiments). The contribution given by the first 4 components can be explained by the use of the CARO naming system [4], which standardizes the 4 components of an AV label. Most AVs follow this convention, even though they heavily rely on extensions to it [3], causing the impacts we demonstrated in this work.

5 Summary and Implications

In this section, we present a summary of our findings to be used as a checklist for future AV label fingerprinting applications.

For AV label grouping:

- Two AVs tend to be always completely separable based on their assigned labels.
- AV labels can be grouped with 100% accuracy for small instances (less than 12 AVs).
- The grouping problem becomes exponential for more than 10K labels per AV.
- Reducing the targeted grouping accuracy to 95% leads to the best cost-benefit.

For random label sampling without confidence guarantees:

- 70% of the first sampled labels tend to be correct.

For label sampling with confidence guarantees:

- Less than 100 queries (less than 1% of the whole dataset) is enough to fingerprint AVs with 99% confidence.
- The problem becomes exponential for more than 5K labels per AV.
- The average value is dominated by the AV outliers.

For considering the effect of time:

- AV label updates over time moderately change the fingerprinting rates.
- AVs are getting moderately easier to fingerprint over time, which means their labels are differentiating even more.
- AV engine merging is reflected in the fingerprinting process. This is clearly observed for Avast and AVG.

For other OSes:

- Android AV labels tend to present a dynamic similar to Windows ones.
- Linux label dynamics are different from Windows and Android and are harder to fingerprint.

6 Discussion

In this section, we discuss the findings and limitations of our experiments to better position our work.

A note on confidence Analysis. Given the probabilistic nature of the AV selection process, we repeated the experiments until reaching the 95% confidence interval. The number of runs varies according to the number of combinations possible for each given set of AVs, i.e., it is possible to combine 72 AVs 2-by-2 in 2556 ways, but 72 AVs can only be combined 72-by-72 in a single way. On top of that, we used the own prediction result as feedback for the sampling, which reduces the number of runs required to achieve the 95% confidence interval in some cases. For instance, since the AVs were 100% separable in the 2-by-2 scenario, it reached the 95% confidence score according to the Wilson interval after only 385 runs.

What were we expecting from the AVs? Our results show that AV labels are easily separable. It means that the AVs do not agree on assigning the same labels to the same malware samples. This label heterogeneity is an undesired effect as it significantly complicates the incident response. For instance, if one searches for information about a malware sample labeled by one AV, one will find information only from the AV research team from that specific AV product, not from other sources that might provide better answers. Ideally,

we would like all AVs to agree on a sample label, which would have the side-effect of making fingerprinting harder (if not impossible), as seen for the ELF/Linux case. Unfortunately, the industry does not seem to be moving toward label uniformity. The most successful label handling approaches so far rely on trying to uniformize the diverse labels after their generation [5–7] than fixing their generation itself.

Does the separation between the labels of two AVs apply to any AV? Our results demonstrate that the labels coming from two different AVs are completely separable. This should not be interpreted as any two different AV solutions are separable, but that two different AV engines are separable. It is known by the literature that many AV solutions share the same AV engines [19], which would make their labels indistinguishable. These cases were not included in any of our experiments.

Contributions & Limitations. Whereas previous works already pointed out the label heterogeneity problem [2, 5, 25, 26], we are the first to invert the research point of view and attempt to fingerprint the AVs based on their assigned labels. The strong point of our work is the unique dataset of Windows samples, labeled for 14 consecutive years. Despite that, our work does not exhaust the subject, and experimental limitations must still be overcome. A major end of our study is the dataset used for APK/Android and ELF/Linux experiments. Whereas they are the best datasets we can provide at the moment, we know that specific phenomena can only be observed at a large scale, thus more studies are warranted.

How does one apply our findings? The discovery of the possibility of label fingerprinting AVs might foster new security applications, such as forensic analysts identifying which AV previously labeled a given set of samples–in case this information is absent (e.g., attributing from which legitimate AV a rogue AV got its labels). Despite this positive side effect, this research does not aim to have immediate applications. Instead, our goal is to present a new view of the label heterogeneity problem. In this sense, the most important finding of our research is to show that the problem is bigger than originally described in the literature, as we demonstrate it is a two-way problem: it happens from AVs to samples and also from samples to AVs. Thus, we expect our insights (e.g., our observations of the most diverse label components) to support future developments on the normalization of AV labels.

Privacy Implications. Using our proposed approach to fingerprint AVs might disclose the defensive strategy of given individuals and/or organizations. However, we do not consider our solution as increasing the privacy risks one is subject to because this same risk is already present in existing fingerprinting mechanisms, such as network scanners, that can be used against public and private networks.

Future Work. We will break down our investigation result to the level of which malware families are harder to classify and thus fingerprint the AVs from them. Previous research has pointed out that labels vary significantly according to the family [27], which motivates further investigations.

7 Related Work

In this section, we present related work on antivirus studies to better position our contributions. Since AVs are well-studied in the literature, we focus on the proposals that allow us to better differentiate our work from previous contributions.

Understanding AV internals is key to having proper knowledge of AV operation. The literature has presented previous work that delves into the AVs implementation details and project decision implications [19]. Whereas this knowledge is fundamental, in this work, we address AV issues at a higher level, looking only at the labels and treating their respective generating engines as black boxes.

Evaluating AVs is another key security task essential to ensure that one is running a system at the highest protection level possible. The literature is rich on AV evaluations, that even propose new metrics to evaluate them [3]. These metrics are important to evaluate AVs in a multi-dimensional perspective, accounting for effects such as regression—when a sample stops being detected after some time [28]. In this paper, we evaluate AVs not for their detection capability, but by their label diversity, complementing previous studies.

The label diversity phenomena has been previously observed in the practice and it is reported

in the literature. Previous work pointed out, for instance, that the use of AV non-agreeing labels may even decrease AV classification accuracy [29]. Whereas previous work presented a complete treatment of the subject from the sample point of view–i.e., observing how AVs assign different labels to the same sample–we here approach the problem from the AV point of view–i.e., verifying if labels are diverse from the AV generation perspective.

Label normalization is an important step to overcome the label diversity phenomena. Multiple solutions are described in the literature to normalize AV labels to an agreeing one (e.g., AVClass [5], AVClass2 [30], and Euphony [31]). In this work, we take the opposite direction. We explore label diversity to fingerprint their generating AV engines.

Label correlation is the process of identifying that similar labels might have been generated by the same AV. Previous work demonstrates that it is possible to observe when different AV companies merge their engines only by observing the label dynamics [20]. In this work, we extend this concept to fingerprint AV engines in general.

8 Conclusions

We investigated the question: *Can AVs be fingerprinted based on the labels that they assign?* We performed experiments with a dataset of 720000 AV-assigned labels for Windows malware spread over 15 years (2006-2020) and discovered that AVs can be fingerprinted by their assigned labels. In most cases, it is possible to achieve 100% group accuracy. AVs can be fingerprinted with a confidence score of 99% using only 1% of the dataset. We show that these results vary over time, as the label changes caused by the AV updates have a key effect on AV recognition. In extreme cases, some AVs lose their ability to recognize their own generated labels over time. We also present results to show that Android AVs can be fingerprinted the same way as Windows AVs, but that Linux labels are harder to be grouped.

Reproducibility. All code developed for this research work is available at https://github.com/marcusbotacin/AV.Label.Uniqueness

Declarations

This manuscript is submitted for publication in accordance with Springer's ethical guidelines for research publication. The author affirms that:

Author Contributions and Originality: All authors have made significant contributions to the conception, design, execution, or interpretation of the research, and the work presented is original, free from plagiarism, and has not been submitted elsewhere for publication.

Data Integrity and Availability: All data generated or analyzed during this study will be made available upon request.

Human and Animal Rights: It does not apply.

Informed Consent: It does not apply.

Acknowledgment of Funding Sources: Marcus Botacin thanks NSF for the support via the CNS 2327427 grant.

Appendix A Individual AV results

All over this paper, we focused our results on the average AV–i.e., on how AVs behave on average, without individualizing AV results. In some cases, however, having individual AV information is desired (e.g., for one selecting an AV to perform experiments similar to the ones reported in this paper). Thus, to streamline reproducibility, we here report individual AV results.

Table A1 classifies the 72 AVs considered in this paper in three different groups: (i) the ones of low complexity, i.e., those which can be fingerprinted with a number of attempts in the magnitude order of individual queries; (ii) the ones of medium complexity, i.e., those which can be fingerprinted with a number of attempts in the magnitude order of dozen queries; and (iii) the ones of high complexity, i.e., those which can be fingerprinted with a number of attempts in the magnitude order of hundred queries.

We notice that, as previously reported for the general case, the majority of AVs require a few queries to be fingerprinted. We believe the obtained results are coherent, as AV products that share the same AV engine are grouped together. For instance, we observe that (i) AVG and Avast; (ii) multiple McAfee versions; and (iii) VIPRE and AVWare are grouped in the same complexity level.

Table A1 Individual AV queries. AVs grouped according to the average number of queries required to fingerprint them. Most AVs can be fingerprinted with a few queries.

AVs				Difficulty Level	Magnitu
AhnLab-V3	Antiy-AVL	APEX	Arcabit	Few	1s
Avira	BitDefenderTheta	ClamAV	CrowdStrike		
Cybereason	Cylance	Cynet	eGambit		
Elastic	Emsisoft	Endgame	ESET-NOD32		
Ikarus	NOD32	Paloalto	Panda		
PCTools	SentinelOne	Symantec	Zillya		
AegisLab	Avast	AVG	Bkav	Moderate	10s
CAT-QuickHeal	CMC	Comodo	Cyren		
DrWeb	FireEye	Fortinet	F-Prot		
F-Secure	GData	K7AntiVirus	K7GW		
Kingsoft	Malwarebytes	McAfee	McAfee-GW-Edition		
Microsoft	Qihoo-360	Sangfor	Sophos		
SUPERAntiSpyware	TheHacker	TotalDefense	VBA32		
ViRobot	Webroot	ZoneAlarm	Zoner		
AntiVir	Ad-Aware	ALYac	AVware	Many	100s
Baidu-International	BitDefender	Kaspersky	MicroWorld-eScan		
NANO-Antivirus	Norman	nProtect	TACHYON		
TrendMicro	TrendMicro-HouseCall	VIPRE	Yandex		

References

[1] Salem, A., Banescu, S. & Pretschner, A. Maat: Automatically analyzing virustotal for accurate labeling and effective malware detection. *ACM Transactions on Privacy and Security (TOPS)* (2021).

[2] Hurier, M., Allix, K., Bissyandé, T. F., Klein, J. & Le Traon, Y. On the lack of consensus in anti-virus decisions: Metrics and insights on building ground truths of android malware (2016).

[3] Botacin, M., Ceschin, F., de Geus, P. & Grégio, A. We need to talk about antiviruses: challenges & pitfalls of av evaluations. *Computers & Security* **95**, 101859 (2020). URL https://www.sciencedirect.com/science/article/pii/S0167404820301310.

[4] CARO. A new virus naming convention. http://www.caro.org/articles/naming.html (1991).

[5] Sebastián, M., Rivera, R., Kotzias, P. & Caballero, J. Avclass: A tool for massive malware labeling (2016).

[6] Kantchelian, A. *et al.* Better malware ground truth: Techniques for weighting anti-virus vendor labels (2015).

[7] Chen, L., He, Z., Wu, H., Gong, Y. & Mao, B. Avminer: Expansible and semantic-preserving anti-virus labels mining method. *arXiv preprint arXiv:2208.14221* (2022).

[8] Pirscoveanu, R.-S., Stevanovic, M. & Pedersen, J. M. Clustering analysis of malware behavior using self organizing map (2016).

[9] Demetrio, L., Biggio, B., Lagorio, G., Roli, F. & Armando, A. Functionality-preserving black-box optimization of adversarial windows malware. *IEEE Transactions on Information Forensics and Security* **16**, 3469–3478 (2021).

[10] Demontis, A. *et al.* Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks (2019).

[11] Apruzzese, G. *et al.* "real attackers don't compute gradients": Bridging the gap between adversarial ml research and practice (2023).

[12] Sharma, Y., Giunchiglia, E., Birnbach, S. & Martinovic, I. To ttp or not to ttp?: Exploiting ttps to improve ml-based malware detection (2023).

[13] Wang, J. *et al.* Mal-lsgan: An effective adversarial malware example generation model (2021).

[14] Bostani, H. & Moonsamy, V. Evadedroid: A practical evasion attack on machine learning for black-box android malware detection. *Computers & Security* **139**, 103676 (2024).

[15] Botacin, M. *et al.* One size does not fit all: A longitudinal analysis of brazilian financial malware. *ACM Trans. Priv. Secur.* **24** (2021). URL https://doi.org/10.1145/3429741.

[16] Nowroozi, E., Jadalla, N., Ghelichkhani, S. & Jolfaei, A. Mitigating label flipping attacks in malicious url detectors using ensemble trees. *IEEE Transactions on Network and Service Management* 1–1 (2024).

[17] Taheri, R. *et al.* On defending against label flipping attacks on malware detection systems. *Neural Comput. Appl.* **32**, 14781–14800 (2020). URL https://doi.org/10.1007/s00521-020-04831-9.

[18] Gashi, I., Sobesto, B., Mason, S., Stankovic, V. & Cukier, M. A study of the relationship between antivirus regressions and label changes (2013).

[19] Botacin, M. *et al.* Antiviruses under the microscope: A hands-on perspective. *Computers & Security* **112**, 102500 (2022). URL https://www.sciencedirect.com/science/article/pii/S0167404821003242.

[20] Zhu, S. *et al.* Measuring and modeling the label dynamics of online anti-malware engines (2020). URL https://www.usenix.org/conference/usenixsecurity20/presentation/zhu.

[21] Monika, Zavarsky, P. & Lindskog, D. Experimental analysis of ransomware on windows and android platforms: Evolution and characterization. *Procedia Computer Science* **94**, 465–472 (2016). URL https://www.sciencedirect.com/science/article/pii/S1877050916318221. The 11th International Conference on Future Networks and Communications (FNC 2016) / The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016) / Affiliated Workshops.

[22] Darabian, H. *et al.* A multiview learning method for malware threat hunting: windows, iot and android as case studies. *World Wide Web* **23**, 1241–1260 (2020).

[23] Cozzi, E., Graziano, M., Fratantonio, Y. & Balzarotti, D. Understanding linux malware (2018).

[24] Galante, L., Botacin, M., Grégio, A. & de Geus, P. Malicious linux binaries: A landscape. https://www.lasca.ic.unicamp.br/paulo/papers/2018-SBSeg-WTICG-lucas.galante-marcus.botacin-malicious.linux.binaries.pdf (2018).

[25] Mohaisen, A. & Alrawi, O. Av-meter: An evaluation of antivirus scans and labels (2014).

[26] Martínez Torres, J., Iglesias Comesaña, C. & García-Nieto, P. J. Machine learning techniques applied to cybersecurity. *International Journal of Machine Learning and Cybernetics* (2019).

[27] Joyce, R. J., Amlani, D., Nicholas, C. & Raff, E. Motif: A malware reference dataset with ground truth family labels. *Computers & Security* **124**, 102921 (2023).

[28] Gashi, I., Sobesto, B., Mason, S., Stankovic, V. & Cukier, M. A study of the relationship between antivirus regressions and label changes (2013).

[29] Carlin, D., Cowan, A., O'Kane, P. & Sezer, S. The effects of traditional anti-virus labels on malware detection using dynamic runtime opcodes. *IEEE Access* **5**, 17742–17752 (2017).

[30] Sebastián, S. & Caballero, J. Avclass2: Massive malware tag extraction from av labels (2020). URL https://doi.org/10.1145/3427228.3427261.

[31] Hurier, M. *et al.* Euphony: Harmonious unification of cacophonous anti-virus vendor labels for android malware (2017).