CPE-Identifier: Automated CPE Identification and CVE Summaries Annotation with Deep Learning and NLP

Wanyu Hu[a] and Vrizlynn L. L. Thing[b]
ST Engineering, Singapore

Keywords: National Vulnerability Database (NVD), Common Vulnerabilities and Exposures (CVE), Common Platform Enumeration (CPE), Named Entity Recognition (NER), Natural Language Processing (NLP).

Abstract: With the drastic increase in the number of new vulnerabilities in the National Vulnerability Database (NVD) every year, the workload for NVD analysts to associate the Common Platform Enumeration (CPE) with the Common Vulnerabilities and Exposures (CVE) summaries becomes increasingly laborious and slow. The delay causes organisations, which depend on NVD for vulnerability management and security measurement, to be more vulnerable to zero-day attacks. Thus, it is essential to come out with a technique and tool to extract the CPEs in the CVE summaries accurately and quickly. In this work, we propose the CPE-Identifier system, an automated CPE annotating and extracting system, from the CVE summaries. The system can be used as a tool to identify CPE entities from new CVE text inputs. Moreover, we also automate the data generating and labeling processes using deep learning models. Due to the complexity of the CVE texts, new technical terminologies appear frequently. To identify novel words in future CVE texts, we apply Natural Language Processing (NLP) Named Entity Recognition (NER), to identify new technical jargons in the text. Our proposed model achieves an F1 score of 95.48%, an accuracy score of 99.13%, a precision of 94.83%, and a recall of 96.14%. We show that it outperforms prior works on automated CVE-CPE labeling by more than 9% on all metrics.

1 INTRODUCTION

As the world advances into the digital era, there is a drastic increase in the number of softwares. These modern softwares depend heavily on open-source libraries and packages. Many organizations spend a lot of resources to monitor the new vulnerabilities in those open source softwares. One of the most well-known open-source vulnerability databases is the National Vulnerability Database (NVD), maintained by the National Institute of Standards and Technology (NIST).

There are thousands of new vulnerabilities disclosed each year. In 2021, an average of fifty [redscan] new vulnerabilities were disclosed daily, and a total of 20169 [NVDCVE] new vulnerabilities were published in 2021. The number of new vulnerabilities is increasing rapidly. More than 8000 new vulnerabilities were published in the first quarter of 2022, around a 25 percent [Comparitech] increase from the same period in 2021. Many organizations' vulnerabilities databases heavily depend on the NVD notifications. Therefore, timely analysis of the new open-source vulnerabilities is crucial for an organization.

However, according to a survey, once a new vulnerability is publicly disclosed by the Common Vulnerabilities and Exposures (CVE) system and published on the NVD, the metadata used in vulnerability management, such as Common Platform Enumeration (CPE) applicability statements, took 35 days on average [Wareus]. This hinders organizations from getting important vulnerability information in time and leaves the organizations vulnerable to zero-day attacks.

Therefore, this paper proposed an automated CPE annotating system, called the CPE-Identifier system, which automates the CPE extraction process in the CVE summaries and, in addition to the improved F1, accuracy, precision, and recall scores, it increases the potential time performance improvements in the analysis. This system serves as a tool to identify CPE entities from new CVE text accurately, quickly and conveniently. Named entity recognition (NER) is used in the system to automatically identify CPEs in CVE de-

[a] https://orcid.org/0000-0002-7600-808X
[b] https://orcid.org/0000-0003-4424-8596

Hu, W. and Thing, V.
CPE-Identifier: Automated CPE Identification and CVE Summaries Annotation with Deep Learning and NLP.
DOI: 10.5220/0012403500003648
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 10th International Conference on Information Systems Security and Privacy (ICISSP 2024)*, pages 67-78
ISBN: 978-989-758-683-5; ISSN: 2184-4356
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

scriptions. This can be useful for security researchers who want to quickly identify the systems and packages affected by a particular CVE. Moreover, while various past machine learning methods were experimented, they require a large amount of annotated training data, which is not publicly available for every CVE summary since the year 1999. Past papers used various methods to annotate the training data. Bridges et al. [Bridges] used ”Database Matching, Heuristic Rules and Relevant Terms Gazetteer” to label the CPEs in the sentence, Wareus E. et al. [Wareus] mapped the words in the sentence to the existing CPEs in the available CPE-list. However, these methods are often costly and not a viable solution. Therefore, we also proposed an idea to automate this data generating and labeling process using deep learning models. Our work outperformed previous methods for automatic CPE labeling in all aspects of performance metrics. Our best model obtains a F1 score of 95.48%, Accuracy score of 99.13%, Precision score of 94.83%, Recall score of 96.14%.

The structure of this paper is as follow: Section 2 introduces the relevant works while discussing their advantages and challenges. Section 3 explores the research methodologies, which include the system design and our proposed five NLP models for different purposes in our system. Section 4 elaborates on all the datasets and data engineering processes, including the automated data annotation, generation, and cleaning. Section 5 explains the three SoTA models: BERT, XLNet and GPT-2. Section 6 describes the implementation of the research, including the Data preprocessing, models training and design of the Graphical User Interface (GUI). Section 7 analyses the experiment result. Section 8 concludes the paper by identifying the best model for the CPE-Identifier system and its usability. We also propose directions for future work.

2 LITERATURE REVIEW

There are past research leveraging NLP methods for CVE summaries. Most of these works are related to Common Attack Pattern Enumeration and Classification (CAPEC), Common Vulnerability Scoring System (CVSS) score, and Common Weakness Enumeration (CWE). K Kanakogi et al. [Kanakogi] have suggested automatically associating the CVE-ID with the CAPEC-ID using the Term Frequency-Inverse Document Frequency (TF-IDF) technique [TFID]. Rostami, S. et al. [Rostami] built 12 Multiple Layer Perceptron (MLP) models to predict and map those missing ATT&CK tactic categories to those CAPEC without tactic types.

Saba Janamian et al. [UC] have introduced the VulnerWatch to predict the eight criteria for calculating the CVSS score. Similarly, Shahid, M. R. et al [Shahid] suggested using multiple BERT classifiers for each metric composing the CVSS vector. While Ohuabunwa, B. C. et al. [Ohuabunwa] proposed Categorical Boosting (CatBoost) to predict the CVSS version 3 scores, Evangelista, J. [Evangelista] compared the predictive performance of Latent Dirichlet Allocation (LDA) and BERT in predicting CVSS version 2 scores. Finally, to convert the metrics from CVSS version 2 to version 3, Nowak, M. et al. [Nowak] compared the three models – Naive Bayes classier (NB), k-nearest Neighbors Algorithm (kNN), and Kernel Support Vector Machine (KSVM). Each model is trained on CVE summaries and predicts the CVSS scores.

When dealing with vulnerabilities, Das, S. S. et al. [Das] maps CVEs to CWEs using BERT model to understand the impact and mitigate the vulnerabilities. Stephan Neuhaus et al. [Neuhaus] experimented with a similar idea using the unsupervised LDA model to classify CVE texts into vulnerabilities. Yang H. et al. [Yang] used multiple ML models to analyze the number of CVE References containing PoC code and how these References affect the CVE exploitability. Similarly, Yin J. et al. [Yin] performed similar research using a Bert-based model with a Pooling layer. Bozorgi M. et al. [Bozorgi] also researched the vulnerability's likelihood and exploitability using the Support vector machines (SVMs) method.

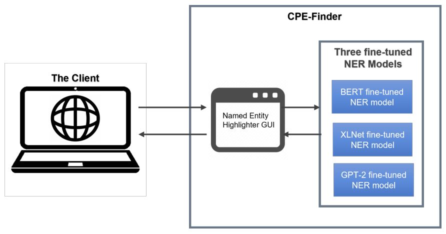
To extract the CPE features from the NVD CVE summaries. Huff P. et al. [Huff] proposed a recommendation system using Random Forest and Fuzzy Matching to automatically match the company's hardware and software inventory in the CVE summary to CPEs' Vendor and Product name. However, the size of the data is insufficient. Jia Y. et al. [Jia] proposed to use multiple datasets and the Stanford NER to extract cybersecurity-related entities and train an extractor to extract cybersecurity-related CPE entities and build a knowledge base. However, the F1 score is only below 80%. Georgescu et al. [Georgescu] uses IBM Watson Knowledge Studio services to train the CVE IoT dataset to enhance the diagnosing and detect possible vulnerabilities within IoT systems. To extract vulnerability types from CVE summaries, Bridges et al. [Bridges] proposed a summarization tool called CVErizer to summarise CVE texts and extract vulnerability types. The author used Auto-Labelling rules to link CVE word with CPE entities through CWE classification. However, the Auto-Labelling rules, like the Heuristic Rules, cannot provide an accurate result. Lastly, Wareus E. et al. [Wareus] have suggested using the Bi-directional Long Short-Term Memory (Bi-LSTM) and a Conditional Random Forest (CRF) layer to identify the CPE labels, such as vendor and product names, in the CVE summaries. The labels are then combined to reconstruct the CPE names. This method is like the approach we use in our paper. However, Wareus E. et al. [Wareus] only considers the top 80% of the most common CPE products and vendors names to ”avoid CPEs with very few mentions”. Those CPEs with very few mentions are eliminated. The disadvantage of this method is that it discourages the prediction of those rare CPE entities names in new CVE summaries, thus compromises the generality of the model.

Despite their efforts to extract CPEs from CVE summaries, achieving a high performance result is challenging. Many of these works fail to produce an accurate and precise CPE identification result. The model used in these researches are not the SoTA NLP model and their performance metrics are not good. Their F1 scores are below 90%. Also, prior researches are not able to find and annotate enough corpora when training the model. Those proposed data annotation rules, such as the Database matching, Relevant Terms Gazetteer, and Heuristic Rules, are not able to provide an accurate annotation result. Hence, we draw insights from the challenges and advantages of the related works and design a system that addresses those disadvantages. Our proposed approach uses the state-of-the-art (SoTA) natural language processing (NLP) technique to automatically find CPE identities from new CVE summaries. We generated and annotated our training data using the SoTA Transformers models with two publicly available cybersecurity NER datasets. We also trained and evaluated the three SoTA Deep Learning NER models – BERT, XLNet, and GPT-2, to perform CPEs entity identification, and found that they outperformed previous methods [Wareus] for automatic CPE labelling. The best model is selected to construct the CPE-Identifier system.

3 RESEARCH METHODOLOGY

3.1 System Design

The system architecture of the CPE-Identifier is a client and server model. The client interacts with the CPE-Identifier using the Named Entity Highlighter GUI. After a string of CVE text is provided to the GUI, the text is encoded, supplied to the model, and then decoded to the prediction result. The Named Entity Highlighter GUI highlights the named entities and displays them to the client through the GUI. The Figure 1 illustrates the proposed CPE-Identifier design and its component.


CPE-Finder
The Client
Named Entity Highlighter GUI
Three fine-tuned NER Models
BERT fine-tuned NER model
XLNet fine-tuned NER model
GPT-2 fine-tuned NER model
Figure 1: CPE-Identifier Design.

3.2 Proposed Approach

The Figure 2 presents the design of the models training process. The research consists of three stages:

1. Data Pre-processing stage: This stage includes data collection, cleaning, annotation, augmentation, and merging. The CVE dataset is downloaded and undergone feature engineering processes. The result is used to generate new dataset and annotate raw dataset in this stage.
2. Build and train models stage: This stage consists of constructing one Data Annotator model, one Data Augmentator model, as well as three SoTA NER models – BERT, XLNet, and GPT-2, which are trained using the pre-processed data.
3. Analysis and build Graphical User Interface (GUI) stage: The predictive performance of each model are compared. A GUI is constructed to visualize the prediction result of the best model.

4 DATASET AND LABELS

To fine-tune a model capable of understanding cybersecurity domain-specific jargon, we need a large set of labeled data specifically for the Named Entity Recognition (NER) task that are tailored to the cybersecurity domain. Moreover, high-quality annotated entities closely related to the CVE-CPE field are also important for identifying new entities in this domain. In a recent paper [Aghaei], Aghaei et al. also suggested that the absence of publicly available domain-specific NER labeled dataset has caused their research on training a NER model in cybersecurity a ”challenging task”. The author employed a relatively small sized dataset called MalwareTextDB [MalwareTextDB]. However, after investigating the MalwareTextDB dataset, it consists of four NER tags, which are not related to our research target. Furthermore,

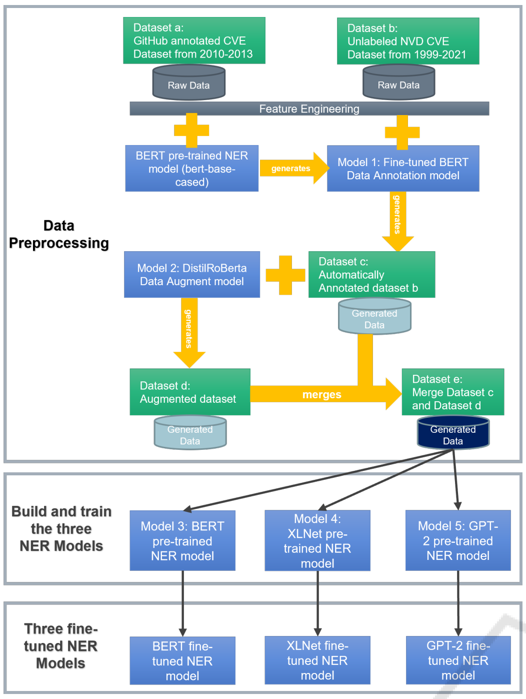

Dataset a: GitHub annotated CVE Dataset from 2010-2013
Dataset b: Unlabeled NVD CVE Dataset from 1999-2021
Raw Data
Raw Data
Feature Engineering
BERT pre-trained NER model (bert-base-cased)
generates
Model 1: Fine-tuned BERT Data Annotation model
generates
Data Preprocessing
Model 2: DistilRoBerta Data Augment model
Dataset c: Automatically Annotated dataset b
Generated Data
generates
Dataset d: Augmented dataset
merges
Dataset e: Merge Dataset c and Dataset d
Generated Data
Generated Data
Build and train the three NER Models
Model 3: BERT pre-trained NER model
Model 4: XLNet pre-trained NER model
Model 5: GPT-2 pre-trained NER model
Three fine-tuned NER Models
BERT fine-tuned NER model
XLNet fine-tuned NER model
GPT-2 fine-tuned NER model

Figure 2: Models Training Process Design.

the size of the dataset is relatively small for training a deep learning model, which may cause the model to overfit. Therefore, it is crucial to collect high-quality publicly available training datasets exclusively compiled for the cybersecurity CVE-CPE domain.

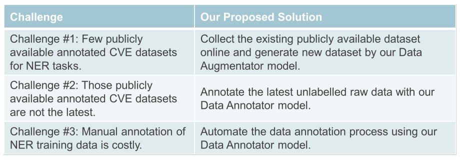
However, large and annotated CVE-CPE datasets are not publicly available. A few possible reasons for this challenge include:

- **Challenge 1:** There are few publicly available annotated CVE datasets for NER tasks.

  Compared to other natural language processing tasks such as text classification or relation extraction, Named Entity Recognition is not common on CVE datasets. There is only a few past research works focused on the NER task of the CVE dataset, making it even more challenging to find a large amount of annotated NER corpus. To the best of our knowledge, there is only one such publicly available dataset from the Stucco project [GitHubdata].

- **Challenge 2:** Those publicly available annotated CVE datasets are not the latest.

  The CVE summary and its corresponding CPE metadata are updated every day. It is difficult for the cybersecurity analysts to keep an eye on every new CVE released and maintain an updated database. Moreover, the CVE NER dataset is not used by any specific application or computing environment but only by developers. Therefore, there is no complete set of the latest annotated CVE NER dataset.



Challenge	Our Proposed Solution
Challenge #1: Few publicly available annotated CVE datasets for NER tasks.	Collect the existing publicly available dataset online and generate new dataset by our Data Augmentator model.
Challenge #2: Those publicly available annotated CVE datasets are not the latest.	Annotate the latest unlabelled raw data with our Data Annotator model.
Challenge #3: Manual annotation of NER training data is costly.	Automate the data annotation process using our Data Annotator model.

Figure 3: Challenges and Proposed Solutions.

- **Challenge 3:** Manual annotation of NER training data is costly.

  The manual annotation of a high-quality CVE dataset for Named Entity Recognition in the cybersecurity domain is time-consuming, requiring experts who are professionals in cybersecurity.

Training a model using a small amount of data or a highly skewed dataset will cause the model to over-fit. The model "sees" a small number of tokens repeatedly and learns to recognize those training data correctly but is not capable of generalizing to the novel inputs in the test dataset. The training error of the model becomes extremely small at the expense of high test-error.

The figure 3 shows our proposed solution to each challenge of the data availability:

4.1 CPE Data

CPE is a standardized identification system for Information and Communications Technology (ICT) products. It provide a more consistent and accurate way to identify products across different security databases, and make it easier for security researchers to collect and share information about vulnerabilities and products. Each CPE is linked to a particular CVE ID, which includes the following features [Andrew]: *cpe_version, part, vendor, product, version, update, edition, and language*. The CPE texts are available at the NVD official website [NVDdata].

4.2 Padding/Trimming the Sentences

Since each of the sentences in the dataset is of a different length, we have padded and trimmed the input sentences to keep all the sentences to be the same length.

Out of 361088 sentences, the length ranges from 3 to 449 words. There are 93.96% of the sentences are shorter than 128 words, therefore we have set the maximum length of the sentences to 128 tokens. Those sentences shorter than 128 are padded with [PAD] tokens. Those sentences longer than 128 are truncated.

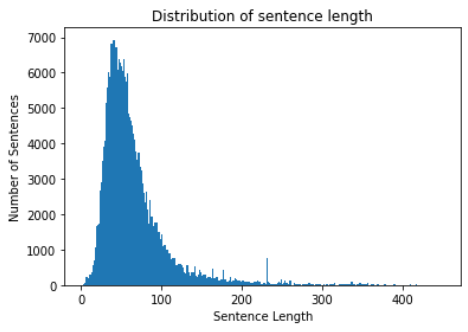


Figure 4: Sentence Length Distribution.

4.3 Tagging Schemes

The training corpora are required to be tagged before performing training. The tagging scheme, BIO (Beginning, Inside, Outside), is used in this paper.

5 MODELS

We trained and used five pre-trained models and one Masked-Language Model (MLM) to construct our CPE-Identifier system. These six models are:

1. **BERT Pre-Trained NER Model**
   - This is a pre-trained bert-base-cased model downloaded from huggingface.co. It is used to fine-tune our own Data Annotator model from scratch.
2. **Model 1: Finetuned BERT Data Annotator Model**
   - This is a fine-tuned bert-base-cased model from training annotated cybersecurity NER texts from GitHub (Dataset a). This model is used to annotate the named entities in NVD CVE Dataset from 1999-2021 (Dataset B) and produces an Automatically Annotated dataset (Dataset C).
3. **Model 2: DistilRoBerta Data Augmentator Model**
   - This is a distilroberta-base Masked Language model. This model generates the Augmented NER dataset (Dataset D) by randomly replacing word tokens in sentences from the Automatically Annotated dataset (Dataset C).
4. **Model 3: BERT NER Model**
   - This is a pre-trained bert-base-cased model.
   - We use the Final Training Corpora (Dataset E) to train this model and analyze its performance against Model 4 and Model 5.
5. **Model 4: XLNet NER Model**
   - This is a pre-trained xlnet-base-cased model.
   - We use Final Training Corpora (Dataset E) to train this model and analyze its performance against Model 3 and Model 5.
6. **Model 5: GPT-2 NER Model**
   - This is a pre-trained gpt2 model.
   - We use Final Training Corpora (Dataset E) to train this model and analyze its performance against Model 3 and Model 4.

5.1 BERT Model

BERT is an AutoEncoder (AE) language model and uses only the Encoder portion of the Transformer architecture. We construed our Data Augmentator model by utilizing the BERT MLM property to generate new sentences. We masked seven tokens in each sentence and asked the model to predict the masked word.

The advantage of the AE language model is that it can see the context in both the forward and backward directions. However, it uses the [MASK] token in the pre-training, which is absent from the real data during fine-tuning and results in the pretrain-finetune discrepancy. It also assumes the masked token is independent of unmasked tokens, which is not always the case. For example, “Microsoft Word” where “Word” is dependent on “Microsoft”. If “Word” is masked and predicted, Bert may output “Microsoft Office”. Therefore, we also tested the XLNet model, which addresses this issue.

5.2 XLNet Model

The XLNet model is a generalized Auto-Regressive (AR) Permutation Language model [Liang]. It is a bi-directional Language model. The AR model can predict the next word using the context word from either forward or backward, but not both directions.

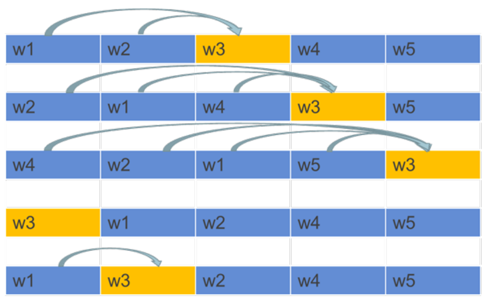


Figure 5: XLNet Permutation Language Modelling (PLM).

XLNet solved this issue by introducing Permutation Language Modelling (PLM), PLM allows for predicting the upcoming words by permutating a sequence and gathering information from all positions on both sides of the token. In the figure, the target token w3 is located at the i-th position, and the i-1 tokens are the context words for predicting w3. There are five patterns in 120 permutations. The words are permutated at each iteration; therefore, every word will be used as the context word. This way, it can avoid the disadvantages of the MASK method in the AE language model.

5.3 GPT-2 Model

GPT-2 is Auto-Regressive (AR) model as well [Liang]. It is good at generative NLP tasks because of its forward direction property. However, it can only use forward or backward contexts, which means it can't use forward and backward context simultaneously. Therefore, GPT-2 is a Uni-directional Language Model, unlike the other two NLP models. Different from the Bert model, GPT-2 uses the Decoder portion of the Transformers architecture.

6 IMPLEMENTATION

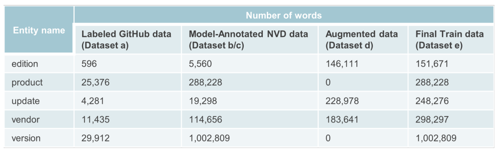
The implementation of the research follows three stages: Data pre-processing, models training, and design and development of the Graphical User Interface (GUI).

6.1 Data Pre-Processing Stage

This stage aims to create a set of clean, annotated, and structured NER corpora, which will be used to train the three SoTA models.

We identified five datasets to be used for our work. They are:

1. Dataset A: GitHub annotated CVE Dataset from Jan 2010 to March 2013 publicly available online [GitHubdata].
2. Dataset B: Raw CVE Dataset from 1999-2021 publicly available online [NVDdata].
3. Dataset C: Self-annotated NER dataset using our proposed Data Annotator model.
4. Dataset D: Self-augmented NER dataset using our proposed distilroberta-base Masked Language model.
5. Dataset E: Final Training Corpora generated by combining both Dataset C and Dataset D, which is used to train the three SoTA models.

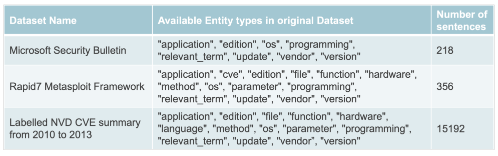


Entity name	Number of words			
	Labeled GitHub data (Dataset a)	Model-Annotated NVD data (Dataset b/c)	Augmented data (Dataset d)	Final Train data (Dataset e)
edition	596	5,560	146,111	151,671
product	25,376	288,228	0	288,228
update	4,281	19,298	228,978	248,276
vendor	11,435	114,656	183,641	298,297
version	29,912	1,002,809	0	1,002,809

Figure 6: The details of all Datasets used and generated.

6.1.1 Data Collection

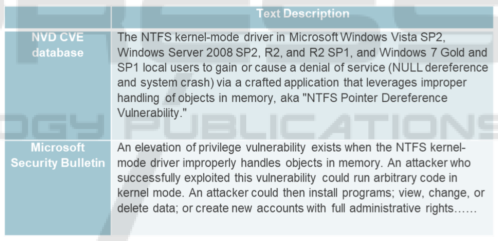
Dataset A is downloaded from GitHub [GitHubdata]. It consists of CVE summaries from three sources: Microsoft Security Bulletin, Metasploit Framework database, and the NVD CVE database. The entity types of each dataset is shown in figure 7:



Dataset Name	Available Entity types in original Dataset	Number of sentences
Microsoft Security Bulletin	"application", "edition", "os", "programming", "relevant_term", "update", "vendor", "version"	218
Rapid7 Metasploit Framework	"application", "cve", "edition", "file", "function", "hardware", "method", "os", "parameter", "programming", "relevant_term", "update", "vendor", "version"	356
Labelled NVD CVE summary from 2010 to 2013	"application", "edition", "file", "function", "hardware", "language", "method", "os", "parameter", "programming", "relevant_term", "update", "vendor", "version"	15192

Figure 7: Dataset A.

While the same CVE-ID describes the same kind of vulnerabilities, the descriptions in each source are different. For example, in Figure 8 the descriptions of CVE-ID: CVE-2013-1293 are different from various sources:



	Text Description
NVD CVE database	The NTFS kernel-mode driver in Microsoft Windows Vista SP2, Windows Server 2008 SP2, R2, and R2 SP1, and Windows 7 Gold and SP1 local users to gain or cause a denial of service (NULL dereference and system crash) via a crafted application that leverages improper handling of objects in memory, aka "NTFS Pointer Dereference Vulnerability."
Microsoft Security Bulletin	An elevation of privilege vulnerability exists when the NTFS kernel-mode driver improperly handles objects in memory. An attacker who successfully exploited this vulnerability could run arbitrary code in kernel mode. An attacker could then install programs; view, change, or delete data; or create new accounts with full administrative rights……

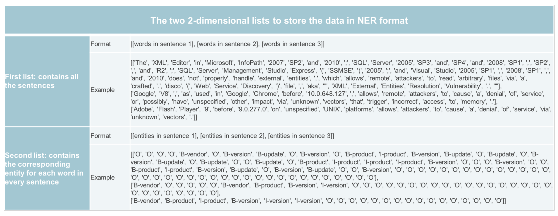
Figure 8: CVE text Description from different sources.

Dataset B contains the raw CVE summaries and CPEs from 1999 to 2021 that is downloaded from [NVDdata].

6.1.2 Data Cleaning

Our proposed CPE-Identifier system only considers the top five popular named entities in the CPEs – "edition", "product", "update", "vendor", and "version", because the remaining named entities are not always available in the CPE identifier strings dataset. These five types are enough for the cybersecurity analyst to quickly narrow down to the vulnerability details. Therefore, We have replaced the unrelated entities in dataset A with the Out-of-Word label ("O"). After investigating the dataset, we have noted that the "application", "hardware" and "os" labels correspond to the "product" label in the CPE identifier strings, so we have renamed the three entities to "product".

We then constructed two 2-dimensional lists to store the data in NER format. The two sample 2-dimensional lists to store the data in NER format is illustrated 9:



The two 2-dimensional lists to store the data in NER format		
First list: contains all the sentences	Format	[[words in sentence 1], [words in sentence 2], [words in sentence 3]]
	Example	[['The', 'XML', 'Editor', 'in', 'Microsoft', 'InfoPath', '2007', 'SP2', 'and', '2010', ';', 'SQL', 'Server', '2005', 'SP3', 'and', 'SP4', 'and', '2008', 'SP1', ',', 'SP2', ',', 'and', 'R2', ';', 'SQL', 'Server', 'Management', 'Studio', 'Express', '(', 'SSMSE', ')', '2005', ';', 'and', 'Visual', 'Studio', '2005', 'SP1', ',', '2008', 'SP1', ',', 'and', '2010', 'does', 'not', 'properly', 'handle', 'external', 'entities', ',', 'which', 'allows', 'remote', 'attackers', 'to', 'read', 'arbitrary', 'files', 'via', 'a', 'crafted', '.', 'disco', '(', 'Web', 'Service', 'Discovery', ')', 'file', ',', 'aka', '"', 'XML', 'External', 'Entities', 'Resolution', 'Vulnerability', '.', '"'], ['Google', 'V8', ',', 'as', 'used', 'in', 'Google', 'Chrome', 'before', '10.0.648.127', ',', 'allows', 'remote', 'attackers', 'to', 'cause', 'a', 'denial', 'of', 'service', 'or', 'possibly', 'have', 'unspecified', 'other', 'impact', 'via', 'unknown', 'vectors', 'that', 'trigger', 'incorrect', 'access', 'to', 'memory', '.'], ['Adobe', 'Flash', 'Player', '9', 'before', '9.0.277.0', 'on', 'unspecified', 'UNIX', 'platforms', 'allows', 'attackers', 'to', 'cause', 'a', 'denial', 'of', 'service', 'via', 'unknown', 'vectors', '.']]
Second list: contains the corresponding entity for each word in every sentence	Format	[[entities in sentence 1], [entities in sentence 2], [entities in sentence 3]]
	Example	[['O', 'O', 'O', 'O', 'B-vendor', 'O', 'B-version', 'B-update', 'O', 'B-version', 'O', 'B-product', 'I-product', 'B-version', 'B-update', 'O', 'B-update', 'O', 'B-version', 'B-update', 'O', 'B-update', 'O', 'O', 'B-update', 'O', 'B-product', 'I-product', 'I-product', 'I-product', 'B-version', 'O', 'O', 'O', 'B-version', 'O', 'O', 'B-product', 'I-product', 'B-version', 'B-update', 'O', 'B-version', 'B-update', 'O', 'O', 'B-version', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O'], ['B-vendor', 'O', 'O', 'O', 'O', 'O', 'B-vendor', 'B-product', 'B-version', 'I-version', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O'], ['B-vendor', 'B-product', 'I-product', 'B-version', 'I-version', 'I-version', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']]

Figure 9: Sample data format after Data Cleaning step.

For dataset B, we combined the dataset from 1999 to 2021 into one text file.

6.1.3 Data Annotation

This step allows an NLP model to automate the named entity labeling process. Manually annotating the 169092 unlabelled CVE summaries is time-consuming and costly and requires expert knowledge in both cybersecurity and natural language processing. Therefore, we built a deep learning model to annotate the unlabelled CVE summaries automatically. It dramatically improves efficiency and accuracy of the data labeling process. The model accuracy is 98.73%, and its F1 score is 95.64%. The time taken to annotate 169092 sentences is only 2 hours. The high F1 score as well as the accuracy score proves that our fine-tuned BERT Data Annotator model is capable of tagging the sentences with NER tags accurately. Furthermore, we have also randomly picked sentences from the output dataset (dataset C) and manually examined its accuracy. The steps to generate and annotate the dataset C is described.

1. Constructed the Data Annotator model by finetuning the bert-base-cased NER model from scratch using dataset A. The resulting model (model 1) learns the domain knowledge from cybersecurity texts and will be used as a tool in the next step.
2. We used model 1 as an annotating tool to identify the named entities in the unlabelled NVD CVE dataset (dataset B) from the Data Cleaning stage.
3. The resulting labeled data (datast C) contains 169092 sentences. The distribution of the entities is shown in the figure 6.

6.1.4 Data Augmentation

The next step in the Data Pre-processing stage is Data Augmentation. The augmented dataset was generated using the NVD CVE summary with BERT Masked Language Model. Data augmentation helps to improve the data skewness problem by increasing the data size of the specific named entity. For example, in our automatically annotated NVD CVE summary dataset, there are only 19298 tokens that are tagged as the “update” entity, while 288228 words are classified as the “product” entity. This highly imbalanced data causes the resulting model to be less generalized on the “edition” tag.

We build a BERT Masked Language Model and augment new sentences by randomly replacing seven words with their synonyms based on the dictionary. There are very few “edition”, “vendor” and “update” named entities in our automatically annotated NVD CVE summary dataset – 5560 words for “edition, 114656 for “vendor” and 19298 for “update”; we plan to generate more sentences containing these three entities.

We extracted sentences containing “edition”, “vendor” and “update” only. There are a total of 13288 sentences. We created 192380 new sentences using our augmentation model, the Data Augmentor. The distribution of the augmented data is shown in the Figure 6.

6.1.5 Data Merging

The last step in the Data Pre-processing stage is to merge the Automatically Annotated NER dataset (dataset C) and the Augmented NER dataset (dataset D). The distribution of the entities is shown in the Figure 6. The Final Training Corpora contains 361472 sentences and will be used to train the three SoTA NER models.

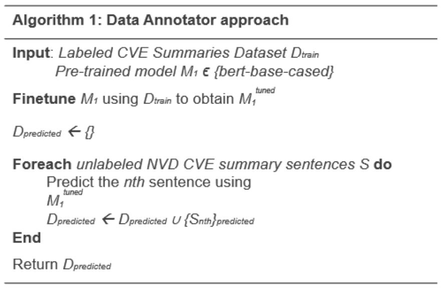
6.2 Models Training Stage

The models are trained in the Ubuntu Kernel, using GeForce RTX 3070 Graphics Card. CUDA version 11.3.

6.2.1 Model 1: Finetuned BERT Data Annotator Model

The fine-tuned BERT Data Annotator model is fine-tuned from the bert-base-cased pre-trained model for 5 epochs, using the annotated CVE summaries (dataset a) from GitHub. The purpose of the Data Annotator model is to serve as a tool to annotate the unlabelled NVD CVE summaries (Dataset B) with the named entities.

The algorithm of the approach is shown in Fig. 10:



```
Algorithm 1: Data Annotator approach

Input: Labeled CVE Summaries Dataset $D_{train}$
         Pre-trained model $M_1 \in$ {bert-base-cased}
Finetune $M_1$ using $D_{train}$ to obtain $M_1^{tuned}$

$D_{predicted} \leftarrow \{\}$

Foreach unlabeled NVD CVE summary sentences S do
    Predict the nth sentence using
    $M_1^{tuned}$
    $D_{predicted} \leftarrow D_{predicted} \cup \{S_{nth}\}_{predicted}$
End
Return $D_{predicted}$
```

Figure 10: Data Annotator Approach.

The F1 score of the model is 95.64%, and the accuracy score is 98.72%. The F1 score is plotted in Fig. 11. This approach improves efficiency and saves the cost of the manual labeling process.

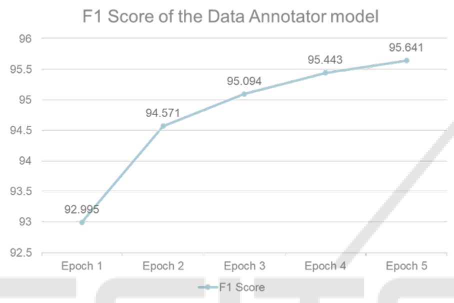
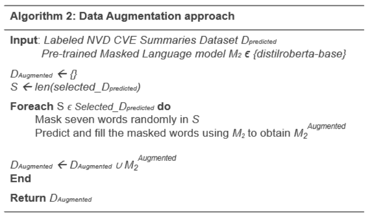


Figure 11: F1 score of Data Annotator model.

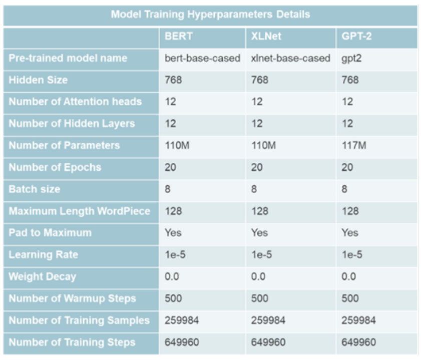
6.2.2 Model 2: DistilRoBerta Data Augmentor Model

The DistilRoBerta Data Augment model is built on top of the DistilRoBerta [Distilroberta] pre-trained model. The purpose of the Data Augmentor model is to serve as a tool to generate new labeled NER corpora. The imbalanced data causes skewness issue in the model, which degrades the model to describe typical cases as it must deal with rare cases on extreme values. The model will predict better on entities with more training words than those with fewer training words [C.]. This approach solves the data skewness issue by generating new sentences with specific entities. The algorithm of the approach is shown in 12:



```
Algorithm 2: Data Augmentation approach

Input: Labeled NVD CVE Summaries Dataset $D_{predicted}$
         Pre-trained Masked Language model $M_2 \in$ {distilroberta-base}

$D_{Augmented} \leftarrow \{\}$
$S \leftarrow len(selected\_D_{predicted})$

Foreach $S \in Selected\_D_{predicted}$ do
    Mask seven words randomly in S
    Predict and fill the masked words using $M_2$ to obtain $M_2^{Augmented}$

$D_{Augmented} \leftarrow D_{Augmented} \cup M_2^{Augmented}$
End
Return $D_{Augmented}$
```

Figure 12: Data Augmentation Approach.



Model Training Hyperparameters Details			
	BERT	XLNet	GPT-2
Pre-trained model name	bert-base-cased	xlnet-base-cased	gpt2
Hidden Size	768	768	768
Number of Attention heads	12	12	12
Number of Hidden Layers	12	12	12
Number of Parameters	110M	110M	117M
Number of Epochs	20	20	20
Batch size	8	8	8
Maximum Length WordPiece	128	128	128
Pad to Maximum	Yes	Yes	Yes
Learning Rate	1e-5	1e-5	1e-5
Weight Decay	0.0	0.0	0.0
Number of Warmup Steps	500	500	500
Number of Training Samples	259984	259984	259984
Number of Training Steps	649960	649960	649960

Figure 13: Model Training Hyperparameters Details.

6.2.3 Model 3, 4, 5: BERT, XLNet and GPT-2 NER models

The three SoTA NER models are trained for 20 epochs each. The figure 13 shows the hyperparameters used when training the models:

We have tested the BERT, XLNet and GPT-2 models and compared their performances and determined the most suitable architecture for the NER task on CVE summaries.

6.3 Analysis and build Graphical User Interface (GUI)

We have also designed a GUI to visualize the prediction result. The interface of the CPE-Identifier is hosted on a web using the Streamlit Python package. The Interface allows the analysts to input the new CVE texts into the "Enter Text Here" text box and select a specific model, or all the three fine-tuned models to retrieve the annotated result. This feature allows the analysts to choose the best model that fits their CVE text.

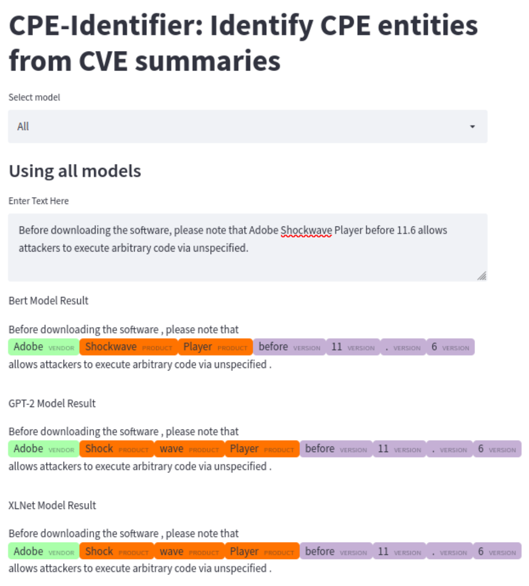
The prediction result is highlighted in different colors, where each color indicates a named entity type. The colors and their corresponding entities are shown in Figure 14

Vendor: Green
Product: Red
Edition: Blue
Update: Yellow
Version: Purple

Figure 14: CPE Entities Colors.

The figure 15 shows the input text box and the corresponding prediction result, where entities are highlighted in the corresponding colors:

In the example, the word "Adobe" is highlighted in green, indicating it belongs to the "VENDOR" category, The word "Shockwave Player" is in orange, which means that it is part of "PRODUCT" category. One interesting finding is that, instead of one



CPE-Identifier: Identify CPE entities from CVE summaries

Select model

All

Using all models

Enter Text Here

Before downloading the software, please note that Adobe Shockwave Player before 11.6 allows attackers to execute arbitrary code via unspecified.

Bert Model Result

Before downloading the software , please note that
Adobe VENDOR Shockwave PRODUCT Player PRODUCT before VERSION 11 VERSION . VERSION 6 VERSION
allows attackers to execute arbitrary code via unspecified .

GPT-2 Model Result

Before downloading the software , please note that
Adobe VENDOR Shock PRODUCT wave PRODUCT Player PRODUCT before VERSION 11 VERSION . VERSION 6 VERSION
allows attackers to execute arbitrary code via unspecified .

XLNet Model Result

Before downloading the software , please note that
Adobe VENDOR Shock PRODUCT wave PRODUCT Player PRODUCT before VERSION 11 VERSION . VERSION 6 VERSION
allows attackers to execute arbitrary code via unspecified .

Figure 15: GUI of CPE-Identifier.

word corresponds to one entity, the same word can be identified as different entities at different positions of the sentence. For example, according to dictionary.com [dict], the word ”before” is defined as ”previous to the time when”. It does not belong to any entities when used alone. However, in the example sentence, the word ”before” is identified as different entities when it is positioned differently. The word ”before” at the beginning of the sentence is not highlighted, which indicates that it is not an entity. However, the word ”before” in the front of the word ”11.6” is highlighted in purple, which indicates that it belongs to the ”VERSION” entity. This finding shows that our fine-tuned BERT model is intelligent enough to identify the meanings of the words based on their syntactic and semantic positions in the sentence, and not purely based on the definitions of that particular words. This is a good indicator of the performance of our fine-tuned model.

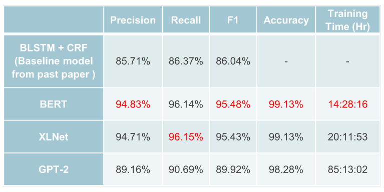
7 RESULT ANALYSIS

The models' performances are evaluated using Accuracy, Precision, Recall, and the F1 score and training time. The results are shown in the table.

7.1 Precision

The formula to calculate the precision score is

$$Precision\_Score = \frac{TruePositive}{TruePositive + FalsePositive}$$



	Precision	Recall	F1	Accuracy	Training Time (Hr)
BLSTM + CRF (Baseline model from past paper)	85.71%	86.37%	86.04%	-	-
BERT	94.83%	96.14%	95.48%	99.13%	14:28:16
XLNet	94.71%	96.15%	95.43%	99.13%	20:11:53
GPT-2	89.16%	90.69%	89.92%	98.28%	85:13:02

Figure 16: Model Results Comparison.

The BERT model achieved the highest Precision score, with a value of 94.83%. XLNet gained 94.71%, and GPT-2 obtained 89.16%.

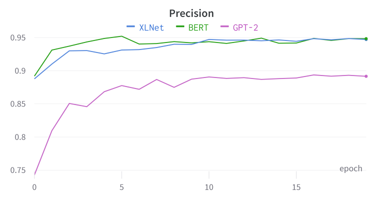


Figure 17: Model Precision Comparison.

7.2 Recall

The formula to calculate the recall score is

$$Recall\_Score = \frac{TruePositive}{TruePositive + FalseNegative}$$

The XLNet model achieved the highest Recall score, with a value of 96.15%. Bert gained 96.14%, and GPT-2 obtained 90.69%.

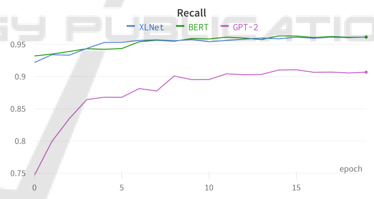


Figure 18: Model Recall Comparison.

7.3 Accuracy

A model with a high accuracy means it can locate most of the named entities correctly. The formula to calculate accuracy is

$$Accuracy\_Score = \frac{TP + TN}{TP + FP + TN + FN}$$

The BERT and XLNet models achieved the highest Accuracy scores, with a value of 99.13%, and GPT-2 obtained 98.28%.

However, although the model achieves an accuracy above 90%, 10% of the cases are the named identities, but the model predicts that they are not. This is obviously too high a cost for the cybersecurity expert to neglect that case. One missed named entity may

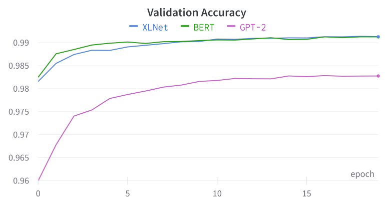


Figure 19: Model Accuracy Comparison.

cause the CPE-Identifier becomes unreliable and results in a single point of failure. Therefore, we also take the F1 score into account.

7.4 F1 Score

The F1 score is the most critical evaluation metric in machine learning. It considers precision and recall and gives a measure of the incorrectly identified cases. F1 score is a good assessment of the model as it considers the distribution of the data. If the data is highly imbalanced, the F1 score better assesses the model's predictive performance.

If our model incorrectly predicts positive named entities as negative or vice versa, the cybersecurity analysts will get confused about the prediction result. It is crucial to ensure there is as few False Positive and False Negative as possible, which means both precision and recall are maximal. However, in practice, it is impossible to maximize the precision and recall together. Improving the precision score will reduce the recall score and vice versa. Therefore, we used the F1 score as it considers both precision and recall.

The F1 score is the weighted average of the model's precision and recall. The range of the F1 score is between 0 and 1. 0 indicates the predictive performance of the model is worst, while 1 represents the predictive performance of the model is the best. The formula of the F1 score is

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The F1 score of the three models is plotted in a graph and compared. The BERT model achieved the highest F1 score, with a value of 95.48%. XLNet gained 95.43%, and GPT-2 obtained 89.92%.

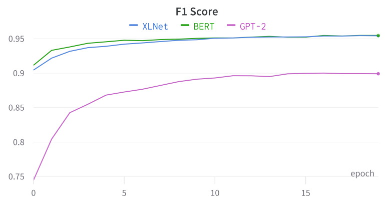
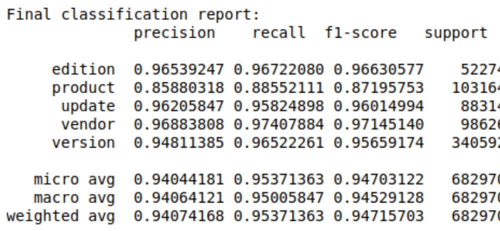


Figure 20: Model F1 Score Comparison.

Hence, from the performance metrics above, the Bert model is the best model which obtained the highest Precision, Accuracy, and F1 score. Its Recall score is 0.01% lower than the XLNet model.

7.5 Error Analysis

For the model with the best performance – the Bert model, we have done an error analysis and checked the number of CVEs in the test set have been correctly labeled for CPE without any error. The classification report of the prediction result is as shown in the Figure 21



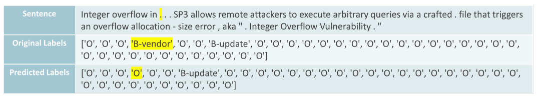
```
Final classification report:
              precision    recall  f1-score   support

     edition  0.96539247 0.96722080 0.96630577    522740
     product  0.85880318 0.88552111 0.87195753   1031640
      update  0.96205847 0.95824898 0.96014994    883140
      vendor  0.96883808 0.97407884 0.97145140    986260
     version  0.94811385 0.96522261 0.95659174   3405920

   micro avg  0.94044181 0.95371363 0.94703122   6829700
   macro avg  0.94064121 0.95005847 0.94529128   6829700
weighted avg  0.94074168 0.95371363 0.94715703   6829700
```

Figure 21: Classification Report.

Since our dataset is imbalanced and every class is equally important, we chose the Macro Averaged F1 score as the performance metric. According to the Classification report21, four out of five classes have achieved a F1 score above 95%. The F1 score for the "product" class is 87%. The reason for this low F1 score may due to the overfitting problem of the model. There are 322344 unique words that are labeled as "product", which is only 30% of the overall "product" class size. The size of the unique data is too small, and there are not enough data samples to accurately represent all possible input data values. Therefore, the model is unable to generalize well on the novel data of the "product" class. This issue is a potential future work we will address in the last section.

Among the 64982 CVE sentences in the test dataset, the Bert model predicts 95% of the CPE entities correctly on average. We have manually checked the dataset with wrong prediction result. There are two main causes for the errors.

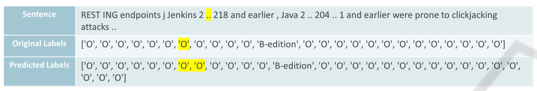
1. The original label is wrong. As shown in the Figure 22, the sentence is "Integer overflow in . . . SP3 allows remote attackers to execute arbitrary queries via a crafted . file that triggers an overflow allocation - size error , aka " . Integer Overflow Vulnerability .". The model predict the "." at the fourth position of the sentence as "O" entity. However, the Ground Truth label of the "." word in the entity is "B-vendor". The model predicts the correct result, while the original dataset is wrong.



Sentence	Integer overflow in . . . SP3 allows remote attackers to execute arbitrary queries via a crafted . file that triggers an overflow allocation - size error , aka " . Integer Overflow Vulnerability . "
Original Labels	['O', 'O', 'O', 'B-vendor', 'O', 'O', 'B-update', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
Predicted Labels	['O', 'O', 'O', 'O', 'O', 'O', 'B-update', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']

Figure 22: Ground Truth is Wrong.

2. The dataset treat the two continuous symbols as one single token, while the model predicts the two continuous symbols as two separate tokens. As shown in the Figure 23, the sentence is ”REST ING endpoints j Jenkins 2 .. 218 and earlier , Java 2 .. 204 .. 1 and earlier were prone to clickjacking attacks ..”. The model treats the two continuous symbols ”..” as two separate tokens and gives two separate labels – ”O” and ”O”. However, the dataset treat the two continuous symbols as one single token, thus corresponds to one single label ”O”. This problem can be solved with better quality of training dataset.



Sentence	REST ING endpoints j Jenkins 2 .. 218 and earlier , Java 2 .. 204 .. 1 and earlier were prone to clickjacking attacks ..
Original Labels	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-edition', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
Predicted Labels	['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-edition', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']

Figure 23: Different number of tokens.

8 CONCLUSION & FUTURE WORK

This paper designs an automated CPE labeling system called CPE-Identifier. The system can help the cybersecurity analysts automatically annotate new CVE summaries and identify key CPE entities, such as edition, product name, vendor name, version number, and update number. We have demonstrated the automated data annotation and augmentation processes by building two deep learning models. We have also compared the predictive performance of the three SoTA NER models when dealing with CVE summary text. We have concluded that the best model is the Bert model, which achieves an F1 score of 95.48% and outperforms the baseline model from past related work. We have also built a Graphical User Interface that allows the cybersecurity analysts to use the CPE-Identifier system conveniently and zoom into the respective CPE entities accurately and efficiently. This paves the way for further discussions on various security operations to accelerate the incident handling processes when a new vulnerability emerges.

Regarding the limitations of the proposed approach, it is noteworthy that while the performance metrics of the Data Annotation model are commendable, the automated labeling process applied to the Named Entity Recognition (NER) dataset might introduces a degree of noise into the resultant dataset. This, in turn, exerts an adverse impact on the training process of the XLNet, BERT, and GPT-2 NER models. Therefore, for future research endeavors, we advocate for a meticulous manual examination of the labeled dataset prior to its utilization in training the aforementioned NER models. Additionally, we recommend enhancing the Data Annotation model by training it on more recent datasets, as it is currently fine-tuned on data spanning from 2010 to 2013. This limited temporal scope may result in suboptimal performance when annotating contemporary data, where novel technical terminologies frequently emerge.

Furthermore, in light of the increasing prominence of Large Language Models (LLMs) such as the GPT series, LlaMa LLMs, and the PaLM LLMs, we propose collaboration with these advanced language models in our future research endeavors to harness their capabilities effectively.

REFERENCES

Stucco. (n.d.). GitHub - stucco/auto-labeled-corpus: Corpus of auto-labeled text for the cyber security domain. GitHub. https://github.com/stucco/auto-labeled-corpus

NVD - Data Feeds. (n.d.). https://nvd.nist.gov/vuln/data-feeds

Wåreus, E., & Hell, M. (2020). Automated CPE labeling of CVE summaries with machine learning. In Detection of Intrusions and Malware, and Vulnerability Assessment: 17th International Conference, DIMVA 2020, Lisbon, Portugal, June 24–26, 2020, Proceedings 17 (pp. 3-22). Springer International Publishing.

Team, R., & Team, R. (2022, October 5). Redscan analysis of NIST NVD reveals record number of critical and high severity vulnerabilities in 2020. Redscan. https://www.redscan.com/news/nist-nvd-analysis/

Browse CVE vulnerabilities by date. (n.d.). https://www.cvedetails.com/browse-by-date.php

O'Driscoll, A., & O'Driscoll, A. (2023, June 14). Cybersecurity vulnerability statistics and facts of 2023. Comparitech. https://www.comparitech.com/blog/information-security/cybersecurity-vulnerability-statistics/

Kanakogi, K., Washizaki, H., Fukazawa, Y., Ogata, S., Okubo, T., Kato, T., ... & Yoshioka, N. (2021). Tracing CVE Vulnerability Information to CAPEC Attack Patterns Using Natural Language Processing Techniques. Information, 12(8), 298.

Understanding TF-ID: A simple introduction. (2019, May 10). MonkeyLearn Blog. https://monkeylearn.com/blog/what-is-tf-idf/

Rostami, S., Kleszcz, A., Dimanov, D., & Katos, V. (2020). A machine learning approach to dataset imputation for software vulnerabilities. In Multimedia Communications, Services and Security: 10th International Conference, MCSS 2020, Kraków, Poland, October 8-9, 2020, Proceedings 10 (pp. 25-36). Springer International Publishing.

Cook, Bryan; Janamian, Saba; Lim, Teck; Logan, James; Ulloa, Ivan; Altintas, Ilkay; Gupta, Amarnath (2021). Using NLP to Predict the Severity of Cyber Security Vulnerabilities. In Data Science & Engineering Master of Advanced Study (DSE MAS) Capstone Projects. UC San Diego Library Digital Collections. https://doi.org/10.6075/J0TX3F89

Shahid, M. R., & Debar, H. (2021, December). Cvss-bert: Explainable natural language processing to determine the severity of a computer security vulnerability from its description. In 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 1600-1607). IEEE.

Ohuabunwa, B. C. (2022). Predicting Cybersecurity Vulnerability Severity via Boosted Machine Learning Ensembles and Feature Ranking (Doctoral dissertation, The George Washington University).

Evangelista, J. F. (2021). Cybersecurity Vulnerability Classification Utilizing Natural Language Processing Methods (Doctoral dissertation, The George Washington University).

Das, S. S., Serra, E., Halappanavar, M., Pothen, A., & Al-Shaer, E. (2021, October). V2w-bert: A framework for effective hierarchical multiclass classification of software vulnerabilities. In 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA) (pp. 1-12). IEEE.

Neuhaus, S., & Zimmermann, T. (2010, November). Security trend analysis with cve topic models. In 2010 IEEE 21st International Symposium on Software Reliability Engineering (pp. 111-120). IEEE.

Yang, H., Park, S., Yim, K., & Lee, M. (2020). Better not to use vulnerability's reference for exploitability prediction. Applied Sciences, 10(7), 2555.

Yin, J., Tang, M., Cao, J., & Wang, H. (2020). Apply transfer learning to cybersecurity: Predicting exploitability of vulnerabilities by description. Knowledge-Based Systems, 210, 106529.

Bozorgi, M., Saul, L. K., Savage, S., & Voelker, G. M. (2010, July). Beyond heuristics: learning to classify vulnerabilities and predict exploits. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 105-114).

Huff, P., McClanahan, K., Le, T., & Li, Q. (2021, August). A recommender system for tracking vulnerabilities. In Proceedings of the 16th International Conference on Availability, Reliability and Security (pp. 1-7).

Jia, Y., Qi, Y., Shang, H., Jiang, R., & Li, A. (2018). A practical approach to constructing a knowledge graph for cybersecurity. Engineering, 4(1), 53-60.

Bridges, R. A., Jones, C. L., Iannacone, M. D., Testa, K. M., & Goodall, J. R. (2013). Automatic labeling for entity extraction in cyber security. arXiv preprint arXiv:1308.4941.

Aghaei, E., Niu, X., Shadid, W., & Al-Shaer, E. (2022, October). SecureBERT: A Domain-Specific Language Model for Cybersecurity. In International Conference on Security and Privacy in Communication Systems (pp. 39-56). Cham: Springer Nature Switzerland.

Statnlp-Research. (n.d.). statnlp-datasets/dataset/MalwareTextDB-1.0.zip at master · statnlp-research/statnlp-datasets. GitHub. https://github.com/statnlp-research/statnlp-datasets/blob/master/dataset/MalwareTextDB-1.0.zip

Andrew Buttner, The MITRE Corporation Neal Ziring, National Security Agency. (2008, January). Common Platform Enumeration (CPE) – Specification. https://cpe.mitre.org/specification/2.1/cpe-specification_2.1.pdf

bert-base-cased · Hugging Face. (2023, June 1). https://huggingface.co/bert-base-cased

distilroberta-base · Hugging Face. (n.d.). https://huggingface.co/distilroberta-base

Cheryl. (2022, March 30). study notes: Handling Skewed data for Machine Learning models. Medium. https://reinec.medium.com/my-notes-handling-skewed-data-5984de303725

Liang, X. (2021, December 10). What is XLNet and why it outperforms BERT - Towards Data Science. Medium. https://towardsdatascience.com/what-is-xlnet-and-why-it-outperforms-bert-8d8fce710335

Nowak, M., Walkowski, M., & Sujecki, S. (2021, June). Machine learning algorithms for conversion of CVSS base score from 2.0 to 3. x. In International Conference on Computational Science (pp. 255-269). Cham: Springer International Publishing.

Georgescu, T. M., Iancu, B., & Zurini, M. (2019). Named-entity-recognition-based automated system for diagnosing cybersecurity situations in IoT networks. Sensors, 19(15), 3380.

Dictionary.com — Meanings & Definitions of English Words. (n.d.). In Dictionary.com. https://www.dictionary.com/browse/before