MalwareTextDB: A Database for Annotated Malware Articles

Swee Kiat Lim and **Aldrian Obaja Muis** and **Wei Lu**
Singapore University of Technology and Design
8 Somapah Road, Singapore, 487372
limsweekiat@gmail.com, {aldrian_muis,luwei}@sutd.edu.sg

Chen Hui Ong
DSO National Laboratories
20 Science Park Drive, Singapore, 118230
ochenhui@dso.org.sg

Abstract

Cybersecurity risks and malware threats are becoming increasingly dangerous and common. Despite the severity of the problem, there has been few NLP efforts focused on tackling cybersecurity.

In this paper, we discuss the construction of a new database for annotated malware texts. An annotation framework is introduced based around the MAEC vocabulary for defining malware characteristics, along with a database consisting of 39 annotated APT reports with a total of 6,819 sentences. We also use the database to construct models that can potentially help cybersecurity researchers in their data collection and analytics efforts.

1 Introduction

In 2010, the malware known as Stuxnet physically damaged centrifuges in Iranian nuclear facilities (Langner, 2011). More recently in 2016, a botnet known as Mirai used infected Internet of Things (IoT) devices to conduct large-scale Distributed Denial of Service (DDoS) attacks and disabled Internet access for millions of users in the US West Coast (US-CERT, 2016). These are only two cases in a long list ranging from ransomeware on personal laptops (Andronio et al., 2015) to taking over control of moving cars (Checkoway et al., 2011). Attacks such as these are likely to become increasingly frequent and dangerous as more devices and facilities become connected and digitized.

Recently, *cybersecurity defense* has also been recognized as one of the "*problem areas likely to be important both for advancing AI and for its long-run impact on society*" (Sutskever et al., 2016). In particular, we feel that natural language processing (NLP) has the potential for substantial contribution in cybersecurity and that this is a critical research area given the urgency and risks involved.

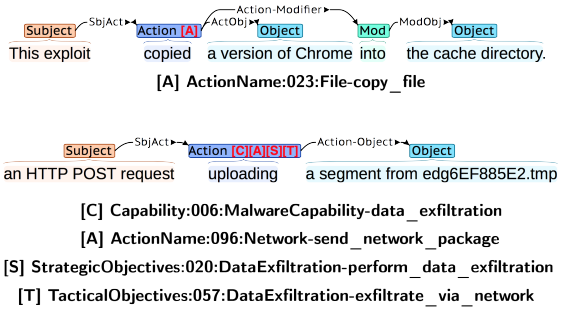

Figure 1: Annotated sentence and sentence fragment from MalwareTextDB. Such annotations provide semantic-level information to the text.

There exists a large repository of malware-related texts online, such as detailed malware reports by various cybersecurity agencies such as Symantec (DiMaggio, 2015) and Cylance (Gross, 2016) and in various blog posts. Cybersecurity researchers often consume such texts in the process of data collection. However, the sheer volume and diversity of these texts make it difficult for researchers to quickly obtain useful information. A potential application of NLP can be to quickly highlight critical information from these texts, such as the specific actions taken by a certain malware. This can help researchers quickly understand the capabilities of a specific malware and search in other texts for malware with similar capabilities.

An immediate problem preventing application of NLP techniques to malware texts is that such

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1557–1567
Vancouver, Canada, July 30 - August 4, 2017.
https://doi.org/10.18653/v1/P17-1143

texts are mostly unannotated. This severely limits their use in supervised learning techniques.

In light of that, we introduce a database of annotated malware reports for facilitating future NLP work in cybersecurity. To the best of our knowledge, this is the first database consisting of annotated malware reports. It is intended for public release, where we hope to inspire contributions from other research groups and individuals.

The main contributions of this paper are:

- We initiate a framework for annotating malware reports and annotate 39 Advanced Persistent Threat (APT) reports (containing 6,819 sentences) with attribute labels from the Malware Attribute Enumeration and Characterization (MAEC) vocabulary (Kirillov et al., 2010).
- We propose the following tasks, construct models for tackling them, and discuss the challenges:
  - Classify if a sentence is useful for inferring malware actions and capabilities,
  - Predict token, relation and attribute labels for a given malware-related text, as defined by the earlier framework, and
  - Predict a malware's signatures based only on text describing the malware.

2 Background

2.1 APTnotes

The 39 APT reports in this database are sourced from APTnotes, a GitHub repository of publicly-released reports related to APT groups (Blanda, 2016). The repository is constantly updated, which means it is a constant source of reports for annotations. While the repository consists of 384 reports (as of writing), we have chosen 39 reports from the year 2014 to initialize the database.

2.2 MAEC

The MAEC vocabulary was devised by The MITRE Corporation as a standardized language for describing malware (Kirillov et al., 2010). The MAEC vocabulary is used as a source of labels for our annotations. This will facilitate cross-applications in other projects and ensure relevance in the cybersecurity community.

2.3 Related Work

There are datasets available, which are used for more general tasks such as content extraction (Walker et al., 2006) or keyword extraction (Kim et al., 2010). These may appear similar to our dataset. However, a big difference is that we are not performing general-purpose annotation and not all tokens are annotated. Instead, we only annotated tokens relevant to malware capabilities and provide more valuable information by annotating the type of malware capability or action implied. These are important differentiating factors, specifically catered to the cybersecurity domain.

While we are not aware of any database catering specifically to malware reports, there are various databases in the cybersecurity domain that provide malware hashes, such as the National Software Reference Library (NSRL) (NIST, 2017; Mead, 2006) and the File Hash Repository (FHR) by the Open Web Application Security Project (OWASP, 2015).

Most work on classifying and detecting malware has also been focusing on detecting system calls (Alazab et al., 2010; Briones and Gomez, 2008; Willems et al., 2007; Qiao et al., 2013). More recently, Rieck et al. (2011) has incorporated machine learning techniques for detecting malware, again through system calls. To the best of our knowledge, we are not aware of any work on classifying malware based on analysis of malware reports. By building a model that learns to highlight critical information on malware capabilities, we feel that malware-related texts can become a more accessible source of information and provide a richer form of malware characterization beyond detecting file hashes and system calls.

3 Data Collection

We worked together with cybersecurity researchers while choosing the preliminary dataset, to ensure that it is relevant for the cybersecurity community. The factors considered when selecting the dataset include the mention of most current malware threats, the range of author sources, with blog posts and technical security reports, and the range of actor attributions, from several suspected state actors to smaller APT groups.

3.1 Preprocessing

After the APT reports have been downloaded in PDF format, the PDFMiner tool (Shinyama, 2004) is used to convert the PDF files into plaintext format. The reports often contain non-sentences, such as the cover page or document header and

footer. We went through these non-sentences manually and subsequently removed them before the annotation. Hence only complete sentences are considered for subsequent steps.

3.2 Annotation

The Brat Rapid Annotation Tool (Stenetorp et al., 2012) is used to annotate the reports. The main aim of the annotation is to map important word phrases that describe malware actions and behaviors to the relevant MAEC vocabulary, such as the ones shown in Figure 1. We first extract and enumerate the labels from the MAEC vocabulary, which we call attribute labels. This gives us a total of 444 attribute labels, consisting of 211 ActionName labels, 20 Capability labels, 65 StrategicObjectives labels and 148 TacticalObjectives labels. These labels are elaborated in Section 3.5.

There are three main stages to the annotation process. These are cumulative and eventually build up to the annotation of the attribute labels.

3.3 Stage 1 - Token Labels

The first stage involves annotating the text with the following *token labels*, illustrated in Figure 2:

Action This refers to an event, such as "*registers*", "*provides*" and "*is written*".

Subject This refers to the initiator of the Action such as "*The dropper*" and "*This module*".

Object This refers to the recipient of the Action such as "*itself*", "*remote persistent access*" and "*The ransom note*"; it also refers to word phrases that provide elaboration on the Action such as "*a service*", "*the attacker*" and "*disk*".

Modifier This refers to tokens that link to other word phrases that provide elaboration on the Action such as "*as*" and "*to*".

This stage helps to identify word phrases that are relevant to the MAEC vocabulary. Notice that for the last sentence in Figure 2, "*The ransom note*" is tagged as an Object instead of a Subject. This is because the Action "*is written*" is not being initiated by "*The ransom note*". Instead, the Subject is absent in this sentence.

3.4 Stage 2 - Relation Labels

The second stage involves annotating the text with the following *relation labels*:

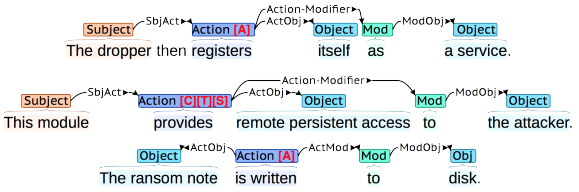

Figure 2: Examples of annotated sentences.

The regime's greatest fear is internal dissent and resulting destabilization.

Songun is North Korea's "military first" doctrine.

Figure 3: Examples of irrelevant sentences.

SubjAction This links an Action with its relevant Subject.

ActionObj This links an Action with its relevant Object.

ActionMod This links an Action with its relevant Modifier.

ModObj This links a Modifier with the Object that provides elaboration.

This stage helps to make the links between the labelled tokens explicit, which is important in cases where a single Action has multiple Subjects, Objects or Modifiers. Figure 2 demonstrates how the relation labels are used to link the token labels.

3.5 Stage 3 - Attribute Labels

The third stage involves annotating the text with the *attribute labels* extracted from the MAEC vocabulary. Since the Action is the main indicator of a malware's action or capability, the attribute labels are annotated onto the Actions tagged in Stage 1. Each Action should have one or more attribute labels.

There are four classes of attribute labels: ActionName, Capability, StrategicObjectives and TacticalObjectives. These labels describe different actions and capabilities of the malware. Refer to Appendix A for examples and elaboration.

3.6 Summary

The above stages complete the annotation process and is done for each document. There are also sentences that are not annotated at all since they do not provide any indication of malware actions or capabilities, such as the sentences in Figure 3. We call these sentences *irrelevant sentences*.

At the time of writing, the database consists of 39 annotated APT reports with a combined total of 6,819 sentences. Out of the 6,819 sentences,

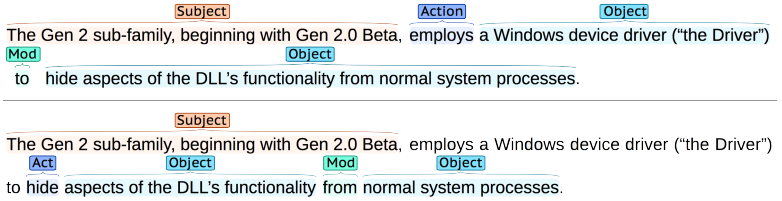

Figure 4: Two different ways for annotating a sentence, where both seem to be equally satisfactory to a human annotator. In this case, both serve to highlight the malware's ability to hide its DLL's functionality.

Token Labels (by label)		**Relation Labels** (by label)		**Attribute Labels** (by class)	
Subj	1,778	SubjAction	2,343	ActionName	982
Obj	4,411	ActionObj	2,713	Capability	2,524
Act	2,975	ActionMod	1,841	StratObj	2,004
Mod	1,819	ModObj	1,808	TactObj	1,592
Total	**10,983**	**Total**	**8,705**	**Total**	**7,102**

Table 1: Breakdown of annotation statistics.

2,080 sentences are annotated. Table 1 shows the breakdown of the annotation statistics.

3.7 Annotators' Challenges

We can calculate the Cohen's Kappa (Cohen, 1960) to quantify the agreement between annotators and to give an estimation of the difficulty of this task for human annotators. Using annotations from pairs of annotators, the Cohen's Kappa was calculated to be 0.36 for annotation of the Token labels. This relatively low agreement between annotators suggests that this is a rather difficult task. In the following subsections, we discuss some possible reasons that make this annotation task difficult.

3.7.1 Complex Sentence Structures

In many cases, there may be no definite way to label the tokens. Figure 4 shows two ways to annotate the same sentence. Both annotations essentially serve to highlight the Gen 2 sub-family's capability of hiding the DLL's functionality. The first annotation highlights the method used by the malware to hide the library, i.e., employing the Driver. The second annotation focuses on the malware hiding the library and does not include the method. Also notice that the Modifiers highlighted are different in the two cases, since this depends on the Action highlighted and are hence mutually exclusive. Such cases occur more commonly when the sentences contain complex noun- and verb-phrases that can be decomposed in several ways. Repercussions surface later in the experiments described in Section 5.2, specifically in the second point under Discussion.

3.7.2 Large Quantity of Labels

Due to the large number (444) of attribute labels, it is challenging for annotators to remember all of the attribute labels. Moreover, some of the attribute labels are subject to interpretation. For instance, should *Capability: 005: MalwareCapability-command_and_control* be tagged for sentences that mention the location or IP addresses of command and control servers, even though such sentences may not be relevant to the capabilities of the malware?

3.7.3 Specialized Domain Knowledge Required

Finally, this task requires specialized cybersecurity domain knowledge from the annotator and the ability to apply such knowledge in a natural language context. For example, given the phrase "*load the DLL into memory*", the annotator has to realize that this phrase matches the attribute label *ActionName: 119: ProcessMemory-map_library_into_process*. The abundance of labels with the many ways that each label can be expressed in natural language makes this task extremely challenging.

4 Proposed Tasks

The main goal of creating this database is to aid cybersecurity researchers in parsing malware-related texts for important information. To this end, we propose several tasks that build up to this main goal.

Task 1 Classify if a sentence is relevant for inferring malware actions and capabilities

Task 2 Predict token labels for a given malware-related text

Task 3 Predict relation labels for a given malware-related text

Task 4 Predict attribute labels for a given malware-related text

Task 5 Predict a malware's signatures based on the text describing the malware and the text's annotations

Task 1 arose from discussions with domain experts where we found that a main challenge for cybersecurity researchers is having to sift out critical sentences from lengthy malware reports and articles. Figure 3 shows sentences describing the political and military background of North Korea in the APT report *HPSR SecurityBriefing_Episode16_NorthKorea*. Such information is essentially useless for cybersecurity researchers focused on malware actions and capabilities. It will be helpful to build a model that can filter relevant sentences that pertain to malware.

Tasks 2 to 4 serve to automate the laborious annotation procedure as described earlier. With sufficient data, we hope that it becomes possible to build an effective model for annotating malware-related texts, using the framework and labels we defined earlier. Such a model will help to quickly increase the size of the database, which in turn facilitate other supervised learning tasks.

Task 5 explores the possibility of using malware texts and annotations to predict a malware's signatures. While conventional malware analyzers generate a list of malware signatures based on the malware's activities in a sandbox, such analysis is often difficult due to restricted distribution of malware samples. In contrast, numerous malware reports are freely available and it will be helpful for cybersecurity researchers if such texts can be used to predict malware signatures instead of having to rely on a limited supply of malware samples.

In the following experiments, we construct models for tackling each of these tasks and discuss the performance of our models.

5 Experiments and Results

Since the focus of this paper is on the introduction of a new framework and database for annotating malware-related texts, we only use simple algorithms for building the models and leave more complex models for future work.

For the following experiments, we use linear support vector machine (SVM) and multinomial Naive Bayes (NB) implementations in the scikit-learn library (Pedregosa et al., 2011). The regularization parameter in SVM and smoothing parameter in NB were tuned (with the values 10^{-3} to 10^{3} in logarithmic increments) by taking the value that gave the best performance in development set.

	P	R	F_1
SVM	69.7	54.0	60.5
NB	59.5	68.5	63.2

Table 2: Task 1 scores: classifying relevant sentences.

For experiments where Conditional Random Field (CRF) (Lafferty et al., 2001) is used, we utilized the CRF++ implementation (Kudo, 2005).

For scoring the predictions, unless otherwise stated, we use the metrics module in scikit-learn for SVM and NB, as well as the CoNLL2000 conlleval Perl script for CRF[1].

Also, unless otherwise mentioned, we make use of all 39 annotated documents in the database. The experiments are conducted with a 60%/20%/20% training/development/test split, resulting in 23, 8 and 8 documents in the respective datasets. Each experiment is conducted 5 times with a different random allocation of the dataset splits and we report averaged scores[2].

Since we focus on building a database, we weigh recall and precision as equally important in the following experiments and hence focus on the F_1 score metric. The relative importance of recall against precision will ultimately depend on the downstream tasks.

5.1 Task 1 - Classify sentences relevant to malware

We make use of the annotations in our database for this supervised learning task and consider a sentence to be relevant as long as it has an annotated token label. For example, the sentences in Figure 2 will be labeled relevant whereas the sentences in Figure 3 will be labeled irrelevant.

A simple bag-of-words model is used to represent each sentence. We then build two models – SVM and NB – for tackling this task.

Results: Table 2 shows that while the NB model outperforms the SVM model in terms of F_1 score, the performance of both models are still rather low with F_1 scores below 70 points. We proceed to discuss possible sources of errors for the models.

[1] www.cnts.ua.ac.be/conll2000/chunking/output.html

[2] Note that therefore the averaged F_1 may not be the harmonic mean of averaged P and R in the result tables.

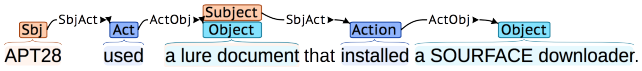

Figure 5: An example of a token ("*a lure document*") labelled as both Subject and Object. In the first case, it is the recipient of the Action "*used*", while in the latter case, it is the initiator of the Action "*installed*".

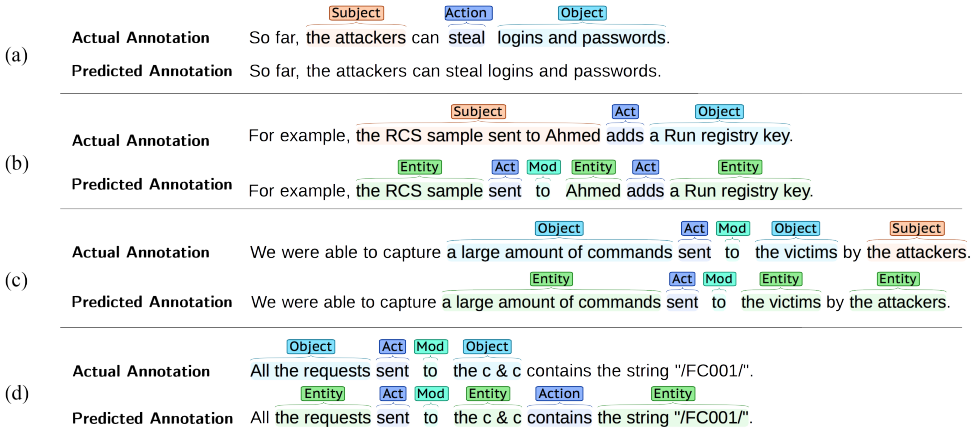

Figure 6: Actual and predicted annotations. For predicted annotations, the Entity label replaces the Subject and Object labels.

Discussion: We find that there are two main types of misclassified sentences.

1. Sentences describing malware without implying specific actions

These sentences often contain malware-specific terms, such as "*payload*" and "*malware*" in the following sentence.

This file is the main payload of the malware.

These sentences are often classified as relevant, probably due to the presence of malware-specific terms. However, such sentences are actually irrelevant because they merely describe the malware but do not indicate specific malware actions or capabilities.

2. Sentences describing attacker actions

Such sentences mostly contain the term "*attacker*" or names of attackers. For instance, the following sentence is incorrectly classified as irrelevant.

This is another remote administration tool often used by the Pitty Tiger crew.

Such sentences involving the attacker are often irrelevant since the annotations focus on the malware and not the attacker. However, the above sentence implies that the malware is a remote administration tool and hence is a relevant sentence that implies malware capability.

5.2 Task 2 - Predict token labels

Task 2 concerns automating Stage 1 for the annotation process described in Section 3.3. Within the annotated database, we find several cases where a single word-phrase may be annotated with both Subject and Object labels (see Figure 5). In order to simplify the model for prediction, we redefine Task 2 as predicting Entity, Action and Modifier labels for word-phrases. The single Entity label is used to replace both Subject and Object labels. Since the labels may extend beyond a single word token, we use the BIO format for indicating the span of the labels (Sang and Veenstra, 1999). We use two approaches for tackling this task: a) CRF is used to train a model for directly predicting token labels, b) A pipeline approach where the NB model from Task 1 is used to filter relevant sentences. A CRF model is then trained to predict token labels for relevant sentences.

The CRF model in Approach 1 is trained on the entire training set, whereas the CRF model in Approach 2 is trained only on the gold relevant sentences in the training set.

For features in both approaches, we use unigrams and bigrams, part-of-Speech labels from the Stanford POStagger (Toutanova et al., 2003), and Brown clustering features after optimizing the cluster size (Brown et al., 1992). A C++ implementation of the Brown clustering algorithm is

	Approach 1			Approach 2		
Token Label	**P**	**R**	**F_1**	**P**	**R**	**F_1**
Entity	48.8	25.1	32.9	42.8	33.8	37.6
Action	55.2	30.3	38.9	50.8	41.1	45.2
Modifier	55.7	28.4	37.3	48.9	37.4	42.1
Average	51.7	27.0	35.2	45.9	36.3	40.3

Table 3: Task 2 scores: predicting token labels.

	Approach 1			Approach 2		
Token Label	**P**	**R**	**F_1**	**P**	**R**	**F_1**
Entity	63.6	32.1	42.3	56.5	46.3	50.6
Action	60.2	31.4	41.0	54.6	42.8	47.7
Modifier	56.4	28.1	37.1	50.1	37.1	42.3
Average	62.7	31.8	41.9	55.9	45.3	49.8

Table 4: Task 2 relaxed/token-level scores.

Relation Label	**P**	**R**	**F_1**
SubjAction	86.3	82.3	84.2
ActionObj	91.6	86.2	88.8
ActionMod	98.5	96.4	97.4
ModObj	98.0	96.7	97.4
Average	89.2	89.4	89.3

Table 5: Task 3 scores: predicting relation labels.

used (Liang, 2005). The Brown cluster was trained on a larger corpus of APT reports, consisting of 103 APT reports not in the annotated database and the 23 APT reports from the training set. We group together low-frequency words that appear 4 or less times in the set of 126 APT reports into one cluster and during testing we assign new words into this cluster.

Results: Table 3 demonstrates that Approach 2 outperforms Approach 1 on most scores. Nevertheless, both approaches still give low performance for tackling Task 2 with F_1-scores below 50 points.

Discussion: There seem to be three main categories of wrong predictions:

1. Sentences describing attacker actions

Such sentences are also a main source of prediction errors in Task 1. Again, most sentences describing attackers are deemed irrelevant and left unannotated because we focus on malware actions rather than human attacker actions. However, these sentences may be annotated in cases where the attacker's actions imply a malware action or capability.

For example, the Figure 6a describes the attackers stealing credentials. This implies that the malware used is capable of stealing and exfiltrating credentials. It may be challenging for the model to distinguish whether such sentences describing attackers should be annotated since a level of inference is required.

2. Sentences containing noun-phrases made up of participial phrases and/or prepositional phrases

These sentences contain complex noun-phrases with multiple verbs and prepositions, such as in Figures 6b and 6c. In Figure 6b, "*the RCS sample sent to Ahmed*" is a noun-phrase annotated as a single Subject/Entity. However, the model decomposes the noun-phrase into the subsidiary noun "*the RCS sample*" and participial phrase "*sent to Ahmed*" and further decompose the participial phrase into the constituent words, predicting Action, Modifier and Entity labels for "*sent*", "*to*" and "*Ahmed*" respectively. There are cases where such decomposition of noun-phrases is correct, such as in Figure 6c.

As mentioned in Section 3.7, this is also a challenge for human annotators because there may be several ways to decompose the sentence, many of which serve equally well to highlight certain malware aspects (see Figure 4).

Whether such decomposition is correct depends on the information that can be extracted from the decomposition. For instance, the decomposition in Figure 6c implies that the malware can receive remote commands from attackers. In contrast, the decomposition predicted by the model in Figure 6b does not offer any insight into the malware. This is a difficult task that requires recognition of the phrase spans and the ability to decide which level of decomposition is appropriate.

3. Sentences containing noun-phrases made up of determiners and adjectives

These sentences contain noun-phrases with determiners and adjectives such as "*All the requests*" in Figure 6d. In such cases, the model may only predict the Entity label for part of the noun-phrase. This is shown in Figure 6d, where the model predicts the Entity label for "*the requests*" instead of "*All the requests*".

Thus, we also consider a relaxed scoring scheme where predictions are scored in token level instead of phrase level (see Table 4). The aim of the relaxed score is to give credit to the model when the span for a predicted label intersects with the span for the actual label, as in Figure 6d.

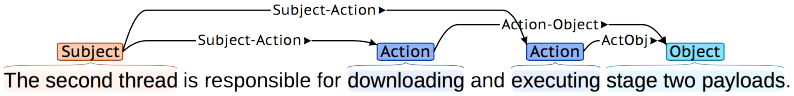

Figure 7: An example of an entity with multiple parents. In this case, *stage two payloads* has two parents by ActionObject relations - *downloading* and *executing*.

5.3 Task 3 - Predict relation labels

Following the prediction of token labels in Task 2, we move on to Task 3 for building a model for predicting relation labels. Due to the low performance of the earlier models for predicting token labels, for this experiment we decided to use the gold token labels as input into the model for predicting relation labels. Nevertheless, the models can still be chained in a pipeline context.

The task initially appeared to be similar to a dependency parsing task where the model predicts dependencies between the entities demarcated by the token labels. However, on further inspection, we realized that there are several entities which have more than one parent entity (see Figure 7). As such, we treat the task as a binary classification task, by enumerating all possible pairs of entities and predicting whether there is a relation between each pair.

Predicting the relation labels from the token labels seem to be a relatively straightforward task and hence we design a simple rule-based model for the predictions. We tuned the rule-based model on one of the documents (*AdversaryIntelligenceReport_DeepPanda_0 (1)*) and tested it on the remaining 38 documents. The rules are documented in Appendix B.

Results: Table 5 shows the scores from testing the model on the remaining 38 documents.

The results from the rule-based model are better than expected, with the average F_1-scores exceeding 84 points for all the labels. This shows that the relation labels can be reliably predicted given good predictions of the preceding token labels.

Discussion: The excellent performance from the rule-based model suggests that there is a well-defined structure in the relations between the entities. It may be possible to make use of this inherent structure to help improve the results for predicting the token labels.

Also, notice that by predicting the SubjAction, ActionObj and ActionMod relations, we are simultaneously classifying the ambiguous Entity labels into specific Subject and Object labels. For instance, Rule 1 predicts a ModObj relation between a Modifier and an Entity, implying that the Entity is an Object, whereas Rule 3 predicts a SubjAction relation between an Entity and an Action, implying that the Entity is a Subject.

Attribute Category	NB			SVM		
	P	R	F_1	P	R	F_1
ActionName	35.2	23.9	28.0	43.9	27.9	33.9
Capability	41.5	39.8	40.6	42.5	41.1	41.8
StrategicObjectives	33.7	24.4	28.3	32.2	23.5	27.2
TacticalObjectives	27.6	17.4	21.1	30.2	18.4	22.7

Table 6: Task 4 scores: predicting attribute labels.

5.4 Task 4 - Predict attribute labels

A significant obstacle in the prediction of attribute labels is the large number of attribute labels available. More precisely, we discover that many of these attribute labels occur rarely, if not never, in the annotated reports. This results in a severely sparse dataset for training a model.

Due to the lack of substantial data, we decide to use *token groups* instead of entire sentences for predicting attribute labels. Token groups are the set of tokens that are linked to each other via relation labels. We extract the token groups from the gold annotations and then build a model for predicting the attribute labels for each token group. Again, we use a bag-of-words model to represent the token groups while SVM and NB are each used to build a model for predicting attribute labels.

Results: Table 6 shows the average scores over 5 runs for the four separate attribute categories. For this task, SVM appears to perform generally better than NB, although much more data seems to be required in order to train a reliable model for predicting attribute labels. The Capability category shows the best performance, which is to be expected, since the Capability attributes occur the most frequently.

Discussion: The main challenge for this task is the sparse data and the abundant attribute labels available. In fact, out of the 444 attribute labels, 190 labels do not appear in the database. For the remaining 254 attribute labels that do occur in the database, 92 labels occur less than five times and 50 labels occur only once. With the sparse data

Features Used	NB			SVM		
	P	R	F_1	P	R	F_1
Text only	58.8	50.8	53.5	49.3	47.0	47.2
Ann. only	64.7	55.0	58.0	62.6	57.2	59.2
Text and Ann.	59.3	50.7	53.6	54.3	51.1	51.6

Table 7: Task 5 scores: predicting malware signatures using text and annotations.

available, particularly for rare attribute labels, effective one-shot learning models might have to be designed to tackle this difficult task.

5.5 Task 5 - Predict malware signatures using text and annotations

Conventional malware analyzers, such as malwr.com, generate a list of signatures based on the malware's activities in a sandbox. Examples of such signatures include *antisandbox_sleep*, which indicates anti-sandbox capabilities or *persistence_autorun*, which indicates persistence capabilities.

If it is possible to build an effective model to predict malware signatures based on natural language texts about the malware, this can help cybersecurity researchers predict signatures of malware samples that are difficult to obtain, using the malware reports freely available online.

By analyzing the hashes listed in each APT report, we obtain a list of signatures for the malware discussed in the report. However, we are unable to obtain the signatures for several hashes due to restricted distribution of malware samples. There are 8 APT reports without any obtained signatures, which are subsequently discarded for the following experiments. This leaves us with 31 out of 39 APT reports.

The current list of malware signatures from Cuckoo Sandbox[3] consists of 378 signature types. However, only 68 signature types have been identified for the malware discussed in the 31 documents. Furthermore, out of these 68 signature types, 57 signature types appear less than 10 times, which we exclude from the experiments. The experiments that follow will focus on predicting the remaining 11 signature types using the 31 documents.

The OneVsRestClassifier implementation in scikit-learn is used in the following experiments, since this is a multilabel classification problem. We also use SVM and NB to build two types of models for comparison.

Three separate methods are used to generate features for the task: a) the whole text in each APT report is used as features via a bag-of-words representation, without annotations, b) the gold labels from the annotations are used as features, without the text, and c) both the text and the gold annotations are used, via a concatenation of the two feature vectors.

Results: Comparing the first two rows in Table 7, we can see that using the annotations as features significantly improve the results, especially the precision. SVM model also seems to benefit more from the annotations, even outperforming NB in one case.

Discussion: The significant increase in precision suggests that the annotations provide a condensed source of features for predicting malware signatures, improving the models' confidence. We also observe that some signatures seem to benefit more from the annotations, such as *persistence_autorun* and *has_pdb*. In particular, *persistence_autorun* has a direct parallel in attribute labels, which is *MalwareCapability-persistence*, showing that using MAEC vocabulary as attribute labels is appropriate.

6 Conclusion

In this paper, we presented a framework for annotating malware reports. We also introduced a database with 39 annotated APT reports and proposed several new tasks and built models for extracting information from the reports. Finally, we discuss several factors that make these tasks extremely challenging given currently available models. We hope that this paper and the accompanying database serve as a first step towards NLP being applied in cybersecurity and that other researchers will be inspired to contribute to the database and to construct their own datasets and implementations. More details about this database can be found at `http://statnlp.org/research/re/`.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work is supported by ST Electronics – SUTD Cyber Security Laboratory Project 1 Big Data Security Analytics, and is partly supported by MOE Tier 1 grant SUTDT12015008.

[3] https://cuckoosandbox.org/

References

Mamoun Alazab, Sitalakshmi Venkataraman, and Paul Watters. 2010. Towards Understanding Malware Behaviour by the Extraction of API Calls. In *2010 Second Cybercrime and Trustworthy Computing Workshop*. IEEE, November 2009, pages 52–59. https://doi.org/10.1109/CTC.2010.8.

Nicoló Andronio, Stefano Zanero, and Federico Maggi. 2015. *HelDroid: Dissecting and Detecting Mobile Ransomware*, Springer International Publishing, Cham, chapter 18, pages 382–404. https://doi.org/10.1007/978-3-319-26362-5.

Kiran Blanda. 2016. APTnotes. https://github.com/aptnotes/.

Ismael Briones and Aitor Gomez. 2008. Graphs, entropy and grid computing: Automatic comparison of malware. In *Virus bulletin conference*. pages 1–12.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based N-gram Models of Natural Language. *Comput. Linguist.* 18(4):467–479. http://dl.acm.org/citation.cfm?id=176313.176316.

Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proceedings of the 20th USENIX Conference on Security*. USENIX Association, Berkeley, CA, USA, SEC'11, pages 6–6. http://dl.acm.org/citation.cfm?id=2028067.2028073.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20(1):37–46. https://doi.org/10.1177/001316446002000104.

Jon DiMaggio. 2015. The Black Vine cyberespionage group. Technical report, Symantec. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-black-vine-cyberespionage-group.pdf.

Jon Gross. 2016. Operation Dust Storm. Technical report, Cylance. https://www.cylance.com/hubfs/2015_cylance_website/assets/operation-dust-storm/Op_Dust_Storm_Report.pdf.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Stroudsburg, PA, USA, SemEval '10, pages 21–26. http://dl.acm.org/citation.cfm?id=1859664.1859668.

Ivan Kirillov, Desiree Beck, Penny Chase, and Robert Martin. 2010. Malware Attribute Enumeration and Characterization. *The MITRE Corporation, Tech. Rep* .

Taku Kudo. 2005. CRF++. https://taku910.github.io/crfpp/.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *International Conference on Machine Learning (ICML 2001)*. pages 282–289. http://dl.acm.org/citation.cfm?id=655813.

R. Langner. 2011. Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Security Privacy* 9(3):49–51. https://doi.org/10.1109/MSP.2011.67.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Master's thesis, Massachusetts Institute of Technology. https://doi.org/1721.1/33296.

Steve Mead. 2006. Unique file identification in the National Software Reference Library. *Digital Investigation* 3(3):138 – 150. https://doi.org/10.1016/j.diin.2006.08.010.

NIST. 2017. National Software Reference Library. http://www.nsrl.nist.gov/.

OWASP. 2015. OWASP File Hash Repository. https://www.owasp.org/index.php/OWASP_File_Hash_Repository.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12:2825–2830. http://dl.acm.org/citation.cfm?id=2078195.

Yong Qiao, Jie He, Yuexiang Yang, and Lin Ji. 2013. Analyzing Malware by Abstracting the Frequent Itemsets in API Call Sequences. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, pages 265–270. https://doi.org/10.1109/TrustCom.2013.36.

Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. 2011. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security* 19(4):639–668. https://doi.org/10.3233/JCS-2010-0410.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing Text Chunks. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Morristown, NJ, USA, page 173. https://doi.org/10.3115/977035.977059.

Yusuke Shinyama. 2004. PDFMiner. https://euske.github.io/pdfminer/.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: A Web-based Tool for NLP-assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, EACL '12, pages 102–107. http://dl.acm.org/citation.cfm?id=2380921.2380942.

Ilya Sutskever, Dario Amodei, and Sam Altman. 2016. Special projects. Technical report, OpenAI, https://openai.com/blog/special-projects/. https://openai.com/blog/special-projects/.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL '03, pages 173–180. https://doi.org/10.3115/1073445.1073478.

US-CERT. 2016. Heightened DDoS Threat Posed by Mirai and Other Botnets. Technical report, United States Computer Emergency Readiness Team. https://www.us-cert.gov/ncas/alerts/TA16-288A.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia* 57.

Carsten Willems, Thorsten Holz, and Felix Freiling. 2007. Toward Automated Dynamic Malware Analysis Using CWSandbox. *IEEE Security and Privacy Magazine* 5(2):32–39. https://doi.org/10.1109/MSP.2007.45.

A Attribute Labels

The following elaborates on the types of malware actions described by each class of attribute labels and gives specific examples.

A.1 ActionName

The ActionName labels describe specific actions taken by the malware, such as downloading a file *ActionName: 090: Network- download_ file* or creating a registry key *ActionName: 135: Registry-create_registry_key*.

A.2 Capability

The Capability labels describe general capabilities of the malware, such as exfiltrating stolen data *Capability: 006: MalwareCapability-data_exfiltration* or spying on the victim *Capability: 019: MalwareCapability-spying*.

A.3 StrategicObjectives

The StrategicObjectives labels elaborate on the Capability labels and provide more details on the capabilities of the malware, such as preparing stolen data for exfiltration *StrategicObjectives: 021: DataExfiltration-stage_data_for_exfiltration* or capturing information from input devices connected to the victim's machine *StrategicObjectives: 061: Spying-capture_system_input_peripheral_data*.

Each StrategicObjectives label belongs to a Capability label.

A.4 TacticalObjectives

The TacticalObjectives labels provide third level of details on the malware's capability, such as encrypting stolen data for exfiltration *TacticalObjectives: 053: DataExfiltration-encrypt_data* or an ability to perform key-logging *TacticalObjectives: 140: Spying-capture_keyboard_input*.

Again, each TacticalObjectives label belongs to a Capability label.

B Rules for Rule-based Model in Task 3

The following are the rules used in the rule-based model described in Section 5.3.

1. If a Modifier is followed by an Entity, a ModObj relation is predicted between the Modifier and the Entity
2. If an Action is followed by an Entity, an ActionObj relation is predicted between the Action and the Entity
3. If an Entity is followed by an Action of token-length 1, a SubjAction relation is predicted between the Entity and the Action
4. An ActionObj relation is predicted between any Action that begins with *be* and the most recent previous Entity
5. An ActionObj relation is predicted between any Action that begins with *is*, *was*, *are* or *were* and ends with *-ing* and the most recent previous Entity
6. An ActionMod relation is predicted between all Modifiers and the most recent previous Action