Better Malware Ground Truth: Techniques for Weighting Anti-Virus Vendor Labels

Alex Kantchelian
UC Berkeley

Michael Carl Tschantz
International Computer Science Institute

Sadia Afroz
UC Berkeley

Brad Miller
UC Berkeley

Vaishaal Shankar
UC Berkeley

Rekha Bachwani
Netflix[*]

Anthony D. Joseph
UC Berkeley

J. D. Tygar
UC Berkeley

ABSTRACT

We examine the problem of aggregating the results of multiple anti-virus (AV) vendors' detectors into a single authoritative ground-truth label for every binary. To do so, we adapt a well-known generative Bayesian model that postulates the existence of a hidden ground truth upon which the AV labels depend. We use training based on Expectation Maximization for this fully unsupervised technique. We evaluate our method using 279,327 distinct binaries from VirusTotal, each of which appeared for the first time between January 2012 and June 2014.

Our evaluation shows that our statistical model is consistently more accurate at predicting the future-derived ground truth than all unweighted rules of the form "k out of n" AV detections. In addition, we evaluate the scenario where partial ground truth is available for model building. We train a logistic regression predictor on the partial label information. Our results show that as few as a 100 randomly selected training instances with ground truth are enough to achieve 80% true positive rate for 0.1% false positive rate. In comparison, the best unweighted threshold rule provides only 60% true positive rate at the same false positive rate.

1. INTRODUCTION

Machine learning provides scalable mechanisms for detecting malware. However, the success of a machine learning based system relies on availability of accurate labels for the training data: prior work has shown that the detection accuracy of a learning system can decrease significantly if the training data is faulty [2, 5, 19, 28, 34] or adversarially corrupted [4]. Unfortunately, in the real world, executable samples often come without trustworthy labels due to the time and expense of manual labeling. In particular, because of the rapidly changing nature of malware, large datasets of executables cannot both have high-confidence manually verified labels and be up-to-date. For example, McAfee received approximately 344,000 unique malware instances per day in the second quarter of 2014 [25].

A common route for obtaining the ground truth about maliciousness for samples is to rely on anti-virus (AV) engines. For instance, the online service VirusTotal [62] accepts any file submission, runs a set of engines provided by anti-virus vendors, and reports on all of the AV decisions. In this paper, we examine the problem of inferring the ground-truth label of an executable instance from the multiple and often conflicting individual AV vendor decisions. In particular, we study and evaluate techniques for assigning individual weights to vendors. In this model, the estimated ground truth is malware if and only if the weighted sum of vendors' (binary) decisions is larger than a specified threshold.

Most prior works assign equal weights to vendor labels, a strategy that we call using an *unweighted threshold.* However, vendors vary in accuracy. Therefore, we propose an unsupervised and a supervised technique for assigning weights to vendors that reflect the vendors' accuracies.

In the unsupervised technique, we take a generative Bayesian model that has proven successful in crowd sourcing (e.g., [10, 40]) and apply it to malware labeling for the first time. The model assumes that each vendor's accuracy is unvarying across various types of executables and that each vendor operates independently of the others. (More formally, it assumes (1) the independence of vendors and instances and (2) independence between vendors given the hidden ground-truth status of the executables.) Despite AV vendors actually having varying detection performance on sub-categories of malware and communicating results with each other, we find that these assumptions simplify the learning and inference aspects of the model and produce acceptable accu-

[*]This author was at Intel Labs while doing most of her work on this paper.

AISec'15, October 16 2015, Denver, CO, USA
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3826-4/15/10$15.00.
http://dx.doi.org/10.1145/2808769.2808780

racy. To adapt the model for malware detection, we encode our problem specific knowledge that vendors favor low false-positive rates and tend to exhibit large false-negative rates by equipping the model parameters with asymmetric Beta priors. These priors act as regularizers for the error rates of the vendors, enforcing both low false positive and high false negate rates. The model can subsequently be trained using Expectation Maximization without requiring any ground truth [10].

In the supervised technique, we allow the learner limited access to ground truth by revealing the ground-truth labels of a small number of instances. We subsequently train a logistic regression model on this labeled data to find a suitable weighting scheme. Both techniques are presented in section 3.

To evaluate the learned models, we use a trustworthy testing dataset based on VirusTotal historical AV scan reports between January 2012 and June 2014, which we construct from a larger dataset [27]. We carefully evaluate our learning approaches so as to respect the temporal ordering of executables. Our dataset is based on the *first-seen* AV scans of 280,000 executables. To create the evaluation ground truth for these instances, we retrieve or otherwise trigger re-scans with updated AV scanners. Our rational is to give enough time to the AV vendors to catch up on their previous false negatives and to a lesser extent false positives. In essence, using historical data enables us to perform time traveling in the future and see what the community consensus will later be like for a completely novel instance. We temporally split this dataset into a training and testing part such that all first-seen dates of the training instances predate the first-seen dates of the testing instances. This ensures temporal consistency, a more realistic and stringent type of evaluation than purely random cross-validation. We carefully explore the dataset to justify our approach to ground-truth labeling in section 4.

Our evaluation shows that both of our label aggregation algorithms outperform the unweighted threshold aggregator. The unsupervised aggregator typically outperformed the unweighted threshold approach by 4 percent points in true positive rate for the same false positive rates obtained by varying the number of detections required in the unweighted threshold method. This improvement shows that simply acknowledging the possibilities of differences in vendor quality is sufficient to improve the labels. The supervised aggregator further improved the true positive rate by 20 percentage points around the 0.1% false positive region, an achievement possible with only a hundred labeled instances. This further improvement shows the value of having a small number of high-confidence labels to rate and prioritize vendors. We present more detailed evaluation results in section 5.

Contributions. We make the following contributions:

1. We study the practice of ground-truth labeling for malware detection using machine learning.

2. Using a large dataset, we measure the change of AV vendor labels over time, and estimate the time required for labels to stabilize.

3. We evaluate two machine learning algorithms for resolving noisy labels in the malware domain.

While learning from noisy labels is a well-studied problem in machine learning, to the best of our knowledge, we are the first to show that unsupervised and supervised models can improve the ground-truth labels for malware detection.

2. RELATED WORK

Ground-truth Approaches for Virus Detection. There is a large body of research on malware detection. Table 1 summarizes a variety of studies on malware detection and their approaches for constructing ground truth. Prior work used four approaches of assigning ground-truth labels for their datasets, each with downsides: 1) label data manually, 2) use labels from a single source, 3) use labels from a single anti-virus vendor and 4) use labels from multiple anti-virus vendor.

First, some manually analyzed their datasets to create their own high confidence labels, an approach that does not scale to large datasets.

Second, the majority of the prior work collected ground-truth labels from well-known malicious and benign file collections, for example, Anubis, VX Heavens, Wildlist Collections and Malfease dataset [1,7,13,15,20,22–24,26,41,43,45–47,49,50,57,59–61]. For benign files, popular data sources are SourceForge and the system files from Windows operating systems. This approach has at least three downsides when using the data for evaluation purposes. Firstly, the diversity of the obtained samples can be very low, a problem especially true for benign samples from the Windows OS. Malware repositories also tend to be small or medium in size, as every malicious label is manually verified. Secondly, and for the same manual verification reasons, the malicious executables are often old. This has the unfortunate effect of evaluating 2014 work on pre-2006 VX Heaven malware samples [31]. Thirdly, the list curators are likely to include only the most clear cut cases (such as Window OS files). Thus, relying upon such lists could inflate a classifier's accuracy by avoiding difficult cases.

Third, the second most popular approach for collecting ground-truth labels is consulting a single anti-virus vendor. Prior work collected labels from Kaspersky [32], KingSoft [65,66], Norton [54], McAfee [14,51] and Microsoft [8]. Using labels from a single vendor can bias the dataset towards detection pattern of that vendor. Furthermore, prior research [2, 5,19,21,28,34,53] showed and we also observe in our own data (fig. 3) that vendors tend to disagree on the labels. To avoid biasing our labels to a single vendor, we collect labels from multiple vendors (≈ 80) participating on VirusTotal, and analyze various approaches to derive a single aggregate label from many vendors.

Fourth, to overcome the problem of single source bias, some studies used the labels from more than one anti-virus vendor to decide instance labels. In particular, they use a threshold for the number of positive signals to decide whether a binary is malicious. Perdisci et al. [35] use the decision of three particular anti-virus vendors (ClamAV, F-Prot, and AVG). Laskov et al. [16] consider a file as malicious if at least 5 AV from VirusTotal detect it as malicious but consider a file as benign if all the AV vendors label it as benign.

Typically, AV vendors tend to be conservative in declaring a binary malicious, preferring false negatives over false positives. However, these approaches differ in how they label binaries as benign. Some simply label all binaries not meeting the threshold for maliciousness as benign, while others use a threshold for benign labels too.

While this second approach can result in higher quality labels, it effectively removes the most difficult to classify binaries from the dataset, which could overestimate the accuracy of algorithms evaluated with it [17]. A limitation of both approaches is that it treats all anti-virus vendors the same despite each having their own false-positive and false-negative rates.

Inconsistencies in AV labels. AV vendors specialize in different types of malware, and thus differ in the types of malware that they most accurately detect. Different vendors use different techniques to detect and analyze malware, and to determine the malware family of different malware instances. These different methods sometimes give different results. As a result, there are widespread inconsistencies among the labels from different AV vendors.

Several prior studies have pointed out inconsistencies in labels across AV vendors [2, 5, 19, 21, 28, 34, 53]. Mohaisen et al. [28] systematically study the inconsistencies among the labels of malware vendors and the impact of these inconsistencies on the overall detection rates using 12K malicious instances. They show that the detection rate of AV vendors varies based on family of the malware being detected, and that an AV engine is consistent (on an average) with other AV engines only about 50% of the time. Bailey et al. performed a similar study on the labels of 5 AV vendors: McAfee, F-Prot, ClamAV, Trend and Symantec [2]. They discovered that 25% of the new malware and around 1% of one year old malware were undetected by some of these five AV vendors.

Our work also finds inconsistency in labeling across AV vendors, but our contributions are significantly different than prior work. We focus on malware detection that is distinguishing malware from benign files, instead of malware clustering which focuses on distinguishing different malware families. We propose a systematic approach to aggregate labels from myriad AV vendors to better predict the future labels for the training set.

Solution for inconsistent labels. Deriving a single accurate label from multiple noisy labels is a common problem when working with real world data. The quality of the label can have significant impact on classification accuracy [53]. The Expectation Maximization (EM) algorithm in conjunction with a generative Bayesian model has been used to estimate the error-rates of the labelers in various crowd-sourced labeling tasks with noisy labels, for example, text classification [29, 38], and image labeling [10, 18, 40, 64]. Whitehill et al. [64] proposed a generative Bayesian model for the labeling process that includes the labelers' areas of expertise, adversarial labelers, and the difficulty of the labeling task. Welinder et al. [63] extended the model by considering labelers' bias. Other work used different models, such as approximate variational methods, to rank labelers [18, 39]. Closely related to the generative Bayesian model we use in our work, Raykar et al. [40] proposed a discriminative approach to learn from the noisy labels of multiple experts. One way that our work differs from is that we only use the generative part of the model, i.e., we do not include additional features outside of the AV labels themselves, and we further constrain the Beta priors to be asymmetrical. We do so to model our prior domain knowledge that vendors have both low false and true positive rates. We also mention two additional models.

Delayed Labeling for Ground Truth. We now discuss work which either incorporates or measures the delay in availability of accurate evaluation labels for malicious content. Evaluations may incorporate labeling delay by inserting a waiting period between the collection and labeling of evaluation data. Some prior work allows a fixed delay for labeling evaluation data, while other work periodically updates labels until labels cease to change. Outside of the context of an evaluation, work may directly measure vendor delay in detecting malicious binaries. Table 2 presents an overview of prior work involving label delay; we discuss each work individually to provide greater insight into current practice and knowledge.

We begin by reviewing work using family labels to distinguish various types of malicious content. Bailey et al. present an unsupervised clustering approach for detection of similar malware instances. Rieck et al. present a supervised approach to detection of new variants of malware families [42]. In further work, Rieck et al. integrate unsupervised methods for the detection of new malware families [44].

Separate from work utilizing labels to distinguish families of malware, prior work also demonstrates a delay in binary labels distinguishing benign and malicious content. Smutz et al. study detection of malicious PDF documents and collect evaluation data over a period of one week [55]. Rajab et al. present an approach to malware detection using instance metadata and evaluate the approach against labels from a proprietary malware detector. Using the same data source as we use to validate their malware detector, Rajab et al. select 2,200 binaries from a single day that are not found in VirusTotal and submit the binaries to VirusTotal [37]. After 10 days, VirusTotal rescans the 2,200 binaries to provide an evaluation labeling for the private malware detector. The evaluation makes no claim that all malware detection occurs in the 10 day period, and does not rescan instances later to check for new detections. Miller et al. use a superset of the executables we use in this paper to evaluate a malware detector [27]. For executables that were not clearly malicious (10 or more positive scans), they use a delay of at least 7 months to create ground-truth labels. They do not explore techniques for weighting anti-virus vendor labels and just use an unweighted threshold.

Ground-truth approach	**Work**
Manual labeling	Chau et al. 2011 [6], Overveldt et al. 2012 [33], Schwenk et al. 2012 [52]
Collection	Kolter et al. 2006 [15], Reddy et al. 2006 [41], Masud et al. 2007 [23], Stolfo et al. 2007 [57], Masud et al. 2008 [24], Gavrilut et al. 2009 [13] , Tabish et al. 2009 [59], Menahem et al. 2009 [26], Schmidt et al. 2009 [50] , Tian et al. 2009 [61] Santos et al. 2010 [46], Rieck et al. 2010 [43], Alazab et al. 2011 [1], Curtsinger et al. 2011 [7], Santos et al. 2011 [47], Maiorca et al. 2012 [20], Sahs et al. 2012 [45], Sanz et al. 2012 [49], Tahan et al. 2012 [60], Markel et al. 2014 [22]
Single AV	Schultz et al. 2001 [51], Shih et al. 2005 [54], Henchiri et al. 2006 [14], Ye et al. 2008 [66], Ye et al. 2009 [65], Nissim et al. 2014 [32], Santos et al. 2009 [48], Dahl et al. 2013 [8], Stringhini et al. 2013 [58]
Multiple AVs with threshold	Laskov et al. 2011 [16], Gascon et al. 2013 [12], Perdisci et al. 2008 [35], Šrndic et al. 2013 [56], Nissim et al. 2014 [30]

Table 1: The ground-truth labeling approach of prior work in file-based malware detection. The majority of the prior work collects data and label from well-known malware collections, such as VX Heavens.

Author	Year	Objective	Collection Period	Delay	# Vendors
Bailey et al. [3]	2007	Family Name	7 Months	13 Days	5
Rieck et al. [42]	2008	Family Name	5 Months	4 Weeks	1
Perdisci et al. [36]†	2010	Detection	6 Months	<1 Month	1
Rieck et al. [44]	2011	Family Name	7 Days	8 Weeks	1
Smutz et al. [55]	2012	Detection	1 Week	10 Days	5
Rajab et al. [37]	2013	Detection	1 Day	10 Days	45
Damballa et al. [9]	2014	Detection	10 Months	6 Months	4
Miller et al. [27]	2015	Detection	30 Months	7 Months	32
This paper	2015	Detection	30 Months	$\geq$ 4 Weeks	34

†Collection occurs from February to July 2009 with detection in August 2009. Perdisci et al. report monthly detection results for the best of three vendors included in the evaluation, demonstrating a decrease in detections in more recent months but not claiming labels are stabilized in any month.

Table 2: Prior work recognizing or examining the delay required for accurate vendor labels. *Objective* indicates the type of label required by the work, with *Family Name* corresponding labels distinguishing various types of malicious behavior, and *Detection* corresponding to works distinguishing malicious from benign content.

Prior work also approaches detection latency from the perspective of evaluating vendors rather than evaluating a detector. As part of a study identifying malware generating HTTP traffic, Perdisci et al. collect a dataset comprised entirely of known malicious instances from February to July 2009 and scan all instances using detectors from three vendors in August 2009 [36]. In a similar approach, Damballa collects instances from January to October 2014 and regularly scans the instances with four vendor products [9]. Both of the work collected useful statistics about anti-virus vendors' detection rate over time. Our work differs from them as we explore different statistical models to weight vendor to improve ground-truth labels.

3. AGGREGATING AV LABELS

We now describe three techniques for aggregating multiple anti-virus vendor labels into a single authoritative label. Let N be the number of instances and M be the number of vendors. We are provided with an $N \times M$ binary matrix X where each entry $X_{i,j}$ represents the label assigned by vendor j to instance i. All three techniques produce aggregation functions which belong to the class of thresholded weighted sum of individual vendors decisions: they assign a positive label to instance i if and only if $\sum_{j=1}^{M} c_j X_{i,j} > \tau$ for fixed model-dependent coefficients $(c_j)_{j \le M}$ and threshold τ.

We present two techniques that *do not* require any additional data aside from the vendors' labels X. We borrow an unsupervised generative Bayesian model from the crowd sourcing literature [10] and equip it with suitable priors. The model attempts to learn vendors' true and false positive rates in the absence of ground truth. We describe an additional simple supervised classification strategy which assumes the availability of at least partial ground-truth labels.

3.1 Unweighted Threshold

The unweighted threshold technique assigns equal weight to all vendors: $c_1 = \cdots = c_M = 1$. The advantage of this technique is that it does not require any learning from X. The only free parameter is τ, which becomes a tunable hyperparameter specifying the required level of agreement. The downside of unweighted threshold is that it assigns an equal importance to all anti-virus vendors. This equal importance can be problematic in several plausible scenarios. First, if one or more vendors have a significant false positive rate, the unweighted threshold decision for small τ values will also suffer a large false positive rate. Second, the unweighted threshold is unable to take advantage of vendors with very low false positive rates. Indeed, if vendor k has exactly zero false positives, then a strictly better aggregation model can be obtained by setting $c_k = \infty$.

3.2 Unsupervised Learning: Generative Model

Even in the absence of ground-truth labels, it is possible to improve the unweighted scheme. Consider the case of three independent vendors, two of which have zero false positive rate and 50% true positive rate, the remaining vendor is pure random noise with 50% true and false positives. Even without looking at the ground truth, the first two vendors will tend to agree much more often between themselves than with the third vendor, which behaves erratically. Given the independence of vendors assumption, the best (maximum likelihood) explanation for this is that the first two vendors are more correlated with the hidden ground truth than the last vendor, and should thus receive larger c weights in the aggregation function. The generative model which follows captures and refines this intuition.

We now describe a simple generative Bayesian model, for recovering the hidden ground-truth labels from X. Figure 1 presents the model in plate notation. The main simplifying assumption for our generative model is that given the ground-truth label of an instance, any two vendor labels are independent. Thus, this model does not account for vendor collusion which can for example happen when vendors share their findings with each other. This approach can be refered to as a Bayesian naive Bayes.

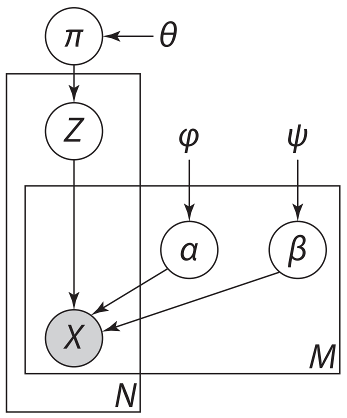


Figure 1: Generative model in plate notation. θ, φ, ψ are constant hyperparameters, π, Z, α, β are hidden random variables, and X is the only observed random variable. There are N instances and M vendors.

In this model, $(X_{i,j})_{i \le N, j \le M}$ is the only observed data. The label $X_{i,j} \in \{0,1\}$ of vendor j on instance i depends on the hidden ground-truth label $Z_i \in \{0,1\}$ for instance i and the vendor's true positive and false positive rates $\alpha_j \in [0,1]$ and $\beta_j \in [0,1]$ respectively. Formally, we have:

$$p(X_{i,j}|\alpha_j, \beta_j, Z_i) = \begin{cases} \alpha_j & \text{if } Z_i = 1 \text{ and } X_{i,j} = 1 \text{ (TP)} \\ 1 - \alpha_j & \text{if } Z_i = 1 \text{ and } X_{i,j} = 0 \text{ (FN)} \\ \beta_j & \text{if } Z_i = 0 \text{ and } X_{i,j} = 1 \text{ (FP)} \\ 1 - \beta_j & \text{if } Z_i = 0 \text{ and } X_{i,j} = 0 \text{ (TN)} \end{cases}$$

The hidden ground-truth label Z_i is taken to be a Bernoulli distribution with unobserved parameter $\pi \in [0,1]$. We have:

$$p(Z_i|\pi) = \pi$$

Finally, we equip this model with three priors on the distribution parameters π, α, β which serve both as regularization devices and encode our prior knowledge for this domain. We regularize π by a symmetric Beta distribution $\text{Beta}(1+\theta, 1+\theta)$ for $\theta \ge 0$. When $\theta = 0$, this corresponds to a uniform prior over $[0,1]$ on the value of π. As θ increases, the prior's mass gets concentrated around $\frac{1}{2}$. All αs and βs are regularized by the same hyperparameters φ and ψ respectively. Since our prior domain knowledge is that the vendors true positive and false positive rates are both expected to be low, we choose the asymmetric Beta priors

$\text{Beta}(1, 1+\varphi)$ and $\text{Beta}(1, 1+\psi)$. Figure 2 shows the effect of different values of φ on $p(\alpha|\varphi)$. Large values of φ correspond to shifting the prior mass towards 0. We have:

$$p(\pi|\theta) = \text{Beta}(\theta+1, \theta+1) \propto \pi^{\theta}(1-\pi)^{\theta}$$
$$p(\alpha_j|\varphi) = \text{Beta}(1, \varphi+1) \propto (1-\alpha_j)^{\varphi}$$
$$p(\beta_j|\psi) = \text{Beta}(1, \psi+1) \propto (1-\beta_j)^{\psi}$$

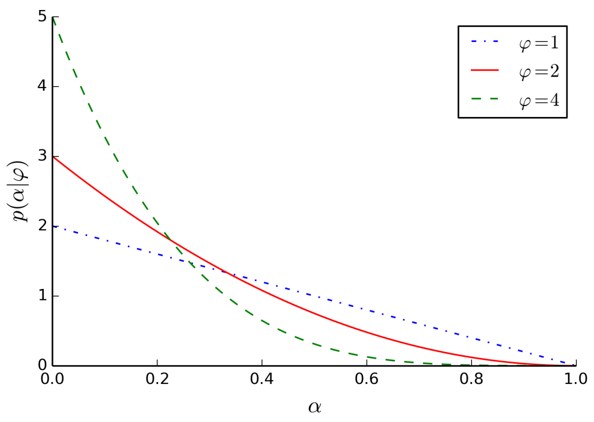


Figure 2: Asymmetric $p(\alpha|\varphi) = \text{Beta}(1, \varphi+1)$ prior densities on the vendors true positive rates αs. Larger values of φ favor smaller true positive rates.

Inference. When the parameters α, β, π are known, inferring the most likely ground-truth label Z_i out of the M labels $X_{i,1}, \ldots, X_{i,M}$ reduces to a simple Naive Bayes. We have:

$$p(Z=1|X,\alpha,\beta,\pi) = \frac{p(X|Z=1,\alpha,\beta)p(Z=1|\pi)}{\sum_{z\in\{0,1\}} p(X|Z=z,\alpha,\beta)p(Z=1|\pi)}$$
$$= \left[1 + \frac{1-\pi}{\pi}\prod_{j\leq M} a_j^{X_{:,j}} b_j^{1-X_{:,j}}\right]^{-1} \quad (1)$$

where $a_j = \frac{\beta_j}{\alpha_j}$ and $b_j = \frac{1-\beta_j}{1-\alpha_j}$ are the vendors likelihood ratios. Here we use the slicing notation $X_{:,j}$ to refer to column j of matrix X, and take all arithmetic operations to be pointwise.

Learning. For learning, we use a standard log-likelihood maximization strategy. Because we have the unobserved variable Z, we use the Expectation Maximization (EM) iterative method [10] for maximizing the log likelihood. Suppose that the ground-truth data Z is observed. The log likelihood $\mathcal{L}_{\theta,\varphi,\psi,X,Z}$ of the model would then be:

$$\mathcal{L}_{\theta,\varphi,\psi,X,Z}(\alpha,\beta,\pi) = \ln p(X,Z|\alpha,\beta,\pi,\theta,\varphi,\psi) \quad (2)$$
$$= \ln p(Z|\pi) + \ln p(\pi|\theta) + \ln p(X|\alpha,\beta,Z) + \ln p(\alpha|\varphi) + \ln p(\beta|\psi)$$

One can further completely separate the log likelihood into three parts:

$$\mathcal{L}_{\theta,\varphi,\psi,X,Z} = \mathcal{L}^{C}_{\theta,Z}(\pi) + \mathcal{L}^{A}_{\varphi,X,Z}(\alpha) + \mathcal{L}^{B}_{\psi,X,Z}(\beta)$$

Dropping the constant subscripts, each part is defined up to a constant additive term as:

$$\mathcal{L}^{C}(\pi) = \sum_i Z_i \ln\pi + (1-Z_i)\ln(1-\pi) + \theta\ln\pi + (1-\theta)\ln(1-\pi)$$
$$\mathcal{L}^{A}(\alpha) = \sum_i Z_i \sum_j X_{i,j}\ln\alpha_j + (1-X_{i,j})\ln(1-\alpha_j) + \varphi\sum_j \ln(1-\alpha_j)$$
$$\mathcal{L}^{B}(\beta) = \sum_i (1-Z_i) \sum_j X_{i,j}\ln\beta_j + (1-X_{i,j})\ln(1-\beta_j) + \psi\sum_j \ln(1-\beta_j)$$

Maximizing $\mathcal{L}$ yields the following regularized "counting" estimators for α, β, π:

$$\pi \leftarrow \frac{\theta + \sum_i Z_i}{2\theta + N} \quad (3)$$
$$\forall j, \alpha_j \leftarrow \frac{\sum_i Z_i X_{i,j}}{\varphi + \sum_i Z_i} \quad (4)$$
$$\forall j, \beta_j \leftarrow \frac{\sum_i (1-Z_i) X_{i,j}}{\psi + \sum_i (1-Z_i)} \quad (5)$$

The idea behind EM is to replace each occurrence of the hidden variable Z in the log likelihood (2) by its expectation $\mathbb{E}[Z|X,\alpha,\beta,\pi]$ for given fixed values of α, β, π. Because Z is (a vector of) Bernoulli variables, we have $\mathbb{E}[Z|X,\alpha,\beta,\pi] = p(Z=1|X,\alpha,\beta,\pi)$, which then can be computed using the Naive Bayes formula (1). Hence, at every iteration t, EM alternates between computing the expected value $\tilde{Z}^t$ of the Z variable for the current model variables $(\alpha^t, \beta^t, \pi^t)$ (E-step) and create the next iteration model variables using updates (3-5) and $\tilde{Z}^t$ in lieu of Z (M-step). The process can be initialized with either a starting Z^0 vector of ground-truth labels, or a starting model $(\alpha^0, \beta^0, \pi^0)$, and finishes when $\|\alpha^{t+1}-\alpha^t\|_1 + \|\beta^{t+1}-\beta^t\|_1 + |\pi^{t+1}-\pi^t| \leq \delta$ for a fixed δ. To alleviate stability issues, we additionally introduce a clipping parameter ϵ such that no model variable is smaller than ϵ or larger than $1-\epsilon$. In our implementation, we use the fixed values $\delta = 10^{-12}$ and $\epsilon = 10^{-3}$.

3.3 Supervised Classification

In some cases, the evaluator can gain access to more trustworthy labels for a subset of the data, for example, by manual analysis. When trustworthy labels are available, it becomes possible to use supervised learning to assign weights to vendors. This approach uses the executables with trustworthy labels as the training set. The vendor labels are the input features and the trustworthy labels are the training labels that the learned classifier attempts to predict from the vendor labels. The evaluator can use the learnt classifier on executables for which trustworthy labels do not exist to assign an aggregate label from the vendor labels.

In this paper, we use a classic a linear separation model for the learning algorithm. In particular, we use a L_2-regularized logistic regression for finding the optimal linear separator. For the training data, we consider the case where the evaluator has ground truth for n_l *randomly* selected instances. We then construct a labeled training dataset consisting of those n_l instances and proceed to learn the linear classifier on those instances. We use n_l to quantify the labeling effort needed for this scheme to succeed. Large values for n_l make this method impractical, but a sufficiently small number of ground-truth-labeled instances is possible in some situations. For instance, a human malware expert might be able to provide the ground truth for a small number of executables.

4. EVALUATION DATASET

In this section we examine the dataset we use for our evaluation. The final dataset is constructed as a subset of a larger datastream we have previously collected. We describe and characterize this datastream first, and then use the observations to construct an evaluation dataset with a high-quality approximation of ground-truth labels.

4.1 The Datastream

We use over one million distinct SHA256 hashes of Windows x86 binary executables (both malware and benign) that were submitted to VirusTotal between January 2012 and June 2014, and collected by Miller et al. [27]. For each file submission or resubmission, VirusTotal scans the file with up-to-date virus detectors and records the individual decisions of the AV vendors. Hence, a given executable hash will not only have multiple sets of multiple AV labels, corresponding to all historical scan events, from the first time the file was submitted, until the most recent resubmission. We collect all historical scan events for all hashes, but we do not have access to the executables.

Although we observe scan results from 80 distinct AV engines, some of these detectors are only sporadically present in the data. To obtain a consistent subset of both executables and vendors such that all vendors are present in all scans of all executables, we focus on the 34 vendors that appears in the most scans: AVG, Antiy-AVL, Avast, BitDefender, ByteHero, CAT-QuickHeal, ClamAV, Comodo, Dr-Web, ESET-NOD32, Emsisoft, F-Prot, F-Secure, Fortinet, GData, Ikarus, K7AntiVirus, Kaspersky, McAfee, McAfee-GW-Edition, Microsoft, Norman, nProtect, Panda, SUPER-AntiSpyware, Sophos, Symantec, TheHacker, TotalDefense, TrendMicro, TrendMicro-HouseCall, VBA32, VIPRE, Vi-Robot. We select the executables for which all of these vendors actually scan executable in all of its submissions. This requirement reduces the number of available distinct executables to about 734,000.

4.2 Deriving Ground-Truth Labels

We turn to the problem of assigning ground truth for future evaluation purposes in our dataset in the absence of actual executable files. To illustrate the difficulty of the problem and lack of consensus between vendors, Figure 3 shows a heat map of the matrix of pairwise correlation coefficients over the first-seen scan of all of the 734,000 executables. Note that the vendors are ordered such that highly correlated vendors appear as close to each other as possible, by agglomerative clustering using complete linkage. While there are a few noticeable square blocks of highly correlated vendors, the average correlation coefficient is only around 0.5, showing a clear lack of consensus between AV vendors. One vendor is barely correlated with the others as shown by the last row being nearly white.

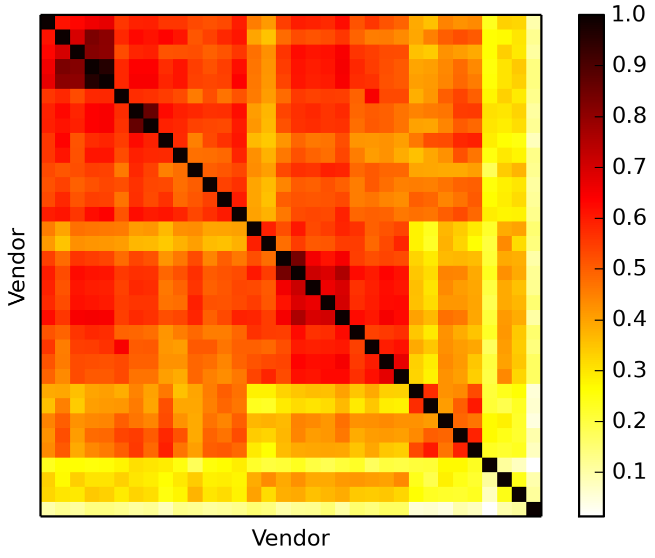


Figure 3: Label correlations between vendors.

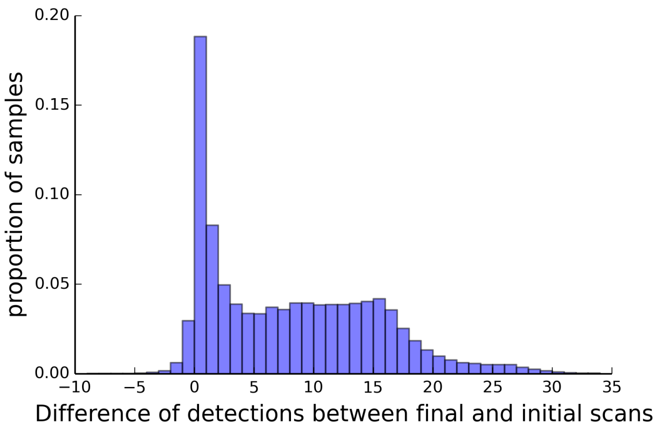


Figure 4: Difference in the number of positive detections between the first and last scans for each instance.

Fortunately, the sub-selected datastream has a temporal dimension that we can exploit to produce ground-truth labels. Figure 4 presents the histogram of the differences in the total number of positive detections between the first and last scan for a given instance. First, notice that there are often changes in the number of positives (malware) between the first and last scans, demonstrating that AV vendors change their labeling over time. Second, notice that the change overwhelmingly favors increasing the number of positive scan results, confirming that vendors catch up on their initial false negatives, and only suffer negligible initial false positive rates.

Figure 5 illustrates this behavior in a different way. For both of the first-seen (hatched red) and last-seen (blue) scan, we plot the histogram of the number of positive detections per instance. The final scan tends to be less likely to have zero positive detections from the 34 vendors, and more likely to exhibit a large number (≥ 10) detections compared to the

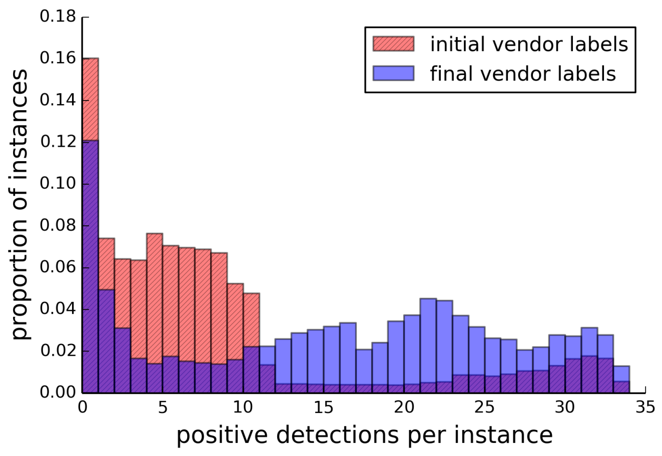


Figure 5: Number of initial and final detections for each instance.

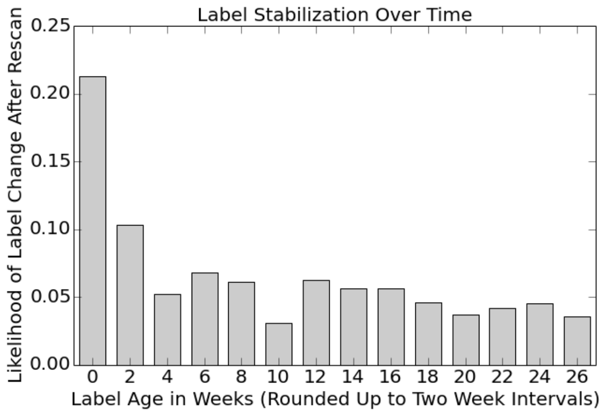


Figure 6: The likelihood of label change during rescan as a function of age of the most recent detection results.

initial scan. Note the relatively few executables in the flat region for 3 to 9 future detections. This region represents a transition zone from those executables that are most likely benign to those that are most likely malicious.

Making use of the observed transition, we select a threshold for declaring a executable to be malicious that lays within it. In particular, if the most up-to-date scan of a file has 3 or fewer detections, we deem it benign. On the other hand, if it receives 4 or more positive scans, we deem it malware for ground-truth purposes. Note that because there are relatively few executables in the flat region of 3 to 9 future detections, choosing any threshold within this range tends to produce a similar ground-truth labeling.

We consider how to know whether the last-seen scan appears long enough after the first-seen scan to give the AV vendors long enough to revise and stabilize their decisions:

Figure 6 shows the likelihood of a vendor changing its label as a function of time from the first scan. Notice that younger labels are more likely to change during a rescan than older labels, confirming that labels stabilize over time.

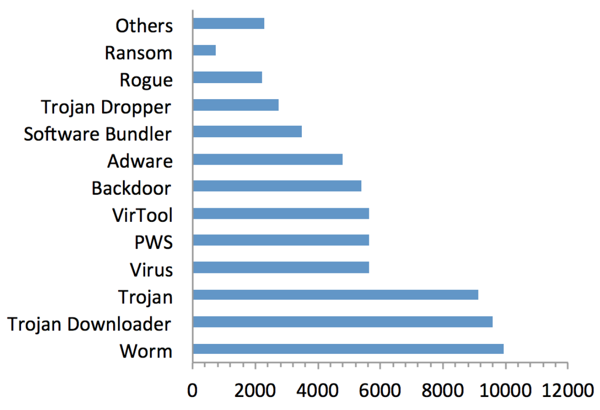


Figure 7: Malware types as detected by Microsoft AV.

Based on this graph, we use only executables for which the last-seen scan appears at least 4 weeks after the first-seen scan. This last constraint reduces our available data down to 279,327 executables for which we can assert ground truth with reasonably high certainty. Figure 7 shows the high degree of diversity of malware for this sample by measuring the proportions of malware types as detected by Microsoft AV.

In summary, we have constructed a subset of the datastream such that every binary is always scanned by all of the 34 vendors and has a last-seen scan at least 4 weeks older than its first-seen scan. For these executables, we set ground truth using threshold of 4 positive detections on the *last-seen* scans. This subset has 279,327 distinct instances, 78% of which are labeled malicious according to our ground-truth approach.

4.3 Evaluation Dataset Construction

To show how well our approaches can predict the ground-truth labels, we divide our dataset of *first-seen* scans into temporally consistent training and testing subsets. To ensure temporal consistency we require that both the training executables and their associated initial scans (and labels) are older than the testing executables and their initial scans. Temporal consistency is important for realistic evaluations as in the real world we can only have access to present and historical data. Note that the last-seen “future” scans are only used for building the ground-truth labels, and that the learning methods are exclusively trained on the first-seen “present” scans data. We choose a time boundary between training and testing such that there are exactly 100,000 instances in the testing set, and thus 179,327 instances available for training.

5. RESULTS

We evaluate the unsupervised model and the supervised classification techniques described in section 3. For each method, we forgo the problem of finding the optimal threshold τ and instead plot the true positive and false positive performance of the method for all possible thresholds τ. Since the focus in the malware detection task is towards methods with low false positive rates, we plot the Receiver Operation Characteristic (ROC) curves on a logarithmic false positive scale between 0.08% and 10% false positive rates. The performance of the unweighted threshold method at different thresholds is always shown for comparison purposes.

5.1 Generative Model

We observe that EM gives remarkably consistent models for a range of initialization choices. In what follows, we choose to initialize EM by setting Z^0 to unweighted threshold at $\tau = 4$ on the training data. Rerunning the experiments using thresholds of $\tau = 2, 3, 5, 6, 7$ lead to the same final model. The number of EM iterations is always smaller than 500, and can drop down to 37 depending on the choice of hyperparameters.

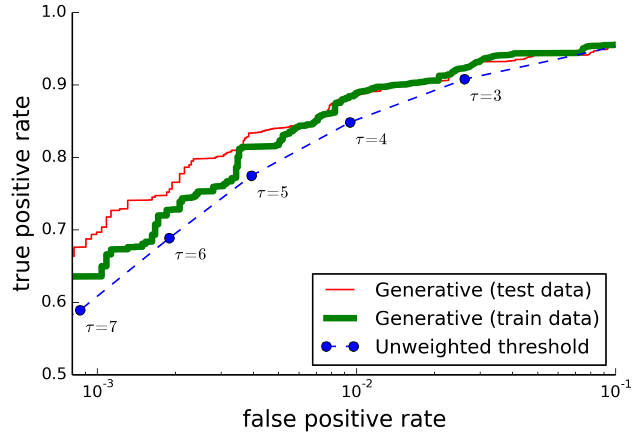
For the hyperparameter θ, we simply fix it to 0 to maximize the uncertainty over π. For φ and ψ, we use a grid search to find the combination of $\varphi \in \{0, 0.1N, 0.2N, \ldots, 2N\}$ and $\psi \in \{0, 0.1N, \ldots, 0.5N\}$ that maximizes the area of the training ROC curve between 0 and 0.1% false positive rate where N is the number of training executables. To keep the learning unsupervised, the false positive rate is not computed using the ground-truth labels, but rather the equally weighted threshold method ($\tau = 4$) using the training-time labels. That is, for hyperparameter selection, training instance i is labeled malicious if and only if $\sum_j X_{i,j} \leq 4$, without appealing to any future information.

The thin red curve on Figure 8a shows the performance of the model at labeling the executables on which it was trained. Here, the labels produced by the model are graded against the testing-time ground-truth labels computed from the future scans as described in Section 4.2.

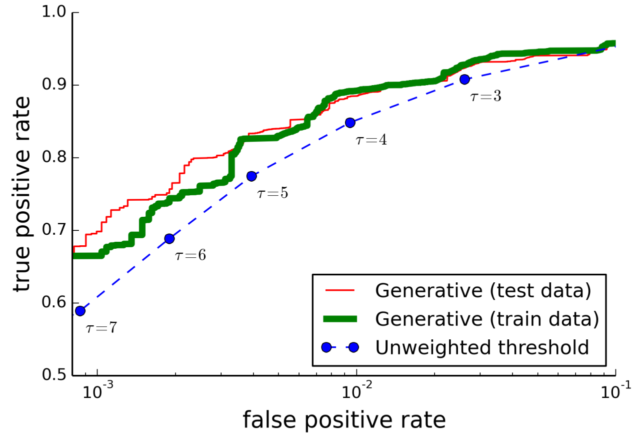
We repeated the above experiment using separate training and testing sets of executables. Since this training set comes from executables earlier in our datastream, this experiment explores how useful parameters selected for labeling one set are for labeling future sets, thereby shedding light on the need for retraining and the stability of vendor quality. In particular, it shows the penalty in accuracy incurred from not retraining for each new executable to be labeled and using an out-of-date model. The thick green curve in Figure 8a shows the performance of the model when trained on old executables. The results show that additional performance gains can be obtained by retraining the generative model on the most recent available data.

To better understand the impact of tuning the hyperparameters φ and ψ, we repeated the experiment using the training ground-truth labels to select the values of the hyperparameters (but not to select the values of the model parameters during training). In these experiments, the hyperparameters are set to our best estimates of their true values, which approximates what a careful analyst could achieve using domain knowledge. Figure 8b shows the results: they barely change. The results suggest that the performance of the EM-trained generative model is stable, even for different choices of priors φ and ψ, which is encouraging for practical situations where finding the "best" hyperparameters is difficult or impossible due to lack of ground truth.

In summary, we observe that in all cases the performance of the model is strictly better than the base line unweighted threshold, with the largest gains (4 to 5 percentage points) occurring in the low false positive region.



(a) Training data alone.



(b) Given training ground truth for validating φ, ψ.

Figure 8: ROC at low false positive rates for the generative model. $\theta = 0$ is fixed. In a, the hyperparameters $\varphi = 1.1N$ and $\psi = 0.4N$ are found by relying on the training data alone (no ground-truth labels). In b, $\varphi = 1.7N$ and $\psi = 0.1N$ are found by relying on ground truth for the training data.

5.2 Logistic Regression

To evaluate label aggregation by logistic regression, we randomly sample $n_l \in \{10, 20, 100\}$ training executables for which we reveal the ground-truth label. We then train a logistic regression model for each of these training subsets. We also train a best case logistic regression model on the whole training data, in which case we have $n_l = 179,327$. In all cases, we use LibLinear [11] with the L_2 regularization parameter C kept constant at $C = 1$.

Figure 9 presents the ROC curves of the logistic regression models at test time. For a sufficiently large sample size n_l, the model improves over any unweighted threshold method. In the limit when ground truth is available for all training executables, the model increases the true positive rate by about 25 percentage points over unweighted threshold at

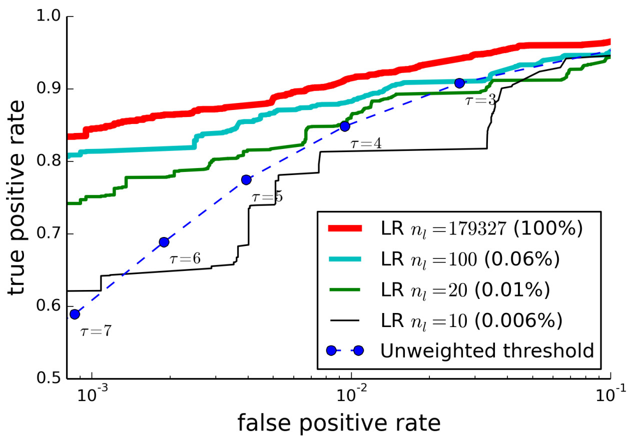


Figure 9: ROC at low false positive rates for logistic regression, with different numbers of labeled executables n_l.

0.1% false positive rate. Even when $n_l = 100$, the performance improvement is still noticeable: 20 percentage points.

The low number n_l of ground-truth labels needed to achieve a large improvement comes from the relatively low dimensionality of the problem: there are only 34 binary features, so we only need about the same order of magnitude random observations to learn a useful model.

6. CONCLUSION AND FUTURE WORK

This paper investigates the problem of combining multiple anti-virus vendor labels into a single authoritative ground-truth label. We present both an unsupervised and a supervised technique to assign confidence weights to vendors. The unsupervised approach uses Expectation Maximization to train a generative Bayesian model from the crowd-sourcing literature. This technique estimates each vendor's true and false positive rates in the absence of ground truth. We also describe a supervised but parsimonious classification scheme for improving label quality.

Using a large-scale real-world dataset, we demonstrate that these approaches can predict the ground-truth label of initially confusing instances. Our evaluation construction assumes that AV vendors eventually correct their choices to reflect the ground truth. The purely unsupervised approach produces modest improvements over the basic equally-weighted voting scheme. The supervised approach improves true-positive rate by 20 percentage points at 0.1% false positives with only a hundred ground-truth training labels.

A potentially significant improvement of the generative model may come from relaxing the vendor independence assumption. For instance, we could augment the model by introducing pairwise vendor interactions and derive a new expectation maximization training algorithm for the resulting model. To further improve the performance of the unsupervised method, we could include additional observed data for the instance. For example, this supplementary data could include the feature vector input from a larger malware detector using machine learning. To accommodate this information, the model could be extended on its generative side by making the features dependent on the ground-truth variable, or on its discriminative side by making the ground-truth variable dependent on the features as in [40].

Finally, one can also consider hybrid approaches. For instance, this problem might benefit from semi-supervised or active learning techniques. However, the small size of our feature space, 34 binary features, makes the fully supervised learning problem already quite efficient in terms of number of labeled instances as demonstrated by the logistic regression experiment.

Improved training labels, as obtained by the techniques presented in this paper, should result in an improved malware detector.

Acknowledgment

This research is supported in part by Intel's ISTC for Secure Computing, NSF grants 0424422 (TRUST) and 1139158, the Freedom 2 Connect Foundation, US State Dept. DRL, LBNL Award 7076018, DARPA XData Award FA8750-12-2-0331, and gifts from Amazon, Google, SAP, Apple, Cisco, Clearstory Data, Cloudera, Ericsson, Facebook, GameOnTalis, General Electric, Hortonworks, Huawei, Intel, Microsoft, NetApp, Oracle, Samsung, Splunk, VMware, WANdisco and Yahoo!. The opinions in this paper are those of the authors and do not necessarily reflect those of any funding sponsor or the United States Government.

7. REFERENCES

[1] Alazab, M., Venkatraman, S., Watters, P., and Alazab, M. Zero-day malware detection based on supervised learning algorithms of API call signatures. In *Ninth Australasian Data Mining Conference - Volume 121* (2011).

[2] Bailey, M., Oberheide, J., Andersen, J., Mao, Z. M., Jahanian, F., and Nazario, J. Automated classification and analysis of internet malware. In *Recent Advances in Intrusion Detection* (2007), Springer, pp. 178–197.

[3] Bailey, M., Oberheide, J., Andersen, J., Mao, Z. M., Jahanian, F., and Nazario, J. Automated classification and analysis of internet malware. In *Recent Advances in Intrusion Detection, 10th International Symposium, RAID 2007, Gold Goast, Australia, September 5-7, 2007, Proceedings* (2007), C. Krügel, R. Lippmann, and A. J. Clark, Eds., vol. 4637 of *Lecture Notes in Computer Science*, Springer, pp. 178–197.

[4] Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. In *In International Conference on Machine Learning (ICML* (2012).

[5] Canto, J., Dacier, M., Kirda, E., and Leita, C. Large scale malware collection: Lessons learned. In *IEEE SRDS Workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems* (2008).

[6] Chau, D., Nachenberg, C., Wilhelm, J., Wright, A., and Faloutsos, C. Polonium: Tera-scale graph mining and inference for malware detection. *SIAM International Conference on Data Mining* (2011).

[7] Curtsinger, C., Livshits, B., Zorn, B., and Seifert, C. ZOZZLE: Fast and precise in-browser JavaScript malware detection. In *Proceedings of the 20th USENIX Conference on Security* (2011), SEC'11, USENIX Association, pp. 3–16.

[8] Dahl, G. E., Stokes, J. W., Deng, L., and Yu, D. Large-scale malware classification using random projections and neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (2013), IEEE, pp. 3422–3426.

[9] Damballa. State of Infections Report: Q4 2014. Tech. rep., Damballa, 2015.

[10] Dawid, A. P., and Skene, A. M. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics* (1979), 20–28.

[11] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research 9* (2008).

[12] Gascon, H., Yamaguchi, F., Arp, D., and Rieck, K. Structural detection of android malware using embedded call graphs. In *Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security* (2013), AISec '13, ACM, pp. 45–54.

[13] Gavrilut, D., Cimpoesu, M., Anton, D., and Ciortuz, L. Malware detection using perceptrons and support vector machines. In *Computation World* (2009), pp. 283–288.

[14] Henchiri, O., and Japkowicz, N. A feature selection and evaluation scheme for computer virus detection. In *Sixth Intl. Conf. on Data Mining* (2006), pp. 891–895.

[15] Kolter, J. Z., and Maloof, M. A. Learning to detect and classify malicious executables in the wild. *J. Machine Learning Research 7* (2006).

[16] Laskov, P., and Šrndić, N. Static detection of malicious JavaScript-bearing PDF documents. In *Proceedings of the 27th Annual Computer Security Applications Conference* (2011), ACSAC '11, ACM, pp. 373–382.

[17] Li, P., Liu, L., Gao, D., and Reiter, M. K. On challenges in evaluating malware clustering. In *Recent Advances in Intrusion Detection* (2010), Springer, pp. 238–255.

[18] Liu, Q., Peng, J., and Ihler, A. T. Variational inference for crowdsourcing. In *Advances in Neural Information Processing Systems* (2012), pp. 692–700.

[19] Maggi, F., Bellini, A., Salvaneschi, G., and Zanero, S. Finding non-trivial malware naming inconsistencies. In *Information Systems Security*. Springer, 2011, pp. 144–159.

[20] Maiorca, D., Giacinto, G., and Corona, I. A pattern recognition system for malicious PDF files detection. In *Proceedings of the 8th International Conference on Machine Learning and Data Mining in Pattern Recognition* (2012), MLDM'12, Springer-Verlag, pp. 510–524.

[21] Mann, G. S., and McCallum, A. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *J. Mach. Learn. Res. 11* (Mar. 2010), 955–984.

[22] Markel, Z., and Bilzor, M. Building a machine learning classifier for malware detection. In *Anti-malware Testing Research (WATeR), 2014 Second Wksp. on* (2014), pp. 1–4.

[23] Masud, M., Khan, L., and Thuraisingham, B. A hybrid model to detect malicious executables. In *Communications, 2007. ICC '07. IEEE International Conference on* (June 2007), pp. 1443–1448.

[24] Masud, M. M., Khan, L., and Thuraisingham, B. M. A scalable multi-level feature extraction technique to detect malicious executables. *Information Systems Frontiers 10*, 1 (2008), 33–45.

[25] McAfee Labs. McAfee Labs Threats Report. Tech. rep., McAfee, 2014.

[26] Menahem, E., Shabtai, A., Rokach, L., and Elovici, Y. Improving malware detection by applying multi-inducer ensemble. *Comput. Stat. Data Anal. 53*, 4 (Feb. 2009), 1483–1494.

[27] Miller, B., Kantchelian, A., Afroz, S., Bachwani, R., Faizullabhoy, R., Huang, L., Shankar, V., Tschantz, M. C., Wu, T., Yiu, G., Joseph, A. D., and Tygar, J. D. Back to the future: Malware detection with temporally consistent labels. Under submission, 2015.

[28] Mohaisen, A., and Alrawi, O. Av-meter: An evaluation of antivirus scans and labels. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2014, pp. 112–131.

[29] Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. Text classification from labeled and unlabeled documents using em. *Machine Learning 39*, 2-3 (2000), 103–134.

[30] Nissim, N., Cohen, A., Moskovitch, R., Shabtai, A., Edry, M., Bar-Ad, O., and Elovici, Y. ALPD: Active learning framework for enhancing the detection of malicious PDF files. In *IEEE Joint Intelligence and Security Informatics Conf.* (2014), pp. 91–98.

[31] Nissim, N., Moskovitch, R., Rokach, L., and Elovici, Y. Novel active learning methods for enhanced pc malware detection in windows os. In *J. Expert Systems with Applications* (2014).

[32] Nissim, N., Moskovitch, R., Rokach, L., and Elovici, Y. Novel active learning methods for enhanced PC malware detection in windows OS. *Expert Systems with Applications 41*, 13 (2014), 5843 – 5857.

[33] Overveldt, T. V., Kruegel, C., and Vigna, G. Flashdetect: Actionscript 3 malware detection. In *Research in Attacks, Intrusions, and Defenses - 15th International Symposium, RAID 2012* (2012), D. Balzarotti, S. J. Stolfo, and M. Cova, Eds., vol. 7462 of *Lecture Notes in Computer Science*, Springer, pp. 274–293.

[34] Perdisci, R., et al. VAMO: Towards a fully automated malware clustering validity analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference* (2012), ACM, pp. 329–338.

[35] Perdisci, R., Lanzi, A., and Lee, W. Mcboost: Boosting scalability in malware collection and analysis using statistical classification of executables. In *Computer Security Applications Conference, 2008. ACSAC 2008. Annual* (2008), pp. 301–310.

[36] Perdisci, R., Lee, W., and Feamster, N. Behavioral clustering of HTTP-based malware and signature generation using malicious network traces. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation* (2010), NSDI'10, USENIX Association, pp. 26–26.

[37] Rajab, M. A., Ballard, L., Lutz, N., Mavrommatis, P., and Provos, N. CAMP: Content-agnostic malware protection. In *20th Annual Network and Distributed System Security Symposium, NDSS* (2013).

[38] Ramakrishnan, G., Chitrapura, K. P., Krishnapuram, R., and Bhattacharyya, P. A model for handling approximate, noisy or incomplete labeling in text classification. In *Proceedings of ICML* (2005).

[39] Raykar, V. C., and Yu, S. Ranking annotators for crowdsourced labeling tasks. In *Advances in Neural Information Processing Systems* (2011), pp. 1809–1817.

[40] Raykar, V. C., Yu, S., Zhao, L. H., Jerebko, A., Florin, C., Valadez, G. H., Bogoni, L., and Moy, L. Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *Proceedings*

of the 26th Annual International Conference on Machine Learning (2009), ACM, pp. 889–896.

[41] Reddy, D. K. S., and Pujari, A. K. N-gram analysis for computer virus detection. *Journal in Computer Virology 2*, 3 (2006), 231–239.

[42] Rieck, K., Holz, T., Willems, C., Düssel, P., and Laskov, P. Learning and classification of malware behavior. In *Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (2008), DIMVA '08, Springer-Verlag, pp. 108–125.

[43] Rieck, K., Krueger, T., and Dewald, A. Cujo: Efficient detection and prevention of drive-by-download attacks. In *Proceedings of the 26th Annual Computer Security Applications Conference* (2010), ACSAC '10, ACM, pp. 31–39.

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

[45] Sahs, J., and Khan, L. A machine learning approach to android malware detection. In *Intelligence and Security Informatics Conference (EISIC), 2012 European* (Aug 2012), pp. 141–147.

[46] Santos, I., Brezo, F., Nieves, J., Penya, Y. K., Sanz, B., Laorden, C., and Bringas, P. G. Idea: Opcode-sequence-based malware detection. In *Engineering Secure Software and Systems, Second International Symposium, ESSoS 2010* (2010), F. Massacci, D. S. Wallach, and N. Zannone, Eds., vol. 5965 of *Lecture Notes in Computer Science*, Springer, pp. 35–43.

[47] Santos, I., Nieves, J., and Bringas, P. G. Semi-supervised learning for unknown malware detection. In *International Symposium on Distributed Computing and Artificial Intelligence, DCAI 2011, Salamanca, Spain, 6-8 April 2011* (2011), A. Abraham, J. M. Corchado, S. Rodríguez-González, and J. F. D. P. Santana, Eds., vol. 91 of *Advances in Soft Computing*, Springer, pp. 415–422.

[48] Santos, I., Penya, Y. K., Devesa, J., and Bringas, P. G. N-grams-based file signatures for malware detection. In *ICEIS 2009: Proceedings of the 11th International Conference on Enterprise Information Systems* (2009), J. Cordeiro and J. Filipe, Eds., pp. 317–320.

[49] Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Bringas, P. G., and Marañón, G. Á. PUMA: permission usage to detect malware in android. In *International Joint Conference CISIS'12-ICEUTE'12-SOCO'12 Special Sessions* (2012), Á. Herrero, V. Snásel, A. Abraham, I. Zelinka, B. Baruque, H. Quintián-Pardo, J. L. Calvo-Rolle, J. Sedano, and E. Corchado, Eds., vol. 189 of *Advances in Intelligent Systems and Computing*, Springer, pp. 289–298.

[50] Schmidt, A.-D., Bye, R., Schmidt, H.-G., Clausen, J., Kiraz, O., Yüksel, K. A., Camtepe, S. A., and Albayrak, S. Static analysis of executables for collaborative malware detection on android. In *Proceedings of the 2009 IEEE International Conference on Communications* (2009), IEEE Press, pp. 631–635.

[51] Schultz, M. G., Eskin, E., Zadok, E., and Stolfo, S. J. Data mining methods for detection of new malicious executables. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy* (2001), SP '01, IEEE Computer Society, pp. 38–49.

[52] Schwenk, G., Bikadorov, A., Krueger, T., and Rieck, K. Autonomous learning for detection of javascript attacks: Vision or reality? In *ACM Wksp. on Artificial Intelligence and Security (AISec)* (2012).

[53] Sheng, V. S., Provost, F., and Ipeirotis, P. G. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), ACM, pp. 614–622.

[54] Shih, D.-H., Chiang, H.-S., and Yen, C. D. Classification methods in the detection of new malicious emails. *Information Sciences 172* (2005), 241 – 261.

[55] Smutz, C., and Stavrou, A. Malicious PDF detection using metadata and structural features. In *Proceedings of the 28th Annual Computer Security Applications Conference* (2012), ACSAC '12, ACM, pp. 239–248.

[56] Šrndic, N., and Laskov, P. Detection of malicious PDF files based on hierarchical document structure. In *Network & Distributed System Security Symp.* (2013).

[57] Stolfo, S., Wang, K., and Li, W.-J. Towards stealthy malware detection. In *Malware Detection*, M. Christodorescu, S. Jha, D. Maughan, D. Song, and C. Wang, Eds., vol. 27 of *Advances in Information Security*. Springer US, 2007, pp. 231–249.

[58] Stringhini, G., Kruegel, C., and Vigna, G. Shady paths: Leveraging surfing crowds to detect malicious web pages. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security* (2013), CCS '13, ACM, pp. 133–144.

[59] Tabish, S. M., Shafiq, M. Z., and Farooq, M. Malware detection using statistical analysis of byte-level file content. In *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics* (2009), CSI-KDD '09, ACM, pp. 23–31.

[60] Tahan, G., Rokach, L., and Shahar, Y. Mal-id: Automatic malware detection using common segment analysis and meta-features. *J. Mach. Learn. Res. 13* (Apr. 2012), 949–979.

[61] Tian, R., Batten, L., Islam, M., and Versteeg, S. An automated classification system based on the strings of trojan and virus families. In *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on* (Oct 2009), pp. 23–30.

[62] VirusTotal. `https://www.virustotal.com/en/statistics/`. Retrieved on July 30, 2014.

[63] Welinder, P., Branson, S., Perona, P., and Belongie, S. J. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems* (2010), pp. 2424–2432.

[64] Whitehill, J., Wu, T.-f., Bergsma, J., Movellan, J. R., and Ruvolo, P. L. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems* (2009), pp. 2035–2043.

[65] Ye, Y., Chen, L., Wang, D., Li, T., Jiang, Q., and Zhao, M. SBMDS: An interpretable string based malware detection system using SVM ensemble with bagging, 2009.

[66] Ye, Y., Wang, D., Li, T., Ye, D., and Jiang, Q. An intelligent pe-malware detection system based on association mining. *Journal in Computer Virology 4*, 4 (2008), 323–334.