SemEval-2018 Task 8: Semantic Extraction from CybersecUrity REports using Natural Language Processing (SecureNLP)

Peter Phandi **Amila Silva** **Wei Lu**
8 Somapah Road, Singapore 487372
Singapore University of Technology and Design
{peter_phandi,amila_silva,wei_lu}@sutd.edu.sg

Abstract

This paper describes the SemEval 2018 shared task on semantic extraction from cybersecurity reports, which is introduced for the first time as a shared task on SemEval. This task comprises four SubTasks done incrementally to predict the characteristics of a specific malware using cybersecurity reports. To the best of our knowledge, we introduce the world's largest publicly available dataset of annotated malware reports in this task. This task received in total 18 submissions from 9 participating teams.

1 Introduction

As a result of the world getting more connected and digitized, cyber attacks become increasingly common and pose serious issues for the society. More recently in 2017, a ransomware called WannaCry, which has the capability to lock down the data files using strong encryption, spread around the world targeting public utilities and large corporations (Mohurle and Patil, 2017). Another example is the botnet known as Mirai, which used infected Internet of Things (IoT) devices to disable Internet access for millions of users in the US West Coast (US-CERT, 2016) through large-scale Distributed Denial of Service (DDoS) attacks. The impact levels of these attacks is ranging from simple ransomware on personal laptops (Andronio et al., 2015) to taking over the control of moving cars (Checkoway et al., 2011).

Along with the importance of cybersecurity in today's context, there is an increasing potential for substantial contribution in cybersecurity using natural language processing (NLP) techniques, even though this has not been significantly addressed. We introduced this task as a shared task on SemEval for the first time with the intention of motivating NLP researchers for this critical research area. Even though there exists a large repository of malware related texts online, the sheer volume and diversity of these texts make it difficult for NLP researchers to quickly move to this research field. Another challenge is that most of the data is unannotated. Lim et al. (2017) has introduced a dataset of annotated malware reports for facilitating future NLP work in cybersecurity. In the light of that, we improved Lim's malware dataset to create, to the best of our knowledge, the world's largest publicly available dataset of annotated malware reports. The aim of our annotation is to mark the words and phrases in malware reports that describe the behaviour and capabilities of the malware and assign them to some certain categories.

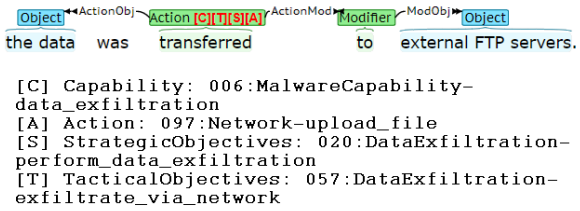

Figure 1: Annotated sentence and sentence fragment from MalwareTextDB. Such annotations provide semantic-level information to the text.

Most of the machine learning efforts in the task of malware detection were based on the system calls. Rieck et al. (2011) and Alazab et al. (2010) proposed models using machine learning techniques for detecting and classifying malware through system calls. Previously, our group has proposed models to predict a malware's signatures based on the text describing the malware (Lim et al., 2017). We defined the same SubTasks mentioned in this paper and used the proposed models as the standard baselines for the shared task. This shared task is hosted on CodaLab[1].

The remainder of this paper is organized as

[1]https://competitions.codalab.org/competitions/17262

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 697–706
New Orleans, Louisiana, June 5–6, 2018.

follows: the information regarding the annotated dataset and its statistics, together with the Sub-Tasks are described in Section 2. Information about the evaluation measures and the baselines is described in Section 3. Different approaches used by the participants are described in Section 4. The evaluation scores of the participating systems and rankings are presented and discussed in Section 5. Finally, the paper concludes with an overall assessment of the task.

2 Data description and Task Definition

In this shared task we expanded upon our previous work, MalwareTextDB (Lim et al., 2017), which was published in ACL 2017. In this paper, we initiated a framework for annotating malware reports and annotated 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). An example of such annotation is shown in Figure 1. During this shared task, we have further annotated 46 APT reports (6,099 sentences), bringing the total number of annotated APT reports to 85 (12,918 sentences). We continue to follow our annotation procedure from the paper, which we will describe in the following subsection.

2.1 Annotation Procedure

This subsection contains the explanation of our annotation procedure.

2.1.1 Data Collection

The APT reports in our dataset are taken from APTnotes, a GitHub repository of publicly-released reports related to APT groups (Blanda, 2016). It provides a constant source of APT reports for annotations with consistent updates. At the time this paper was written, the repository contains 488 reports. We have chosen 85 reports from year 2014 and 2015 for annotation.

We consulted the cybersecurity team from DSO National Laboratories[2] when selecting the APT reports in order to ensure that the preliminary dataset will be relevant for the cybersecurity community.

2.1.2 Preprocessing

The APT reports from APTnotes are in PDF format, hence we used the PDFMiner tool (Shinyama, 2004) to convert the PDF files into plaintext format. We also manually removed the non-sentences, such as the cover page or document header and footer, before the annotation. Hence only complete sentences were considered for subsequent steps.

2.1.3 Annotation

The annotation was performed using the Brat Rapid Annotation Tool (Stenetorp et al., 2012) . In this annotation, our aim is to mark the words and phrases that describe malware behaviors and map them to the relevant *attribute labels*, which are the labels we extracted from the MAEC vocabulary. There are a total of 444 attribute labels, consisting of 211 *ActionName* labels, 20 *Capability* labels, 65 *StrategicObjectives* labels and 148 *TacticalObjectives* labels. The annotation was performed by a team of research assistants and student interns. The annotation work done by the student interns was further reviewed by the research assistants to ensure the quality.

The annotation was performed in three main stages:

2.1.4 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 "*implements*", "*deploy*", and "*transferred*".

Subject This refers to the initiator of the Action such as "*Babar*" and "*they*".

Object This refers to the recipient of the Action such as "*an obfuscation technique*", "*the data*", and "*privilege escalation tools*"; it also refers to word phrases that provide elaboration on the Action such as "*hide certain API names*" and "*external FTP servers*".

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

This stage helps to identify word phrases that are relevant to the MAEC vocabulary.

2.1.5 Stage 2 - Relation Labels

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

SubjAction links an Action with its relevant Subject.

[2] https://www.dso.org.sg/

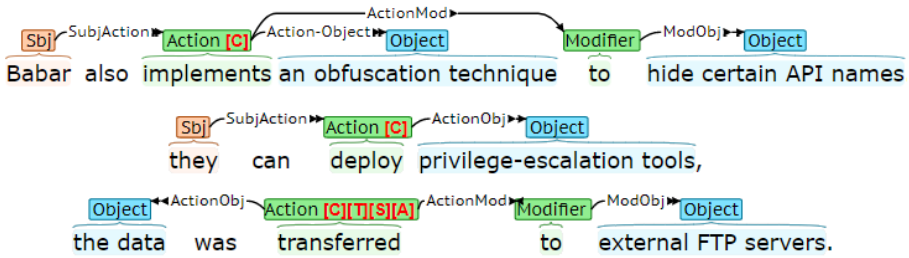

Figure 2: Examples of annotated sentences.

```
The myth of Babar has been around for a while in
the intelligence community.

The drawing of personal educated conclusions is
left as an exercise for the interested reader.
```

Figure 3: Examples of irrelevant sentences.

ActionObj links an Action with its relevant Object.

ActionMod links an Action with its relevant Modifier.

ModObj links a Modifier with the Object that provides elaboration.

This stage indicates the links between the labeled tokens. Such annotations are important in cases where an Action has more than one Subjects, Objects or Modifiers. The illustration on how the relation label links token labels is shown in Figure 2.

2.1.6 Stage 3 - Attribute Labels

The third stage involves annotating the text with the *attribute labels* extracted from the MAEC vocabulary. We decided to annotate the attribute labels onto the Action tokens tagged in the first stage. This is because Action is usually the main indicator of the malware's behaviour. This scheme requires each Action token to be annotated with at least one attribute label.

The attribute labels are categorized into four classes: ActionName, Capability, StrategicObjectives and TacticalObjectives. These classes describe different kinds of actions and capabilities of the malware.

2.1.7 Irrelevant Sentences

The document also contains sentences that provide no indication of malware action or capability. We call these sentences *irrelevant sentences* and do not annotate them. Examples of such sentences can be seen in Figure 3.

2.1.8 Annotation Challenges

We took a portion of the dataset and calculated the agreement for the token labels annotation based on Cohen's kappa (Cohen, 1960). The agreement between annotators is quite low at 0.36, suggesting that this is a difficult task. The main challenges the annotators faced are:

Multiple ways of annotating the same sentence
There might be multiple ways of annotating the same sentence that are equally valid. An example of this is demonstrated in Figure 4. Both annotations highlight the malware ability to *conduct profiling*.

Large amount of annotation labels
There are 444 attribute labels and it is very challenging for the annotators to remember all of them. There are also some attribute labels that are very similar to each other, such as *ActionName 084: load library* and *ActionName 119: map library into process*.

Required special domain knowledge
The annotation requires the annotator to have some cybersecurity domain knowledge. For example, given the phrase *"conduct profiling"*, the annotator must be able to classify it as *Capability 015: probing*.

2.2 SubTask Description

We focus on the evaluations for the following 4 different SubTasks, which are formulated as follows:

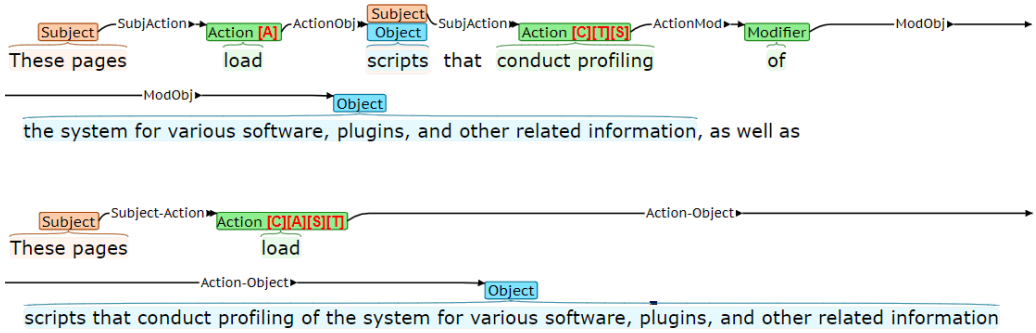

Figure 4: Two different ways of annotating an example sentence.

- SubTask 1: Classify relevant sentences for inferring malware actions and capabilities
- SubTask 2: Predict token labels for a malware related text
- SubTask 3: Predict relation labels between tokens for a malware-related text
- SubTask 4: Predict attribute labels for a malware-related text

In SubTask 1, participants were asked to solve the challenge of sifting out critical sentences from lengthy malware reports and articles. This is modeled as a binary classification task, where each sentence had to be labeled as either relevent or irrelevant. The participants are provided with a list of sentences.

In SubTask2, special tokens in a relevant sentence had to be identified and labeled with one of the following ***token labels*** (examples are taken from Figure 2):

- **Action** This refers to an event, such as "*implements*", "*deploys*", and "*transferred*".
- **Entity** This refers to the initiator of the Action such as "*Babar*" and "*They*" or the recipient of the Action such as "*an obfuscation technique*", "*privilege-escalation tools*", and "*the data*"; it also refers to word phrases that provide elaboration on the Action such as "*hide certain API names*" and "*external FTP servers*".
- **Modifier** This refers to tokens that link to other word phrases that provide elaboration on the Action such as "*to*".

	#Documents	#Sentences
Train	65	9,424
Dev	5	1,213
SubTask1,2 test	5	618
SubTask3 test	5	668
SubTask4 test	5	995
Total	85	12,918

Table 1: Statistics of the MalwareTextDBv2.0.

The formulation is similar to the *token labels* in section 2.1.4. The only difference is the Entity label, which is a combination of the Subject and Object labels. This is to accommodate cases where a single word-phrase is annotated as both the initiator and the recipient of an Action (as seen in Figure 5). This SubTask uses the same list of sentences used in SubTask 1.

In SubTask 3, participants were asked to identify the relation between the tokens. We decided to provide the gold labels for the tokens here due to the low performance of our initial models on SubTask 2. The relation labels are as we described in section 2.1.5.

In SubTask 4, participants were asked to label each action token with the corresponding attribute label(s). In our ACL paper, we did the evaluation on token groups (a set of token labels connected by relation labels) instead of action tokens. However, we decided to evaluate on action tokens in order to encourage the participants to make use of the surrounding context, not limiting themselves just to the tokens in the token group. We also provided the gold labels for the token and relation labels here, following the same consideration as we described for SubTask 3.

In our ACL paper, we also had the experiments

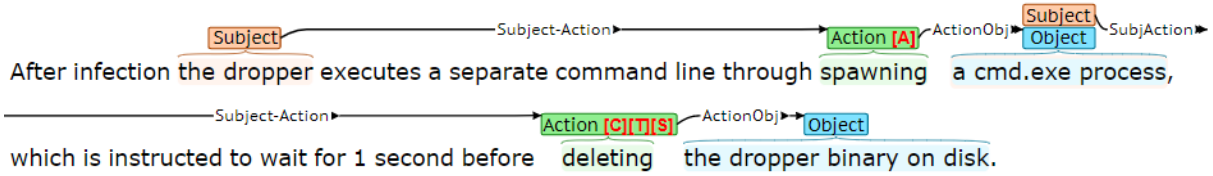

Figure 5: An example of a token (a cmd.exe process) labelled as both Subject and Object. In the first case, it is the recipient of the Action spawning, while in the latter case, it is the initiator of the Action deleting.

	#Relevant	#Irrelevant	#Sentences
Train	2,204	7,220	9,424
Dev	79	1,134	1,213
Test	90	528	618

Table 2: Data distribution of SubTask 1.

on predicting the malware signatures for each document. The list of malware signatures are taken from Cuckoo Sandbox[3]. We excluded such an evaluation at this stage as precise information like malware signatures might be easily obtained from external resources such as malware information websites.

2.3 Data Statistics

We decided to call the dataset we used for this shared task MalwareTextDBv2.0[4], which has twice the number of documents compared to MalwareTextDB. The total statistics are shown in Table 1. The training data for this shared task contains 9,424 sentences, the dev set contains 1,213 sentences, and each test set has various amount of sentences. SubTask 1 and 2 share the same test set, while SubTask 3 and 4 use different test sets. This is because the gold labels from the previous annotation stages are provided for SubTask 3 and 4.

The data distribution for SubTask 1, 2, 3, and 4 can be seen in Table 2, 3, 4, and 5 respectively.

We can see from the distribution of SubTask 1 that the dataset mostly contains *irrelevant* sentences. This shows the importance of SubTask 1 in which the participants filter out the *irrelevant* sentences. Our preliminary result in the ACL paper also shows that removing the *irrelevant* sentences can improve the score for SubTask 2.

From the distribution of SubTask 2, an interesting observation is that the number of *Entity* tokens is roughly double the number of *Action* tokens. This is quite intuitive since *Entity* token refers to either *Subject* or *Object* token and an *Action* usually has one *Subject* and one *Object*.

In the distribution table for SubTask 3, we can observe that the number of *ActionMod* is roughly the same as the number of *ModObj*. This is inline with our observation that a *Modifier* is usually connected to an *Action* and an *Object*.

For SubTask 4, we can see that the *Capability* attribute class has the highest count in the dataset. This is also the category that has the least amount of unique labels (with only 20 different labels). On the other hand, *ActionName* class appears the least in the dataset but has the highest number of unique labels (with 211 different labels).

3 Evaluation Measures and Baselines

Our baseline and evaluation measures follow our ACL paper (Lim et al., 2017). We used F1 score for the evaluation metric for all the SubTasks. Simple baselines were utilized, such as linear support vector machines (SVM) and multinomial Naive Bayes (NB) implementation from the scikit-learn library (Pedregosa et al., 2011). For the conditional random fields (CRF) (Lafferty et al., 2001) models, we used the CRF++ implementation (Kudo, 2005). For the feature extraction, we used spaCy[5] to extract the part-of-speech (POS) features and a C++ implementation (Liang, 2005) of the Brown clustering algorithm.

For SubTask 1, our baseline models are the SVM and NB baselines with bag-of-words features. We also performed some hyper-parameter tuning based on the development set. Other simple baselines, such as random uniform and stratified, are also included as a comparison.

For SubTask 2, we used the CRF baseline with unigrams, bigrams, POS, and Brown clustering features (Brown et al., 1992). CRF model was trained only on the malware related sentences in

[3] https://cuckoosandbox.org/
[4] http://www.statnlp.org/research/resources
[5] https://spacy.io/

	Action	Entity	Modifier	Total
Train	3,202	6,875	2,011	12,088
Dev	122	254	79	455
Test	125	249	79	453

Table 3: Data distribution of SubTask 2.

	#Root	#ActionMod	#ActionObj	#ModObj	#SubjAction	Total
Train	3,378	1,859	2,552	1,760	2,307	11,856
Dev	111	74	110	74	82	451
Test	97	52	86	53	72	360

Table 4: Data distribution of SubTask 3.

	#ActName	#Capability	#StratObj	#TactObj	Total
Train	1,154	2,817	2,206	1,783	7,960
Dev	46	102	77	63	288
Test	34	88	70	64	256

Table 5: Data distribution of SubTask 4.

the training set. The Brown clustering features for words were trained on the 84 additional unannotated APT reports provided with the training materials.

For SubTask 3, a simple rule-based model was utilized. The rules are listed in the Appendix section of our ACL paper. They consist of simple rules, such as connecting a *Modifier* token to the nearest *Action* token with *ActionMod* relation.

Finally, for SubTask 4, we trained SVM and NB model with bag-of-words features. The features for SubTask 4 are extracted from token groups, which are the set of tokens connected via relation labels. In creating the token groups, we only traverse the direction of Action $\rightarrow$ Subj, Action $\rightarrow$ Mod, Action $\rightarrow$ Obj, and Mod $\rightarrow$ Obj. This will prevent multiple *Action* tokens from having the same token group when they are connected to a common *Subject* or *Modifier*.

4 Participants

We received 18 submissions from 9 different teams; 9 submissions to SubTask 1, 8 submissions to SubTask 2, and 1 submission to SubTask 4. Unfortunately, none of the teams submitted to SubTask 3. Participants generally submitted to both SubTask 1 and 2. Here is the list of the participants who submitted a system description paper together with a brief summary of the method they used:

Villani (Loyola et al., 2018) submitted only to SubTask 1. They used word-embeddings initialized using Glove vectors (Pennington et al., 2014) trained on Wikipedia text to represent the tokens. In addition to that, they also used an LSTM to get another token representation from the characters. After that, they trained a binary classifier using Bi-directional Long Short-Term Memory network (BiLSTM) (Graves et al., 2013). They made use of attention mechanism (Luong et al., 2015) to weigh the importance of the tokens.

Flytxt_NTNU (Sikdar et al., 2018) submitted to both SubTask 1 and SubTask 2. They constructed an ensemble of CRF and NB classifiers for SubTask 1. The CRF model used lexical-based and context-based features. The same CRF model was also used to predict the answers for SubTask 2. If the CRF predicts any token labels for the sentence, the sentence is considered relevant in SubTask 1. They did SubTask 2 in 2 steps. First, they detect whether a token is either an *Action*, *Entity*, or *Modifier* (Mention identification). After that, they classify the tokens into one of the three types (Token identification).

DM_NLP (Ma et al., 2018) also submitted to

SubTask 1 and 2, but focuses on SubTask 2 and just used the predicted output labels from SubTask 2 to get the predictions for SubTask 1. They model this task as a sequence labeling task and used a hybrid approach with BiLSTM-CNN-CRF following the method of Ma and Hovy (2016). The CNN layer was used to extract char-level feature representation. They then added other features, such as POS, dependency labels, chunk labels, NER labels, and brown clustering labels as the input to BiLSTM layer. They also made use of word-embeddings, pre-trained using unlabeled data. The output of the BiLSTM layer is then fed into a CRF layer that makes the entity label prediction.

HCCL (Fu et al., 2018) submitted to SubTask 1 and 2. They performed a very similar approach to team DM_NLP using the same BiLSTM-CNN-CRF architecture. The main difference is that they just used POS features, instead of the more complicated linguistic features used by team DM_NLP. They aim to build an end-to-end system that does not require any feature engineering or data preprocessing. Their output for SubTask 1 was also generated from their predictions for SubTask 2.

Digital Operatives (Brew, 2018) participated in SubTask 1 and 2. They utilized a passive aggressive classifier (Crammer et al., 2006), which has similar cost and performance with the linear SVM classifier, for SubTask 1. The features they used include POS, lemma, dependency links, and bigrams. For SubTask 2, they implemented a linear CRF approach using a window of words and POS tags surrounding the focus token as features.

TeamDL (R et al., 2018) made the submissions for SubTask 1 and 2. For SubTask 1, they built a convolutional neural network with original glove embeddings. Their model followed the work of Kim (2014). They also used a CRF for SubTask 2 with features like N-grams (N$\in\{1,2,..6\}$), POS tags, word lemmas, word shape features, etc. In order to tackle unknown malware entities, they used additional set of features taken from malware documents from the web and the training corpus.

UMBC (Padia et al., 2018) participated in SubTask 1, 2 and 4. They are the only team participated in SubTask 4. They used a Multi-Layer Perceptron model for the submission of SubTask 1. After the submission deadline, they have explored other methods for SubTask 1 like LSTM. For SubTask 2, they used a CRF model with features similar to TeamDL. The main difference is that their model had less features compared to TeamDL's model. For SubTask 4, they mainly focused on learning better word embedding features. They developed an Annotation Word Embedding (AWE) model that is capable of incorporating domain-specific knowledge to the embeddings.

5 Results and Discussion

5.1 SubTask 1 Results

Table 6 shows the scores of the submissions to SubTask 1. We also added the precision, recall, and accuracy scores as additional metrics. All 9 participating teams submitted to SubTask 1. This might be because SubTask 1 is the simplest and can be done as a by-product of doing SubTask 2. We can see that by guessing randomly we get an F1 score of 25.06%. However, this does not mean that this SubTask is not challenging as we can see that the scores of top systems are far from perfect. We submitted the NB baseline result in the competition page since it achieved a better performance compared to the SVM baseline in the development data.

Most of the teams used neural network models to tackle this task, which were shown to perform quite well. However, approaches using classifiers such as naive Bayes are still competitive. Team Villani achieved the best F1 score of 57.14% using a neural approach and Flytxt_NTNU reached the second place with an F1 score of 56.87% using an ensemble of naive Bayes and CRF approach.

Some of the teams utilized their results from SubTask 2 to generate predictions for SubTask 1. This method seems to have performed quite well too, with 3 of the top-5 teams using it. Flytxt_NTNU is notable for combining this method with a naive Bayes approach as an ensemble system.

	Prec	Recall	F1	Acc
Our baselines				
Our SVM baseline	**49.55**	62.22	**55.17**	**80.58**
Our NB baseline	38.17	**78.89**	51.45	78.32
Random uniform baseline	16.09	56.67	25.06	50.65
Random stratified baseline	11.45	16.67	13.57	69.09
Participants Outputs				
Villani	47.76	71.11	**57.14**	84.47
Flytxt_NTNU	49.59	66.67	56.87	85.28
DM_NLP	39.43	**76.67**	52.08	79.45
HCCL	**53.57**	50.00	51.72	**86.41**
Digital Operatives	39.31	75.56	51.71	79.45
TeamDL	38.46	72.22	50.19	79.13
NLP_Foundation	36.13	**76.67**	49.11	76.86
UMBC	11.14	43.33	17.73	41.42
NanshanNLP	13.56	17.78	15.38	71.52

Table 6: SubTask 1 results sorted by F1 score, the highest score in each column from the baselines and the participants outputs are marked in **bold.**

5.2 SubTask 2 Results

The scores of the submissions for SubTask 2 are shown in Table 7. This task attracted 8 teams and 4 teams were able to outperform our baseline which is a CRF model with unigrams, bigrams, POS, and Brown clustering features. Though all participants have used the CRF model as the final layer of their models, 3 teams used neural architectures like Bi-LSTM and CNN-BiRNN architectures to generate better embeddings for the features.

Team DM_NLP achieved the best F1 score of 29.23%. In addition, we considered a relaxed scoring scheme where predictions are scored at token level instead of phrase level to give credit to the model when the span for a predicted label intersects with the span for the actual label. The model from team DM_NLP still achieved the highest F1 score of 39.18% under this scoring scheme. Team HCCL showed significant improvement in their scores for the relaxed scoring schemes for their model based on CNN-BiRNN-CRF architecture.

5.3 SubTask 3 Results

The results of our baselines for SubTask 3 can be seen in Table 8. As we mentioned in an earlier section, no participant submitted to this SubTask. From our baselines, we can see that this task cannot be done using random prediction. However, our rule-based method still works well on this new test set.

5.4 SubTask 4 Results

We summarized the results for SubTask 4 in Table 9. The main challenges to this SubTask are the data sparsity and the number of attribute labels. The only participant who submitted to this SubTask is from team UMBC. They used a domain-specific word embedding model trained on APT reports and their automatically generated text annotations to train an SVM classifier.

6 Conclusion and Future Work

In this work, we have presented the results of SemEval 2018 shared task on Semantic Extraction from CybersecUrity REports using Natural Language Processing (SecureNLP). This new SemEval task attracted 9 participating teams with 18 submissions. We have provided a new dataset on annotated malware report and also the evaluation criteria for the 4 SubTasks that we proposed. We also described the methods that the participants used to tackle this shared task. We hope that this shared task can spark the interest of the research community to use NLP techniques for cybersecurity purposes.

The participants have improved the state-of-the-art results for SubTask 1 and 2. They explored many interesting methods to tackle the SubTasks that we proposed. Since the post evaluation phase is still ongoing on the competition website, hopefully other people will be interested in testing their models.

	Normal Scores			Relaxed Scores		
	Prec	**Recall**	**F1**	**Prec**	**Recall**	**F1**
Our baselines						
CRF baseline	24.05	22.30	23.14	31.22	30.80	31.01
Participants Outputs						
DM_NLP	23.35	**39.07**	**29.23**	30.14	**55.98**	**39.18**
Flytxt_NTNU	25.98	29.36	27.56	32.96	40.06	36.17
NLP_Foundation	25.57	29.80	27.52	35.42	42.46	38.62
TeamDL	22.90	28.26	25.30	30.64	43.08	35.81
UMBC	18.19	28.48	22.20	24.42	46.31	31.98
HCCL	7.64	17.88	21.72	**38.39**	36.84	37.60
NanshanNLP	**26.96**	17.44	21.18	34.03	23.84	28.03
Digital Operatives	16.58	14.57	15.51	23.65	26.43	24.96

Table 7: SubTask 2 results sorted by F1 score, the highest score in each column from the baselines and the participants outputs are marked in **bold.**

	Prec	**Recall**	**F1**
Rule-based baseline	**85.60**	**85.83**	**85.71**
Random uniform baseline	3.24	14.17	5.27
Random stratified baseline	3.14	2.22	2.60

Table 8: SubTask 3 baseline results sorted by F1 score.

	Prec	**Recall**	**F1**
Our baselines			
SVM baseline	**40.30**	**31.64**	**35.45**
NB baseline	36.77	32.03	34.24
Participants Outputs			
UMBC	**15.23**	**26.17**	**19.25**

Table 9: SubTask 4 results sorted by F1 score, the highest score in each column from the baselines and the participants outputs are marked in **bold.**

Acknowledgments

We would like to thank all the teams who participated in this shared task. Special mention to Chris Brew and Arpita Roy for their valuable feedback. We also thank the authors of the ACL 2017 paper: Swee Kiat Lim, Aldrian Obaja Muis for doing the initial work, and Chen Hui Ong from DSO for her feedback during the annotation.

References

Mamoun Alazab, Sitalakshmi Venkataraman, and Paul Watters. 2010. Towards understanding malware behaviour by the extraction of api calls. In *Proc. of CTC*, pages 52–59.

Nicoló Andronio, Stefano Zanero, and Federico Maggi. 2015. Heldroid: Dissecting and detecting mobile ransomware. In *Proc. of RAID*, pages 382–404.

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

Chris Brew. 2018. Digital Operatives at SemEval-2018 Task 8: Using dependency features for malware NLP. In *Proc. of SemEval*.

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.*, 18:467–479.

Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, et al. 2011. Comprehensive experimental analyses of automotive attack surfaces. In *Proc. of USENIX Security Symposium*. San Francisco.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Mingming Fu, Xuemin Zhao, and Yonghong Yan. 2018. HCCL at SemEval-2018 Task 8: An End-to-End System for Sequence Labeling from Cybersecurity Reports. In *Proc. of SemEval*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proc. of ICASSP*, pages 6645–6649.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. of EMNLP*.

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 *Proc. of ICML*, pages 282–289.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Swee Kiat Lim, Aldrian Obaja Muis, Wei Lu, and Chen Hui Ong. 2017. MalwareTextDB: A Database for Annotated Malware Articles. In *Proc. of ACL*, volume 1, pages 1557–1567.

Pablo Loyola, Kugamoorthy Gajananan, Yuji Watanabe, and Fumiko Satoh. 2018. Villani at SemEval-2018 Task 8: Semantic Extraction from Cybersecurity Reports using Natural Language Processing (SecureNLP). In *Proc. of SemEval*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*, pages 1412–1421.

Chunping Ma, Huafei Zheng, Pengjun Xie, Chen Li, Linlin Li, and Si Luo. 2018. DM_NLP at SemEval-2018 Task 8: neural sequence labeling with linguistic features. In *Proc. of SemEval*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proc. of ACL*, volume 1, pages 1064–1074.

Savita Mohurle and Manisha Patil. 2017. A brief study of Wannacry Threat: Ransomware Attack 2017. *International Journal of Advanced Research in Computer Science*, 8(5).

Ankur Padia, Arpita Roy, Taneeya Satyapanich, Francis Ferraro, Shimei Pan, Anupam Joshi, and Tim Finin. 2018. UMBC at SemEval-2018 Task 8: Understanding Text about Malware. In *Proc. of SemEval*.

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.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*, pages 1532–1543.

Manikandan R, Krishna Madgula, and Snehanshu Saha. 2018. TeamDL at SemEval-2018 Task 8: Cybersecurity Text Analysis using Convolutional Neural Network and Conditional Random Fields. In *Proc. of SemEval*.

Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. 2011. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 19:639–668.

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

Utpal Kumar Sikdar, Biswanath Barik, and Björn Gambäck. 2018. Flytxt_NTNU at SemEval-2018 Task 8: Identifying and Classifying Malware Text Using Conditional Random Fields and Nave Bayes Classifiers. In *Proc. of SemEval*.

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 *Proc. of EACL*, pages 102–107.

US-CERT. 2016. Heightened ddos threat posed by mirai and other botnets.