Predicting Malware Attributes from Cybersecurity Texts

Arpita Roy[1], **Youngja Park**[2], **Shimei Pan**[1]
[1]University of Maryland, Baltimore County
Baltimore, Maryland, USA
{arpita2, shimei}@umbc.edu
[2]IBM T. J. Watson Research Center
Yorktown Heights, New York, USA
young_park@us.ibm.com

Abstract

Text analytics is a useful tool for studying malware behavior and tracking emerging threats. The task of automated malware attribute identification based on cybersecurity texts is very challenging due to a large number of malware attribute labels and a small number of training instances. In this paper, we propose a novel feature learning method to leverage diverse knowledge sources such as small amount of human annotations, unlabeled text and specifications about malware attribute labels. Our evaluation has demonstrated the effectiveness of our method over the state-of-the-art malware attribute prediction systems.

1 Introduction

Securing computer systems has become a necessity for both organizations and individuals, as many cyber attacks result in devastating consequences. An outbreak of the WannaCry ransomware in 2017 affected more than 200,000 computers across 150 countries, with total damages ranging from hundreds of millions to billions of dollars (Berr, 2017). Detection of malware often relies on an understanding of the characteristics of malware behavior. To establish a standard for unambiguously characterizing malware, MAEC (Malware Attribute Enumeration and Characterization), a community-based project organized by MITRE, has specified a set of standard malware attributes (Kirillov et al., 2011). Based on MAEC, the actions of a malware can be categorized by four attributes: *ActionName*, *Capability*, *StrategicObjectives* and *TacticalObjectives*. *ActionName* specifies the actions taken by a malware. For example, "delete file" is a malware action that deletes existing files from affected systems. *Capability* defines the general capabilities of a malware. For example, "anti-removal" is a malware capability that prevents itself from being removed from a system. *StrategicObjectives* and *TacticalObjectives* are subcategories of *Capability* to capture more details. For example, a malware can have a *StrategicObjective* of "staging data for exfiltration" and a *TacticalObjective* of "moving data to a staging server". In total, MAEC specified 211 *ActionNames*, 20 *Capabilities*, 65 *StrategicObjectives*, and 148 *TacticalObjectives*.

The goal of this research is to automatically assign malware attribute labels based on cybersecurity texts. The task is challenging. The system needs to assign a large number of labels (444 in total). However, it is difficult to obtain sufficient training examples for each label since malware attribute labeling requires extensive cybersecurity knowledge and only domain experts can do this reliably. Given a large number of possible labels and a small number of training examples, typical supervised text classification techniques do not work well. In this work, we focus on incorporating additional knowledge sources to improve feature learning. The main contributions of this work include

1. Develop a novel ***malware attribute prediction system*** with the state of the art performance to automatically characterize malware behavior based on cybersecurity text.

2. Propose a novel ***Word Annotation Embedding (WAE) algorithm*** to encode diverse information from heterogeneous knowledge sources such as human annotations, raw texts and MAEC specifications.

Proceedings of NAACL-HLT 2019, pages 2857–2861
Minneapolis, Minnesota, June 2 - June 7, 2019.

3. Since WAE generates embeddings for both words and malware attribute labels, we construct ***high-quality predicting features*** based on both types of embeddings.

2 Related Work

Cybersecurity researchers have recently recognized the benefits of leveraging information extracted from security documents. Most prior work utilizing NLP for cybersecurity has focused on privacy policy analysis and Android security (Peng et al., 2012; Pandita et al., 2013; Qu et al., 2014; Slavin et al., 2016; Zhu and Dumitras, 2016). These systems aim to map written policies and the actual permission requests from Android applications and assess the risk level of these applications.

Lim et al. (2017) represents the first major effort to apply NLP techniques for general text-based malware behavior analysis. It processes reports by cybersecurity companies (e.g., FireEye, IBM X-Force, Symantec and Trend Micro) on malware or campaigns associated with Advanced Persistent Threat (APT) groups (Blanda and Westcott, 2018) and assign attributes to identified malware actions. They use word unigrams as predicting features. SVM and Naive Bayes are used to build classifiers for attribute label prediction.

To extend this effort, SemEval organized a shared task (called SecureNLP) on semantic analysis for cybersecurity texts. It adopted the same dataset and task definitions as (Lim et al., 2017). There are four subtasks in SemEval SecureNLP: (1) identifying sentences containing malware actions from APT reports; (2) identifying "Malware Action", "Subject of Action", "Object of Action" and "Modifier of Action"in the identified sentences; (3) identifying four relations,"Subject-Action", "Action-Object", "Modifier-Action" and "Action-Modifier" in identified sentences; (4) assigning attribute labels to each identified action based on the MAEC specification. Figure 1 shows an annotated example for these tasks. This paper describes our approach to solve subtask 4. The input to our system includes all the sentences identified in subtask 1 with additional labels for the entities identified in subtask 2. Each training and testing instance used in SemEval SecureNLP only contains a single malware action.

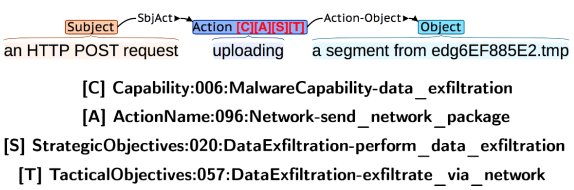

Figure 1: Annotated sentence fragment for SemEval shared task.

3 System Overview

Figure 2 shows the high-level system architecture. First, we augment all the raw APT reports with annotations to encode the knowledge from both the training data and MAEC. This allows us to design a unified representation for both types of knowledge. The annotated texts are then used by WAE to simultaneously learn embeddings for both words and malware attribute labels. The learned word and attribute label embeddings are then used to construct high qualify prediction features. Finally, we employ supervised machine learning to predict malware attribute labels. We build four classifiers, one for each malware attribute. Each classifier performs n+1-way classification, where n is the number of possible labels for each attribute and 1 is for 'no value' when the value of an attribute is not conveyed in the text.

4 Annotation Generation

To annotate text with additional information, the first step is to map each attribute label to a set of keywords based on both MAEC and the human annotations in the training data.

Identify Keywords from MAEC: Figure 3 shows a snippet of the MAEC specification. Each malware attribute label in MAEC includes a description and a few keywords. The malware action 004 in Figure 3 has a name 'emulate driver', a description 'specify the defined action of emulating an existing driver on a system' and two keywords: 'driver' and 'emulate'. Since the keywords carry the most essential information about a malware at-

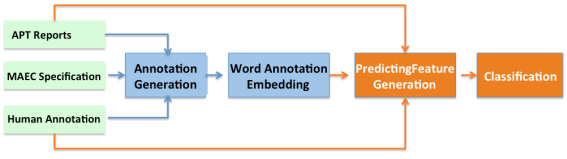

Figure 2: System Architecture

004	emulate driver	Specifies the defined Action of emulating an existing driver on a system.	[driver][emulate]
005	load and call driver	Specifies the defined Action of loading a driver into a system and then calling the loaded driver.	[driver][load][call]
006	load driver	Specifies the defined Action of loading a driver into a system.	[driver][load]
007	unload driver	Specifies the defined Action of unloading a driver from a system.	[driver][unload]

Figure 3: A Snippet of the MAEC Specification

tribute label, we link each label with these keywords (e.g., ActionName004: 'driver', 'emulate').

Identify Keywords from Training Data: Since a malware attribute label in the training data is at the sentence level, to extract keywords for each attribute label, we extract all the sentences associated with the same label and consider them one document. To select the most relevant keywords, we only keep those conveying "Malware Action", "Subject of Action", or "Object of Action" (which were identified in subtask 2).

Keywords Ranking: For the same attribute label, we merge the keywords from MAEC and those from the training data to form a single document. We then use TF-IDF scores to select the most informative keywords to differentiate these labels. In our experiments, we use the top 25 keywords based on their TF-IDF scores.

Text Annotation Generation Finally, for all the APT documents, we annotate the text with malware attribute labels. Specifically, for any word in the APT documents, if it is a keyword associated with K different labels, we annotate the word with K attribute labels.

5 Word Annotation Embedding (WAE)

Similar to word embedding (Mikolov et al., 2013), we want to learn features that capture the semantic relations between words. In addition, to facilitate attribute classification, we want to capture the semantic relations between words and their labels. Specifically, we want the words and their attribute labels to be close to each other in the embedding space and the embeddings of different labels to be far away from each other.

To achieve the goals, we developed a novel Word Annotation Embedding method. As shown in Figure 4, the target word is used to predict not only its context words but also its labels. To further strengthen their relations, the labels of the target word are also used to predict the target word. Specifically, given a sequence of T words (W_1,..,W_t,...,W_T) and their annotations (($A_{1,1}$,...,A_{1,M_1}),...,($A_{T,1}$,...,A_{T,M_T})), the objective of the WAE model is to maximize the average log probability shown in Equation 1, where C is the size of the context window, W_t is the target word, W_{t+j} is a context word, M_t is number of annotations W_t has and $A_{t,k}$ is the k-th annotation of W_t.

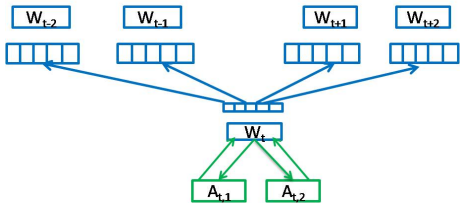

Figure 4: Architecture of WAE Model

$$\frac{1}{T}\sum_{t=1}^{T}\Big(\sum_{-C\le j\le C, j\neq 0}\log P(W_{t+j}|W_t) + \sum_{0\le k\le M_t}\left(\log P(A_{t,k}|W_t)+\log P(W_t|A_{t,k})\right)\Big) \quad (1)$$

Label-aware Negative Sampling: Negative sampling (Mikolov et al., 2013) was introduced as an approximation method in word2vec to improve the efficiency of model training. Previously, negative samples were selected either randomly or based on popularity. The method, however, is insensitive to class labels. Here, we propose a new annotation-aware negative sampling method to (1) keep different annotations apart in the embedding space and (2) to keep words associated with different labels apart, in addition to bringing words and their associated labels closer to each other via positive samples. To achieve this, WAE randomly selects (1) a word as a negative sample if it does not share the same annotations as the target word; (2) an attribute label as a negative sample if it is not the same as the labels of the target word.

6 Feature Generation and Classification

We construct six sets of features to train the classifiers.

(S1) WAE_W*+Sim:* Assume the average word embeddings for a given data instance generated by WAE is WE_{wae}, and the malware attribute label

Models	ActionName	Capability	StratObj	TactObj
(B1) (Lim et al., 2017)	0.33	0.41	0.27	0.22
(B2) Word2vec	0.40	0.57	0.41	0.36
(B3) Word2vec+Cap	NA	NA	0.41	0.39
(S1) WAE_W+Sim	0.44	0.61	0.43	0.41
(S2) $w.WAE_W$+Sim	**0.46**	0.62	0.44	0.43
(S3) WAE_W+WAE_L+Sim	0.45	**0.63**	0.46	0.43
(S4) $w.WAE_W$+WAE_L+Sim	0.45	**0.63**	0.45	0.43
(s5) WAE_W+WAE_L+Sim+Cap	NA	NA	**0.47**	**0.45**
(S6) $w.WAE_W$+WAE_L+Sim+Cap	NA	NA	**0.47**	**0.45**
Δ1 : over (Lim et al., 2017)	↑**39%** ($p < 0.05$)	↑**54%** ($p < 0.05$)	↑**74%**($p < 0.05$)	↑**105%** ($p < 0.05$)
Δ2 : over word2vec	↑**15%** ($p < 0.05$)	↑**11%**($p < 0.05$)	↑**15%**($p < 0.05$)	↑**25%**($p < 0.05$)
Δ3 : over word2vec+Cap	NA	NA	↑**15%** ($p < 0.05$)	↑**15%**($p < 0.05$)

Table 1: Evaluation Results. B1-B3 are the three baselines. S1-S6 are our models with six different sets of predicting features. Δ1-Δ3 are the improvements of our best models over the three baselines.

embedding learned by WAE is $LabelE_i$ for each label i. For each $LabelE_i$, we compute SIM_i, which is the cosine similarity between WE_{wae} and $LabelE_i$. WAE_W+*Sim* is the concatenation of WE_{wae} and all the SIM_i. For example, to predict *ActionName*, the model will include 100 word embedding features learned by WAE plus 211 similarity features, one for each *ActionNames*.

(S2) $w.WAE_W$+*Sim:* This feature set is similar to WAE_W+*Sim* except when computing WE_{wae}, we assign twice as much weight for a word with a label as one without a label. The intuition is words with labels are important keywords based on either MAEC or the training data.

(S3) WAE_W+WAE_L+*Sim:* It is similar to WAE_W+*Sim* except we also include the average embeddings of attribute labels associated with the instance.

(S4) $w.WAE_W$+WAE_L+*Sim:* This is the weighted version of (S3).

(S5) WAE_W+WAE_L+*Sim+Cap:* Since the label of *StrategicObjective* and *TacticalObjective* depends on the label of *Capability*, we added the *capability* label in the feature set. We use a 1-hot vector with 20 elements to encode a *Capability* label. We use the ground truth and the predicted label of *Capability* during training and testing respectively.

(S6) $w.WAE_W$+WAE_L+*Sim+Cap:* This is the weighted version of (S5)

7 Experiments

7.1 Dataset

The dataset used in the experiments was provided as a part of the SemEval shared task. It contains 456 APT reports (Blanda and Westcott, 2018), 39 of them were annotated by humans. Among them, 2975 sentences contain malware actions, which are the data instances used in this study. The annotated data are very sparse. Out of the 444 attribute labels, 190 labels do not appear in the labeled data. For the remaining 254 attribute labels, 92 labels occur less than five times, and 50 labels occur only once. In our experiments, we used the raw text in all the APT reports to train WAE. There are 16423 unique tokens and a total of 2544645 tokens in the dataset. We trained both word2vec and WAE with context window size 5 and 100 dimension vectors. The 39 annotated documents were divided into a training set (23 documents), a validation set (8 documents) and a test set (8 documents). Only the training dataset was used to generate annotations for WAE.

7.2 Evaluation Results

For classification, we tried both SVM and neural network-based models such as multilayer perceptron. After experimenting with different model parameters, we found that the best SVM model with a linear kernel performed slightly better than the best neural network models. We speculate that this might be because SVM is less likely to overfit when the training data are sparse. Table 1 shows the average F-scores over 5 runs by the SVM models on the test data. We compare our models with three baseline systems: (B1) (Lim et al., 2017), (B2) word2vec and (B3) word2vec + cap. Among them, (B1) represents the best published results on the same dataset. (B2) and (B3) are all the comparable models with embeddings learned by word2vec.

As shown in Table 1, all our models outperformed all the baseline systems. The improve-

ment over the word2vec model is 15%, 11%, 15% and 25% respectively, and, the improvement over (Lim et al., 2017), a previous state of the art, is 39%, 54%, 74% and 105% respectively. The improvement over $Word2vec + Cap$ is 15% and 15% respectively for *StrategicObjective* and *TechnicalObjective*. We also conducted t-tests to verify the significance of the improvements. The t-test results confirmed that our models significantly outperformed the baseline models with $p<0.05$. Moreover, the value of "Capability" seems to help the prediction of *StrategicObjective* and *TechnicalObjective*.

8 Conclusion

In this paper, we present a novel method to predict malware attribute labels from cybersecurity text. Given a large number of attribute labels and limited training data, we propose a new feature learning method to incorporate knowledge from diverse knowledge sources such as raw text, MAEC specifications and human annotations. We tested our system using the SemEval shared task data and our evaluation demonstrates that the features learned by our models are much more effective than an existing state of the art as well as embedding features learned by word2vec. Our investigation has highlighted the importance of incorporating diverse knowledge sources in complex classification tasks when human annotations are sparse.

References

Jonathan Berr. 2017. "wannacry" ransomware attack losses could reach $4 billion. *CBSNews*.

Kiran Blanda and David Westcott. 2018. Aptnotes. https://github.com/aptnotes/.

Ivan Kirillov, Desiree Beck, Penny Chase, and Robert Martin. 2011. Malware attribute enumeration and characterization.

Swee Kiat Lim, Aldrian Obaja Muis, Wei Lu, and Chen Hui Ong. 2017. Malwaretextdb: A database for annotated malware articles. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1557–1567.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Rahul Pandita, Xusheng Xiao, Wei Yang, William Enck, and Tao Xie. 2013. Whyper: Towards automating risk assessment of mobile applications. In *USENIX Security Symposium*, volume 2013.

Hao Peng, Chris Gates, Bhaskar Sarma, Ninghui Li, Yuan Qi, Rahul Potharaju, Cristina Nita-Rotaru, and Ian Molloy. 2012. Using probabilistic generative models for ranking risks of android apps. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12.

Zhengyang Qu, Vaibhav Rastogi, Xinyi Zhang, Yan Chen, Tiantian Zhu, and Zhong Chen. 2014. Autocog: Measuring the description-to-permission fidelity in android applications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1354–1365. ACM.

Rocky Slavin, Xiaoyin Wang, Mitra Bokaei Hosseini, James Hester, Ram Krishnan, Jaspreet Bhatia, Travis D Breaux, and Jianwei Niu. 2016. Toward a framework for detecting privacy policy violations in android application code. In *Proceedings of the 38th International Conference on Software Engineering*, pages 25–36. ACM.

Ziyun Zhu and Tudor Dumitras. 2016. Featuresmith: Automatically engineering features for malware detection by mining the security literature. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 767–778. ACM.