Automatic Mapping of Threat Information to Adversary Techniques Using Different Datasets

Otgonpurev Mendsaikhan*
ogo@net.itc.nagoya-u.ac.jp

Hirokazu Hasegawa†
hasegawa@icts.nagoya-u.ac.jp

Yukiko Yamaguchi*
yamaguchi@itc.nagoya-u.ac.jp

Hajime Shimada*
shimada@itc.nagoya-u.ac.jp

*Information Technology Center, Nagoya University, Furocho, Chikusa Ward, Nagoya, Aichi 464-8601, Japan
†Information Security Office, Nagoya University, Furocho, Chikusa Ward, Nagoya, Aichi 464-8601, Japan

***Abstract*—Along with the growth in the usage of software in almost every aspect of human life, the risks associated with software security vulnerabilities also increase. The number of average daily published software vulnerabilities exceeds the human ability to cope with it; hence, various threat models to generalize the threat landscape have been developed. The most prevalent threat model MITRE ATT&CK proved to be a valuable tool for the security analyst to perform cyber threat intelligence, red and blue teaming, and so on. However, the security analyst must prioritize his/her defense by manually mapping the daily published threat information to the adversarial techniques listed in MITRE ATT&CK for his/her day-to-day operation. This paper proposes a method to automatically map the cyber threat information using a multi-label classification approach. We conducted four experiments using three publicly available datasets to train and test seven multi-label classification methods and one pre-trained language model in six evaluation measures. According to our estimate, the LabelPowerset method with Multilayer Perceptron as the base classifier performs best in our experiment.**

Keywords–Multi-label classification; MITRE ATT&CK; Cyber Threat.

I. INTRODUCTION

In our previous work [1] we experimented with various multi-label classification methods to evaluate the performance to automatically map the vector representations of vulnerability description to adversary techniques and tactics. In this work, we extend the scope to map cyber threat information, including the vulnerability description to the same adversarial techniques.

The digital age has presented various opportunities to society along with different challenges. One of the biggest challenges comes with the risk of cyber-attack, data breaches and loss of intellectual property, etc. Software security vulnerability is one of the most significant factors behind these challenges. According to the US National Vulnerability Database (NVD), the total number of reported vulnerabilities as of June 2020 is 146,000 [2], and this number is increasing year by year. In 2019 alone, 20,362 vulnerabilities are reported on NVD, which is a 17.6% increase from 2018 (17,308) and 44.5% increase from 2017 (14,086), and the trend is likely to be upwards [3].

Given this large number of reported vulnerabilities, tracking individual vulnerabilities is nearly impossible. Hence, various approaches and threat models are developed to generalize the threat landscape and ease the burden of a security analyst. One of the most commonly used approaches is a curated knowledge base called MITRE ATT&CK® that enlists adversary behaviors, including their tactics and techniques based on real-world observations. It is a robust framework commonly used as a threat model in adversary emulation, red and blue teaming, and cyber threat intelligence practices [4]. MITRE ATT&CK generalizes the adversary attack techniques and tactics based on the common weaknesses of the systems without mentioning specific products or vulnerabilities.

Even though the MITRE ATT&CK proved to be a helpful framework, the need to identify the specific threat that individual vulnerability poses in the adversarial landscape still exists. In layperson's terms, MITRE ATT&CK is the playbook of steps that house robbers would take to rob a house (e.g., find open access), and software security vulnerability is the weaknesses of the house security (e.g., an unlocked door or broken window). For effective defense, the house owner needs to combine this information, the most common approaches that house robbers use, and the weaknesses of his/her house to better understand the situation and prioritize his/her defenses.

This paper proposes a method to automatically map the cyber threat information, including vulnerability description to adversary techniques and tactics. Since a specific threat or vulnerability can be associated with more than one adversarial technique, we believe developing a multi-label classification model that can infer the adversarial techniques to a given threat would be suitable. Since every vulnerability has associated textual description, we believe using the features of this text; a classic multi-label classification algorithm could produce a result that could be useful for a practical purpose.

Mapping threat information to adversarial tactics and techniques requires a certain level of expertise and domain knowledge. Thus, it may consume a considerable amount of time for the security analyst. In our previous work [1] we experimented

with a limited dataset to demonstrate the method, and the LabelPowerset method with Multilayer Perceptron as base classifier performed best. In this paper, we extend the work by including an additional and more comprehensive dataset and state-of-the-art language model Bidirectional Encoder Representations from Transformers (BERT) and compared the results. By utilizing such existing tools and data, we believe the task of mapping threat information could be automated to spare the human analyst from manual labor. Hence, the paper aims to seek the possibilities to automate the mapping of threat information to adversarial techniques by exploring the existing tools and evaluating them with different datasets.

This work aims to extend the scope of the [1] with additional datasets and methods and explore the possibilities to gain further insight into the current knowledge in the security domain.

The specific contributions of the paper are as follows:

1) To propose an approach to automate the mapping of threat information to adversarial technique.
2) Explore and experiment with various multi-label classification methods and language models to compare the performance.

The remainder of this paper is organized as follows. Section II will review the related research and how this paper differs in its approach. In Section III, we will briefly discuss the background information to be used for this research. In Section IV, the experimental dataset used in this study will be described, and in Section V, the experimental setup of the proposed multi-label classification approach will be discussed. In Section VI, we will show the experimental result and corresponding analysis, and finally, we will conclude by discussing the implications of this result in Section VII.

II. Related work

There have been various studies related to the content of this work. But to the best of our knowledge, using a language model on multi-label classification task of text-based threat information to map to adversarial technique has not been published yet.

A. Threat intelligence and MITRE ATT&CK framework

There has been an attempt to use a multi-label classification approach to map cyber threat intelligence reports to adversarial techniques and tactics. Legoy et al. implemented a tool called rcATT. This system predicts tactics and techniques related to given cyber threat reports and outputs the results using Structured Threat Information eXpression (STIX) format [5]. They focused on extracting MITRE ATT&CK techniques and tactics from cyber threat reports. They used more straightforward approaches for text representation and classification algorithms. In contrast, we focused on mapping the threat information to the same framework, though using more neural and deep learning approaches.

Also, extracting general Tactics, Techniques, and Procedures (TTP) from cyber threat information is gaining some attention. Husari et al. developed a system to automate Cyber Threat Intelligence (CTI) analytics that learns attack patterns [6]. They combined Natural Language Processing (NLP) and Information Retrieval (IR) techniques to extract threat actions from threat reports based on their semantic relationships. Their focus was to extract actionable TTP from threat reports, whereas our focus is to identify the adversarial techniques that can exploit the specific threat.

Apart from extracting an adversarial technique from textual documents, some studies have directly mapped the malware behavior to the MITRE ATT&CK framework. Oosthoek et al. did the automated analysis of 951 unique families of Windows malware and mapped them onto the MITRE ATT&CK framework [7]. They generated a behavior signature of the malware in the sandbox and mapped the signature to the corresponding MITRE ATT&CK technique. Their work focused on mapping the malware based on its behavior to the adversarial techniques defined in the MITRE ATT&CK framework. In contrast, our focus is to map the threat information that the adversary could exploit to the same techniques through its textual representation.

Some researchers have been working on the information provided by the MITRE ATT&CK framework to improve the adversarial predictions. Al-Shaer et al. presented their statistical machine learning analysis on Advanced Persistent Threat (APT) and software attack data reported by MITRE ATT&CK to infer and predict the techniques the adversary might use [8]. They associated adversarial techniques using hierarchical clustering with 95% confidence, providing statistically significant and explainable technique correlations. Our focus is to correlate individual threat information to the adversarial techniques and create a model that can be used to map new threats to the MITRE ATT&CK framework automatically.

There have also been works on classifying the vulnerability information based on its textual description. Huang et al. proposed an automatic vulnerability classification model built on Term Frequency-Inverse Document Frequency (TF-IDF), Information Gain (IG), and deep neural network [9]. They validated their model with CVE descriptions of the National Vulnerability Database and compared them to the performances of SVM, Naive Bayes, and kNN algorithms. We are also attempting to classify the vulnerability information based on its textual description. Still, Huang et al. focused on a multi-class classification that each vulnerability belongs to a specific category. In contrast, we attempt to classify a vulnerability or threat information into multiple adversarial techniques simultaneously.

Hemberg et al. proposed an open-source, relational graphing tool BRON which links MITRE ATT&CK, CWE, CAPEC, and CVE information to gain further insight from available threat intelligence in [10]. Their proposed method correlated those publicly accessible information sources to make it more usable for the analysts and systematically hunt the threat. Our approach is similar by associating the available threat information with a publicly accessible knowledge base of MITRE ATT&CK to assist the human analysts.

Zhou et al. proposed a method to automatically identify Indicators of Compromise (IOC), an essential artifact of cyber threat intelligence using a neural-based sequence labeling model to identify IOCs from cybersecurity reports [11]. The model used attention mechanism and token spelling features to identify low-frequency IOCs from long sentences of the cybersecurity reports. We are also using an attention-based pre-trained model to determine the associated technique in the threat information.

B. Multi-label classification using language models

Medina et al. created and analyzed a text classification dataset from United Nation's Sustainable Development Goals [12]. They experimented with various multi-label classification methods and pre-trained language models, and according to their estimation, the pre-trained language model BERT showed the best performance. Similarly, Sovrano et al. proposed to utilize the Text Similarity Approach using Term Frequency-Inverse Document Frequency (TF-IDF) and pre-trained sentence embedding framework Universal Sentence Encoder (USE) [13]. In this work, we use a similar approach of labeling threat information in text format by embedding it through USE and applying multi-label classification models on top of it. Also, we compared those results with the performance of the BERT language model, which used its own embedding layers instead of USE.

Liu et al. proposed to utilize deep learning to Extreme Multi-label Text Classification (XMTC) task with a family of new Convolutional Neural Network (CNN) models tailored for multi-label classification task [14]. Their approach successfully scaled to the most extensive datasets of 6 benchmark datasets and consistently produced the best or the second-best results on all the datasets. Their focus was the multi-label classification performance on the extreme setting, i.e., too many labels to predict. Pal et al. proposed to correlate the labels in multi-label classification task through attention-based graph neural network [15]. The attention in their model allowed the system to assign different weights to neighbor nodes per label, thus, allowing it to learn the dependencies among labels.

Since the inception of the BERT language model, there have been various attempts to train and utilize it for specific domains, especially in a multi-label classification task. Lee et al. described a patent classification system created by fine-tuning the BERT language model [16]. Their model outperformed the state-of-the-art results in classifying the patent claims in multi-label settings. Similarly, Adhikari et al. presented DocBERT for document classification task [17]. By minimizing cross-entropy and binary cross-entropy loss, DocBERT achieved state-of-the-art results across four popular datasets in single-label and multi-label tasks.

Although there have been various attempts to utilize different language models in a multi-label classification setting, its potential has yet to be fully explored in the cybersecurity domain.

III. Background

Since this study is at the intersection of different fields, the theoretical background knowledge is briefly explained in this section.

A. Vulnerability Modeling

There have been several attempts to standardize the reporting and modeling of software security vulnerabilities or weaknesses and threat landscape in general. In this section, we will discuss a few relevant schemes for this study.

1) Common Vulnerabilities and Exposures: Common Vulnerabilities and Exposures (CVE) is a list of entries, each containing an identification number, description, and at least one public reference for publicly known cybersecurity vulnerabilities [18]. CVE was launched in 1999 and now became the standard naming convention to address interoperability and disparate databases and tools. CVE entries, also called CVEs, CVE IDs, and CVE numbers by the community, provide common reference points so that cybersecurity products and services can speak the same language. CVE is an international cybersecurity community effort, and each new CVE entry is assigned by CVE Numbering Authorities (CNAs).

The majority of the disclosed vulnerabilities are stored at the NVD for centralized vulnerability management purposes. The NVD is the U.S. government repository of standards-based vulnerability management data and is known as the de facto central database of software security vulnerabilities [19]. CVEs stored at NVD proved to be a valuable resource for vulnerability management and overall cybersecurity-related research.

2) Common Attack Pattern Enumeration and Classification: Common Attack Pattern Enumeration and Classification (CAPEC) efforts provide a publicly available catalog of common attack patterns that helps users understand how adversaries exploit weaknesses in applications, and other cyber-enabled capabilities [20]. CAPEC was established by the U.S Department of Homeland Security in 2007 and continuously evolved to include public participation and contributions. CAPEC defines "Attack Patterns" as descriptions of adversaries' common attributes and approaches to exploit known weaknesses in cyber-enabled capabilities. Each attack pattern captures knowledge about how specific parts of an attack are designed and executed and provides guidance on mitigating the attack's effectiveness.

CAPEC differs from the MITRE ATT&CK framework to focus on application security and enumerates exploits against vulnerable systems. In contrast, the MITRE ATT&CK framework focuses on network defense and provides a contextual understanding of malicious behavior. CAPEC is mainly used for application threat modeling and developer training and education, whereas ATT&CK is used to compare network defense capabilities and hunt new threats.

3) MITRE ATT&CK framework: Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) was created at MITRE corporation to systemically categorize adversary behavior in September 2013 [4]. It was initially designed to document and categorize post-compromise adversary TTPs against Microsoft Windows systems and later added other platforms called ATT&CK for Enterprise and publicly released in May 2015. Subsequently, complementary models such as PRE-ATT&CK, ATT&CK for Mobile, and ATT&CK for ICS have been published in 2017 and 2020. The ATT&CK framework consists of the following components:

- **Adversary group**: Known adversaries that are tracked and reported in threat intelligence reports.
- **Tactics**: Tactics represent the adversary's tactical objective: the reason for performing an action.
- **Technique/Sub-Technique**: Techniques represent "how" an adversary achieves its tactic, whereas Sub-technique further breaks down techniques into more specific descriptions of actions to reach the goal.
- **Software**: Software represents an instantiation of a technique or sub-technique at the software level.

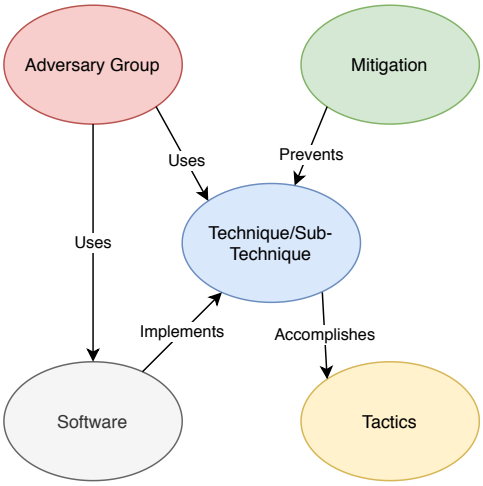

Figure 1. MITRE ATT&CK components and their relationship.

- **Mitigation**: Mitigation represents security concepts and technologies to prevent a technique or sub-technique from being successfully executed.

The relationship between the components is visualized in Figure 1.

The MITRE ATT&CK framework is constantly enriched with techniques and sub-techniques. At the time of writing, there are 266 techniques/sub-techniques of 12 tactics in the MITRE ATT&CK Enterprise model, 174 techniques of 15 tactics in the PRE-ATT&CK and 79 techniques of 13 tactics in ATT&CK for Mobile model.

B. Multi-label classification

Classification is the task of learning to classify the set of examples that are from a set of disjoint labels L, $|L|> 1$. If $|L|= 2$, then the learning problem is called a binary or single-label classification and if $|L|> 2$, it is a multi-class classification. In the case of multi-class classification, the example should correspond to a single class or label. In contrast, in multi-label classification, the examples are associated with a set of labels $Y \subseteq L$ [21]. According to Madjarov et al., the multi-label classification methods could be of the following categories [22].

1) **Algorithm adaptation methods:** The existing machine learning algorithms that are adapted, extended, and customized for multi-label classification problems. The examples include: boosting, k-nearest neighbors, decision trees, and neural networks.
2) **Problem transformation methods:** This method transforms the multi-label classification into one or more single-label classification or regression problems. It is further divided into categories as binary relevance, label power-set, and pair-wise methods.
3) **Ensemble classification:** The ensemble methods are developed on top of existing problem transformation or algorithm adaptation methods. The examples include Random k-label sets (RAkEL) and ensembles of pruned sets (EPS) etc.

C. Evaluation measures of multi-label classification

Since the multi-label classification task is different from the traditional binary classification, the evaluation metrics to measure the method's performance also differ. The multi-label classification measures generally fall into the following categories according to [22].

1) Example based measures
2) Label based measures
3) Ranking based measures

The evaluation measures used in this study are briefly discussed below. In the definitions, y_i denotes the set of true labels for example x_i and $h(x_i)$ denotes the set of predicted labels for the same examples. N is the number of examples, and Q denotes the total number of possible class labels.

1) Subset Accuracy: Subset Accuracy, also called as Exact Match Ratio is the most strict metric, indicating the percentage of samples that have all their labels classified correctly. It can be calculated as shown in (1):

$$Accuracy(h) = \frac{1}{N}\sum_{i=1}^{N} I(h(x_i) = y_i) \tag{1}$$

where $I(true) = 1$ and $I(false) = 0$.

2) Micro averaged F1 score: Since the classification is on multiple labels, the results have to be averaged out. Micro-precision and micro-recall are the measures averaged over all the example/label pairs. In the definitions below TP_j, TN_j denote the number of True Positive and True Negative, FP_j, FN_j denote the number of False Positive and False Negative examples per label λ_j when considered as binary classification.

$$Precision = \frac{\sum_{j=1}^{Q} TP_j}{\sum_{j=1}^{Q} TP_j + \sum_{j=1}^{Q} FP_j} \tag{2}$$

$$Recall = \frac{\sum_{j=1}^{Q} TP_j}{\sum_{j=1}^{Q} TP_j + \sum_{j=1}^{Q} FN_j} \tag{3}$$

The Micro averaged F1 Score is the harmonic mean between micro-precision and micro-recall.

$$F1 = \frac{2 \times microPrecision \times microRecall}{microPrecision + microRecall} \tag{4}$$

3) Macro averaged F1 score: Macro-precision and macro-recall are the measures averaged across all labels and defined as shown in (5) and (6).

$$Precision = \frac{1}{Q}\sum_{j=1}^{Q} \frac{TP_j}{TP_j + FP_j} \tag{5}$$

$$Recall = \frac{1}{Q}\sum_{j=1}^{Q} \frac{TP_j}{TP_j + FN_j} \tag{6}$$

Macro-F1 is the harmonic mean between precision and recall, where the average is calculated per label and then averaged across all labels. If P_j and R_j are the precision and recall for all $\lambda_j \in h(x_i)$ from $\lambda_j \in y_i$ then Macro F1 is defined as in (7):

$$F1 = \frac{1}{Q}\sum_{j=1}^{Q} \frac{2 \times P_j \times R_j}{P_j + R_j} \tag{7}$$

4) Hamming loss: Hamming loss evaluates how many times an example-label pair is misclassified, i.e., a fraction of incorrectly predicted labels.

$$HammingLoss(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{Q} \mid h(x_i)\Delta y_i \mid \quad (8)$$

where Δ stands for the symmetric difference between two sets. The smaller the Hamming loss better the model performance.

5) Ranking loss: Ranking loss evaluates the average fraction of label pairs that are reversely ordered for the particular example.

$$RankingLoss(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{\mid D_i \mid}{\mid y_i \mid\mid \bar{y_i} \mid} \quad (9)$$

where

$D_i = \{(\lambda_m, \lambda_n) \mid f(x_i, \lambda_m) \leq f(x_i, \lambda_n), (\lambda_m, \lambda_n) \in y_i \times \bar{y_i})\}$, while $\bar{y_i})\}$ denotes the complementary set of y in L. The smaller the Ranking loss better the model performance.

6) Micro averaged Receiver Operating Characteristic: Receiver Operating Characteristic (ROC) is a measure used mainly in binary classification to study the output of a classifier. It is a probability curve that plots the True Positive Rate (TPR) against False Positive Rate (FPR) at the various threshold and essentially separates the signal from noise. ROC is usually represented as the Area Under Curve (AUC) graph. In order to use ROC in a multi-label setting, examples are binarized per each label and averaged for aggregated contributions of all classes to compute the Micro averaged metric. Biggest data science online community Kaggle [23] uses Micro averaged ROC score to evaluate the competitions in multi-label problems. Hence, we decided to adopt this measure to evaluate the performances of different models.

D. Language models in Natural Language Processing

Since natural language can not be absolutely formalized in specific rules in a similar way as programming languages, computational linguists approached it by specifying the model of the language from the example texts. Thus, language modeling became a crucial component of Natural Language Processing (NLP) and used various statistical and probabilistic techniques to determine the probability of a given sequence of text to appear in a particular position.

The latest trend in NLP includes the utilization of a pre-trained language model in a transfer learning setting. A pre-trained language model is a language model which has been fed a large amount of unannotated data. As a result, the model learns the usage of various words and general rules of the language. Then the model is transferred to a downstream task where it is fed a smaller, task-specific dataset, through which the model is fine-tuned to perform well on the task. It is basically creating a machine equivalent of a "well-read" human being.

Another significant advancement is transformer-based language models. The attention mechanism is used to selectively attend to some part of the input example, similar to humans focusing on the object's detail. The transformer is a novel architecture introduced in [24] that was considered a breakthrough in NLP. It relies on the self-attention mechanism to compute the input and output representations by handling the dependencies between them.

Current state-of-the-art results in NLP belong to the models such as Generative Pre-trained Transformer-3 (GPT-3) [25], Bidirectional Encoder Representations from Transformers (BERT) [26] and XLNet [27] that were based on the transfer learning and transformer architecture.

This study used two pre-trained and one transformer-based language model, namely Universal Sentence Encoder (USE) and Bidirectional Encoder Representations from Transformers (BERT). The pre-trained USE model is used to embed or convert the text document into vector representations that could be used to apply traditional multi-label classification methods. In contrast, the pre-trained, transformer-based BERT model is used as a standalone model that performs embedding and classification in itself.

IV. Experimental Dataset

The amount and quality of the dataset used are essential to the performance of the trained model. Hence, data has been collected from different sources to represent the diverse nature of the threat information.

A. Data sources

We collected data from three different repositories that are publicly available. The data sources are as follows:

1) ENISA: The European Union Agency for Cybersecurity (ENISA) published a report in December 2019 titled State of Vulnerabilities 2018/2019 [28]. The report aimed to provide an insight into both the opportunities and limitations of the vulnerability ecosystem. They collected in total 27,471 vulnerability information published during 1st January 2018 to 30th September 2019 from various data sources. As part of the analysis of the collected data, the authors mapped the CVEs to the MITRE ATT&CK technique using the common CAPEC information found in both NVD and ATT&CK. The authors generously made available the dataset they have analyzed [29] and we utilized the CVE information mapped to MITRE ATT&CK tactics and techniques for training and testing the multi-label classification model.

The ENISA report dataset represents the vulnerability information in the form of CVE descriptions. It has 8,077 CVEs that are mapped to 52 unique MITRE ATT&CK techniques or, in this instance, labels. The mean length of the example CVE description is 368 characters, and minimum/maximum lengths are 40 and 3,655 characters long. For the purpose of this experiment, this dataset will be denoted as ENISA in short.

2) TRAM: Threat Report ATT&CK Mapping (TRAM) is a tool developed by MITRE to aid the analyst in mapping finished reports to ATT&CK. TRAM uses a Logistic Regression model to predict the mapping of the ATT&CK technique for a given report. MITRE generously released the source code and the corresponding dataset used to train the model [30]. The dataset contains example sentences or phrases representing specific techniques and maps them to one or more techniques.

The TRAM dataset represents the short threat information in the form of sentences or phrases. It has 3,005 example sentences mapped to 188 unique MITRE ATT&CK techniques. The mean length of the example sentence is 84 characters, and minimum/maximum lengths are 15 and 465 characters long.

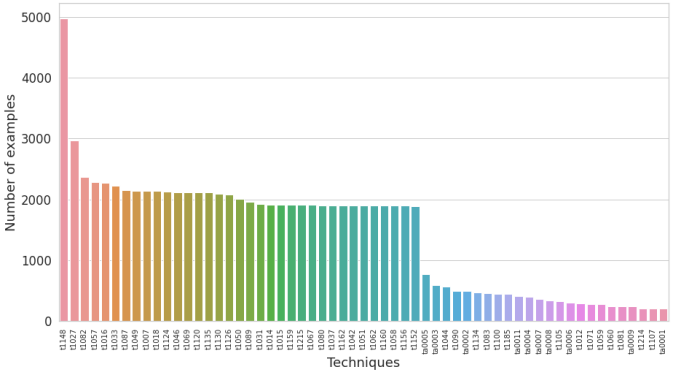

Figure 2. Most common techniques that have at least more than 200 examples associated with it.

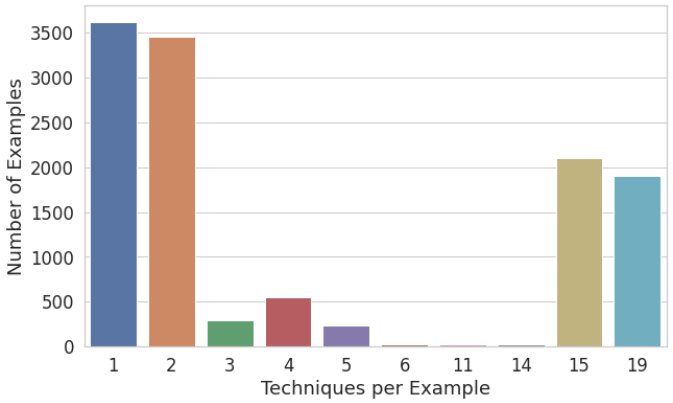

Figure 3. Number of techniques associated per example (top 10).

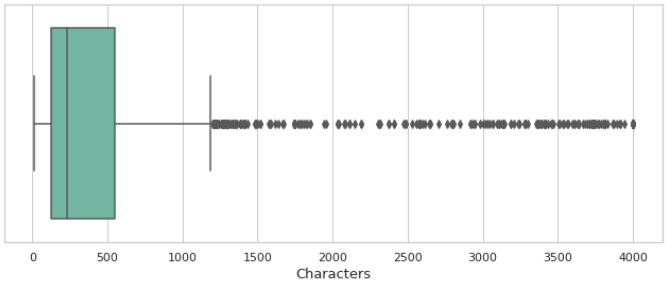

Figure 4. Text length distribution of the dataset.

For the purpose of this experiment, this dataset will be denoted as TRAM in short.

3) rcATT: Legoy et al. implemented a tool called rcATT, a system that predicts tactics and techniques related to given cyber threat reports and described it in [5]. They collected the threat reports referenced in the original MITRE ATT&CK framework per each individual technique to train the tool. They generously made their source code and the parsed threat reports publicly available, and we believe it is the most accurate data we could utilize in this experiment.

The rcATT represents the long descriptive information in the form of threat reports. It has 1,490 example reports mapped to 227 unique MITRE ATT&CK techniques. The mean length of the example report is 19,270 characters, and minimum/maximum lengths are 625 and 457,759 characters long. For the purpose of this experiment, this dataset will be denoted as rcATT in short.

Since the examples of the rcATT dataset are too long for embedding framework, we trimmed the examples to a maximum of 4,000 characters long to ease the computational burden.

B. Dataset analysis

From 3 repositories, we collected a total of 12,572 examples mapped to 239 unique techniques. The characteristics of each data repository and the combined dataset are shown in Table I.

TABLE I. DATASET PROPERTIES.

Dataset	*Examples*	*Techniques*	*Length*	*Type of example*
ENISA	8,077	52	Medium	CVE descriptions
TRAM	3,005	188	Short	Sentences
rcATT	1,490	227	Long	Threat reports
Combined	12,572	239	Mixed	Mixed

The combined dataset is unevenly distributed in terms of the technique association, and there are only 58 techniques with more than 200 examples associated. In Figure 2, the techniques with more than 200 examples associated are shown, and we can see that technique T1148 - HISTCONTROL has 4,978 examples associated with it.

The examples of the combined dataset also have varying characteristics. Figure 3 shows the top 10 instances of label associations. From Figure 3, we can see that majority of the examples, i.e., 7,072 examples out of 12,572, has either 1 or 2 labels associated with them.

The length of the text in the combined dataset also greatly varies. Figure 4 shows this variation in terms of the boxplot graph. Note that the rcATT examples have been trimmed to a maximum of 4,000 characters to ease the computational burden. From Figure 4, we can see that at least 75% (third quartile) of the examples are a less than 550 characters long, and the estimated maximum value would be around 1,200 characters long.

In terms of the content of the examples, a quick analysis using the wordcloud reveals that common words used in the vulnerability descriptions or threat reports are the words that are most repeated in the dataset. Figure 5 shows the wordcloud analysis.

V. Experimental Setup

Using the background information of Section III and the experimental dataset discussed in Section IV, we conducted the experiment to map the threat information to adversarial techniques. Depending upon the experimental setup, we designed the following two separate experiments.

- Converting the text information into its vector representation and apply traditional multi-label classification methods.
- Feed the BERT model with raw text to convert it into its internal representation and performs multi-label classification.

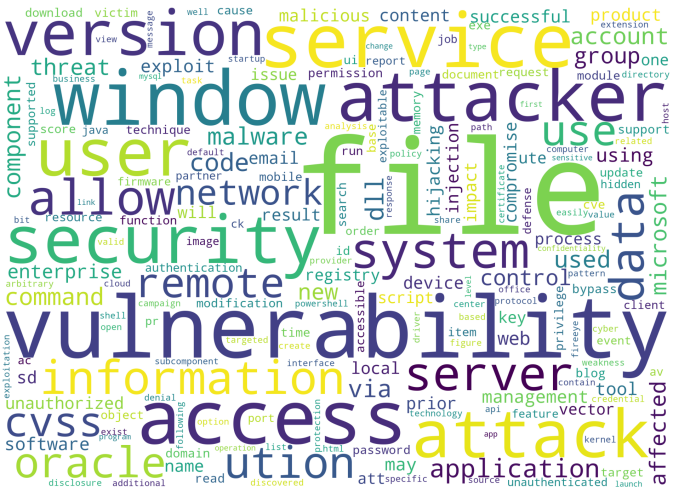

Figure 5. Most common words in the dataset.

In this section, we will discuss the details of these environmental differences.

A. Text representation in multi-label classification

To conduct the multi-label classification, we need to convert the given text into numerical vectors, also known as embeddings. Conventionally, vector embeddings were achieved through shallow algorithms such as Bag of Words (BoW) or Term Frequency-Inverse Document Frequency (TF-IDF). These approaches have been superseded by predictive representation models such as Word2Vec [31], GloVe [32], and so on. The utilization of deep neural networks has been proven to be superior in different fields. Thus, various studies have adopted deep neural models to embed the text into vector space, such as Facebook's InferSent [33] and Universal Sentence Encoder (USE) from Google Research. Perone et al. evaluated different sentence embeddings, and Universal Sentence Encoder outperformed InferSent in terms of semantic relatedness and textual similarity tasks [34]. Therefore, for this research, Universal Sentence Encoder has been utilized to generate the vector embeddings of the text.

The sentence embeddings from USE produce good task performance with little task-specific training data. Thus, we decided to utilize a Deep Averaging Network (DAN)-based USE model introduced in [35] to represent the threat information in numerical vectors so that traditional multi-label classification methods could be applied. The model takes English sentences of variable lengths as input and produces 512 fixed-dimensional vector representations of the sentences as output [36].

B. Multi-label classification model selection

The 512 fixed-dimensional vectors generated by USE are treated as features for the classifier. To determine the suitable model to map the threat information to MITRE ATT&CK techniques, we experimented with 1 Algorithm Adaptation, 3 Problem Transformation, and 1 Ensemble multi-label classification methods using the open-source library scikit-multilearn [37]. The experimented methods are listed below.

- **Multi-label k-Nearest Neighbors (MlkNN)** is the adaptation of the popular k-nearest neighbors (kNN) algorithm to the multi-label classification task and an example of the Algorithm adaptation method. We estimated the number of neighbors k to be most optimal when $k = 3$ where $1 \leq k \leq 30$ when optimized for macro-average F1 measure.
- **LabelPowerset** is a Problem Transformation method that transforms a multi-label problem into a multi-class problem with one multi-class classifier trained on all unique label combinations. It maps each combination to a unique id number and performs multi-class classification using the classifier as a multi-class classifier and combination ids as classes.
- **ClassifierChain** is also a Problem Transformation method. The classifiers are linked along a chain where the i-th classifier deals with the binary relevance problem associated with its label. The feature space of each link in the chain is extended with the 0/1 label associations of all previous links.
- **BinaryRelevance** is the well known one-against-all method. It learns one classifier for each label using all the examples labeled with that label as positive and remaining as negative. And while making a prediction, each binary classifier predicts whether its label is relevant for the given example or not. It is an example of the Problem Transformation method.
- **RAndom k-labELsets multi-label classifier (RAkELd)** is an Ensemble method that divides the label space into equal partitions of size k, trains a LabelPowerset classifier per partition, and predicts by summing the result of all trained classifiers.

The methods mentioned above use traditional classification algorithms for the multi-label classification task. Since neural networks have been proven to be superior in almost every task, we also experimented with more neural approaches as multi-label classification algorithms. Since the multi-label classification task of our experiment does not require the sequential input or memory state of the input, we experimented with a simple Multilayer Perceptron (MLP) neural model to conduct the classification. Szymański et al. included a wrapper in a scikit-multilearn library that allows any Keras or PyTorch compatible backend to be used to solve multi-label problems through problem-transformation methods [38]. We utilized it to conduct the same experiment with neural methods. The following lists the neural methods we used along with their basic parameters.

- **LabelPowerset (neural)** LabelPowerset method with the Multilayer Perceptron as the base classifier. It has two hidden layers, and the softmax function is used for activation.
- **BinaryRelevance (neural)** BinaryRelevance method with the Multilayer Perceptron as base classifier. It has two hidden layers, and the sigmoid function is used for activation.

In Section VI, we will discuss the evaluation results of these different methods.

C. BERT

BERT is designed to learn deep bidirectional representations from an unlabeled text by jointly conditioning both the

left and right contexts in contrast to the previous attempts of predicting a token in a unidirectional (left-to-right, right-to-left) way [26]. BERT achieves bidirectionality by using a pre-training objective called a Masked Language Model (MLM). Before feeding the text sequence to a model, BERT replaces 15% of the words in each training example with a [MASK] token. Then, the task of the model is to predict the original token based on the non-masked tokens. In addition to the MLM task, BERT also employs the Next Sentence Prediction (NSP) task, where the model takes a pair of sentences and then tries to predict whether the second sentence is subsequent to the first. The model is fed 50% of the subsequent sentences during the training while the other 50% of sentences are ordered randomly.

To successfully train with MLM and NSP tasks, BERT preprocesses the input text according to the following steps.

1) A special [CLS] token is placed at the beginning of the first sentence, and a [SEP] token is placed right before the second.
2) Token embeddings, where dense embeddings for each token, including [CLS] and [SEP], will be learned.
3) Sentence embeddings indicate which tokens belong to which sentence. This process is similar to token embeddings; however, vocabulary size is limited to only two words.
4) Positional embeddings are borrowed from the original transformer paper [24]. Since the transformer is not recurrent, it needs to learn the sequential information with the help of positional embeddings.

BERT has been pre-trained on BookCorpus (800M words) and English Wikipedia (2,500M words) with the goal of minimizing the combined loss of MLM and NSP tasks. Fine-tuning BERT on a classification task is relatively straightforward by simply adding a linear layer on top of the transformer output for the first [CLS] token. Applying BERT to downstream tasks involves fine-tuning for the task-specific data.

For the purpose of this experiment, we used BERT-Base uncased model, which contains 30,522 words. Since BERT uses the WordPiece tokenization method, out of vocabulary words are split into subwords, and a group of subwords represents the word. We adapted BERT's sequence classification class for the multi-label classification task using binary cross-entropy with a logits loss function. The final model consisted of the Input Embedding layer, 12 BERT attention layer, and output layer as per the number of labels to be predicted. We trained the model for ten epochs with a batch size of 8 and sequence length of 512. The learning rate was kept to 3e-5, as recommended in the original paper.

VI. Experimental result

Using the experimental dataset discussed in Section IV, we conducted four separate experiments, with each dataset and the combined dataset of 12,572 examples. All the models have been trained with randomly selected 66.6% examples of the dataset and tested on the remaining 33.3% examples. Since the dataset is limited in size, we evaluated the models with a 10-fold cross-validation method. The only exception is in the case of the BERT model due to its computationally costly nature. In this section, we will review the results of each experiment and analyze them.

A. ENISA dataset

The evaluation results for the ENISA dataset is listed in Table II.

From the results listed in Table II, we could see that LabelPowerset using the neural model as a base classifier, has the best scores in 3 of the 6 measures, and BERT has the best results in the remaining 3 measures. The nature of the ENISA dataset is that it consists of quite uniform medium-length text (CVE descriptions) and makes up more than 64% of the total dataset. Hence, the results of this experiment may have the most influence over the final outcome of the Combined dataset.

B. TRAM dataset

The evaluation results for the TRAM dataset is listed in Table III.

From the results listed in Table III, we could see that LabelPowerset using the neural model as a base classifier, has the best scores in every measure except Ranking loss, in which the neural BinaryRelevance model has a lead. Unfortunately, we were not able to produce any meaningful result with the BERT model using this dataset. Since the nature of the TRAM dataset is short sentences and phrases, we assume that BERT may not work very well with the short text. We reduced the sequence length of the BERT model from 512 characters to 64 characters, but it did not yield any improvements. Since the dataset has fewer examples associated with more labels than the ENISA dataset, the resulting performances also seem to have deteriorated.

C. rcATT dataset

The evaluation results for the rcATT dataset is listed in Table IV.

From the results listed in Table IV, we could see that no single model has superior performance in all the measures. Since the nature of the rcATT dataset is long threat reports, it could potentially improve the model performance. However, it is clear that all the multi-label classification models perform poorly compared to the previous two datasets in every except one measure. This poor performance could be because the size of the training and test data is small compared to the other datasets (999 and 491 examples, respectively), and fewer examples are associated with more techniques (1,490 examples with 227 techniques).

D. Combined dataset

The evaluation results for the Combined dataset is listed in Table V.

From the results listed in Table V, we could see that LabelPowerset using the neural model as a base classifier, has 3 out of 6 best results and BERT has 2, and neural BinaryRelevance has the best score in Hamming loss only. Since the combined dataset is the combination of 3 datasets, the performances of the models fall within a range of the best and worst results. We believe the results of the combined dataset are reasonable performance and considered it as the final experimental result.

TABLE II. ENISA DATASET EXPERIMENTAL RESULT.

		Micro Average			Macro Average					
Algorithm	*Accuracy score*	*Prec.*	*Rec.*	*F1 Score*	*Prec.*	*Rec.*	*F1 Score*	*Hamm. loss*	*Rank. loss*	Micro ROC
MlkNN	0.6079	0.7310	0.6193	0.6700	0.6225	0.4960	0.5403	0.1089	0.3617	0.7848
LabelPowerset	0.6028	0.7233	0.5640	0.6320	0.5756	0.5283	0.5014	0.1159	0.3754	0.7588
ClassifierChain	0.4993	0.5997	0.6201	0.6094	0.4219	0.4678	0.4224	0.1423	0.4330	0.7647
BinaryRelevance	0.3757	0.5937	0.6508	0.6205	0.4727	0.6096	0.4875	0.1424	0.3283	0.7765
RakelD	0.4235	0.6374	0.6133	0.6243	0.5070	0.5801	0.4937	0.1310	0.3446	0.7688
LabelPowerset (neural)	**0.7363**	0.7491	0.7464	0.7470	0.6681	0.6213	**0.6316**	0.0907	**0.2503**	0.8455
BinaryRelevance (neural)	0.5526	0.7821	0.7062	0.7415	0.6921	0.5873	0.6236	0.0882	0.2910	0.8313
BERT	0.7201	0.8269	0.7384	**0.7801**	0.6646	0.5930	0.6247	**0.0742**	0.2643	**0.8524**

TABLE III. TRAM DATASET EXPERIMENTAL RESULT.

		Micro Average			Macro Average					
Algorithm	*Accuracy score*	*Prec.*	*Rec.*	*F1 Score*	*Prec.*	*Rec.*	*F1 Score*	*Hamm. loss*	*Rank. loss*	Micro ROC
MlkNN	0.5095	0.7194	0.5196	0.6031	0.2437	0.2045	0.2117	0.0037	0.4753	0.7593
LabelPowerset	0.6370	0.6470	0.6370	0.6420	0.2439	0.2544	0.2371	0.0039	0.3579	0.8175
ClassifierChain	0.0519	0.0746	0.1054	0.0873	0.0149	0.0486	0.0185	0.0120	0.8975	0.5491
BinaryRelevance	0.2612	0.3481	0.6983	0.4642	0.1787	0.2840	0.1974	0.0088	0.3048	0.8456
RakelD	0.2639	0.3533	0.6976	0.4686	0.1804	0.2840	0.1989	0.0086	0.3053	0.8453
LabelPowerset (neural)	**0.6902**	0.7019	0.6938	**0.6978**	0.2856	0.2897	**0.2763**	0.0033	**0.3013**	**0.8461**
BinaryRelevance (neural)	0.5271	0.7853	0.5803	0.6671	0.2764	0.2311	0.2420	**0.0031**	0.4156	0.7897
BERT	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0054	1.0000	0.5000

TABLE IV. RCATT DATASET EXPERIMENTAL RESULT.

		Micro Average			Macro Average					
Algorithm	*Accuracy score*	*Prec.*	*Rec.*	*F1 Score*	*Prec.*	*Rec.*	*F1 Score*	*Hamm. loss*	*Rank. loss*	Micro ROC
MlkNN	0.0174	0.3920	0.2730	**0.3153**	0.0724	0.0572	0.0559	0.0349	0.7509	0.6303
LabelPowerset	0.0121	0.2394	0.0822	0.1170	0.0302	0.0147	0.0147	0.0364	0.8898	0.5371
ClassifierChain	0.0007	0.1381	0.5541	0.2192	0.0432	0.2077	0.0604	0.1173	0.6798	**0.7224**
BinaryRelevance	0.0000	0.0988	0.5894	0.1678	0.0574	0.2796	**0.0826**	0.1724	0.6946	0.7108
RakelD	0.0000	0.1232	0.4974	0.1947	0.0598	0.1926	0.0780	0.1197	**0.6777**	0.6940
LabelPowerset (neural)	**0.0302**	0.2824	0.2195	0.2434	0.0918	0.0680	0.0687	0.0402	0.7848	0.6015
BinaryRelevance (neural)	0.0054	0.4737	0.2296	0.3058	0.0717	0.0431	0.0484	0.0316	0.7896	0.6109
BERT	0.0000	0.6042	0.0679	0.1222	0.0163	0.0063	0.0086	**0.0187**	0.9757	0.5335

TABLE V. COMBINED DATASET EXPERIMENTAL RESULT.

		Micro Average			Macro Average					
Algorithm	*Accuracy score*	*Prec.*	*Rec.*	*F1 Score*	*Prec.*	*Rec.*	*F1 Score*	*Hamm. loss*	*Rank. loss*	Micro ROC
MlkNN	0.5460	0.7367	0.6104	0.6675	0.3183	0.2235	0.2501	0.0182	0.4044	0.8018
LabelPowerset	0.5468	0.6884	0.5162	0.5898	0.2975	0.1926	0.2124	0.0214	0.4255	0.7545
ClassifierChain	0.0230	0.0598	0.3319	0.1013	0.0439	0.2396	0.0444	0.1763	0.7215	0.5854
BinaryRelevance	0.1293	0.2448	0.7424	0.3681	0.1099	0.5133	0.1537	0.0762	0.3464	0.8359
RakelD	0.1258	0.3067	0.7139	0.4289	0.1313	0.4172	0.1738	0.0568	0.3356	0.8321
LabelPowerset (neural)	**0.6631**	0.7093	0.6948	0.7018	0.3353	0.2524	0.2701	0.0177	**0.3123**	**0.8430**
BinaryRelevance (neural)	0.4850	0.7776	0.6754	0.7229	0.3424	0.2480	**0.2735**	0.0155	0.3548	0.8347
BERT	0.4464	0.8420	0.6817	**0.7534**	0.1676	0.1363	0.1453	**0.0130**	0.5274	0.8389

E. Result analysis

The best results of the experiments measured by Micro ROC are listed in Table VI along with the models which have shown the best performance in at least two measures. The results of Table VI and the dataset properties listed in Table I show that the performances of the multi-label classification deteriorate when the number of labels, in this instance, techniques to be predicted, increases and could depend highly on the number of examples used. Also, the length of the text matters in which shorter text has a higher probability of correct technique to be predicted.

We could conclude from the listed results that the LabelPowerset method with Multilayer Perceptron as base classifier performs best in most cases, and BERT comes in second. Since running a BERT model requires a computationally expensive environment, the neural LabelPowerset model would be an ideal choice for predicting the adversarial techniques in cyber threat information.

TABLE VI. COMBINED DATASET EXPERIMENTAL RESULT.

Dataset	Best Micro ROC	Best model
ENISA	0.8524	LabelPowerset (neural), BERT
TRAM	0.8461	LabelPowerset (neural)
rcATT	0.7224	None
Combined	0.8430	LabelPowerset (neural), BERT

VII. CONCLUSION

This paper proposed an approach to automatically map the cyber threat information to adversary techniques in the cybersecurity context. We converted various threat information into vector space and experimented with different multi-label classification methods, as well as a state-of-the-art language model to identify the most suitable method to map the threat information into MITRE ATT&CK adversarial techniques. We used 12,572 examples from 3 open datasets to conduct 4 independent experiments to train and test 7 multi-label classification methods and 1 pre-trained language model in 6

evaluation measures.

According to the results of the experiments converting threat information into vector space using a pre-trained USE model and applying the neural LabelPowerset method to conduct multi-label classification to predict adversarial techniques yielded the best results. State-of-the-art pre-trained language model BERT performed second in which BERT's internal embedding represents the threat information, and the final layer is fine-tuned for multi-label purposes.

Based on the experiment results, we believe that by embedding the given threat information using Universal Sentence Encoder and applying the LabelPowerset method with Multilayer Perceptron as a base classifier, we could effectively predict the adversarial techniques.

However, we acknowledge the limitations of the paper, including the lack of explainability in the models so that the labels it predicted could not be entirely analyzed with current settings. Also, there has been some progress with the NLP and language models since the experiments were first designed, thus not reflected in this work. Those limitations are to be addressed in future work.

REFERENCES

[1] O. Mendsaikhan, H. Hasegawa, Y. Yamaguchi, and H. Shimada, "Automatic mapping of vulnerability information to adversary techniques," in The Fourteenth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE). IARIA, 2020, pp. 53–59.

[2] National Vulnerability Database, 2020 (accessed June 1, 2020). [Online]. Available: https://nvd.nist.gov/general/nvd-dashboard

[3] D. Bekerman and S. Yerushalmi, The State of Vulnerabilities in 2019, 2020 (accessed June 10, 2020). [Online]. Available: https://www.imperva.com/blog/the-state-of-vulnerabilities-in-2019/

[4] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK®: Design and philosophy," The MITRE Corporation, Tech Report, 2020.

[5] V. Legoy, M. Caselli, C. Seifert, and A. Peter, Automated Retrieval of ATT&CK Tactics and Techniques for Cyber Threat Reports, 2020, vol. abs/2004.14322.

[6] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources," in Proceedings of the 33rd Annual Computer Security Applications Conference, ser. ACSAC 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 103–115. [Online]. Available: https://doi.org/10.1145/3134600.3134646

[7] K. Oosthoek and C. Doerr, SoK: ATT&CK Techniques and Trends in Windows Malware, 2019, pp. 406–425.

[8] R. Al-Shaer, J. M. Spring, and E. Christou, Learning the Associations of MITRE ATT&CK Adversarial Techniques, 2020.

[9] G. Huang, Y. Li, Q. Wang, J. Ren, Y. Cheng, and X. Zhao, "Automatic classification method for software vulnerability based on deep neural network," IEEE Access, vol. 7, 2019, pp. 28 291–28 298.

[10] E. Hemberg, J. Kelly, M. Shlapentokh-Rothman, B. Reinstadler, K. Xu, N. Rutar, and U.-M. O'Reilly, "Linking threat tactics, techniques, and patterns with defensive weaknesses, vulnerabilities and affected platform configurations for cyber hunting," 2021.

[11] S. Zhou, Z. Long, L. Tan, and H. Guo, "Automatic identification of indicators of compromise using neural-based sequence labelling," 2018.

[12] S. R. Medina, "Multi-label text classification with transfer learning for policy documents," 2019.

[13] F. Sovrano, M. Palmirani, and F. Vitali, "Deep learning based multi-label text classification of unga resolutions," in Proceedings of the 13th International Conference on Theory and Practice of Electronic Governance, ser. ICEGOV 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 686–695. [Online]. Available: https://doi.org/10.1145/3428502.3428604

[14] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang, "Deep learning for extreme multi-label text classification," in Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 115–124. [Online]. Available: https://doi.org/10.1145/3077136.3080834

[15] A. Pal, M. Selvakumar, and M. Sankarasubbu, "Multi-label text classification using attention-based graph neural network," in ICAART, 2020.

[16] J. Lee and J. Hsiang, "Patentbert: Patent classification with fine-tuning a pre-trained BERT model," CoRR, vol. abs/1906.02124, 2019. [Online]. Available: http://arxiv.org/abs/1906.02124

[17] A. Adhikari, A. Ram, R. Tang, and J. Lin, "Docbert: BERT for document classification," CoRR, vol. abs/1904.08398, 2019. [Online]. Available: http://arxiv.org/abs/1904.08398

[18] Common Vulnerabilities and Exposures, 2020 (accessed June 25, 2020). [Online]. Available: https://cve.mitre.org/

[19] National Vulnerability Database, 2020 (accessed June 22, 2020). [Online]. Available: http://nvd.nist.org

[20] About CAPEC, 2020 (accessed June 25, 2020). [Online]. Available: https://capec.mitre.org/about/index.html

[21] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," IJDWM, vol. 3, 2007, pp. 1–13.

[22] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Deroski, "An extensive experimental comparison of methods for multi-label learning," Pattern Recogn., vol. 45, no. 9, 2012, p. 3084–3104. [Online]. Available: https://doi.org/10.1016/j.patcog.2012.03.004

[23] Kaggle: Your Machine Learning and Data Science Community, 2021 (accessed June 20, 2021). [Online]. Available: https://www.kaggle.com/

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," CoRR, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[25] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[26] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," CoRR, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[27] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," 2020.

[28] V. Katos, S. Rostami, P. Bellonias, N. Davies, A. Kleszcz, S. Faily, A. Spyros, A. Papanikolaou, C. Ilioudis, and K. Rantos, "State of vulnerabilities 2018/2019," European Union Agency for Cybersecurity (ENISA), Tech Report, 2019.

[29] Threat Report ATT&CK Mapping (TRAM) is a tool to aid analyst in mapping finished reports to ATT&CK, 2020 (accessed May 1, 2021). [Online]. Available: https://github.com/mitre-attack/tram

[30] TRAM, 2019 (accessed May 10, 2020). [Online]. Available: https://github.com/enisaeu/vuln-report/

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in 1st International Conference on Learning Representations, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[32] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[33] InferSent, 2018 (accessed January 10, 2020). [Online]. Available: https://github.com/facebookresearch/InferSent

[34] C. S. Perone, R. Silveira, and T. S. Paula, "Evaluation of sentence embeddings in downstream and linguistic probing tasks," CoRR, vol. abs/1806.06259, 2018. [Online]. Available: http://arxiv.org/abs/1806.06259

[35] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil, "Universal sentence encoder," CoRR, vol. abs/1803.11175, 2018. [Online]. Available: http://arxiv.org/abs/1803.11175

[36] universal-sentence-encoder, 2018 (accessed January 11, 2020). [Online]. Available: https://tfhub.dev/google/universal-sentence-encoder/2

[37] Multi-Label Classification in Python, 2018 (accessed May 15, 2020). [Online]. Available: http://scikit.ml/

[38] P. Szymański and T. Kajdanowicz, "A scikit-based Python environment for performing multi-label classification," ArXiv e-prints, 2017.