Adaptive Semantics-Aware Malware Classification

Bojan Kolosnjaji, Apostolis Zarras, Tamas Lengyel,
George Webster, and Claudia Eckert

Technical University of Munich
{kolosnjaji,zarras,tklengyel,webstergd,eckert}@sec.in.tum.de

Abstract. Automatic malware classification is an essential improvement over the widely-deployed detection procedures using manual signatures or heuristics. Although there exists an abundance of methods for collecting static and behavioral malware data, there is a lack of adequate tools for analysis based on these collected features. Machine learning is a statistical solution to the automatic classification of malware variants based on heterogeneous information gathered by investigating malware code and behavioral traces. However, the recent increase in variety of malware instances requires further development of effective and scalable automation for malware classification and analysis processes.
In this paper, we investigate the topic modeling approaches as semantics-aware solutions to the classification of malware based on logs from dynamic malware analysis. We combine results of static and dynamic analysis to increase the reliability of inferred class labels. We utilize a semi-supervised learning architecture to make use of unlabeled data in classification. Using a nonparametric machine learning approach to topic modeling we design and implement a scalable solution while maintaining advantages of semantics-aware analysis. The outcomes of our experiments reveal that our approach brings a new and improved solution to the reoccurring problems in malware classification and analysis.

1 Introduction

Malware has evolved over the years to the point where it generates a global threat for our digital lives. Nowadays, the amount of malware that arises every day has increased exponentially. Security companies currently need to analyze hundreds of thousands of malicious samples on a daily basis, which directly affects their performance. In some cases, this number can be larger than one million distinct files per day [34]. Meanwhile, malware classification is becoming increasingly critical as new malware instances integrate sophisticated techniques to deceive the signature-based detectors and operate under the radar for longer period. This fact, along with the rapid increase in the number of malware samples, presents a very real challenge that cannot be met by manual reverse engineering efforts or by generating static signatures. Specifically, while it is relatively easy for antivirus and other security companies to obtain large numbers of malicious samples, it requires significant effort to successfully classify them.

To address this problem, researchers proposed statistical machine learning methods that can enable analysts to focus on new and previously unseen attacks by classifying malware as being part of a larger family [29]. These methods leverage gathered static and behavioral malware data to generate statistically confident knowledge. Towards this direction, Schultz et al. [29] used statistical methods to detect malicious executables based on n-grams of instructions. Rieck et al. [27], on the other hand, utilized behavioral features of malware for both detection and classification, and proved the superiority of this approach against the traditional signature-based methods. This performance improvement is explained with the inherent advantages of statistical methods in capturing the variety of malware samples.

Nevertheless, statistical malware classification systems are not without their own problems. Foremost, there is a scarcity of reliable labels for fully supervised malware classification systems. Malware analysts could potentially retrieve antivirus results and use them to label malware samples. Although this approach seems ideal, unfortunately, many times it is difficult to provide confident labels this way. Antivirus companies offer malware signatures, which are mostly used in the academic community for testing the malware classification systems. However, we have observed by manual inspection of antivirus results that the reliability of those signatures is not always high. Every antivirus program has its own system of labeling malware, and although sometimes the signatures match between different antivirus programs, very often they are different or even contradictory. Furthermore, there is a limited public information about the process by which companies assign these signatures and how accurate these signatures are. Yet another problem is the very high data dimensionality when the execution logs contain whole system behaviors [6]. Finally, the malware analysis tools provide different features of malware with respect to static and dynamic analysis [1–3, 19]. Using multiple tools ensures that all the information is considered, yet, there exist only few efforts that try to join this information [4]. This problem is non-trivial because data retrieved from the analysis tools is heterogeneous, which means that different machine learning models might be optimal for different data. For example, dynamic malware analysis results have a sequential nature, while metadata from static analysis, such as code entropy or size of code sections, do not always have such interdependencies.

The number and variety of malware samples that need to be processed has surpassed the ability of the classical approaches that analyze the static and behavioral characteristics of malicious samples and create signatures. Automation of detection and classification procedures that take into account the aforementioned approaches is becoming less effective when dealing with large amount of data, let alone extracting useful knowledge about malware. Since the problem is essentially the automatic analysis of high amounts of noisy data, statistical machine learning methods constitute a superior approach. These methods, however, need to be adapted to online setting, where a high influx of samples imposes a necessity for retraining of machine learning models in order to maintain accurate label predictions.

In this paper, we evaluate and improve the use of statistical topic modeling with respect to the curse of dimensionality of long execution sequences. Further, combined with semi-supervised learning methods of exploiting unlabeled samples, we effectively overcome the problem of the lack of labeled data. Finally, we show how the use of data obtained from static and dynamic analysis increases the reliability of the classification results, demonstrating that data heterogeneity can in fact boost confidence in classification. In essence, we use a nonparametric machine learning approach, where parameter set is not set up in advance, but depends on the training data. Nonparametric approach is, to the best of our knowledge, novel in malware classification problems. This enables a more stable approach, where semantic interpretation is automatically updated on arrival of new malware samples. Our evaluation reveals that our model achieves over 90% precision and recall in classification for most of the tested malware families, while it retains stability in classification performance and retraining speed.

In summary, we make the following main contributions:

- We create a semi-supervised malware classification system that unifies views of static and dynamic malware analysis.
- We perform an automatic extraction of semantic behavioral features from the results of dynamic malware analysis.
- We design and evaluate a nonparametric model that is adaptive in a setting of online training.

2 Background

The key concepts from the area of machine learning that constitute the lifeblood of our approach are *topic modeling*, *semi-supervised learning*, and *nonparametric learning*. In this section, we briefly introduce the aforementioned concepts.

2.1 Topic Modeling

As behavioral malware execution data is a sequence of tokens taken from a predefined dictionary, it closely resembles text documents by structure. Therefore, methods of information retrieval designed for extracting latent properties of text can be of great importance. In machine learning, data is very often organized in long sequences. Most explored examples of this kind of data are audio and video recordings, genetic sequences, and text documents. For instance, it has been determined that very often news articles belong to a smaller set of latent topics such as *Basketball*, *Tour De France*, *Hollywood*, *FBI Investigation*, etc [22]. On a higher level, topics could be *sports*, *culture*, and *finance*. Furthermore, words in text documents belong to these topics with certain probability, where one word can be attributed with multiple different topics as well. If topics are semantically interpretable, created model also has a semantic meaning. This text modeling problem and vocabulary can be translated to problems with other types of data.

Topic modeling methods are mostly constructed as generative methods: they are not designed just for classification but also for generation of data based on the probability distributions inferred from the model. In essence, the topics are constructed in such a way that the training documents can be generated with high probability using just the topics inferred from the model. Given a reasonable assumption that our documents can be confidently described by a smaller set of topics, we can determine these topics and their distribution by training a topic model. We do not need to know the topics in advance, as they can be inferred from the data (i.e., from the documents and the words contained in them).

Overall, topic modeling is a method to statistically explain a large set of documents using a small set of clusters (topics), based on frequency of different words in these documents. Note that it counts the words independently without a specific interest of their sequences inside the documents. This approach has been often called *bag-of-words* and it greatly simplifies document analysis.

One of the most adequate random processes used for topic modeling is the *Dirichlet* process. This is a suitable model especially for datasets where only few latent topics can describe a large set of documents. The notion of latent topics was popularized with the development of *Latent Dirichlet Allocation* (LDA) method [7]. In this method the topic structure is sampled from a Dirichlet distribution as prior, which gives more flexibility in training the generative model. Although there exist related methods of topic modeling [10], LDA is the most used regarding document information retrieval because of its flexibility and modular structure. This method has been further adapted to discriminative learning, i.e., classification [25]. In its standard form, LDA uses a bag-of-words assumption, which means that it does not capture the sequential nature of the document: it only counts words independently.

2.2 Semi-Supervised Learning

The lack of proper labeling has already been defined as an important problem in malware research [5]. Consequently, one would benefit from a method that offers maximum utilization of a minimal number of highly confident labels. This setting is known in machine learning as semi-supervised learning and is halfway between supervised and unsupervised algorithms. While supervised learning is a paradigm that encompasses machine learning methods where the training data is labeled and the purpose of the algorithm is to optimize the classification of data on the test dataset, unsupervised learning discovers the underlying structure in the data such as locating clusters in the dataset. We use unsupervised learning when we do not have information about labels in the time of training. Since in semi-supervised setting we do have labeled data, but it is scarce, we combine the advantages of two separate methods to overcome this limitation. More specifically, in semi-supervised learning we leverage the property of data that it forms natural clusters. Even if we only have a small number of labeled data that identifies the clusters that exist in the dataset, we can propagate these labels in the neighborhood of the labeled data, by considering the clusters detected in the dataset.

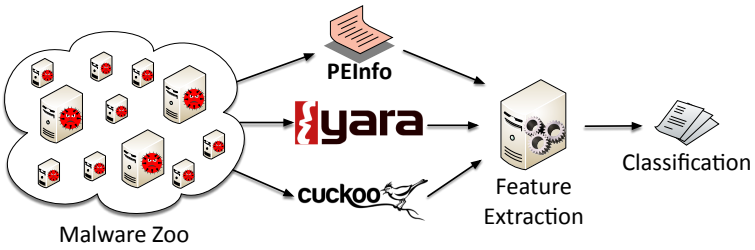

Figure 1. Malware classification architecture.

2.3 Nonparametric Learning

In many scenarios the parameter set of machine learning models cannot be always defined in advance. This is also the case with malware classification, as high influx of malware samples imposes a need to adapt the model incrementally. This can be done using a nonparametric approach, where parameter set grows with the size of the dataset. Since this growth of the parameter set increases complexity of the model, additional effort is needed to stabilize the classifier. We utilize an improved approach in order to maintain this stability.

3 Methodology

We propose a classification scheme aimed at solving the problems indicated in the introductory sections. In particular, we want to be able to discover semantic features of malware classes, maintain an adaptive topic model, and maximize the utilization of a semi-labeled dataset from heterogeneous data sources. To do so, we first emphasize on extracting semantic features from high-dimensional and noisy data. Second, we optimize the classification mechanism under the setting where low number of labeled samples is available. To this end, we join results of static and dynamic malware analysis to unify these different views on properties of malware samples. Finally, we design an architecture that is adaptive in the online training setting. In summary, our malware classification architecture complies with the scheme displayed on Figure 1.

3.1 Experimental Environment

To perform data extraction against malware samples we employ a *malware zoo*, in which we can execute malicious samples while monitoring their behavior. The zoo back-end infrastructure is composed of a custom version of CRITs [33] that utilizes large scale analysis concepts proposed by Hanif et al. [15]. Specifically, our modifications use custom CRITs Services to extract API call information from Cuckoo [1] and execute the requested work in a distributed fashion. The malware samples were collected over multiple months from three primary sources: Virus Share [28], Maltrieve [21], and private collections. We chose these sources to provide a large and diverse volume of samples for evaluation.

Data acquisition is done using widely available tools for the static and dynamic malware analysis. On the one hand, static analysis provides us with features extracted from the code of the malware samples. For this purpose, we use two sources aimed for static analysis: PEInfo [33, 38] and Yara [3]. We leverage PEInfo to extract entropy, size of different PE sections, and the collection of imported libraries. Similarly, Yara provides us with a list of used function calls to the Windows kernel API and other custom signatures extracted from the code.

On the other hand, dynamic analysis enables us to gather reliable behavioral data without the need for deobfuscation. There exist various tools that enable tracing the execution of malware and gather logs of execution sequences [1, 19]. We select the Cuckoo Sandbox, which provides a controlled environment for executing malware. During the execution of malware samples we record calls to the kernel API that we later use to characterize malware activity. For each sample we obtain a sequence of API calls, which is preprocessed by removing subsequences where one API call is repeated multiple times in a row. We cut these subsequences by using only one kernel API call instance as representative in the resulting sequence. In multiple samples we have noticed the repetition of one API call; for example, when malware repeatedly tries to open a file.

In addition, we leverage VirusTotal [2] by extracting antivirus signatures from its web service, for each malware sample we use. Users can upload MD5 hashes of malware executables to VirusTotal and retrieve results from multiple antivirus engines through the VirusTotal API. These engines are signature-based and compare the submitted hash to the data in their own database. By using the VirusTotal services we access malware analysis results and signatures, out of which we are mostly interested in retrieving ground truth labels for our classification. In a lack of other label sources, we use antivirus signatures in label construction for training and testing our classification scheme. Since antivirus programs use customized strategies for signature generation, we need to find a way to extract one numerical training label per unique sample using the diverse antivirus signatures. We use signature clustering to achieve this goal.

3.2 Signature Clustering

To get more confident training and testing labels, we perform a selection process that uses a simplified version of signature clustering method introduced in VAMO [23]. Specifically, we create signature vectors for every malware sample that contains signatures given by different antivirus engines. We use boolean features to generate these vectors, where each feature reveals presence or absence of a certain antivirus signature. Our assumption is that the malware samples of the same family will have the same or similar boolean feature vector. Next, we use a variant of *cosine distance* as a measure of difference among signature vectors for our clustering process. We cluster the samples using DBSCAN [12], as we do not know their number in advance. Finally, we select ten clusters with the highest number of members as classes for classification. This way we cover most of our labeled dataset. Since the classes assigned to our malware resemble the families defined by antivirus engines, we use the terms *class* and *family* interchangeably.

3.3 Feature Selection

Static analysis tools provide us with a high number of features. In detail, we retrieve 23,060 features from PEInfo and 3805 features from Yara extracted from the malware binary files. Using a high number of features makes the classification problem ill-posed and therefore we choose to utilize feature selection methods to obtain an optimal feature set. We use *univariate feature selection* approach and perform a χ^2 test for all training samples. This way we can extract the features that are most relevant to our classification problem and reduce the computational effort needed for the training process. For our purpose we achieve best results by selecting 10,000 features for PEInfo and 1000 features for Yara.

3.4 Topic Modeling Algorithms

To extract features from the kernel API call sequences we utilize the topic modeling approach, which includes a well-developed set of methods already heavily used for automatic information retrieval from text and image data.

General Approach. As we already mentioned, topic modeling is a method based on the fact that a collection of tokens (words) from documents can be grouped to a limited set of topics. More specifically, we apply topic modeling to process data from dynamic malware analysis, as we consider that a list of API calls can be divided into a smaller number of latent activities. In our case, documents are malware execution logs and words are elements of malware execution sequences—calls to the Windows Kernel API. Additionally, topics are groups of these elements that constitute an elementary operation, for instance, registry access and modification, file manipulation, process creation and invocation.

This analogy justifies the attempt to adapt the topic modeling approach for the malware classification problem. The general topic modeling scheme can be represented with the following formulas:

$$G \sim DP(\alpha, H) \tag{1}$$

$$\theta_i \mid G \sim G \tag{2}$$

$$x_{j,i} \mid \theta_i \sim F(\theta_i) \tag{3}$$

where parameter G (a Dirichlet distribution) controls the topics and generates the parameter θ_i. Words ($x_{j,i}$) are generated based on this parameter. Dirichlet process is actually a distribution of distributions. The draws from Dirichlet processes are probability distributions, which are inferred for the next parameter in the chain. This parameter controls the word distribution for single topics. Topic modeling based on a Dirichlet process enables us to define a generative model, where each document is a mixture of a small number of topics. It is important to note that topics are not known in advance, but are inferred by the topic modeling methods. This enables us to uncover previously unknown semantics from the malware execution logs. Parameters are approximately determined using variational inference and Markov Chain Monte Carlo methods [35], as exact

inference is not tractable. This also enables fast retraining in case of need for online update of the model. Figure 2 contains a graphical model used for topic inference, where the directed edges show the process of word generation.

Topic models can be essential for classification performance, as important latent structure is inferred and noise canceling is implicitly executed by extracting the important topics. However, even more crucial is the possibility of semantic interpretation. Although malware analysts are able to get a rich set of information from the dynamic malware analysis tools, this information needs to be further analyzed and significant expert knowledge is required to extract the important information out of the logs retrieved from these tools. Thus, it would be extremely useful to automate this procedure and to extract relevant data about the malware activity. This would enable analysts to achieve their goals faster and with statistically confident results. Therefore, we develop a more efficient alternative to the cumbersome deterministic manual analysis procedure. Even if the topics do not have an obvious semantic meaning, comparing the topic structure among different malware families can enhance the classification process and provide new knowledge about the dataset in use.

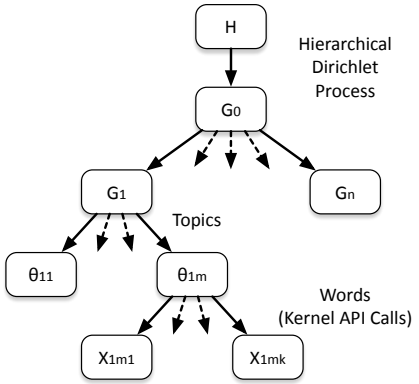

Figure 2. Graphical model for our Hierarchical Dirichlet Process.

Previous work demonstrated the utility of topic modeling for extracting semantics out of kernel API call logs by using LDA, where topic parameters are drawn out of the Dirichlet distribution [7]. Furthermore, this method was adapted from a bag-of-words method to a new scheme that takes account of sequential data ordering [39]. However, this approach is not scalable on large sets of malware and online learning, and is sensitive to noisy sequences. It also requires a predefined number of topics, which would need to be manually updated by the malware analyst as new data is acquired. In case of an organization that maintains its own malware dataset and receives a high amount of submissions on a daily basis, this kind of setting may not be satisfactory.

Hierarchical Dirichlet Processes. Given the limitations of LDA, we take a different approach, using methods that bring the required improvement to online learning. More precisely, we utilize an adaptive method called *Hierarchical Dirichlet Process* (HDP) [32]. In this method the topic distributions are also determined by Dirichlet processes, yet there exist different processes for each document. These processes, however, are not independent. They are drawn from a prior Dirichlet process, which depends on parameters that control the growth of topics and their distribution as the dataset grows in time:

$$G_0 \sim DP(\alpha_0, H) \tag{4}$$

$$G_j \mid G_0 \sim DP(\alpha_j, G_0) \tag{5}$$

where Dirichlet processes G_j are conditioned by the prior G_0.

Overall, the general setting of the topic modeling remains the same: documents belong to multiple topics and words depend on topic distributions. HDP is an instance of nonparametric machine learning methods. As a difference from parametric methods, like LDA, nonparametric methods are used when we want the parameter set to change with the dataset. HDP introduces a more flexible approach, which is also more computationally demanding. Actually, this is the case with all the nonparametric machine learning methods. Nevertheless, there exist modifications that trade the accuracy of the method for performance in an online setting [36]. We use these modifications to create a scalable approach with respect to the computational demand. Our implementation is based on the *GenSim* library [26], developed for the estimation of text document similarity.

3.5 Semi-Supervised Malware Classification

Accurate malware classification is often difficult due to lack of confident label sources. We can find proper signatures only for a small subset of malware samples, even by utilizing services such as VirusTotal. To deal with the scarceness of labeled data, we use semi-supervised learning, where we influence the usual malware clustering procedure with high-confidence labels. Figure 3 displays our semi-supervised classification scheme. Our system unifies advantages of topic modeling and semi-supervised learning. To this end, data retrieved from static and dynamic analysis tools are run through feature extraction and forwarded to the classification stage.

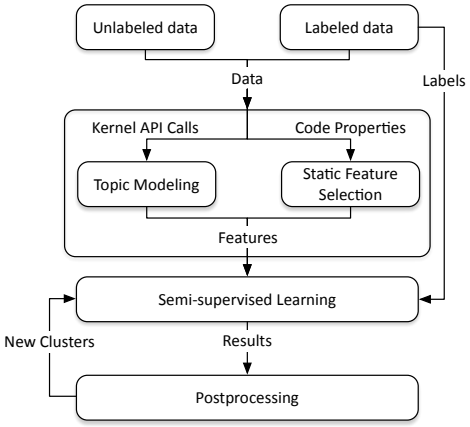

Figure 3. Semi-supervised learning scheme.

To achieve an effective semi-supervised learning model, we take two separate approaches to classify the static and dynamic analysis results. For results retrieved from static analysis we use label propagation. This method uses labeled data and density-based clustering to propagate the given labels through the dataset. The propagation of labels is conditioned by the similarity structure between data samples. In particular, we use a regularized variant of label propagation, to take account of the possible noise in labeling [42].

For dynamic analysis results we use another alternative. In a semi-supervised setting we can use unlabeled data for initial pretraining of topic models before using the actual labeled data. To discriminate between classes of malware, we make use of a *maximum-a-posteriori* (MAP) approach. This approach is used in machine learning very often when estimating distributions and parameters of a model. As a result, when classifying, we assign the class label to the data that is inferred with a highest probability.

We create a topic model for every existing class, based on the available logs of API calls. For each new log we evaluate the likelihood that its API call sequence would be generated from each topic model ($P(D = x \mid y = c_i)$). We also estimate independent prior probability of a certain class ($P(y = c_i)$) by simply calculating the share of certain class in the labeled dataset. Using the MAP approach, we evaluate the conditional probability of a certain sample belonging to the class i:

$$P(y = c_i \mid D = x) = \frac{P(D = x \mid y = c_i)P(y = c_i)}{\sum_i P(D = x \mid y = c_i)P(y = c_i)} \tag{6}$$

After computing the conditional probabilities for all classes, we find the most probable class by maximization:

$$CLASS(x) = \underset{i}{\operatorname{argmax}}(P(y = c_i \mid D = x)) \tag{7}$$

Malware sample is classified to the class to which it belongs with the highest probability.

Once the separate classification procedures finish for the outputs of available static and dynamic malware analysis tools, we forward the classification results to the aggregation and postprocessing stage.

3.6 Result Aggregation and Postprocessing

Our semi-supervised learning method returns probabilities of malware belonging to the predefined classes. These probabilities are results of separate classification using our three data sources (i.e., PEInfo, Yara, and Cuckoo). We combine these results to get a reliable class probability estimation. In machine learning-based classification it is often beneficial to combine multiple data sources and different classifiers to reduce model overfitting and use advantages of different methods in one system [18]. This approach is called *ensemble learning*. Multiple methods of various sophistication exist for combining different classifiers. We argue that, since we do not have a large set of classifiers, there is no need for complicated ensemble learning approaches. In case of a larger number of data sources, an approach such as mixture of experts can be used, however, we did not notice any advantage of this approach in our case. For our experiments we use median and average of probability values, and majority voting of class assignments resulting from the three data sources. By aggregating the classification results, we get a more robust classifier. In fact, we combine the advantages of the static and dynamic analysis, to get a better classification performance. The combination of multiple views on data makes our results more reliable.

During the online classification procedure our system can detect the appearance of a new cluster. This can happen in one of the following cases: (*i*) new data has been put in the learning algorithm that contains a previously unknown label, or (*ii*) there is a new region of high local density that is detected during the execution of the learning algorithm. With the postprocessing algorithm, it is determined if the new sample can be confidently assigned to one of the existing classes, or a new class needs to be defined. Introduction of new classes can be done automatically by tuning the machine learning model, and in addition the new labels can be approved by a malware analyst. If we do not expect the new classes to appear very often, we can assign this job to the analyst, who can give a reliable estimation and help avoid possible mistakes in labeling. If indeed a new cluster is confirmed, the algorithm must be retrained in order to include this new fact into the machine learning model.

4 Evaluation

In this section we evaluate our approach. The extracted results prove advantages of topic models, semi-supervised methods, and combining results of static and dynamic malware analysis into a unified classification procedure.

For this purpose, we took ten recurring malware families from our labeled dataset of 2000 malware samples. The class titles were directly extracted from VirusTotal, where we manually chose signatures from multiple antivirus programs that were most prominent in our dataset. In addition to the labeled samples, we had 15,000 samples that we used as unlabeled, as the results of VirusTotal did not provide us with signatures for them. We then divided the dataset into training and test sets using a variant of three-fold cross-validation. More precisely, the dataset is divided randomly in three parts, where two parts are used for training and the last part for evaluation. This division and accuracy experiment were repeated ten times and we took the average of the results. The distribution of samples in our dataset is mostly uniform, except for one significantly bigger and three smaller families (see Figure 4). However, we take this into account in cross-validation when determining the training and test set, as well as during the evaluation of our approach.

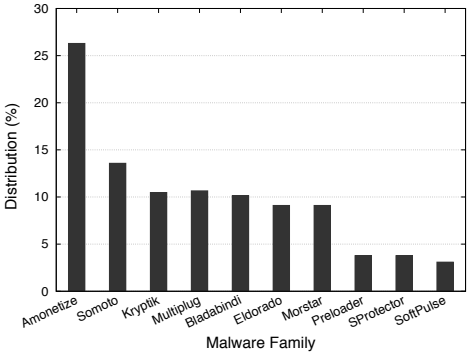

Figure 4. Samples distribution by family.

Family	LDA for a different number of topics						HDP
	1(%)	5(%)	10(%)	20(%)	40(%)	80(%)	
Amonetize	0.0	0.0	10.0	100.0	100.0	100.0	100.0
Somoto	0.0	0.0	0.1	30.3	20.4	30.0	99.8
Kryptik	0.0	18.0	30.0	70.0	60.0	30.5	91.5
Multiplug	0.0	57.4	80.0	30.0	40.0	69.4	80.0
Bladabindi	0.0	1.7	5.7	4.0	7.0	10.3	93.0
Eldorado	0.0	0.0	0.0	0.0	0.0	0.0	54.4
Morstar	0.0	0.0	0.0	0.0	0.0	0.0	100.0
Preloader	0.0	0.0	7.5	71.0	50.0	60.0	100.0
SProtector	100.0	100.0	100.0	100.0	100.0	100.0	100.0
SoftPulse	0.0	4.2	4.1	6.7	5.0	6.9	86.2

Table 1. Accuracy evaluation of LDA for different number of topics.

4.1 Topic Models

Using the training set, we created a topic model for each class using HDP. We computed the statistical likelihood of drawing each particular sample from the model. Based on this likelihood we classified the samples using the already described MAP approach. Next, we executed ten cross-validation tests with a random division into training and test set and averaged the obtained results. We also executed the equivalent tests for LDA with different number of topics in order to compare our work with this approach. Table 1 displays the averaged results for different malware families. These results are obtained using the supervised learning approach, however, the distribution is similar in the semi-supervised case. The outcomes justify the use of Hierarchical Dirichlet Processes over the Latent Dirichlet Allocation. More specifically, the classification accuracy is higher for most classes in case of using HDP, and in the worst case the performance is equal. This result along with the property that the HDP can automatically optimize the number of topics, gives us an adaptive and accurate classification component.

An interesting aspect of using LDA is that depending on the malware family we want to detect, we should apply different number of topics. Although the overall results reveal a significant advantage when using a higher number of topics on average, the correlation is not clear for all the families we tested. An example of this is Multiplug which exhibits better detection accuracy by selecting just ten topics, while Amonetize offers better accuracy when selecting 20 or more topics. Unfortunately, we could not detect any samples that belong to Eldorado and Morstar families using LDA. One possible reason is that we did not find the optimal number of topics for these samples.

In our experiments we noticed that the topics that were results of our topic modeling experiment often have an obvious semantic meaning. This makes our classification approach semantics-aware. Some examples of semantically meaningful topics are presented in Table 2. It is worth to mention that some kernel API calls belong to different topics simultaneously, which is a useful property of topic models, since activities represented by topics can consist of overlapping sets of operations.

Registry manipulation	Memory management	File manipulation	Process Handling
NtWriteFile	VirtualAllocEx	NtReadFile	OpenProcess
RegOpenKeyExW	VirtualQueryEx	NtWriteFile	ReadProcessMemory
RegCloseKey	VirtualQuery	NtDelayExecution	WriteProcessMemory
RegEnumValueW	VirtualFreeEx	LdrGetProcedureAddress	CloseHandle
RegQueryValueExW	VirtualFree	NtSetInformationFile	LocalAlloc
LdrGetProcedureAddress	LdrGetProcedureAddress	NtCreateFile	LocalFree
RegOpenKeyExA		NtQueryDirectoryFile	

Table 2. Overview of main semantically relevant topics.

Family	[Cuckoo + HDP](%)	[Yara + LP](%)	[PEInfo + LP](%)	Average(%)	Median(%)	Majority(%)
Amonetize	100.0	99.6	100.0	100.0	100.0	100.0
Somoto	99.0	100.0	51.0	100.0	100.0	100.0
Kryptik	100.0	100.0	100.0	100.0	100.0	100.0
Multiplug	99.2	100.0	100.0	100.0	100.0	100.0
Bladabindi	93.2	96.6	100.0	96.6	96.6	99.0
Eldorado	56.6	80.2	83.0	84.9	86.8	81.0
Morstar	100.0	40.0	100.0	40.0	100.0	100.0
Preloader	100.0	100.0	100.0	100.0	100.0	100.0
SProtector	100.0	100.0	100.0	100.0	100.0	100.0
SoftPulse	86.1	88.8	0.0	88.8	88.8	77.8
Average	93.4	90.5	83.4	91.0	97.2	95.8

Table 3. Comparative accuracy test using results from static and dynamic malware analysis data, separately and combined.

4.2 Static and Dynamic Analysis Combination

Table 3 illustrates a comparison of classification accuracy of our three data sources, determined by executing cross-validation with these sources separately, using a semi-supervised procedure. More specifically, we combined on the one hand the Cuckoo sandbox with HDP, and on the other hand Yara and PEInfo with label propagation. It is evident from the results that even in cases with a small number of labeled samples we can achieve a sufficient accuracy. Furthermore, we notice that each separate data source is significant for the overall performance, as none of the data sources gives maximal classification accuracy for all classes. The maximal accuracy is, however, achieved when combining the three data sources. All the methods of combining results give a high accuracy for most of the families, with slight advantage for median and majority voting. These results justify our motivation for combining multiple data sources in order to get a better performance.

4.3 Comparing Supervised and Semi-Supervised Learning

We gathered results from semi-supervised and fully supervised learning. Table 4 shows the comparison of results of both approaches, when taking median class probability from all available data sources as classification criterion. The two colons for semi-supervised learning represent two separate experiments that we

Family	Supervised(%)			Semi-supervised(%)					
				1^{st} Experiment			2^{nd} Experiment		
	ACC	PR	RC	ACC	PR	RC	ACC	PR	RC
Amonetize	100.0	100.0	100.0	100.0	88.3	100.0	100.0	98.4	100.0
Somoto	100.0	100.0	100.0	93.6	72.2	93.3	100.0	96.8	100.0
Kryptik	100.0	100.0	100.0	100.0	86.4	100.0	100.0	100.0	100.0
Multiplug	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Bladabindi	99.4	98.1	96.0	83.5	95.4	82.9	96.6	95.8	96.6
Eldorado	75.6	26.3	86.0	31.4	98.1	31.6	86.8	98.9	86.8
Morstar	100.0	98.5	100.0	99.2	97.5	99.2	100.0	99.0	100.0
Preloader	100.0	100.0	100.0	57.1	100.0	55.4	100.0	100.0	100.0
SProtector	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
SoftPulse	64.4	75.4	87.0	49.5	51.1	50.8	88.9	86.5	88.9
Average	93.9	89.8	96.9	81.4	88.9	81.3	**97.2**	**97.5**	**97.2**
Rieck et al. [27]	**88.0**	-	-	-	-	-	-	-	-
Dahl et al. [9]	**90.5**	-	-	-	-	-	-	-	-

Table 4. Performance experiment with fully supervised and semi-supervised classification models regarding the accuracy, precision, and recall.

executed in order to evaluate the advantages and disadvantages of a partially labeled sample set.

The first experiment for supervised and semi-supervised methods is done using the same set of labeled examples, with the difference that in the semi-supervised case two thirds of the labeled data are used as unlabeled. We can notice that despite of using only a small number of labeled examples, we can get an adequate performance in classification. This performance is provided by our label propagation procedure, where we used the local density around the labeled points to propagate the class affiliation through the affinity matrix.

For the second experiment we used the samples for which we do not have antivirus labels as unlabeled samples and attempt to improve the classification performance. This approach shows that in most classes we can obtain a marginal improvement in the classification performance, as the unlabeled data helps in inferring the high density regions in the dataset.

Finally, we compared our classification performance with the results from related papers. Our results on average represent a significant improvement with respect to the related work in terms of average accuracy.

4.4 Open World vs. Closed World

We measured the performance of our closed world experiment to an open world situation, where not all classes are known in advance. We did this by executing the cross-validation test, always leaving out one class from the training set. In the test phase, we classified the samples that belonged to one of the training classes with probability higher than 50% into the appropriate training class. We put the samples which did not belong to any classes with such a high probability into the "outlier" class. Our hypothesis is that the "outlier" samples will be the

ones belonging to the class that is missing from the training set. This method was previously used by Rieck et al. [27], where the drop in accuracy in the open world was around 20 %. Our experiments showed that in our case, for most of the families, the performance dropped by 10 % or less. However, our system could not reliably detect the family *Eldorado* in the open setting, as the performance drop was over 40 %. This may be due to the comparatively shorter system call sequences, which makes the discrimination against other classes more difficult.

4.5 Time of Training

In our last experiment we wanted to measure the time of training of our approach. Therefore, we executed various number of samples and measured the time frame in which the training was complete. Figure 5 illustrates the distribution of the time it takes to retrain the topic models on arrival of new data points. It is noticeable that training time growth is linear, which is acceptable in online setting, considering that usually computational complexity of topic models grows not only with the number of documents, but also with the number of topics [7].

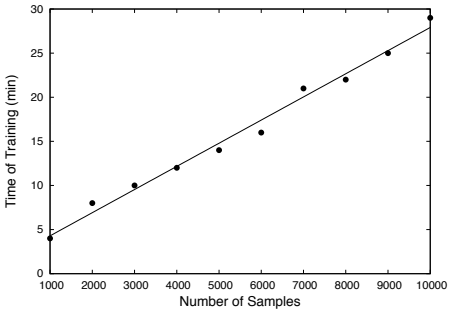

Figure 5. Time of training.

4.6 Summary of Findings

The outcomes of our experiments reveal that our methodology is adaptive, as our topic model can deal with varying number of topics and with this outperforms the standard LDA approach. Additionally, we show the semantic awareness of our method by displaying topics retrieved from system call sequences. Furthermore, we justify our approach by showing performance advantages of semi-supervised learning and joining static and dynamic analysis results. Finally, we compare the performance of our approach to previous works and show improvement in classification accuracy. Overall, our approach can assist analysts by offering them a more accurate malware classification.

5 Discussion

The experiments provide an insight into the performance of the methods used in our classification system. Our classification accuracy experiment on topic models shows a comparison of HDP and LDA, where HDP outperforms LDA in most classes. Nevertheless, it is also noticeable that the overall accuracy varies between different classes. For most of the classes the accuracy is high, yet there exist

outliers. An explanation for this would be the overall limited reliability of the ground truth labels based on the antivirus signatures and lack of possibility of explicit evaluation of label confidence. The results could be more reliable if a more trustworthy source of labels was available. For instance, it would be useful to initially perform unsupervised learning with unknown number of clusters and in addition enhance the results with custom labeling using the analyst domain knowledge. As a difference from work done by Xiao et al. [40], we use the sequence of kernel API calls as a bag-of-words (i.e., we ignore the information about order between the calls). This gives flexibility to our model, however, it may reduce accuracy. A further study is needed to experimentally compare these two approaches. Aside from the obtained accuracy of our classification, we are able to add another feature to our approach. This feature is the ability to extract semantics out of kernel API logs using inferred topics of Hierarchical Dirichlet Processes. Although a minority of the extracted topics has such an obvious semantic interpretation as in the presented examples, it can be very useful for a malware analyst to have such an insight.

In our evaluation, we compared results of static and dynamic malware analysis. While both static and dynamic analysis data were very useful for malware classification, the combination of the two methods proves to be the best of both worlds. Unfortunately, we had two data sources for static analysis data and only one source of dynamic analysis results. Therefore, the utilization of more data sources that provide additional data related to the program execution path, such as Drakvuf [19], would enhance the inference capability of our method.

Finally, we evaluated precision and recall of our classification and compared it with related work. Overall our system achieves a significant improvement over the previously published work in terms of classification performance, while retaining semantic model interpretation.

6 Related Work

This section contains the description of the research efforts that precede our work. These efforts are mostly divided into research dedicated to (*i*) application of machine learning methods in malware analysis and (*ii*) designing systems to support the malware analysis process. Therefore we explain the evolution and current state of those two groups of methods separately. Furthermore, we explain how the methodology used in our approach takes into account the related papers and builds a new approach upon this work.

6.1 Machine Learning Methods for Malware Detection

Machine learning has been used in multiple research efforts as a malware detection and classification method. Various features that characterize program behavior have been used as input data for the machine learning-based procedures: system calls [37], registry accesses [16], and network packets [31]. These event sequences are analyzed using unsupervised (e.g., clustering), semi-supervised, or

supervised learning (classification) methods. Static program code features have also been deployed for malware classification [29]. The classification methods can be further divided into one-class anomaly detection [16], binary classification [24], and multiclass learning [27]. One-class classification is used in case that we want to create a model for normal behavior (benign samples) and detect malware as a deviation from that model. In binary classification we optimize the classification boundary between benign and malicious samples. Multiclass classification methods are able to differentiate between different—previously known and defined—classes of malware instead or in addition to differentiating between benign and malicious samples.

Researchers that perform malware detection, usually maintain a sample set from different malware families with their static and behavioral patterns, and use them as a baseline to properly classify the suspicious applications. For instance, in the case of sequential data, automatic methods for extraction of relevant features can be used to cope with the possibly noisy and high-dimensional data. An example of this is given by recent application of statistical topic modeling approaches to the classification of system call sequences [40]. This approach could be extended by taking system call arguments as additional information and including memory allocation patterns and other traceable operations [39]. Support vector machines with string kernels represent an another novel methodology, where a standard classification scheme is augmented to work robustly with system call sequences of variable length [24]. However, most of these approaches only consider malware detection, and do not focus on classifying malware samples into families. Another example of sequential data is the network traffic. Towards this direction, the network traffic produced by the analyzed samples can be classified by taking into account the frequency and length of different types of packets or generating n-gram features out of packet payloads. As a matter of fact, researchers have already proposed various approaches to model the network data and design anomaly detection procedures for network infrastructures with purpose of network security [13, 20, 30, 41].

Previous works have considered many potential solutions for semantics-aware malware classification and analysis, including topic modeling. However, they have not dealt with the typical setting in malware analysis systems where a high number of samples is acquired online and models must be updated to give an accurate result. Therefore their methodology is only adequate in a scenario of offline malware analysis.

6.2 Big Data Malware Analysis Systems

Since security companies get overwhelmed with hundreds of thousands of malware samples on a daily basis, the problem of malware classification can be defined as a *Big Data* problem. Recently, there have been many efforts to create Big Data platforms for malware analysis. Examples of such systems are BinaryPig [14], Polonium [8], BitShred [17], and WINE [11]. BinaryPig is a system for distributed processing of data obtained by static malware analysis, leveraging

the recent advances in tools for Big Data domain. It uses Hadoop File System, MapReduce, and ElasticSearch as building blocks for scalable processing of static analysis data. Polonium is an another system for large-scale mining of malware. It leverages graph mining approaches to build a reputation-based system to identify malware among terabytes of anonymously submitted suspicious files. BitShred, on the other hand, is an attempt to design and build a scalable malware analysis system. It focuses on increasing efficiency of similarity analysis with feature hashing and uses similarity information for clustering. Finally, WINE is an approach that leverages Big Data and creates a scalable reputation-based security intelligence system, which also includes intrusion detection for network-based attacks.

These systems use machine learning-based technology and represent advances in scalability of malware detection and feature extraction. However, they do not emphasize on the development of statistical methods and do not consider semantic interpretability of the statistical models. Machine learning models very often need tuning and the absence of semantics can make such efforts extremely difficult for malware analysts. It is very important for analysts to be able to interpret the model in order to focus their efforts properly. In our approach, we do not only consider advanced topic modeling methodology for semantics-aware modeling, but we also take into account the scenario that a high influx of malware induces changes in the dataset and requires adaptation of the classification model. We automate this adaptation in order to maintain topic modeling feature extraction, using the nonparametric modeling methodology. Furthermore, our approach joins results of static and dynamic malware analysis and acknowledges the case where labeled examples are scarce.

7 Conclusion

In this paper, we presented an improved semi-supervised malware classification approach that joins the results from static and dynamic malware analysis to give an optimal classification performance. It uses separate algorithms for classification of static and dynamic analysis results: static analysis results are classified using a semi-supervised label propagation procedure, while the results from dynamic malware analysis are preprocessed by statistical topic modeling, which uncovers the latent semantically interpretable topics that capture the important properties of malware families. The method used for topic modeling is flexible and offers automatic adjustment of the topic set in case of online learning. Overall, our nonparametric approach creates an adaptive online system for malware classification that outperforms previous approaches.

Acknowledgments

The research was supported by the German Federal Ministry of Education and Research under grant 16KIS0328 (IUNO) and by the Bavarian State Ministry of Education, Science and the Arts as part of the FORSEC research association.

References

1. The Cuckoo Sandbox. `https://www.cuckoosandbox.org/`.
2. VirusTotal. `http://www.virustotal.com`.
3. V. M. Alvarez. Yara. `http://plusvic.github.io/yara/`.
4. B. Anderson, C. Storlie, and T. Lane. Improving Malware Classification: Bridging the Static/Dynamic Gap. In *Workshop on Security and Artificial Intelligence (AISec)*, 2012.
5. M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario. Automated Classification and Analysis of Internet Malware. In *International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2007.
6. U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, Behavior-Based Malware Clustering. In *ISOC Network and Distributed System Security Symposium (NDSS)*, 2009.
7. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
8. D. H. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos. Polonium: Tera-Scale Graph Mining and Inference for Malware Detection. In *SIAM International Conference on Data Mining (SDM)*, 2011.
9. G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu. Large-Scale Malware Classification Using Random Projections and Neural Networks. In *IEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.
10. S. T. Dumais. Latent Semantic Analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.
11. T. Dumitras and D. Shou. Toward a Standard Benchmark for Computer Security Research: The Worldwide Intelligence Network Environment (WINE). In *Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, 2011.
12. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases With Noise. In *Kdd*, 1996.
13. P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges. *Computers & Security*, 28(1):18–28, 2009.
14. Z. Hanif, T. Calhoun, and J. Trost. Binarypig: Scalable Static Binary Analysis Over Hadoop. *Black Hat USA*, 2013.
15. Z. Hanif, T. K. Lengyel, and G. D. Webster. Internet-Scale File Analysis. *Black Hat USA*, 2015.
16. K. Heller, K. Svore, A. D. Keromytis, and S. Stolfo. One Class Support Vector Machines for Detecting Anomalous Windows Registry Accesses. In *Workshop on Data Mining for Computer Security (DMSEC)*, 2003.
17. J. Jang, D. Brumley, and S. Venkataraman. Bitshred: Feature Hashing Malware for Scalable Triage and Semantic Analysis. In *Conference on Computer and Communications Security (CCS)*, 2011.
18. L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, 2004.
19. T. K. Lengyel, S. Maresca, B. D. Payne, G. D. Webster, S. Vogl, and A. Kiayias. Scalability, Fidelity and Stealth in the Drakvuf Dynamic Malware Analysis System. In *Annual Computer Security Applications Conference (ACSAC)*, 2014.
20. K. Leung and C. Leckie. Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. In *Australasian Conference on Computer Science*, 2005.

21. K. Maxwell. Maltrieve. `https://github.com/krmaxwell/maltrieve`.
22. D. Newman, C. Chemudugunta, P. Smyth, and M. Steyvers. Analyzing Entities and Topics in News Articles Using Statistical Topic Models. In *Intelligence and Security Informatics*, 2006.
23. R. Perdisci and M. C. U. VAMO: Towards a Fully Automated Malware Clustering Validity Analysis. In *Annual Computer Security Applications Conference (ACSAC)*, 2012.
24. J. Pfoh, C. Schneider, and C. Eckert. Leveraging String Kernels for Malware Detection. In *International Conference on Network and System Security*, 2013.
25. D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled LDA: A Supervised Topic Model for Credit Attribution in Multi-Labeled Corpora. In *Conference on Empirical Methods in Natural Language Processing*, 2009.
26. R. Řehůřek and P. Sojka. Software Framework for Topic Modelling With Large Corpora. In *Workshop on New Challenges for NLP Frameworks*, 2010.
27. K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov. Learning and Classification of Malware Behavior. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2008.
28. J.-M. Roberts. Virus Share. `https://virusshare.com/`.
29. M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo. Data Mining Methods for Detection of New Malicious Executables. In *Symposium on Security and Privacy*, 2001.
30. G. Stringhini, M. Egele, A. Zarras, T. Holz, C. Kruegel, and G. Vigna. B@bel: Leveraging Email Delivery for Spam Mitigation. In *USENIX Security Symposium*, 2012.
31. F. Tegeler, X. Fu, G. Vigna, and C. Kruegel. Botfinder: Finding Bots in Network Traffic Without Deep Packet Inspection. In *International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2012.
32. Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet Processes. *Journal of the american statistical association*, 101(476), 2006.
33. The MITRE Corporation. CRITS. `https://crits.github.io/`.
34. VirusTotal. File Statistics. `https://www.virustotal.com/en/statistics/`.
35. M. J. Wainwright and M. I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 2008.
36. C. Wang, J. W. Paisley, and D. M. Blei. Online Variational Inference for the Hierarchical Dirichlet Process. In *International Conference on Artificial Intelligence and Statistics*, 2011.
37. C. Warrender, S. Forrest, and B. Pearlmutter. Detecting Intrusions Using System Calls: Alternative Data Models. In *Symposium on Security and Privacy*, 1999.
38. G. Wicherski. Pehash: A Novel Approach to Fast Malware Clustering. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2009.
39. H. Xiao and C. Eckert. Efficient Online Sequence Prediction With Side Information. In *IEEE International Conference on Data Mining (ICDM)*, 2013.
40. H. Xiao and T. Stibor. A Supervised Topic Transition Model for Detecting Malicious System Call Sequences. In *Workshop on Knowledge Discovery, Modeling and Simulation*, 2011.
41. A. Zarras, A. Papadogiannakis, R. Gawlik, and T. Holz. Automated Generation of Models for Fast and Precise Detection of HTTP-Based Malware. In *Annual Conference on Privacy, Security and Trust (PST)*, 2014.
42. D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning With Local and Global Consistency. *Advances in Neural Information Processing Systems*, 16(16):321–328, 2004.