Proceedings of the 6th International Conference on Computing and Data Science
DOI: 10.54254/2755-2721/71/20241647

Refining CVE-to-CWE mapping with enhanced attention in BERT-based models

Jingyi Su[1,2], **Yan Wu**[1,3,*]

[1]Bowling Green State University, OH, USA

[2]jsu@bgsu.edu
[3]yanwu@bgsu.edu
*Corresponding author

Abstract. In this study, we introduce CySecBERT-ARD, an advanced approach for classifying software vulnerabilities that maps Common Vulnerabilities and Exposures (CVE) to Common Weakness Enumerations (CWE). Our approach is to use a pretrained transformer-based model CySecBERT tailored for cybersecurity contexts, the model is enhanced with additive attention and relative position encoding which allow for a deeper understanding of the vulnerability descriptions of CVE by capturing the contextual relationships. Our approach achieves an impressive accuracy of 91.34% and F1-score of 91.32% during the evaluation and testing phase compared to the base models. The results demonstrate the potential of CySecBERT-ARD in enhancing the efficiency and effectiveness of vulnerability classification.

Keywords: BERT, Transformers, Cybersecurity, Vulnerability Classification.

1. Introduction

As the eld of cybersecurity rapidly evolving, it is crucial to have accurate and timely vulnerability classification for securing software and systems. CVE and CWE are integral to cybersecurity systems and known vulnerabilities are published and categorized by their weaknesses. Mapping CVEs to CWEs efficiently is beneficial for software engineers and developers to manage the vulnerabilities, since understanding the vulnerabilities with their underlying categories and related weaknesses helps in developing defensive strategies. Traditional methods which are mostly based on manual categorization and rely on security experts are often time and resource costly and may fail to recognize the complexities of some vulnerability descriptions. They are also prone to errors and lag behind the volume and evolving nature of threats. This indicates the need for more complex automated approaches that are able to handle large volumes of data with high accuracy and efficiency. Our model, CySecBERT-ARD, utilizes a transformer-based architecture enhanced with additive attention, relative position encoding, and dual pooling, achieving substantial performance improvements. With an overall F1 of 91.32%, this model outperforms the base model and variants across different vulnerability categories showing a strong ability to handle the complexities of CVE descriptions. The contributions of this study are:

Integration of CySecBERT [1] for Cybersecurity Texts: We utilize the pre-trained CySecBERT model specifically for cybersecurity contexts, to generate contextual embeddings from preprocessed CVE descriptions as input texts. This allows our model to more effectively collect the nuances of some of the keyword meanings and their relationships with each other in the dataset about cybersecurity data.

Dual Pooling with Attention Mechanism: Our model contains a dual pooling layer, which is a combination of average pooling and maximum pooling operations. We employ this mechanism to try to extract overall feature activations and peak feature activations, thus improving the robustness and accuracy of extracted features in the classification task. We incorporate an additive attention mechanism in the model for dynamically prioritizing important text features. Using this attention mechanism in conjunction with relative positional coding can be more helpful in drawing out positional relationships in CVE descriptions, and such attempts lead to more accurate CWE classification in terms of CVE descriptions alone.

Comprehensive Classification Framework: Our model is structured to include multiple high-level layers, such as a step-by-step approach to learn the word meanings of CVE descriptions more deeply using BERT embedding, relative position encoding, additive attention, and double pooling.

This structure improves the original model's ability to categorize different CVE descriptions into appropriate CWE categories with higher accuracy and reliability during the experiments. These innovations enhance the performance of our model for CVE and CWE categorization, with significant improvements over the base model and other similar variants of the model, ensuring efficiency and effectiveness in handling complex CVE and CWE data.

2. Background

As more software is developed and utilized, the vulnerabilities represent significant threats to modern information systems, as cyber attackers can make use of the weaknesses to access and modify sensitive data. They can also take control of devices and spread malware which leads the users' system to be more vulnerable and exposed to dangerous environments. The attackers are able to reach other hosts within the same network just by accessing one node [2]. Attackers can also use these hosts with weaknesses to do malicious activities, increasing the overall impact of the initial vulnerability [3].

To make these threats less intimidating, the National Vulnerability Database (NVD) was launched in 2004, the world's most comprehensive repository of publicly disclosed vulnerabilities in commercial and open-source software and enhances the CVE list previously developed by the nonprofit MITRE Corporation in 1999. While the NVD and CVE are closely related and often used interchangeably, the NVD adds important analytics and metadata to CVE entries, providing a more detailed resource for understanding vulnerabilities. Each CVE entry includes a unique identifier, a detailed description, and public references, covering affected products, vendors, impacts, access required for exploitation, and compromised code components [4]. This extensive information makes the NVD a valuable resource for cybersecurity professionals [5].

Complementing the CVE system, CWE further categorizes the types of software vulnerabilities by using them as root causes that can be referenced and queried to focus on security issues. This system quickly began to help developers and security professionals to easily understand, learn, and use to try to address potential weaknesses in their systems. By identifying common patterns of flaws, the CWE facilitates the development of preventive measures against widespread vulnerabilities [6]. This categorization also aids in prioritizing security efforts and resources more effectively [7].

Since CVEs and CWEs play an important role in cybersecurity, our research is directed toward building models to accurately categorize CVE textual descriptions into the correct CWE categories, which are essential for accurately locating and mitigating security risks in software systems.

As mentioned in the previous section, this task is crucial for accurately locating and mitigating security risks in software systems. By learning and extracting the known vulnerabilities found in the software components in the CVE details, they are categorized into CWEs as broader vulnerability categories. Understanding this classification is vital for devising effective security strategies and enhancing the resilience of systems against potential attack s [8]. This classification also assists in the development of targeted mitigation strategies to address specific types of vulnerabilities [9].

Initially, CVE to CWE classification was conducted manually by cybersecurity experts who meticulously reviewed vulnerability reports and mapped them to CWE identifiers using their expert knowledge and established criteria [10]. While this method ensured high accuracy, it proved unscalable with the increasing volume of vulnerabilities. Basic NLP techniques have also been leveraged in traditional classification approaches. Kanakogi et al. (2021) utilized TF-IDF and Doc2Vec to identify CAPEC attack patterns from CVE descriptions [11], enabling automated mapping of vulnerabilities to attack patterns and improving classification accuracy. Recent advancements have introduced sophisticated techniques like V2W-BERT, a Transformer-based learning frame- work that integrates natural language processing, link prediction, and transfer learning [12]. This framework outperforms previous methods, showcasing the potential of advanced machine learning techniques in cybersecurity.

3. Methodology

This section describes our approach to building the model and running the model data preparation methodology, the model architecture and flow, and the effects of using the model and categorizing CVEs as CWEs if experiments are conducted and evaluated.

3.1. Data Acquisition and Representation

In our prep work, we used data collected from MITRE and NVD, a dataset that spans from 2002 to 2021 according to the data description. It contains rich information about publicly disclosed cybersecurity vulnerabilities and their categorization, and we adopted the software development view to categorize vulnerabilities around concepts frequently encountered in software development as an initial attempt at modeling. To ensure the reliability and relevance of the dataset, we decided to focus our categorization on the most prevalent CWEs (top 50), especially those CWE categories with more than 100 CVEs. This approach highlights the common vulnerabilities and improves the generalization ability of the model. We divided the data into training, evaluation, and test sets in a ratio of 7:1.5:1.5 (the number of data entries for train, eval and test: 38547, 8238, 8314) and maintained an even distribution of CWEs across these sets in a randomized grouping approach to ensure a balanced assessment of model performance with no missing CWEs.

3.2. Data Preprocessing

Our preprocessing pipeline enhances the model's ability to interpret general as well as security-related texts with greater accuracy.

3.2.1. Text Normalization and Cleaning

Initial text preprocessing involves several operations:

- Normalization and Contraction Expansion: For uniformity, we standardize tokenized text input by converting all textual vocabulary to lowercase, and acronyms are expanded to their full form to reduce lexical ambiguity.
- Tokenization and Lemmatization: We used the Natural Language Toolkit (NLTK) to tokenize text into individual words and lemmatize each token into its base form. This step simplifies the language and focuses on the core meaning of the words.
- Removal of Non-Informative Text: We also removed non-alphabetic characters and numbers and stripped out URLs to focus on plain text information. In addition, our approach includes the removal of stop words of low semantic value, allowing us to focus on more meaningful content that is more relevant to software security.
- Handling of Rare Categories: We also set up the option to filter out infrequent CWEs (when there are less than 100 relevant CVEs) in order to focus training on more and more supported data points.

3.2.2. Validation and Error Handling

We ensure the validity and reliability of the data through:

- Validation Checks: We perform checks for invalid entries, such as CVSS scores outside the acceptable range or incorrectly formatted entries, to maintain data quality for training.

- Logging and Documentation: Detailed logs are kept throughout the preprocessing steps to document data transformations, errors, and dropped rows, providing transparency and traceability.
- Removal of Duplicates and CVEs with Missing Description: Further checks for duplicates are done before sending the preprocessed data into models. These methods of preprocessing improve the quality and consistency of the data fed into the model, greatly affecting the effectiveness and efficiency of subsequent classification tasks.

During the data preprocessing phase, we cleaned and rearranged the original data with very satisfying results that contributed to all the models we evaluated.

3.3. *Transformer-Based Model Architecture*

In this section, we present the architecture of the transformer-based model CySecBERT-APR, which, as mentioned earlier, incorporates advanced features such as additive attention, relative position encoding, and dual pooling to enhance the processing of cybersecurity texts and to ensure improved model performance in the task of classifying CVEs.

- BERT Embeddings: We first generate contextual embeddings from the input text using a pre-trained BERT model (CySecBERT which is further pretrained on CVE descriptions) that captures the differences in word meanings in the CVE descriptions and their relationships in the cybersecurity context.
- Relative Position Coding: We also use sinusoidal coding to supplement the relative position information of the embeddings, which helps us understand the order-dependent features of the language in the CVE descriptions.
- Additive Attention: Our model applies an additive attention mechanism after embedding to pay more attention to the relevant parts of the text and uses a trainable scoring system to prioritize critical information for classification.
- Dual Pooling: Based on the attentional mechanism, we use the dual pooling technique, where we combine average pooling and maximum pooling to capture overall feature activations and peak feature activations in this layer to enhance the robustness of extracted features.
- Classifier: The final feature set is passed through a culling layer for regularization and a linear layer for classifying features into CWE categories.

3.4. *Training Protocol and Hyperparameter Optimization*

Our model is trained in batches of 16, using a cross-entropy loss function to optimize parameters and an Adam optimizer to adjust learning rates. Training spans 10 epochs with a learning rate of 1×10^{-5} and a dropout rate of 0.1.

3.5. *Evaluation Metrics and Validation*

During our experiments, we chose to evaluate the effectiveness of the model using key performance metrics such as precision, recall, and F1 scores, which are critical for measuring the model's ability to classify and generalize across different cybersecurity vulnerabilities due to the variety of different CWEs in the dataset. The performance comparison table below shows the performance of the various model variants in Table 1.

Table 1. Comparison of Model Performance

Model	Accuracy(%)	F1-Score(%)	ROC AUC	PR AUC
CySecBERT-ARD	**91.34**	**91.32**	**99.47**	**83.67**
CySecBERT-AAR	91.28	91.13	99.32	82.33
CySecBERT-AR	90.93	90.85	99.43	82.15
CySecBERT	90.99	90.92	99.32	80.10
BERT	90.39	90.02	99.02	79.04

This table highlights the performance of the CySecBERT-ARD model in terms of precision, recall, and F1-score, showing its robustness and accuracy in classifying CVE into CWE categories. In the table, we also have other evaluation results from model variants including CySecBERT-AR, which features additive attention and relative position encoding; and CySecBERT-AAR, which combines an attention pool, additive attention, and relative position encoding. By comparing the performance based on the test dataset involved with the same CWEs, CySecBERT-ARD is the one that integrated the best combination of such techniques to capture the most accurate CVE contexts.

Table 2. Performance Metrics for CySecBERT-ARD for Top 10 CWEs Based on Support

CWE	Precision(%)	Recall(%)	F1-Score(%)	Support
79	99.04	98.70	98.87	2615
89	99.05	98.86	98.95	1050
78	87.63	90.97	89.27	288
94	87.25	85.33	86.28	409
125	91.18	89.14	90.15	580
190	83.46	83.46	83.46	260
787	84.84	87.34	86.07	782
476	93.36	92.59	92.98	243
434	84.87	83.77	84.31	154
59	93.04	90.68	91.85	118

The dataset was filtered with CWE categories which are associated with more than 100 CVEs relevant to software development. To ensure a thorough and balanced evaluation, we divided the dataset into training, evaluation, and test sets using a 7:1.5:1.5 ratio. The partial model performance for CWE with more than 100 CVEs is shown in Table 2.

4. Future Work

Although our model has enhanced the classification task of CVE into CWE categories, there are several promising aspects for future work toward better cybersecurity classification performance. These opportunities aim to refine our existing framework and explore new dimensions of vulnerability classification.

Firstly, the multi-label classification offers a good direction for future iterations. In our evaluation setup, each CVE is linked to a single CWE category, but many vulnerabilities are multifaceted and related so that they cannot be neatly categorized into one specific CWE. The exploration of a multi-label classification task would address this complexity, enabling a single CVE to be linked with various CWEs, the understanding of CVEs would be further studied due to the need to extract more accurately reflecting their true nature.

Secondly, the potential enhancement could be based on the CWE hierarchy. The CWE taxonomy often presents a hierarchy of weaknesses, and leveraging this structure could improve both the accuracy and specificity of vulnerability categorization. The hierarchies are officially published in different views and our model focuses on the software development view, which is the most popular view. Thus, implementing hierarchical classification models would capitalize on the inherent relationships among different classes of weaknesses, potentially leading to more precise and context-aware classifications.

Lastly, enhancing our approach to handle the rare CWEs. With less information provided by the CVE, the model will suffer from an imbalanced data set. And in the real world, it is hard to balance the number of CWEs for all categories. This would be one of our next studies which will not only improve the model's overall accuracy but also ensure that it delivers reliable predictions across all CWEs.

5. Conclusion

In conclusion, we introduced a Bert-based framework CySecBERT-ARD which is refined with additive attention, relative position encoding, and dual pooling. Based on the evaluation metrics, the combination of attention techniques usage improves the CVE to CWE category classification accuracy. We also integrated steps with Bert model pretrain using CVE descriptions, and data preprocessing with natural language processing techniques tailored for cybersecurity texts. In this way, our approach our performed by classifying the language of cybersecurity vulnerability descriptions as CVE. It efficiently recognizes the differences between CWEs as weakness categories based on the CVE description, which is beneficial for devising precise mitigation strategies and enhancing predictive precision in software weaknesses. We anticipate further advancements, including additional CVE features and CWE descriptions integration, hierarchical classifications, and improved handling of rare classes. These developments aim to maintain the relevance and enhance the performance of our variants. Our work demonstrates how machine learning techniques can benefit the field of software security and has the honor of providing broader ideas for future research in the field.

Acknowledgments
Sincerely thank Dr. Yan Wu for her guidance and support throughout this research. We also extend our gratitude to the NVD for the public dataset and resources.

References

[1] Bayer M, Kuehn P, Shanehsaz R, Reuter C. Cysecbert: A domain-adapted language model for the cybersecurity domain. ACM Transactions on Privacy and Security. 2024 Apr 8;27(2):1-20.
[2] Elmishali A, Stern R, Kalech M. Diagnosing software system exploits. IEEE Intell Syst. 2020;35:7-15. doi: 10.1109/MIS.2020.2965496.
[3] Charmanas K, Mittas N, Angelis L. Predicting the existence of exploitation concepts linked to software vulnerabilities using text mining. In: Proceedings of the 25th Pan-Hellenic Conference on Informatics; 2021. doi: 10.1145/3503823.3503888.
[4] Younis A A, Malaiya Y. Using software structure to predict vulnerability exploitation potential. In: 2014 IEEE Eighth International Conference on Software Security and Reliability-Companion; 2014. p. 13-18. doi: 10.1109/SERE-C.2014.17.
[5] Bullough B L, Yanchenko A K, Smith CL, Zipkin J R. Predicting exploitation of disclosed software vulnerabilities using open-source data. In: Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics; 2017. doi: 10.1145/3041008.3041009.
[6] Bhatt N, Anand A, Yadavalli V. Exploitability prediction of software vulnerabilities. Qual Reliab Eng Int. 2020;37:648-663. doi: 10.1002/qre.2754.
[7] Younis A A, Malaiya Y, Ray I. Assessing vulnerability exploitability risk using software properties. Softw Qual J. 2016;24:159-202. doi: 10.1007/s11219-015-9274-6.
[8] Almukaynizi M, Nunes E, Dharaiya K, Senguttuvan M, Shakarian J, Shakarian P. Patch before exploited: An approach to identify targeted software vulnerabilities. In: AI in Cybersecurity. Springer; 2018. doi: 10.1007/978-3-319-98842-9_4.
[9] Iannone E, Guadagni R, Ferrucci F, De Lucia A, Palomba F. The secret life of software vulnerabilities: A large-scale empirical study. IEEE Trans Softw Eng. 2023;49:44-63. doi: 10.1109/TSE.2022.3140868.
[10] Aghaei S, Others. Manual CVE to CWE mapping: Challenges and expert solutions. J Cybersecur Res. 2023;9(1):45-59.
[11] Kanakogi H, Others. Applying NLP techniques to classify CVE entries into CAPEC categories. J Inf Secur Appl. 2021;59:102717.
[12] Das S S, Serra E, Halappanavar M, Pothen A, Al-Shaer E. 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); 2021. p. 1-12. doi: 10.1109/DSAA53316.2021.9564227.