Rebuilding the Tower of Babel[*] : Towards Cross-System Malware Information Sharing

Ting Wang
IBM Research
Yorktown Heights, NY
tingwang@us.ibm.com

Shicong Meng
Facebook, Inc
Menlo Park, CA
shicong@fb.com

Wei Gao
University of Tennessee
Knoxville, TN
weigao@utk.edu

Xin Hu
IBM Research
Yorktown Heights, NY
huxin@us.ibm.edu

ABSTRACT

Anti-virus systems developed by different vendors often demonstrate strong discrepancies in how they name malware, which signficantly hinders malware information sharing. While existing work has proposed a plethora of malware naming standards, most anti-virus vendors were reluctant to change their own naming conventions. In this paper we explore a new, more pragmatic alternative. We propose to exploit the correlation between malware naming of different anti-virus systems to create their *consensus classification*, through which these systems can share malware information without modifying their naming conventions. Specifically we present `Latin`, a novel *classification integration* framework leveraging the correspondence between participating anti-virus systems as reflected in heterogeneous information sources at instance-instance, instance-name, and name-name levels. We provide results from extensive experimental studies using real malware datasets and concrete use cases to verify the efficacy of `Latin` in supporting cross-system malware information sharing.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; D.4.6 [**Security and Protection**]: Invasive software

Keywords

Malware Naming; Classification Integration; Consensus Learning

1. INTRODUCTION

Mitigating and defending against malware threats (e.g., viruses, worms, rootkits, and backdoors) has been a prominent topic for security research for decades. A plethora of anti-virus (AV) systems have been developed by different vendors (e.g., `Kaspersky`, `Symantec`, `Norton`). These systems often feature drastically different expertise (details in § 2); conceivably it is beneficial to integrate malware intelligence across different systems, thereby leading to more comprehensive understanding of the entire "malware universe" and helping build defense systems with broader coverage and more informative diagnosis. In reality, however, the sharing of malware threat information is still limited to the scale of a few systems [8], due to challenges such as conflict of interest, discrepancy of information, and incompatibility of data schemas.

Here we focus on one potent challenge - the discrepancy of malware naming in different AV systems. Conceptually an AV system comprises two core parts, malware taxonomy and underlying classification rules. The *name* assigned to an malware instance specifies its *class* in the taxonomy. For example, "*Trojan.Spy*" in Figure 1 represents a class of malware spying on user activities. Moreover names of a set of instances essentially dictate a *partitioning* of the set into different classes. Due to lack of standards, AV systems developed by different vendors often demonstrate strong discrepancies in how they name malware instances, both syntactically (e.g., "*Spy-Trj*" versus "*Trojan.Spy*") and semantically (e.g., whether two instances belong to a same class). Hence sharing malware information solely based on malware names may lead to unpredictable consequences, because similar names could refer to completely distinct concepts by different systems.

Existing work has attempted to remedy this issue by proposing various malware naming standards; however most major AV vendors were reluctant to change their naming conventions [4, 10, 25]. We argue that even if they did, the semantic discrepancy would still exist; for example, different vendors may hold conflicting opinions regarding how a set of malware instances should be classified, although they might fully agree on the syntactic representation of taxonomy. One may suggest use more intrinsic features (e.g., MD5 hash of malware binaries) instead of names to identify malware instances. This option however has the drawbacks such as lacking interpretability and eradicating all the semantic relationships between malware (e.g., variation [23]).

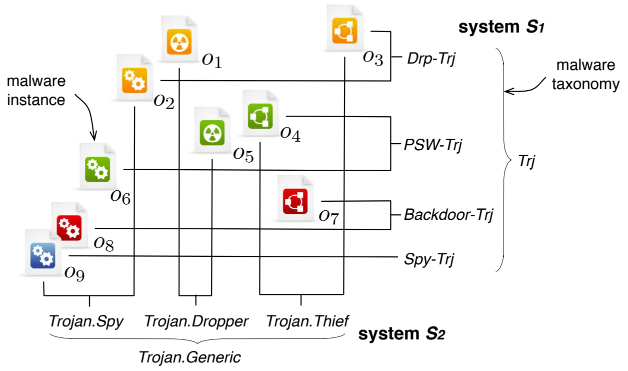


Figure 1: Schematic example of malware classification by two AV systems, which demonstrate discrepancies in both syntactic representation of taxonomy and semantic entailment of categorization.

[*]The narrative of the tower of Babel is recorded in *Genesis* 11: everyone in the tower spoke the same language.

CIKM'14, November 3–7, 2014, Shanghai, China.
Copyright 2014 ACM 978-1-4503-2598-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2661829.2662086.

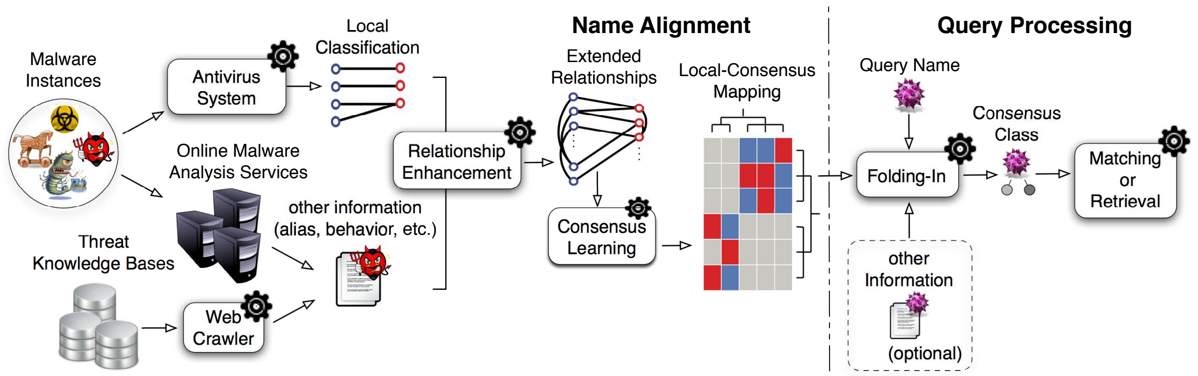


Figure 2: `Latin`: flowchart of classification integration, which comprises name alignment and query processing.

Motivated by this, we explore a new, more pragmatic alternative premised on the fundamental assumption below: regarding a given set of malware instances, despite the disparate classification by individual AV systems, there exists certain "consensus classification" agreed by a majority of them. Therefore, via the probabilistic mappings between their "local" classification and consensus classification, individual AV systems are able to share malware information with each other without modifying their own naming conventions. Note that we are not arguing that the consensus classification corresponds to the ground truth; rather, we consider it as a common ground to facilitate malware information sharing.

To this end, we present `Latin`, a novel *classification integration* framework which exploits the correlation between malware naming of different AV systems to create the consensus classification. Specifically (i) for participating AV systems, `Latin` exploits the correspondence between their local classification as reflected in various heterogenous information at instance-instance, instance-name, and name-name levels; (ii) it then applies novel consensus learning methods over such multifaceted correspondence to derive the consensus classification; and (iii) with the help of local-consensus mappings, `Latin` is capable of answering several fundamental types of queries including: *matching* - determining whether given names from different systems refer to similar instances, and *retrieval* - identifying counterparts of given names in other AV systems. Our proposal is illustrated in Figure 2.

With extensive experimental studies using real malware datasets and concrete use cases, we show that:

- `Latin` effectively exploits various heterogenous information including: AV system scan results, malware encyclopedia records, and online malware analysis reports.
- `Latin` has direct impact on a range of real-world applications. For example, it allows analysts to search for relevant information in multiple AV systems when a newly found malware variant is only partially understood; it supports efficient integration of malware analysis reports generated by different AV systems; it also helps calibrate performance measures of malware analysis tools conducted using self-composed malware datasets.
- `Latin` scales up to large-scale malware datasets (e.g., millions of instances) under modest hardware configuration.

The remainder of the paper proceeds as follows. § 2 presents an empirical study on the current state of malware naming discrepancies and motivates the design of `Latin`. § 3 and § 4 describe in detail the design and implementation of core components of `Latin`. The proposed solution is empirically evaluated in § 5. § 6 surveys relevant literature. The paper is concluded in § 7.

2. CHAOS OF MALWARE NAMING

To understand and characterize the issue of malware naming discrepancies, in this section we conduct an empirical study on a set of representative AV systems.

2.1 Study Setup

We prepare a large malware corpus (235,974 distinct instances) from `VX Heavens`[1], a well-known malicious code repository. The corpus covers most major types of malware (e.g., viruses, worms, backdoors, rootkits). We feed these instances to `VirusTotal`[2], an online malware scanner which generates analysis reports by querying multiple AV systems. We intend to understand malware naming discrepancies via analyzing the scan results.

We observe that except for generic- (e.g., "*Malware.Generic*") and heuristic-type names (e.g., "*BehavesLike-Win32-Trojan*"), most malware names assigned by target AV systems contain two common fields: $\langle type \rangle$ (primary category, e.g., "*Trojan*") and $\langle family \rangle$ (secondary category, e.g., "*Spy*"). In the following we focus on these two fields and consider their combination as malware *names*.

We identify two primary types of malware naming discrepancies: *syntactic* discrepancy and *semantic* discrepancy. The former refers to the disparate syntactic representation of malware names used by different AV systems; while the latter refers to the conflicting opinions of different systems regarding the relationships between given malware instances. Below we detail both types of discrepancies observed in the scan results.

2.2 Syntactic Discrepancy

To quantify syntactic discrepancy, we apply approximate string matching over malware names (i.e., $\langle type \rangle.\langle family \rangle$) given by four AV systems[3] (S_1, S_2, S_3, and S_4). Figure 3 shows for each AV system the cumulative distribution of names matching certain counterparts in another system under varying minimum edit distance constraint. For each AV system the average length of malware names is shown as the right boundary.

We observe that in all the cases the edit distance between "matchable" names from different systems is small (i.e., $1 \sim 4$ characters), in contrast of the average name length. This implies that most AV systems adopt similar taxonomies. This also suggests one possible solution to addressing syntactic discrepancy (as proposed in previous studies [4, 10, 25]): finding corresponding malware names in different AV systems and imposing unified naming standards.

[1]http://vxheaven.org

[2]https://www.virustotal.com

[3]We intentionally mask specific names of AV systems to avoid possible controversies.

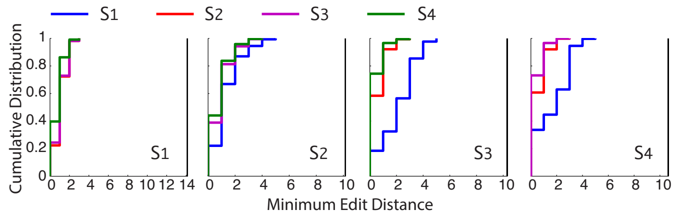


Figure 3: Cumulative distribution of matchable malware names between different AV systems with respect to varying edit distance constraint. The right boundary of horizontal axis represents the average length of names in the corresponding system.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
S1	1.00	0.15	0.42	0.32	0.23	0.25	0.32	0.25	0.09	0.18	0.19	0.34
S2	0.15	1.00	0.13	0.09	0.12	0.19	0.10	0.16	0.08	0.12	0.10	0.11
S3	0.42	0.13	1.00	0.23	0.29	0.46	0.23	0.45	0.09	0.22	0.12	0.17
S4	0.32	0.09	0.23	1.00	0.54	0.39	0.98	0.30	0.05	0.22	0.14	0.19
S5	0.23	0.12	0.29	0.54	1.00	0.35	0.55	0.22	0.08	0.17	0.22	0.30
S6	0.25	0.19	0.46	0.39	0.35	1.00	0.39	0.40	0.12	0.31	0.52	0.47
S7	0.32	0.10	0.23	0.98	0.55	0.39	1.00	0.30	0.05	0.21	0.14	0.19
S8	0.25	0.16	0.45	0.30	0.22	0.40	0.30	1.00	0.11	0.17	0.21	0.47
S9	0.09	0.08	0.09	0.05	0.08	0.12	0.05	0.11	1.00	0.06	0.07	0.12
S10	0.18	0.12	0.22	0.22	0.17	0.31	0.21	0.17	0.06	1.00	0.16	0.26
S11	0.19	0.10	0.12	0.14	0.22	0.52	0.14	0.21	0.07	0.16	1.00	0.22
S12	0.34	0.11	0.17	0.19	0.30	0.47	0.19	0.47	0.12	0.26	0.22	1.00

Figure 4: Semantic similarity matrix of twelve representative AV systems, with each cell representing the Jaccard similarity of equivalent instance pairs in two AV systems.

2.3 Semantic Discrepancy

Unfortunately the syntactic discrepancy is only a part of the story. More subtle are various semantic discrepancies. First, similar names may refer to completely distinct instances by different AV systems, for designers and developers of different systems may hold conflicting opinions about the connotation of these names. Thus even if one could completely eradicate syntactic discrepancy via standardizing naming formats, this semantic discrepancy would still exist. Second, recall that given a set of instances, their names essentially partition the set into different classes; any two instances in a same class are considered "equivalent". Indeed the semantic discrepancy reflects at the disparity of such equivalence relationships in different AV systems. In the following we focus on the second type of semantic discrepancy.

To quantify this semantic discrepancy, we compare the pairs of equivalent malware instances in different AV systems. More specifically, consider two AV systems S_k $(k = 1, 2)$; denote by P_k the set of equivalent instance pairs in S_k. We measure the semantic discrepancy between S_1 and S_2 by calculating the Jaccard similarity of P_1 and P_2: $J(P_1, P_2) = |P_1 \cap P_2|/|P_1 \cup P_2|$.

We apply this metric to scan results given by twelve top-ranked AV systems [7], with results shown in Figure 4 wherein each cell represents the Jaccard similarity of equivalent pairs in two systems. It is observed that for the majority of systems, their pairwise similarity is below 0.5. This implies that (i) the semantic discrepancy is substantial and prevalent in major AV systems and (ii) the feasibility of imposing unified naming standards is questionable given the magnificence of discrepancy.

2.4 LATIN In A Nutshell

Instead of attempting to directly reconcile either syntactic or semantic discrepancy, the objective of Latin is to create the "consensus classification" maximally agreed by participating AV systems and to use it as a common ground to support cross-system malware information sharing.

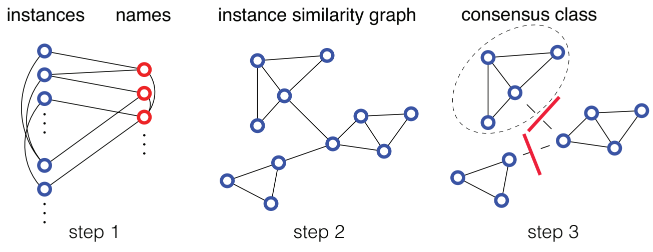


Figure 5: Illustration of name alignment process.

Figure 2 illustrates the system architecture of Latin, which comprises two major components: *name alignment* and *query processing*. Informally name alignment is the process of constructing the consensus classification using all available (heterogenous) information (e.g., AV system scan results, malware encyclopedia records, online malware analysis reports). The way the consensus classification is constructed also entails its mappings with the "local" classification of each individual AV system. These mappings are often probabilistic (i.e., "soft correspondence") due to the possible conflicting opinions from different AV systems, e.g., some may divide a set of malware instances into multiple classes whereas others may prefer lumping them together.

With the consensus classification as reference, Latin is capable of answering several fundamental types of queries. It works in two phases: (i) *folding-in*, projecting names in the query to the space of consensus classification via local-consensus mappings in conjunction with other available information relevant to the query (e.g., malware behavior description), and (ii) *query execution*, e.g., reasoning about the relationships between query names on the consensus classification (matching) or further projecting query names to target AV systems (retrieval).

The details of name alignment and query processing are presented in § 3 and § 4, respectively.

3. NAME ALIGNMENT

As illustrated in Figure 5, name alignment entails three steps: (1) *relationship extraction*, wherein it derives all types of relationships (i.e., instance-name, instance-instance, name-name) from available heterogenous information sources; (2) *ISG construction*, in which it creates a structure called *instance similarity graph* (ISG), which maximally preserves all relationships derived in the previous step; and (3) *consensus classification derivation*, in which it partitions ISG into high-quality clusters (corresponding to consensus classes) and derives probabilistic local-consensus mappings. Next we elaborate each step.

3.1 Relationship Extraction

In this step, Latin identifies all types of relationships among given malware instances and their names assigned by participating AV systems. To be succinct yet reveal the important issues, we use three heterogenous information sources as example to show how to extract such relationships, namely, AV *system scan results*, *online malware analysis reports*, and *malware encyclopedia records*.

3.1.1 AV System Scan Results

We feed the corpus of malware instances to each AV system and obtain their scan results, which list the name assigned to each instance. We then incorporate this local classification (i.e., instance-name relationships) into a weighted bipartite graph as shown in Figure 6 (a), wherein nodes on the left side represent malware instances, ones on the right side represent names appearing in the classification results, and instance i and name c are adjacent only if

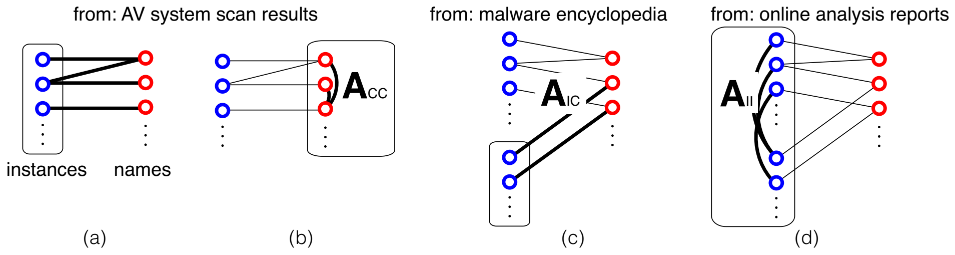
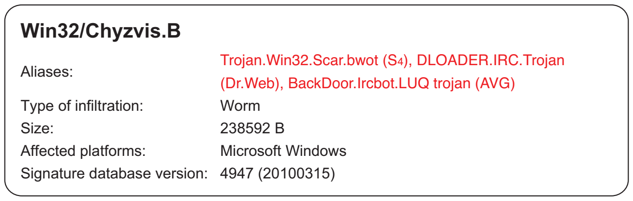

from: AV system scan results
instances
names
(a)
$\mathbf{A}_{CC}$
(b)
from: malware encyclopedia
$\mathbf{A}_{IC}$
(c)
from: online analysis reports
$\mathbf{A}_{II}$
(d)

Figure 6: Incorporation of instance-instance, instance-name, and name-name relationships.



Win32/Chyzvis.B

Aliases:	Trojan.Win32.Scar.bwot (S4), DLOADER.IRC.Trojan (Dr.Web), BackDoor.Ircbot.LUQ trojan (AVG)
Type of infiltration:	Worm
Size:	238592 B
Affected platforms:	Microsoft Windows
Signature database version:	4947 (20100315)

Figure 7: A sample entry of threat encyclopedia.

i is categorized as c by a certain AV system. Note that we differentiate identical names from different AV systems as they may have varying semantics.

As constructed over a common set of instances, this graph encodes correlation between names from different systems.

EXAMPLE 1. *An instance classified as* "Trojan-Proxy.Ranky" *by* $\mathtt{S}_4$ *and* "Backdoor.Bot" *by* $\mathtt{S}_2$ *correlates these two names.*

Moreover from the scan results we can also derive name-name relationships. Recall that an malware name c consists of two parts, $\langle type \rangle$ and $\langle family \rangle$, denoted by c_t and c_f. We compare the type and family fields of malware names c and c' separately. Specifically let $\mathtt{ngram}(c_t)$ denote the set of n-grams of the type field of c ($n = 2$ in our implementation). The similarity of c_t and c'_t is measured using their Jaccard coefficient:

$$\mathtt{jac}(c_t, c'_t) = \frac{|\mathtt{ngram}(c_t) \cap \mathtt{ngram}(c'_t)|}{|\mathtt{ngram}(c_t) \cup \mathtt{ngram}(c'_t)|}$$

A similar definition applies to the family field. The similarity of names c and c' is then defined as:

$$\mathtt{k}(c, c') = \theta\mathtt{jac}(c_t, c'_t) + (1 - \theta)\mathtt{jac}(c_f, c'_f) \quad (1)$$

where θ balance the importance of type and family fields.

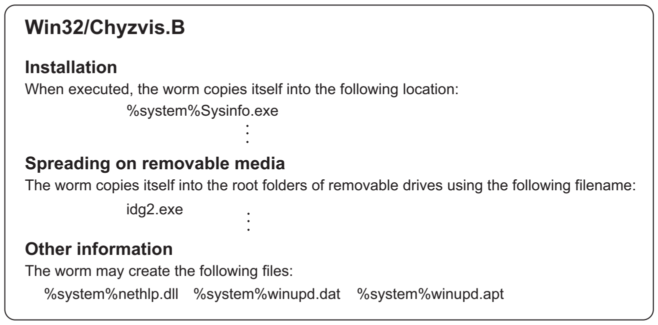
Now let $\mathbf{A_{CC}} \in \mathbb{R}^{m \times m}$ denote the name-name similarity matrix with $[\mathbf{A_{CC}}]_{c,c'} = \mathtt{k}(c, c')$ as shown in Figure 6 (b).

Unfortunately for a particular AV system, the coverage of its local classification could be fairly limited or biased, highly dependent on the available malware corpus. Moreover the instance-name and name-name relationships alone are insufficient for constructing optimal consensus classification; rather, instance-instance relationships also need to be taken into account. Next we show how `Latin` integrates other forms of malware intelligence, thereby providing coverage beyond the corpus of malware instances and the instance-name and name-name relationships.

3.1.2 Malware Encyclopedia Records

Many AV system vendors today make available online *threat encyclopedias* that provide information of a number of malware instances well-studied by analysts. For example, Figure 7 shows a sample entry of malware "*Worm.Win32/Chyzvis.B*" from $\mathtt{S}_2$.

In each encyclopedia entry, the field of *aliases* (i.e., "also-known-as") is particularly valuable, for it specifies equivalent names of the malware instance in other AV systems as identified by security analysts; therefore they also reflect instance-name relationships, which complement the corpus of malware instances.



Win32/Chyzvis.B

Installation
When executed, the worm copies itself into the following location:
%system%Sysinfo.exe
⋮

Spreading on removable media
The worm copies itself into the root folders of removable drives using the following filename:
idg2.exe
⋮

Other information
The worm may create the following files:
%system%nethlp.dll %system%winupd.dat %system%winupd.apt

Figure 8: A sample entry of behavior profile.

EXAMPLE 2. *As highlighted in Figure 7, the aliases of malware instance* "Worm.Win32/Chyzvis.B" *indicate that it is classified as* "Worm.Chyzvis" *by* $\mathtt{S}_2$ *but* "Trojan.Scar" *by* $\mathtt{S}_4$.

As shown in Figure 6 (c), we incorporate these instance-name relationships to the graph created using the malware scan results. Let $\mathcal{I}$ and $\mathcal{C}$ denote the set of instances and names in the 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-name relationships, with $[\mathbf{A_{IC}}]_{i,c} = 1$ if i is classified as c in a certain system and 0 otherwise.

3.1.3 Online Malware Analysis Reports

Another valuable information source is online malware analysis services (e.g., `Anubis`[4]) which execute submitted malware binaries in sandbox environments and report observed activities (called "behavior profiles"), as shown in Figure 8. `Latin` queries `Anubis` with the malware corpus and collects their analysis reports.

The behavior profile of an malware instance gives its detailed technical description, such as how it gets installed, how it spreads, and how it damages an infected system. Thus by comparing behavior profiles of two instances, we are able to derive their semantic similarity, i.e., instance-instance relationships, which complements scan results and malware encyclopedia.

However the challenge lies in that: (i) the schema of behavior profile varies from case to case, e.g., certain sections present in one profile may be missing in another, or some profiles may use free text description while others may use pseudo-code like languages; (ii) the detail level of documentation also varies, e.g., different parts may be missing for different instances, reflecting the varying depth of understanding analysts have regarding the behavior of different instances. Such inconsistency makes directly comparing 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 representation, the keywords of behavior profiles share a

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

common vocabulary, specific to security research community. We resort to security glossaries (e.g., SANS[5] and RSA[6] glossaries) to build such a vocabulary, onto which each behavior profile is projected. More formally, assuming a vocabulary $\mathcal{T}$, the behavior profile of instance $i \in \mathcal{I}$ is transformed into a bag of words: $\mathtt{p}_i = \{\ldots, w_{i,t}, \ldots\} (t \in \mathcal{T})$ where $w_{i,t}$ denotes the frequency of term t for i's behavior profile.

EXAMPLE 3. *In Figure 8, the behavior profile contains the following set of security terms from the vocabulary:* {worm, removable media, backdoor, root, UPX}.

The for each behavior profile, we consider its probability distribution over the latent topics generated by Latent Dirichlet Allocation (LDA) [3] as its feature vector. The computation of the similarity of feature vectors is fairly straightforward, e.g., using cosine distance. We consider this similarity measure encodes the instance-instance relationship from behavior profile perspective, which complement the instance-name and name-name relationships, as shown in Figure 6 (d). In the following we use $\mathbf{S^p} \in \mathbb{R}^{n \times n}$ to denote the instance-instance relationships derived from their behavior profiles, where $[\mathbf{S^p}]_{i,i'} = \mathtt{k}(\mathtt{p}_i, \mathtt{p}_{i'})$.

3.2 Instance Similarity Graph Construction

In this step, Latin constructs *instance similarity graph* (ISG), a structure maximally preserving all the relationships derived in the previous step. Intuitively ISG embeds all instance-name and name-name relationships in the space of instance-instance relationships.

More specifically, given instance-name relationship matrix $\mathbf{A_{IC}}$, two malware instances are considered similar if they are associated with similar names. We therefore derive the similarity matrix $\mathbf{S^r} \in \mathbb{R}^{n \times n}$ for malware instances based on their relationships with different names: $\mathbf{S^r} = \mathbf{A_{IC}} \mathbf{A_{IC}}^T$.

Given name-name relationship matrix $\mathbf{A_{CC}}$, we populate it to the space of malware instances by following a procedure similar to that of deriving $\mathbf{S^p}$ from behavior profiles. The details are omitted here due to the space limitations. Below we use $\mathbf{S^n} \in \mathbb{R}^{n \times n}$ to represent the instance-instance similarity matrix derived from the name-name relationships.

Now we have collected evidence of instance-instance similarities in forms of three similarity matrices $\mathbf{S^r}$, $\mathbf{S^p}$, and $\mathbf{S^n}$. All these evidences are integrated into an overall similarity measure using a linear combination:

$$\mathbf{S} = \alpha \mathbf{S^n} + \beta \mathbf{S^p} + (1 - \alpha - \beta) \mathbf{S^r} \tag{2}$$

where α, β ($\alpha, \beta \geq 0$ and $\alpha + \beta \leq 1$) control the relative weight of different evidences (their setting will be discussed in § 5). For mathematical convenience, we assume the diagonal elements of $\mathbf{S}$ are set as zero. Indeed $\mathbf{S}$ encodes ISG.

3.3 Consensus Classification Derivation

The previous steps extract comprehensive relationships between malware instances and names. Next we aim at applying consensus learning methods over these relationships to derive the consensus classification. However we face the major challenges of scalability. As most existing consensus learning methods rely heavily on techniques such as spectral graph analysis [18], thereby scaling poorly for our case (e.g., over 235K instances and over 25K classes in our experiments).

We derive a novel consensus learning algorithm based on power iterative methods [15], which significantly reduces the running time. Moreover we integrate this algorithm with clustering quality metrics to automatically determine the optimal number of classes in one run. Specifically we achieve this goal as follows: (i) finding an extremely low-dimensional embedding of malware instances that captures all instance-instance, instance-name, and name-name relationships; (ii) identifying the optimal partitioning of the embedding as the consensus classification; and (iii) deriving the mappings between local and consensus classification through the relationships between instances and names.

3.3.1 Optimal Consensus Classification

Denote by $\mathbf{W}$ the row-normalized version of $\mathbf{S}$:

$$\mathbf{W} = \mathtt{diag}^{-1}(\mathbf{S} \cdot \mathbf{1})\mathbf{S}$$

It is known that the top eigenvectors of $\mathbf{W}$ give a lower dimensional embedding of instances that preserves their similarity structures. We thus apply the power iteration method to deriving a one-dimensional vector $\mathbf{v}^*$, which not only well approximates this embedding but also is efficiently computable.

The extreme low dimensionality of embedding $\mathbf{v}^*$ allows us to readily apply sophisticated clustering methods to find high-quality partitioning of $\mathbf{v}^*$, which corresponds to majority-agreed grouping of malware instances. We devise a simple, dynamic programming algorithm for this task that requires no prior knowledge of the number of clusters. The details are referred to Appendix A.

3.3.2 Local-Consensus Mappings

The bijection between embedding $\mathbf{v}^*$ and malware instances $\mathcal{I}$ implies that the partitioning of $\mathbf{v}^*$ corresponds to the classification of $\mathcal{I}$, which we consider as the consensus classification. Then by leveraging instance-name relationships $\mathbf{A_{IC}}$, we identify the probabilistic mappings between local and consensus classification. Specifically let c and c' denote a name in the local classification and a class in the consensus classification, respectively. The *forward correspondence* $[\mathbf{F}]_{c,c'}$ is the probability that an instance with name c belongs to class c', which is calculated as the fraction of instances associated with both c and c', $|c \cap c'|/|c|$.

Similarly we derive the *reverse correspondence* from the consensus classification to each local classification. However as the consensus classification may contain instances not covered by the local classification, we have to exclude those instances in the computation. It is worth emphasizing that while it is possible to derive the soft correspondence between a pair of local classifications directly, the drastically varying coverage of AV system often lead to mappings of inferior quality.

4. QUERY PROCESSING

Equipped with both forward and reverse correspondence between local and consensus classification as reference, Latin effectively processes several fundamental types of queries.

4.1 Folding-In

The prerequisite of query processing is to first map malware instances in queries (from specific AV systems) to the space of consensus classification.

Given instance q and class c in the consensus classification, the *class membership* $[\mathbf{M}]_{q,c}$ specifies the probability that q belongs to c. Intuitively the class memberships of instance q with respect to all classes in the consensus 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^* in the consensus classification and show how to estimate $[\mathbf{M}]_{q,c^*}$.

[5]www.sans.org/security-resources/glossary-of-terms
[6]www.rsa.com/glossary

4.1.1 Query with Only Name

We start with the simple case that only the name c_q of q is available. We assume c_q has been normalized into the format of $\langle$*type*$\rangle$ + $\langle$*family*$\rangle$. If c_q appears in the local classification, the folding-in of q is straightforward, which is the forward correspondence of the class indicated by c_q, i.e., $[\mathbf{M}]_{q,c^*} = [\mathbf{F}]_{c_q,c^*}$.

In the case that c_q is not found in the local classification, we need to first derive the relationships between c_q and malware names $\{c\}$ in the local classification using the name similarity metric defined in Eqn.(1) and then integrate the forward correspondence between $\{c\}$ and c^* to estimate $[\mathbf{M}]_{q,c^*}$. More formally,

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

Here the summation iterates over all names in the local classification which c_q associates with.

4.1.2 Query with Alias

With (optional) other information relevant to q available, we can further improve the estimation quality of $\mathbf{M}_{q,c^*}$. Consider the case that the aliases of q are available. We first compute $\mathbf{M}_{q,c^*}$ according to Eqn.(3) for each of its aliases with respect to the corresponding local classification and then use their average value as the overall estimation of $\mathbf{M}_{q,c^*}$.

4.1.3 Query with Behavior Profile

In this case that the behavior profile $\mathtt{p}_q$ of q is available, for each name c in the local classification, we incorporate the behavior profile similarity between $\mathtt{p}_q$ and all instances associated with 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 use their average value as the overall estimation.

Finally we integrate the estimation from both name (and alias) and behavior profile. Let $[\mathbf{M}^\mathbf{n}]_{q,c^*}$ and $[\mathbf{M}^\mathbf{p}]_{q,c^*}$ denote the name- and profile-based estimation respectively. The overall estimation is defined as $[\mathbf{M}]_{q,c^*} = \frac{\alpha}{\alpha+\beta}[\mathbf{M}^\mathbf{n}]_{q,c^*} + \frac{\beta}{\alpha+\beta}[\mathbf{M}^\mathbf{p}]_{q,c^*}$, where α and β are the same parameters in Eqn.(2) controling the relative importance of names and behavior profiles.

4.2 Matching and Retrieval

Once the malware instances in queries are projected to the space of consensus classification, the query processing is fairly straightforward. Here we consider the nontrivial case that the instances are not found in any local classification.

4.2.1 Matching

In a matching query, a pair of instances (q_s, q_t) from different AV systems are given, the analyst intends to know whether q_s and q_t belong to the same class in the consensus classification. Recall that we use the distribution $\mathbf{M}_q$ to represent the membership of q with respect to each class in the consensus classification. Let $\mathbf{M}_{q_s}$ and $\mathbf{M}_{q_t}$ denote 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; in current implementation, we use Jensen-Shannon divergence [16] as the metric.

4.2.2 Retrieval

In a retrieval query, an instance q from one specific AV system is given, the analyst intends to identify 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. Concretely consider a malware name c in the target classification, the probability of q associating with 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 name c with the largest probability as the most likely name of q; or one can sum up the probability of names with the same type (e.g., "*Trojan*") and reason about the membership of q at the granularity of malware type.

system	# instances	# names	# aliases
$\mathtt{S}_1$	158,745	500	*N/A*
$\mathtt{S}_2$	209,040	470	23
$\mathtt{S}_3$	157,428	458	*N/A*
$\mathtt{S}_4$	153,506	666	8,678

Table 1: Summarization of datasets used in experiments.

5. EVALUATION

In this section, we present an empirical analysis of `Latin` centering around three metrics: (i) its effectiveness in reconciling the malware naming discrepancies of participating AV systems, (ii) its impact on real-world security applications, and (iii) its operation efficiency and sensitivity to training data. We start with describing the experimental setting.

5.1 Experimental Setting

Our experiments use a set of datasets collected from real AV systems. The first one is the collection of 235,974 distinct malware instances with binaries obtained from `VX Heavens`. We feed this malware corpus to four AV systems ($\mathtt{S}_1$, $\mathtt{S}_2$, $\mathtt{S}_3$, and $\mathtt{S}_4$) and collect their classification results, from which we remove the results with generic- and heuristic-type malware names. The numbers of remaining instances and names are summarized in Table 1. Meanwhile among these AV systems, $\mathtt{S}_2$ and $\mathtt{S}_4$ have online threat encyclopedias available, from which we retrieve all available alias information. Furthermore we feed the corpus of malware instances to the online malware analysis service `Anubis` 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.

All the algorithms are implemented using Python. All the experiments are conducted on a Linux workstation running Intel i7 2.2 GHz processor and 8 GB RAM. By default we set $\theta = 0.6$ (Eqn.(1)) and determine α and β (Eqn.(2)) for concrete tasks (matching or retrieval) using ten-fold cross validation.

5.2 Reconciliation of Naming Discrepancies

In the first set of experiments, we take a close examination of the name alignment component of `Latin` on reconciling the aforementioned semantic discrepancies. While the design of `Latin` allows it to flexibly consume most available intelligence (e.g., aliases, behavior profiles, etc.), here we solely rely on the most commonly available information - the scan results of malware instances by different AV systems - as input to train `Latin` and evaluate the quality of learned consensus classification.

Specifically the consensus classification integrates the scan results of 231,663 distinct instances in the malware corpus (after excluding generic and heuristic-type names). We contrast the consensus and local classifications from these three aspects.

Coverage. The consensus classification provides much broader coverage than any local classification as it integrates intelligence

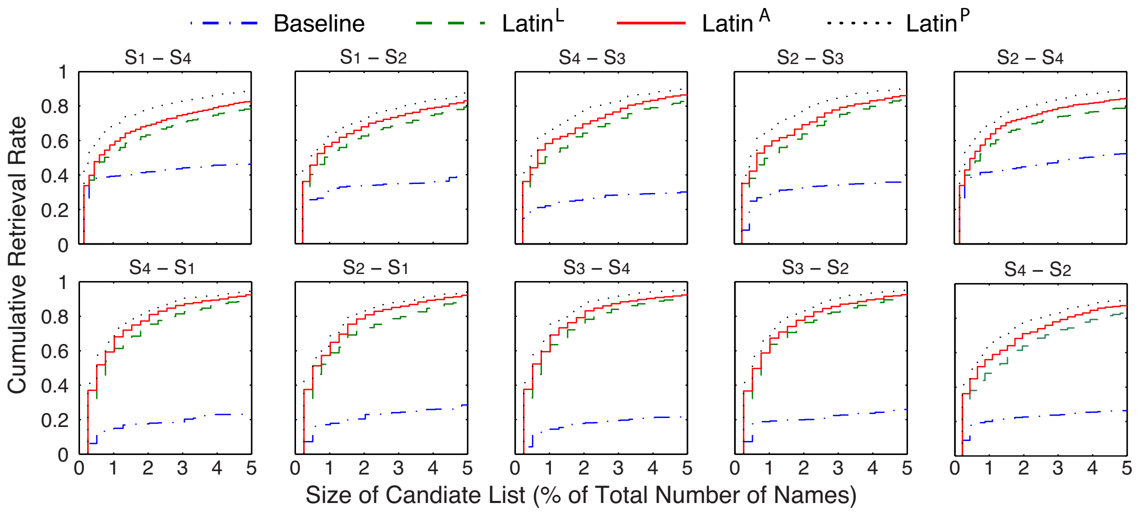


Figure 9: Cumulative retrieval rate of name retrieval models with respect to varying candidate list size.

	S_1	S_2	S_3	S_4	S
size of largest class	16,694	22,864	16,976	16,343	**8,813**
# 5+% classes	3	3	3	3	**1**

Table 2: Statistics of large-size classes in local and consensus classifications (S - consensus classification).

from all individual systems. Specifically with respect to the total number of instances in corpus (235,974), the coverage of individual AV systems varies from 65% to 88% (Table 1), while the consensus classification achieve coverage above 98%.

Granularity. We then measure the statistics of large-size classes in local and consensus classification. Table 2 lists the size of the largest class and the number of classes containing at least 5% of total number of instances in the corpus. The consensus classification (S) features the lowest numbers in both categories, indicating that intuitively the consensus classification prefers detailed over coarse-grained classification.

	S_1	S_2	S_3	S_4	S
S_1		11.1%	12.2%	16.3%	**38.5%**
S_2	17.6%		18.1%	21.3%	**39.5%**
S_3	10.5%	10.3%		13.1%	**35.4%**
S_4	22.1%	13.6%	20.7%		**46.8%**

Table 3: Confusion matrix of equivalent instance pairs agreed by both row and column systems (percentage relative to the total number of equivalent pairs in the row system).

Classification. Further we validate the effectiveness of Latin in reconciling classification conflicts between different AV systems. Table 3 lists the percentage of equivalent instance pairs (i.e., both instances classified to a same class) agreed by two AV systems (represented by respective row and column) with respect to the total number of equivalent pairs in the row AV system. It is clear that in contrast of severe conflicts between local classifications, the consensus classification achieves the maximum agreement with each local classification. This is explained by that for each local classification, the consensus classification inherits all its classification results that are missed by other local classification and that are agreed by a majority of others. This also implies that the consensus classification is able to serve as an ideal common ground for different AV systems to share malware information.

5.3 Real Use Cases

Next we demonstrate the application of Latin in three concrete use cases and evaluate its empirical efficacy.

Use Case 1: Searching Others' Systems Using Your Own Words

One functionality crucial for information sharing between AV systems is to support the retrieval of content from one system using the terms of another. For example, the analyst may wish to search different AV systems for information relevant to a particular malware variant using any of its available information (e.g., name, alias, behavior profile, etc.). We show how well Latin supports this scenario in terms of its efficacy of retrieving names of given malware instances in other AV systems, which can be easily generalized to other types of malware information.

Specifically we randomly sample 60% (144,918) of malware instances from the dataset created in the preparation phase together with their behavior profiles for training and regard the remainder 40% (96,612) as the test set (we will discuss the impact of training data size over the evaluation results shortly). For comparison purpose, we implement a baseline method that directly uses names of query instances in search (Baseline) and three variants of Latin: one that leverages both names of query instances and the consensus classification (Latin^L), one that exploits both names and aliases information as well as the consensus classification (Latin^A), and one that further takes account of behavior profiles of query instances (Latin^P). Particularly in Latin^A and Latin^P, for each instance, only one of its aliases (randomly selected) is used.

For each pair of AV systems we generate 5,000 retrieval queries at random, wherein each query is an malware instance with its name from the source system, its alias in another system (different from the target system), and its behavior profile, while the ground truth is its name in the target system. Recall that rather than suggesting a single name in the target system, Latin gives a probability distribution over all names of the target system representing their likelihood of being the true name for the query instance. We rank these names in descending order in terms of their likelihood and pick the top ones to form the candidate list. We measure the accuracy of Latin in answering retrieval queries using the cumulative rate that the correct name appears in the candidate list as the size of the candidate list grows. It is worth emphasizing that the retrieval accuracy is inherently bounded by the naming discrepancy between two systems, as there may not exist an equivalent counterpart in another system for a given name.

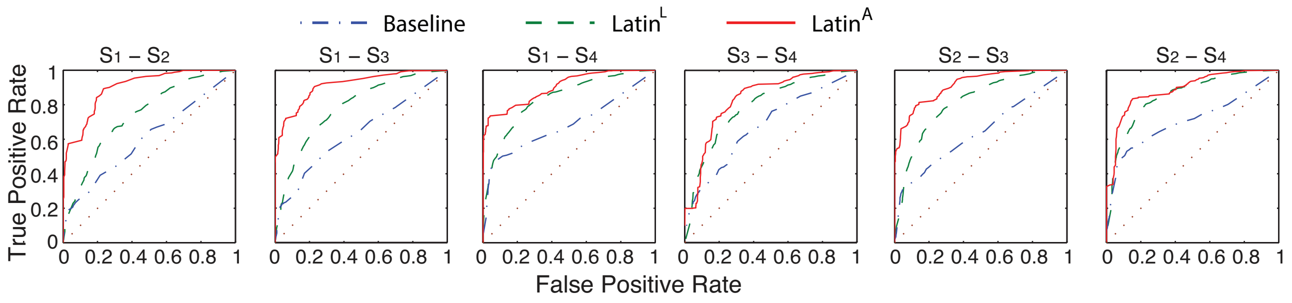


Figure 10: ROC curves of name matching models for six pairs of AV systems.

Figure 9 illustrates how the retrieval accuracy of different models increases as the size of candidate list varies from 0% to 5% of the total number of names (ranging from 393 to 500) of the target system. As expected, the models that leverage the consensus classification (Latin$^{\text{L}}$, Latin$^{\text{A}}$, and Latin$^{\text{P}}$) all significantly outperform the baseline method (Baseline) that only compares names syntactically. For example, when the size of candidate list is fixed as 2.5% (about 10 names), all variants of Latin achieve accuracy ranging from 75% to 97% in all the test cases. Meanwhile it is observed that the incorporation of alias and behavior profile information improves the retrieval accuracy over the basic version. For example, in the case of S$_1$-S$_4$, the first guess of Latin$^{\text{P}}$ achieves accuracy over 42%, in comparison of that around 34% and 30% by Latin$^{\text{A}}$ and Latin$^{\text{L}}$, respectively.

Use Case 2: Caliberating Measures of Malware Analysis Tools

Due to lack of standard benchmarks for AV systems, currently it is still fairly difficult to interpret performance measures of malware analysis tools evaluated over self-composed test sets. Here we show that with the help of Latin it is now possible to benchmark the performance measures of these tools provided that only the (local) malware names of the test sets are available.

Specifically we consider the case of malware clustering which groups malware instances exhibiting similar behavior patterns. Typically the quality of clustering is measured by *precision* and *recall*. Precision measures the exactness of clustering, i.e., how well the clustering divides dissimilar malware into different clusters; while recall measures the completeness of clustering, i.e., how well the clustering groups similar malware into a same cluster. Formally let $(C_1, C_2, \ldots, C_c)$ and $(R_1, R_2, \ldots, R_r)$ be the set of clusters and reference classes respectively, we have (n is the total number of malware instances):

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

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

We apply Malheur [24], one of the state-of-the-art malware clustering tools, to analyzing our malware corpus and use names given by each AV system as reference. Conceivably the quality measures vary significantly with such reference names. Table 4 lists the precision and recall scores of Malheur with respect to each AV system. It is observed that even with the same clustering results, the precision and recall scores vary from 0.37 to 0.48 and from 0.63 to 0.70, respectively. Now by leveraging the local-consensus mappings, one can "recalibrate" the performance measures in the space of consensus classification. More specifically, we construct the calibrated reference as follows. For each instance classified as class c in local classification, we consider it is classified as c' in the consensus classification according to the definition of *forward correspondence* $[\mathbf{F}]_{c,c'}$. The accuracy measures after calibration are listed in Table 4. It is clear that both precision (from 0.32 to 0.34) and recall (from 0.65 to 0.68) scores are more consistent across all four AV systems.

	S$_1$	S$_2$	S$_3$	S$_4$
precision (before)	0.4043	0.3813	0.3675	0.3942
recall (before)	0.6794	0.6827	0.6295	0.7047
precision (after)	0.3398	0.3259	0.3423	0.3235
recall (after)	0.6669	0.6723	0.6532	0.6843

Table 4: Clustering accuracy of Malheur measured against different reference systems before and after calibration.

Use Case 3: Integrating Reports by Multiple AV Systems

It is not uncommon in today's enterprise networks that multiple AV systems are deployed together to improve the protection. However the task of integrating the results reported by these systems is nontrivial [5]. For example, the analyst may wish to understand whether two instances detected by two systems essentially belong to a same malware class. To this end, we introduce the functionality of *matching*, which determines the similarity of malware instances (reported by different AV systems) based on any of their available information (e.g., names, aliases, behavior profiles).

We follow the same partitioning of training and test data as in Use Case 1. For each pair of AV systems, we use malware instances in the test set to randomly generate 5,000 positive and 5,000 negative cases, in which a positive case comprises the aliases of an instance in the two AV systems, while a negative case consists of a pair of names corresponding to two distinct instances from the two systems. Note that we use instance-level ground truth (whether two instances are identical) to evaluate Latin in determining whether two instances belong to the same class; thus the false positive and false negative rates here are overestimation.

We use receiver operating characteristic (ROC) curves 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 consensus classification (see § 4.2), which varies from 0 to 1. Similar to Use Case 1, we implement a baseline method (Baseline) and two variants of Latin, Latin$^{\text{L}}$ and Latin$^{\text{A}}$, where Baseline uses the Jaccard coefficient as the similarity metric (Eqn.(1)). Here we focus on the name/alias information and consider comparing two malware instances based on their behavior profiles as an ongoing direction.

Figure 10 illustrates the ROC curves of different matching models for each pairs of AV systems. It is observed that in all the cases Latin$^{\text{L}}$ and Latin$^{\text{A}}$ significantly outperform the baseline method. It is also noticed that incorporating alias information considerably improves the matching accuracy. For example, with false positive

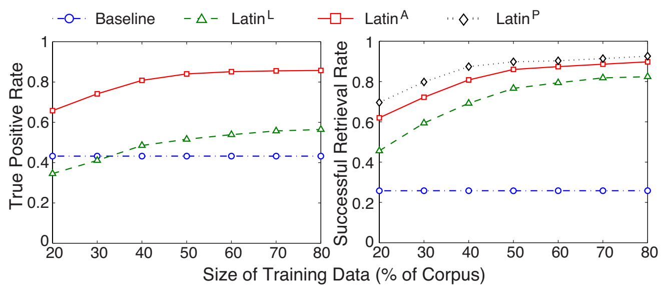


Figure 11: Impact of training data size over matching and retrieval.

rate fixed as 0.05, true positive rates of `Latin`^{L} and `Latin`^{A} differ about 0.3 in most cases. This is explained by the diverse expertise of different AV systems: combining evidences from multiple systems generally helps more accurately pinpoint the classes of query instances in the consensus classification. The only exception is the case of S_3-S_4 wherein `Latin`^{A} performs slightly worse than `Latin`^{L} at the beginning. This can be explained by the nature of local classification (Table 3). For example, S_3 is much more consistent with S_4 than S_2 on malware classification; thus taking account of their aliases in S_2 when matching names from S_3 and S_4 may slightly "dilute" the matching accuracy. This exception however is well justified by the overall superior performance of `Latin`^{A}. The ROC curves also shed light on how to set the optimal discrimination threshold for name matching models to determine the equivalence of two names. We can select the "elbow point" of an ROC curve as the threshold for it maximizes the ratio of true positive over false positive rates. For instance in the case of S_4-S_2, we may choose the threshold corresponding to the false positive rate around 0.05.

5.4 Other Experiments

Impact of Training Data

We now discuss the impact of training data on the quality of query processing by `Latin`. Due to space limitations, we focus our discussion on the size of training data. Figure 11 shows the performance of different matching and retrieval models as the size of training data varies from 20% to 80% of the entire malware collection. In particular, Figure 11 (a) illustrates the true positive rates of `Baseline`, `Latin`^{L}, and `Latin`^{A}, where the false positive rate is fixed as 0.2 and the results are averaged over every pair of AV systems. It is observed that the true positive rate of `Baseline` stays around 0.42 as it is not subject to the consensus classification, while the performance of both `Latin`^{L} and `Latin`^{A} improve significantly as the training data size grows from 20% to about 50%, which is explained by that during this early stage the coverage of consensus classification over all the classes of participating AV systems increases fast. After this coverage reaches a certain threshold, both `Latin`^{L} and `Latin`^{A} achieve stable true positive rates above 0.5 and 0.8 respectively. We can thus conclude that (i) `Latin` is fairly tolerant to the availability of training data and (ii) in the case that training data is extremely limited we can complement `Latin` with syntactic-based approaches. Similar conclusions can be drawn for the case of retrieval in Figure 11 (b) as well.

Operation Scalability

Finally we measure the scalability of `Latin` in processing queries. We evaluate the average processing time for each incoming query, under varying size of consensus classification, with and without additional threat intelligence. The size of consensus classification is controlled by the size of training set. Note that here we only evaluate the processing time for queries that require nontrivial folding-in (with class names not appearing in the local classification). Figure 12 illustrates the average processing time per query by `Latin`^{A} and `Latin`^{P} in matching and retrieval tasks, respectively, as the size of consensus classification increases. We break down the processing time into four parts, name processing, alias processing, behavior profile processing (corresponding to folding-in), and matching (or retrieval). We observe that (i) overall `Latin` scales almost linearly with the size of consensus classification; (ii) the execution time for matching or retrieval is barely noticeable; (iii) the behavior profile processing accounts for the major overhead as it involves LDA operation.

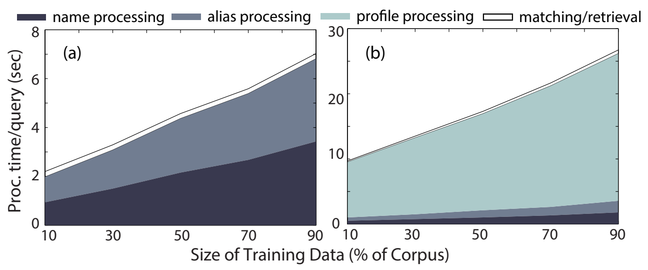


Figure 12: Execution time of matching and retrieval.

6. RELATED WORK

The issues of malware naming discrepancies have been debated in security research communities for decades [1, 5, 12, 14, 20]. Bailey et al. [1] measured such inconsistency in terms of the capability of AV systems to identify similar or identical malware in the same way and found that consistency is never a design goal of most AV systems, e.g., they assign the same name to identically behaved malware only from 31% to 61% of the time. Recently Maggi et al. [20] quantitatively showed that high degrees of inconsistency in terms of naming distance and scatter score exist across different AV systems. Many factors may account for the malware naming discrepancies. The first and foremost may be the exponentially escalated number of new malware instances created nowadays [4, 12, 14], which makes it practically impossible to examine every new malicious program and to agree on a reasonable name. Second, AV systems increasingly rely on generic signatures and other heuristics (behaviors, rare packers, etc.); ad hoc rules for naming the detected malware instances are often used and the cost is prohibitive to correct or change such names afterwards [4].

Several existing attempts have been made to standardize the disparate naming conventions in well defined formats (e.g., *platform, family, group, major variant, minor variant*) [4, 10, 25]. Unfortunately most major AV system vendors were reluctant to accept these proposals. Even if they did, the naming discrepancy would still exist, as there are no regulations on how these names would be semantically used. `VGrep`[7] and `VirusTotal` offer online services for users to search names from different AV systems for given malware instances, yet providing no systematic approaches to reconcile the inconsistent or even conflicting names. To our best knowledge, this work represents one of the first attempts to systematically resolve the malware naming discrepancies from both syntactic and semantic perspectives.

One line of research related to our work is schema matching [2] which studies how to reconcile schemas of different data sources. Particularly rule-based methods [11, 19, 21] exploit schema-level information (e.g., element names, data types) to discover similar elements in different schemas, while learning-based methods [9,

[7]www.virusbtn.com/resources/vgrep/index

26] apply machine learning techniques over both schema- and data-level information to identify correspondence between schemas. Another line of related work is object matching (also known as record linkage) [22, 6] which attempt to find records referring to the same entity across different relational databases. Our work differs in that it addresses not only disparate syntactic representation but also conflicting classification by different AV systems.

There are several other lines of work we build upon. Consensus learning has been applied to malware behavior analysis [27, 17], while power iteration method has been applied to large-scale clustering [15]. To the best of our knowledge, the present work however is the first that considers the application of power iteration method in the context of consensus learning.

7. CONCLUSION

The malware naming discrepancies across different AV systems remain one potent barrier that prevents these systems from effectively sharing malware intelligence. To the best of our knowledge, this work represents one of the first few attempts of systematically reconciling both syntactic and semantic naming discrepancies at a large scale. We presented `Latin`, a novel classification integration framework which exploits the correspondence between different AV systems reflected in various heterogenous information and creates optimal consensus classification to support cross-system information sharing. We implemented and evaluated a prototype system, which demonstrates the potential of this methodology to reconcile malware naming discrepancy issues to a certain extent.

8. REFERENCES

[1] M. Bailey, J. Andersen, Z. M. Mao, and F. Jahanian. Automated classification and analysis of internet malware. In *RAID*, 2007.

[2] P. A. Bernstein, J. Madhavan, and E. Rahm. Generic schema matching, ten years later. In *VLDB*, 2011.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[4] V. Bontchev. Current status of the caro malware naming scheme. www.people.frisk-software.com/∼bontchev /papers/naming.html.

[5] P.-M. Bureau and D. Harley. A dose by any other name. In *VB*, 2008.

[6] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD*, 2003.

[7] CNET. Most popular security software: www.cnet.com.au/software/security/most-popular.htm, 2012.

[8] Damballa. Integration partners: www.damballa.com/solutions/integration_partners.php.

[9] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *SIGMOD*, 2001.

[10] N. FitzGerald. A virus by any other name: Towards the revised caro naming convention. In *AVAR*, 2002.

[11] F. Giunchiglia and P. Shvaiko. Semantic matching. *Knowl. Eng. Rev.*, 18(3):265–280.

[12] D. Harley. The game of the name malware naming, shape shifters and sympathetic magic. In *CFET*, 2009.

[13] J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., 1975.

[14] T. Kelchner. The (in)consistent naming of malcode. *Computer Fraud & Security*, 2010(2):5–7.

[15] F. Lin and W. W. Cohen. Power iteration clustering. In *ICML*, 2010.

[16] J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. Inf. Theor.*, 37(1):145–151.

[17] B. Long, Z. M. Zhang, and P. S. Yu. Combining multiple clusterings by soft correspondence. In *ICDM*, 2005.

[18] B. Luo, R. C. Wilson, and E. R. Hancock. Spectral clustering of graphs. In *GbRPR*, 2003.

[19] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *VLDB*, 2001.

[20] F. Maggi, A. Bellini, G. Salvaneschi, and S. Zanero. Finding non-trivial malware naming inconsistencies. In *ICISS*, 2011.

[21] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, 2002.

[22] H. B. Newcombe and J. M. Kennedy. Record linkage: Making maximum use of the discriminating power of identifying information. *Commun. ACM*, 5(11):563–566.

[23] M. D. Preda, M. Christodorescu, S. Jha, and S. Debray. A semantics-based approach to malware detection. In *POPL*, 2007.

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

[25] G. Scheidl. Virus naming convention 1999 (vnc99). http://members.chello.at/erikajo/vnc99b2.txt.

[26] T. Wang and R. Pottinger. Semap: a generic mapping construction system. In *EDBT*, 2008.

[27] Y. Ye, T. Li, Y. Chen, and Q. Jiang. Automatic malware categorization using cluster ensemble. In *KDD*, 2010.

APPENDIX

A. OPTIMAL CLUSTERING

We assume that $\mathbf{v}^* = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ has been sorted in non-descending order. Our goal is to partition $\mathbf{v}^*$ into a set of tight clusters. Since $\mathbf{v}^*$ is sorted, a cluster containing x_s and x_t $(s < t)$ must also include $\{x_j\}_{j=s+1}^{t-1}$. We use the squared deviation to measure the intra-cluster dispersion of a cluster of values: $\mathtt{d}[s,t] = \sum_{j=s}^{t} x_j^2 - (\sum_{j=s}^{t} x_j)^2/(t-s+1)$.

Let $\mathtt{w}(t,k)$ denote the minimum sum of intra-cluster dispersion when we partition $[x_1, \ldots, x_t]$ into k clusters. The following optimal substructure exists:

$$\mathtt{w}(t,k) = min_{k \leq s \leq t}(\mathtt{w}(s-1, k-1) + \mathtt{d}[s,t]) \tag{4}$$

Note that $\mathtt{w}(n,k)$ corresponds to an optimal k-clustering of $\mathbf{v}^*$. Meanwhile we are also interested in simultaneously determining the optimal k. We apply the *Hartigan index* [13] to evaluate the clustering quality, which is defined as

$$\mathtt{h}(k) = (n-k-1)\frac{\mathtt{w}(n,k) - \mathtt{w}(n,k+1)}{\mathtt{w}(n,k+1)} \tag{5}$$

Intuitively $\mathtt{h}(k)$ captures the relative improvement when increasing the cluster number from k to $k+1$. The optimal cluster number is the smallest k which produces $\mathtt{h}(k) \leq \eta$ (typically $\eta = 10$). Moreover, tnstead of computing $\mathtt{w}(n,k)$ for every possible k, the algorithm can stop immediately once it finds the minimum k satisfying this condition.