Reconciling Malware Labeling Discrepancy via Consensus Learning

Ting Wang Xin Hu Shicong Meng Reiner Sailer
IBM T.J. Watson Research Center
Email: {tingwang, huxin, smeng, sailer}@us.ibm.com

***Abstract*—Anti-virus systems developed by different vendors often demonstrate strong discrepancy in the labels they assign to given malware, which significantly hinders threat intelligence sharing. The key challenge of addressing this discrepancy stems from the difficulty of re-standardizing already-in-use systems. In this paper we explore a non-intrusive alternative. We propose to leverage the correlation between the malware labels of different anti-virus systems to create a "consensus" classification system, through which different systems can share information without modifying their own labeling conventions. To this end, we present a novel *classification integration* framework LATIN which exploits the correspondence between participating anti-virus systems as reflected in heterogeneous information at instance-instance, instance-class, and class-class levels. We provide results from extensive experimental studies using real datasets and concrete use cases to verify the efficacy of LATIN in reconciling the malware labeling discrepancy.**

I. INTRODUCTION

Mitigating and defending against the malware threats (e.g., virus, worms, and backdoors) has been a prominent topic for security research for decades. A plethora of anti-virus systems have been developed by different vendors (e.g., *Kaspersky*, *Symantec*, and *Norton*). Conceivably, these systems demonstrate fairly different expertise; it would be beneficial to share intelligence across different systems, thereby helping build defense systems with broader coverage and more informative diagnosis. In reality however, the integration of malware threat information is still limited to a few systems [1], due to a number of challenges, including: conflict of interest, discrepancy of information, and incompatibility of data schemas.

We focus on one such challenge, namely, the malware labeling discrepancy. Specifically, each anti-virus system follows certain malware taxonomy. The *label* of a malware instance identifies its *class* in this taxonomy. For example, the label *"Trojan.Spy"* represents a class of malware spying on users' activities. Moreover, the labels of a set of malware instances essentially partition this set into different classes. Due to lacking malware classification standards, systems developed by different vendors often demonstrate strong discrepancy in the labels they assign to given malware, both syntactically (e.g., *"Spy-Trj"* versus *"Trojan.Spy"*) and semantically (e.g., whether two instances belong to a same class). Hence, sharing information based on labels may lead to unpredictable consequences, because similar labels could refer to completely distinct concepts by different systems.

Existing work has attempted to remedy this discrepancy by standardizing malware label formats; however most major anti-virus vendors were reluctant to change their conventions [2], [3], [4]. Even if they did, the semantic discrepancy would still exist; that is, vendors may hold conflicting opinions regarding whether two malware instances are classified into a same class, though they may be unanimous on the syntactic representation of taxonomy. One may suggest use more intrinsic features (e.g., md5 of binary code) instead of labels to identify malware. This option however has the drawbacks such as lacking interpretability and eradicating all the semantic relationships between malware (e.g., variation [5]).

Instead we consider a more pragmatic alternative based on the fundamental observation: for given malware instances, despite the disparate classification by individual systems, there exists certain "consensus" classification agreed by a majority of the systems. Thus, bridged by the probabilistic mappings between the "local" classification of individual systems and this mediated classification, different systems can communicate with each other without changing their own classification. Note that we are not arguing that this mediated classification corresponds to the ground truth; rather we use it as a common ground to facilitate information sharing.

We present a novel *classification integration* framework LATIN that fulfills this idea. Specifically, (i) for participating systems, LATIN exploits the correspondence between their "local" classification as reflected in heterogenous information at instance-instance, instance-class, and class-class levels; (ii) it then applies novel consensus learning methods over such multifaceted correspondence to derive the mediated classification; (iii) equipped with the local-mediated mapping, LATIN is able to answer various interesting queries, including: *matching* - determining whether two given labels from different systems refer to similar instances; and *retrieval* - identifying the counterpart of a given label in another system. This proposal is illustrated in Fig.1. Using large-scale real datasets and concrete use cases, we show that:

- LATIN effectively exploits various heterogenous information, including: scan results of malware instances, malware specification in threat encyclopedia, and behavior description from malware analysis services.
- LATIN has impact on a range of applications: (1) it allows analysts to search for relevant information in multiple anti-virus systems when a newly discovered malware variant is only partially understood; (2) it supports efficient integration of malware detection reports generated by different systems; and (3) it enables to calibrate the performance measures of malware analysis tools conducted on self-labeled malware datasets.
- LATIN easily scales up to large-size malware datasets under modest hardware configuration.

The remainder of the paper proceeds as follows. Section II and Section III describe in detail the design and implementation of the core components of LATIN. The proposed solution is experimentally evaluated in Section IV. Section V surveys relevant literature. The paper is concluded in Section VI.

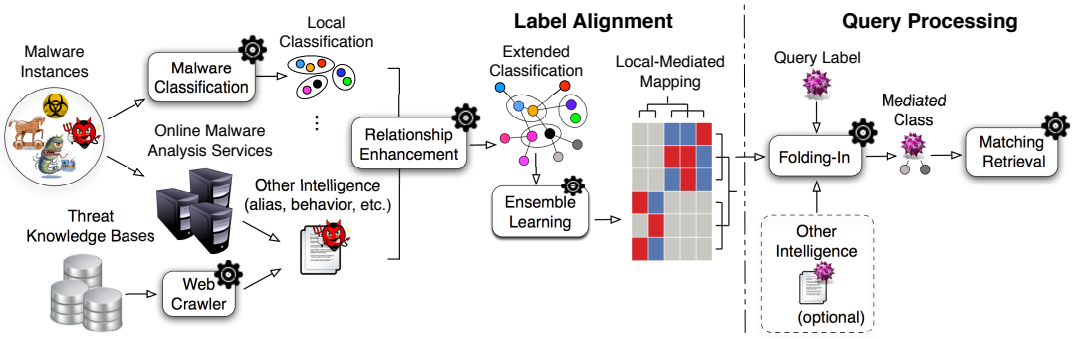

Fig. 1: LATIN: System architecture of classification integration (label alignment and query processing).

II. Label Alignment

We refer to the process of deriving the consensus classification as "label alignment", which works in three steps: *malware classification*, *relationship enhancement*, and *ensemble learning*, each we elaborate below.

A. Malware Classification (basic instance-class relationships)

LATIN incorporates all local classification into a weighted bipartite graph (Fig.2 (a)) wherein the left nodes represent the malware instances, the right ones the classes appearing in the classification results, and instance i and class c are adjacent only if i is classified as c by some system. We differentiate classes from different systems even if they have identical labels because they may have varying semantics. Constructed over a common set of instances, this graph encodes correlations between classes from different systems.

Example 1: One instance classified as *"Trojan.Spy"* and *"Backdoor.Bot"* by two systems indirectly correlates classes *Trojan.Spy* and *Backdoor.Bot*.

B. Relationship Enhancement (additional instance-class, instance-instance, and class-class relationships)

For a particular system, the coverage of its local classification could be limited or biased, highly dependent on the given malware instances. Moreover, instance-class relationships alone are insufficient for constructing optimal mediated classification; rather, instance-instance and class-class relationships also need to be taken into account.

Next LATIN integrates other forms of threat intelligence, thereby providing coverage beyond the available malware corpus. Specifically, many antivirus vendors today make available online *threat encyclopedias* that provide information of malware instances well-studied by analysts, including *alias* (i.e., "also-known-as" names of malware used by other vendors) and *behavior profile* (e.g., how the malware instance infects a system, what damage it causes to infected systems, etc.). Fig.3 shows a sample entry of malware *Worm.Chyzvis*.

Another source of valuable information is online malware analysis service (e.g., Anubis[1]) which execute submitted malware binaries in sandbox environments and report observed activities (similar to behavior profiles). LATIN queries such services with the malware binaries and collects their analysis reports. Combining the information of threat encyclopedias and analysis reports, LATIN obtains behavior profiles for all the malware instances concerned.

Here we focus on three types of threat intelligence, namely, alias, behavior profile, and class label. Below we discuss how we leverage the additional intelligence to enhance the bipartite instance-class graph built in the previous step.

1) **Threat Alias (additional instance-class relationships)***:* The aliases of malware instances are essentially equivalent labels in different antivirus systems identified by analysts; therefore they can also be used to construct instance-class relationships, complementing the corpus of malware instances.

Example 2: In Fig.3, the aliases of *Worm.Chyzvis* indicate that it is also classified *Trojan.Scar* by another system.

We incorporate these instance-class relationships to the bipartite graph created in the probing step (Fig.2 (b)). Let $\mathcal{I}$ and $\mathcal{C}$ denote the set of malware instances and classes in this extended bipartite graph, with $|\mathcal{I}| = n$ and $|\mathcal{C}| = m$. We use a matrix $\mathbf{A_{IC}} \in \mathbb{R}^{n \times m}$ to represent the instance-class relationships, with $[\mathbf{A_{IC}}]_{i,c} = w_{i,c}$. Intuitively two malware instances are considered similar if they are categorized into similar classes. We can therefore derive a similarity matrix $\mathbf{S^r} \in \mathbb{R}^{n \times n}$ for malware instances based on their relationships with different classes: $\mathbf{S^r} = \mathbf{A_{IC}}\mathbf{A_{IC}}^T$.

2) **Behavior Profile (instance-instance relationships)***:* The challenge of leveraging behavior profile information lies in that: (i) The threat encyclopedias of different systems have drastically different schemas for behavior profiles, e.g., certain sections present in one encyclopedia may be missing in another, or some profiles may use free text description while others may use pseudo-code like languages. (ii) Even in the same encyclopedia, the detail level of documentation varies from case to case, e.g., different parts may be missing for different instances, which reflects the varying levels of understanding analysts have regarding the behavior of different instances. Such inconsistency makes directly comparing the behavior profiles extremely difficult.

We address the challenge by transforming different forms of behavior profiles into a unified model. We observe that despite their diverse representations, behavior profiles (particularly key words) share a common vocabulary, specific to security research community. We resort to security glossaries (e.g., SANS glossary[2] and RSA glossary[3]) to build such a

[1] http://anubis.iseclab.org

[2] www.sans.org/security-resources/glossary-of-terms

[3] www.rsa.com/glossary

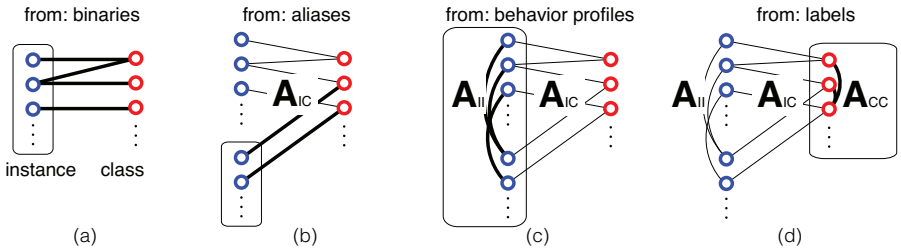

Fig. 2: Augmentation of bipartite instance-class graph.

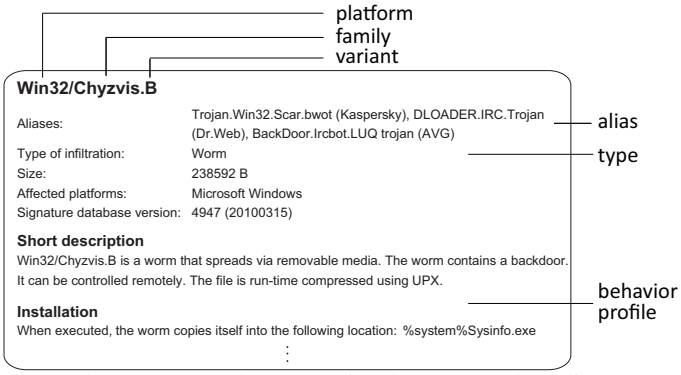

Fig. 3: A sample entry of a threat encyclopedia.

vocabulary, onto which each behavior profile is projected. Formally, assuming a vocabulary $\mathcal{T}$, the behavior profile of a malware instance $i \in \mathcal{I}$ is modeled as a bag of words: $\mathtt{p}_i = \{\ldots, w_{i,t}, \ldots\}(t \in \mathcal{T})$ where $w_{i,t}$ is the frequency of term t for i's behavior profile.

Example 3: In Fig.3, the behavior profile may contain the following set of security terms from the vocabulary: {*worm, removable media, backdoor, root, UPX*}.

We then use the probability distribution of a profile over the latent topics generated by Latent Dirichlet Allocation [6] as its feature vector. For given instances i and i', let $\mathtt{k}(\mathtt{p}_i, \mathtt{p}_{i'})$ denote the similarity of the feature vectors of their profiles, which complement the bipartite instance-class graph (Fig.2 (c)). Let $\mathbf{S^P} \in \mathbb{R}^{n \times n}$ denote the instance-instance similarity derived from their behavior profiles, where $[\mathbf{S^P}]_{i,i'} = \mathtt{k}(\mathtt{p}_i, \mathtt{p}_{i'})$.

3) **Class Label (class-class relationships)***:* Another important type of relationships missed in the instance-class graph thus far is that between class and class. In our implementation we rely on the class labels to derive such relationships. Let $\mathtt{l}_c$ be the label of a class c and $\mathtt{gram}(\mathtt{l}_c, n)$ be its set of n-grams (n = 2 in our implementation). The similarity of $\mathtt{l}_c$ and $\mathtt{l}_{c'}$ is measured by their Jaccard coefficient:

$$\mathtt{k}(\mathtt{l}_c, \mathtt{l}_{c'}) = \frac{|\mathtt{gram}(\mathtt{l}_c, n) \cap \mathtt{gram}(\mathtt{l}_{c'}, n)|}{|\mathtt{gram}(\mathtt{l}_c, n) \cup \mathtt{gram}(\mathtt{l}^t_{c'}, n)|} \quad (1)$$

Let $\mathbf{A_{CC}} \in \mathbb{R}^{m \times m}$ denote the class-class similarity matrix with $[\mathbf{A_{CC}}]_{c,c'} = \mathtt{k}(\mathtt{l}_c, \mathtt{l}_{c'})$ (Fig.2 (d)). We intend to populate such class-class similarity to the space of malware instances. We follow a procedure similar to that of deriving $\mathbf{S^P}$ from behavior profiles. The details are omitted here due to the space constraint. Below we use $\mathbf{S^l} \in \mathbb{R}^{n \times n}$ to represent the instance-instance similarity matrix derived from class labels.

C. Consensus Learning

Next we present a scalable consensus learning method that integrates these relationships to derive the mediated classification: it first (i) finds an extremely low-dimensional embedding of malware instances that captures all instance-instance, instance-class, and class-class relationships, (ii) then identifies the optimal partitioning of the embedding as the mediated classification, and (iii) finally derives the mappings between local and mediated classifications through the relationships between instances and classes.

1) **Power Iteration Embedding:** Recall that in the previous step we collect evidence of instance-instance similarity from the perspectives of instance-class relationships, behavior profiles, and class labels, leading to three similarity matrices $\mathbf{S^r}$, $\mathbf{S^P}$, and $\mathbf{S^l}$. Here all these evidences are integrated into an overall similarity measure.

$$\mathbf{S} = \lambda \mathbf{S^l} + \eta \mathbf{S^P} + \mathbf{S^r} \quad (2)$$

where λ, η balance the relative weight of different evidences (their setting is discussed in Section IV). The row-normalized version $\mathbf{W}$ is derived as $\mathbf{W} = \mathtt{diag}^{-1}(\mathbf{S} \cdot \mathbf{1})\mathbf{S}$. It is known that the top eigenvectors of $\mathbf{W}$ give a lower dimensional embedding of instances that preserves their similarity structure. We apply the power iteration method [7] to derive a one-dimensional vector $\mathbf{v}^*$ which well approximates such embedding and can be efficiently computed.

2) **Optimal Mediated classification:** The very low dimensionality of the embedding $\mathbf{v}^*$ allows us to readily apply any sophisticated clustering methods to find high-quality partition of $\mathbf{v}^*$, which corresponds to the majority-agreed grouping of malware instances.

3) **Local-Mediated Mappings:** The bijection between $\mathbf{v}^*$ and instances $\mathcal{I}$ implies that the partitioning of $\mathbf{v}^*$ corresponds to the mediated classification of $\mathcal{I}$. By leveraging the instance-class relationships $\mathbf{A_{IC}}$ (Fig.2), we identify the probabilistic mapping between local and mediated classification. Specifically, for two classes c and c' in local and mediated classification, respectively, the *forward correspondence* $[\mathbf{F}]_{c,c'}$ is the probability that an instance in c also belongs to c', estimated by the fraction of instances shared by c and c', $|c \cap c'|/|c|$.

Similarly, we derive the *reverse correspondence* from the mediated to local classification. However as the mediated classification may contain instances not covered by local classification, we have to exclude those instances in the computation.

III. Query Processing

Equipped with the local-mediated mappings as reference, LATIN is able to effectively answer various interesting queries.

A. Folding-In

The prerequisite of query processing is to map the malware instance referred in the query (from specific anti-virus systems) to the space of mediated classification.

Given instance q and class c in the mediated classification, the *class membership* $[\mathbf{M}]_{q,c}$ specifies the probability that q belongs to c. Intuitively $[\mathbf{M}]_{q,c}$ w.r.t. to all the classes $\{c\}$ in the mediated classification form a probability distribution. The process of inferencing this distribution is referred to as *folding-in*, which we detail below. Without loss of generality, we fix a particular class c^* and intend to estimate $[\mathbf{M}]_{q,c^*}$.

Query with only label: We start with the simple case that only the label $\mathtt{l}_q$ of q is available. If $\mathtt{l}_q$ appears in the local classification, the folding-in of q is straightforward, which is the forward correspondence of the class indicated by $\mathtt{l}_q$ (denoted by c_q), i.e., $[\mathbf{M}]_{q,c^*} = [\mathbf{F}]_{c_q,c^*}$.

We first derive the relationships between $\mathtt{l}_q$ and the labels of all the classes $\{c\}$ in the local classification using the label similarity metric (Eq.(1)), and integrate the forward correspondence between $\{c\}$ and c^* to estimate $[\mathbf{M}]_{q,c^*}$. Formally,

$$[\mathbf{M}]_{q,c^*} = \frac{\sum_c \mathtt{k}(\mathtt{l}_q, \mathtt{l}_c)[\mathbf{F}]_{c,c^*}}{\sum_{c'} \mathtt{k}(\mathtt{l}_q, \mathtt{l}_{c'})} \tag{3}$$

Here the summation iterates over all the classes in the local classification involving $\mathtt{l}_q$.

Query with alias: With additional information relevant to q available, we can further improve the estimation of $\mathbf{M}_{q,c^*}$. Let us first consider the case that the aliases of $\mathtt{l}_q$ are available. We first compute $\mathbf{M}_{q,c^*}$ according to Eq.(3) for each alias w.r.t. its corresponding local classification and then use their average as the overall estimation of $\mathbf{M}_{q,c^*}$.

Query with behavior profile: In the case that the behavior profile $\mathtt{p}_q$ of q is available, for each class c in the local classification, we incorporate the behavior profile similarity between $\mathtt{p}_q$ and all the instances in c:

$$[\mathbf{M}]_{q,c^*} = \frac{\sum_c (\sum_{i \in c} \mathtt{k}(\mathtt{p}_q, \mathtt{p}_i))[\mathbf{F}]_{c,c^*}}{\sum_{c'} \sum_{i' \in c'} \mathtt{k}(\mathtt{p}_q, \mathtt{p}_{i'})}$$

We compute $[\mathbf{M}]_{q,c^*}$ for each local classification and then use their average value as the overall estimation.

Further we integrate the estimation from both label and behavior profile. Let $[\mathbf{M}^{\mathbf{l}}]_{q,c^*}$ and $[\mathbf{M}^{\mathbf{p}}]_{q,c^*}$ denote the label- and profile-based estimation, respectively. The overall estimation is given by: $[\mathbf{M}]_{q,c^*} = \frac{\lambda}{\lambda+\eta}[\mathbf{M}^{\mathbf{l}}]_{q,c^*} + \frac{\eta}{\lambda+\eta}[\mathbf{M}^{\mathbf{p}}]_{q,c^*}$, where λ and η are the parameters in Eq.(2).

B. Matching and Retrieval

Once the malware instance referred in the query is mapped to the space of mediated classification, LATIN is capable of answering various fundamental questions.

Matching: Within a matching query, a pair of instances (q_s, q_t) from different systems are given, the analyst intends to know whether q_s and q_t belong to the same class in the mediated classification. Recall that we use the distribution $\mathbf{M}_q$ (membership vector) to represent the membership of q with respect to each class in the mediated classification. Let $\mathbf{M}_{q_s}$ and $\mathbf{M}_{q_t}$ be the membership vectors of q_s and q_t. The probability that q_s matches q_t is estimated by any distance metrics for distributions, e.g., Jensen-Shannon divergence.

system	# instances	# classes	# aliases	# profiles
KP	153,506	666	8,678	5,529
ND	209,040	470	23	1,204
AT	158,745	500	*N/A*	*N/A*
BD	157,428	458	*N/A*	*N/A*

TABLE I: Summarization of datasets used in experiments.

Retrieval: Given the label of an instance q from one specific system, the analyst intends to find its alternative names in another system. Similar to folding-in but in the opposite direction, with the help of reverse correspondence, we translate the class memberships of q into the space of target local classification. Specifically consider a particular class c in the target classification, the probability of q belonging to c is estimated by $\sum_{c^*} [\mathbf{M}]_{q,c^*} [\mathbf{F}]_{c^*,c}$. This result can be interpreted in multiple ways. For example, one can select the class c with the largest probability as the most likely class of q.

IV. EVALUATION

In this section we present an empirical analysis of LATIN using real datasets and concrete use cases. We start with introducing the experiment settings.

A. Experiment Settings

Our experiments use the following datasets collected from real-life anti-virus systems. The first dataset is the collection of 235,974 malware instances with binary digests obtained from *VX Heaven*[4]. We feed them to four top-ranked anti-virus systems (which we refer to using the acronyms *KP*, *AT*, *ND*, and *BD* for anonymity) and collect their classification results. The numbers of detected instances and classes are summarized in Table I. Meanwhile among these anti-virus systems, *KP* and *ND* have online threat encyclopedia available, from which we retrieve all available aliases and behavior profile information. Furthermore we feed the corpus of binary digests to the online malware analysis service *Anubis*[5] and collect their behavior description information. After combining all these datasets and de-duplicating instances appearing in multiple sources, we collect a total of 241,530 distinct malware instances associated by both aliases and behavior profiles. We randomly sample 60% (144,918) instances from the dataset together with their behavior profiles as training set and regard the rest 40% (96,612) as the test set.

We implement a baseline method which solely relies on the labels of query instances (B) and three variants of LATIN: one that uses both the labels and the mediated classification (Latin^L), the second one that additionally exploits the alias information (Latin^A), and the third one that further accounts for their behavior profiles (Latin^P). In both Latin^A and Latin^P, for each instance, only one of its aliases is used.

All the algorithms are implemented in Python. The experiments are conducted on a Linux box running Intel i7 2.6 GHz processor and 16 GB memory. We determine λ and η in Eq.(2) for concrete tasks using ten-fold cross validation.

B. Experimental Results

We demonstrate the application of LATIN in three concrete use cases and evaluate its empirical efficacy.

[4]http://vxheaven.org

[5]http://anubis.iseclab.org

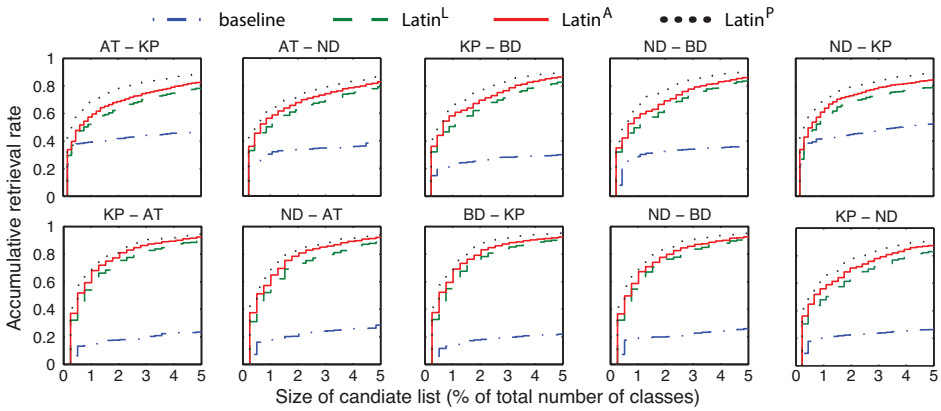

Fig. 4: Cumulative retrieval rate of label retrieval models with respect to varying candidate list size.

Use Case 1: Searching Others' Systems using Your Own Words: One crucial functionality for cross-system information sharing is to support the retrieval of content from one system using the terms of the other. For example, an analyst may wish to search other knowledge bases for information relevant to a particular malware variant using any of its available information (e.g., label, alias, or behavior profile). We show how well LATIN supports this scenario in terms of effectiveness of retrieving the labels of given malware instances in other knowledge bases, which can be easily generalized to other types of information.

For each pair of anti-virus systems, we construct 5,000 retrieval queries; each query is an instance with its label from the source system, its alias in another system (different from the target system), and its behavior profile, and the ground truth is its class label in the target system. Recall that LATIN gives a probability distribution over all the classes in the target system indicating their likelihood of being the true label for the query instance. We thus rank these suggested labels in descending order of their likelihood and pick the top ones to form the candidate list. We measure the accuracy of LATIN using the cumulative rate that the correct class label appears in the candidate list as the list size varies.

Fig.4 illustrates the results as the candidate list size varies from 0% to 5% of the total number of classes of the target system. As expected, the models that leverage mediated classification (Latin^L, Latin^A, and Latin^P) all significantly outperform the baseline method (B) that only compares the labels syntactically. For example, when the list size is fixed as 2.5% (about 10 classes), all variants of LATIN achieve accuracy ranging from 75% to 97% in all the cases. Meanwhile we observe that the incorporation of alias and behavior profile information improves the retrieval accuracy over the basic version. For example, in the case of *KP-AT*, the first guess of Latin^P achieves accuracy over 42%, in comparison of that around 34% and 30% by Latin^A and Latin^L, respectively.

Use Case 2: Integrating Threat Reports by Multiple Systems: It is not uncommon in today's enterprise environments that multiple anti-virus systems are deployed together to improve the protection coverage. However the task of integrating the results reported by these systems is not trivial [8]. For example, an analyst may wish to understand whether two instances detected by two systems essentially belong to a same class. To this end, we introduce the functionality of *matching*, which determines the similarity of two malware instances (reported by two systems) given any of their available information (e.g., labels, aliases, behavior profiles).

For each pair of anti-virus systems, we use the instances in the test set to generate 5,000 positive and 5,000 negative cases, wherein a positive case comprises the aliases of an instance in the two systems, while a negative case consists of a pair of labels corresponding to two different instances from the two systems. Note that we use instance-level ground truth (whether two instances are the same) to evaluate LATIN in determining whether two instances belong to a same class; thus the false positive and false negative rates here are over-estimated.

We use the receiver operating characteristic (ROC) curve to measure the matching accuracy of LATIN. Intuitively the ROC curve shows the relationship of true positive and false positive rates of a classifier as the discrimination threshold varies. We set the discrimination threshold on the distance between the projection of two instances in the space of mediated classification, which varies from 0 to 1. As in Use Case 1, we implement a baseline method (B) and two variants of LATIN, Latin^L and Latin^A, wherein B uses the the similarity metric defined by Eq.(1). Note that here we focus on the label/alias information and consider comparing two malware instances based on their behavior as an orthogonal direction.

In Fig.5 we observe that in all the cases both Latin^L and Latin^A significantly outperform the baseline method. For example, with false positive rate fixed as 0.05, the true positive rates of Latin^A and Latin^L differ about 0.3 in most cases. This is explained by the diverse expertise of different anti-virus systems: combining evidences from multiple systems can help pinpoint the classes of query instances in the mediated classification. The ROC curves also shed a light on how to set the discrimination threshold for label matching models to determine the equivalence of two labels. We can select the "elbow point" of an ROC curve as the threshold for it maximizes the ratio of true positive and false positive rates.

Use Case 3: Calibrating Performance Measures of Malware Analysis Tools: Due to the lack of standardized malware labeling, it is difficult to interpret the performance measures of malware analysis tools evaluated over self-labeled test sets. Here we show that LATIN is able to calibrate the performance of these tools provided that only the (local) malware labels of the testing sets are available.

Specifically we consider malware *clustering* which groups malware instances that exhibit similar behavior patterns. Typ-

$$\text{Precision} = \frac{1}{n}\sum_{i=1} \max(|C_i \cap R_1|, |C_i \cap R_2|, \ldots, |C_i \cap R_r|)$$

$$\text{Recall} = \frac{1}{n}\sum_{i=1}^{r} \max(|C_1 \cap R_i|, |C_2 \cap R_i|, \ldots, |C_c \cap R_i|)$$