CVSS PREDICTION: BERT CLASSIFIER FOR EVALUATING COMPUTER SECURITY SEVERITY OF VULNERABILITY

Dr. S Gunasekaran[1*], A Happy[2], A Abhinandh[3],M Amal Mohamed[3], K Chaithra[3], C S Harshan[3]

[1*]Professor, Computer Science and Engineering, Ahalia School of Engineering and Technology, Palakkad, Kerala, India - 678557
[2]Asst. Professor, Computer Science and Engineering, Ahalia School of Engineering and Technology, Palakkad, Kerala, India – 678557
[3]Students, Computer Science and Engineering, Ahalia School of Engineering and Technology, Palakkad, Kerala, India - 678557
*E-mail: gunaphd@yahoo.com

Abstract. Security vulnerabilities form a significant risk to organizations, which leads to effective prioritization to mitigate potential threats. For assessing the severity of the vulnerability, the Common Vulnerability Scoring System (CVSS) provides a standardized method, but manual scoring is time-consuming and demands expert knowledge. This project leverages Natural Language Processing (NLP) and deep learning techniques, specifically a BERT-based model, to automate CVSS vector generation from textual vulnerability descriptions. By training multiple classifiers to predict individual CVSS metrics, the system computes severity scores efficiently, aiding in risk prioritization. The proposed approach enhances cybersecurity defences by providing accurate, scalable, and explainable vulnerability assessments, ultimately improving organizational response times to emerging threats.

1. Introduction

Computer security vulnerability is a flaw that attackers can exploit that compromises a system's confidentiality, integrity, or availability. Organizations face challenges managing thousands of vulnerabilities that are disclosed annually. It is difficult to address them as limited resources are available. Effective prioritization is necessary for focusing on high-severity vulnerabilities to reduce risks while balancing operational demands. Among all vulnerability rating methods, the Common Vulnerability Scoring System (CVSS) remains the most utilized evaluation system. Security experts use CVSS as a standardized evaluation framework to measure security weaknesses to inform business decisions about operational remediations. MITRE maintains a Common Vulnerabilities and Exposures (CVE) system that lists publicly disclosed vulnerabilities, while the NIST National Vulnerability Database (NVD) [7] enhances this data with severity scores and impact ratings using the Common Vulnerability Scoring System (CVSS) [3].

CVSS evaluates vulnerabilities based on exploitability and impact; still, assessing and scoring vulnerabilities manually is time-consuming and requires expert knowledge. To handle this, CVSS scoring is automated using Natural Language Processing (NLP). A system based on BERT (Bidirectional Encoder Representations from Transformers) is trained to predict CVSS

metrics from the textual descriptions in CVE entries [4]. Each classifier focuses on a specific CVSS metric, and their predictions are combined to compute a severity score. This speeds up the evaluation process, allowing users to understand and justify the model's decisions. By automating this process, organizations can efficiently assess vulnerabilities, prioritize critical issues, and enhance their cybersecurity defenses.

This paper overall focuses how a BERT-based classifier can help to predict the CVSS score. The process includes gathering data from the NVD source, cleaning and preprocessing it, training the BERT model, and evaluating the performance metrics (F1 Score, Accuracy, Balanced Accuracy). Finally, the model is fine-tuned, tested and analysed for real world applications.

2. Related Work

Rishabb Singla et al. [1] propose a system using NVD data to classify and analyze vulnerabilities automatically in compute, IoT, and network devices. TF-IDF is used to process textual data, and a multinomial Naïve Bayes algorithm is used to classify vulnerabilities by product groups (compute, IoT, and network) and domains (application, OS, and hardware). The study shows rising vulnerabilities in IoT and network devices, especially in hardware and OS, while application vulnerabilities decline. Buffer overflows and improper authentication are identified as prolonged issues across categories. To monitor trends and prioritize vulnerabilities, the system provides metrics to help security professionals in managing risks and improving cybersecurity strategies.

Xiaodan Li et al. [2] propose a new model to classify software vulnerabilities, which is based on their nature, exploitation complexity, and attack likelihood. Vulnerabilities are grouped into Bohr Vulnerabilities (easy to detect and exploit), Non-Aging-Related Mandel Vulnerabilities (complex dependencies like timing issues or design flaws), Aging-Related Vulnerabilities (resource exhaustion or numeric errors), and Unknown Vulnerabilities (novel or unclear). Analysis of Google Project Zero data shows non-aging-related vulnerabilities as the most common, highlighting the importance of better memory management and input validation. The framework links vulnerabilities to attack patterns, providing insights to enhance mitigation strategies and guide future research.

The Common Vulnerability Scoring System version 4.0 specification document [3] provides a comprehensive guide to assessing software vulnerability severity using the updated Common Vulnerability Scoring System (CVSS). CVSS classifies metrics into four groups: Base Metrics, which evaluate intrinsic vulnerability characteristics; Threat Metrics, which assess the current threat landscape; Environmental Metrics, which adjust scores based on organizational context; and Supplemental Metrics, offering additional context without affecting the score. Key metrics include exploitability (e.g., attack complexity, attack vector, etc.) and impact (integrity, availability, and confidentiality). The document explains how these metrics combine to calculate a base score, ranging from 0 to 10, helping organizations prioritize vulnerabilities based on their potential risk.

Ion Babalau [4] et al. present a deep learning approach with a pre-trained BERT model using multi-task learning to predict the severity of software vulnerabilities based on their text descriptions. The model is trained on a dataset from the Common Vulnerabilities and Exposures (CVE) repository, achieving a mean absolute error of 0.86 in severity score prediction and 71.55% accuracy in severity classification. The model predicts key CVSSv3 metrics such as exploitability, impact, and attack complexity, outperforming baseline models

like TF-IDF and Naive Bayes. By automating vulnerability assessments, the method helps system administrators prioritize threats without waiting for official CVSS scores. Future improvements include experimenting with other language models and developing a real-time monitoring application for CVE feeds.

3. Proposed Work

This work proposes an approach to predict software vulnerability severity scores to automate the generation of CVSS vector scores from textual vulnerability descriptions. Multiple BERT classifiers are used for predicting a specific metric of the CVSS vector, such as Attack Complexity (AC), Attack Vector (AV), User Interaction (UI), Privileges Required (PR), Confidentiality Impact (C), Integrity Impact (I), and Availability Impact (A). Each classifier will take vulnerability description as input, provided by the CVE system, and predict the corresponding value for the assigned metric. By combining the predictions of all classifiers, a complete CVSS vector will be generated, from which the severity score can be calculated. This process will help prioritize vulnerabilities based on their predicted risk level, facilitating more efficient and targeted vulnerability remediation efforts. The effectiveness of this approach will be evaluated using standard performance metrics such as balanced accuracy, accuracy, and F1-score. By training classifiers for each CVSS metric, we aim to address challenges created by emerging cyber threats and the growing complexity of modern software. This method will automate the CVSS scoring process, ensuring consistency and scalability, especially as new vulnerabilities are reported daily. The results of this work will help to prioritize high-severity vulnerabilities, improving response times and the overall security posture of organizations. Additionally, this project aims to explore the impact of dataset imbalances and refine the model's ability to generalize across different software domains, ensuring that the predictions are reliable in a wide range of real-world scenarios.

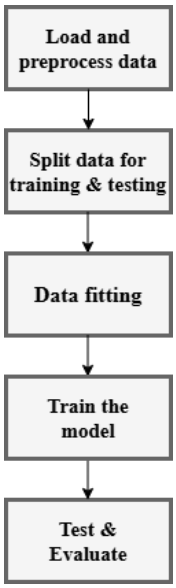

Figure 1: Workflow Diagram

3.1 Dataset

The NVD (National Vulnerability Database) is a repository of publicly known software vulnerabilities, identified by CVE (Common Vulnerabilities and Exposures) identifiers. It provides detailed information, including text descriptions, CVSS scores, and severity ratings for each vulnerability, linked to their corresponding CVE entries. It is well suited for training machine learning models due to its structured format, including BERT for CVSS prediction, using CVE-related data.

3.2 BERT model

Initially, collect vulnerability text descriptions and their CVSS scores as training data. Pre-process the text and fine-tune a pre-trained BERT model to predict CVSS metrics. Train the model to learn relationships between text descriptions and corresponding scores. The trained model is used to predict CVSS scores for new vulnerability descriptions.

3.3 Tokenization

Tokenization of CVE descriptions for BERT involves breaking down the text into smaller units [1], such as words or sub-words, to make it suitable for processing by the model. After tokenization, the text is converted into input embedding, including attention masks and token type IDs, to guide the BERT model in processing the CVE descriptions effectively.

3.4 CVSS Base Score

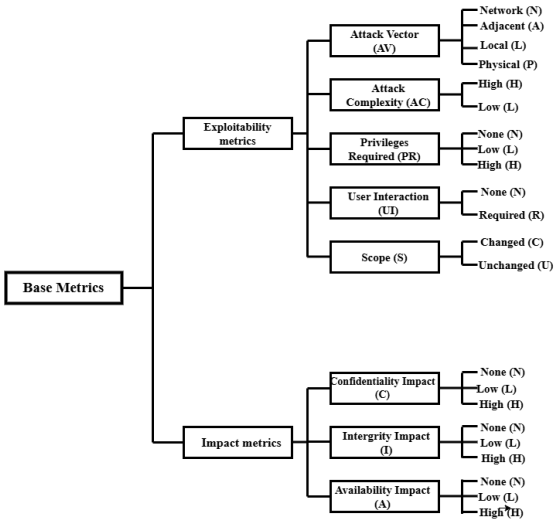

Figure 2: Base Metrics

Exploitability Sub score, E = 8.22 x AV x AC x PR x UI
Impact Sub score:
If Scope (S) is changed: I= 7.52 x (C+ I + A - C x I - C x A - I x A)
If Scope(S) is unchanged: I = 6.42 x C x I x A
Base Score = round (min (I + E,10),1)

3.5 Generate Final Predictions

Use the fine-tuned BERT model to generate the final predictions of the CVSS score for the given CVE description.

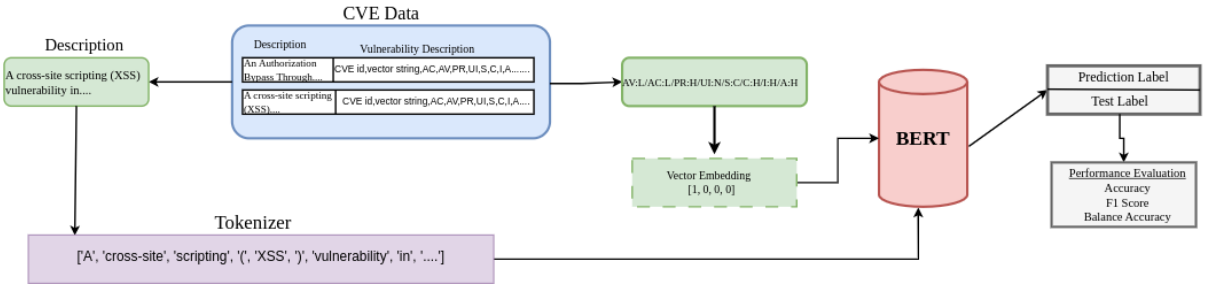

Figure 3: Architecture Diagram

4. Implementation

4.1 Data Collection

A total of 45676 vulnerability descriptions from the year 2022-2023 are taken for this purpose from NVD [5]. The dataset includes details about each vulnerability, such as publication and modification dates, CVE ID, description, CVSS v3 metrics (version, vector string, AV, AC, PR, UI, S, C, I, A, base score, and severity), exploitability and impact scores, and CWE-related information. The classification of vulnerabilities based on CVSS v3 Base Severity is as follows: Medium: 19881, High: 17506, Critical: 7535, Low: 754.

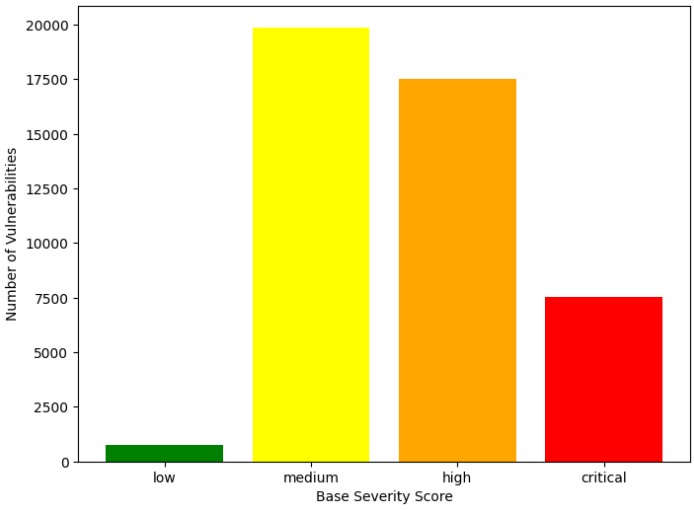

Figure 4: Number of vulnerabilities based on their severity

4.2 Data Preparation

The code processes CVE data and transforms it into a more structured format. First, the first element from the list in the description column is extracted and normalized to JSON-like data

to flatten the structure. It then creates new columns (nb_CWE, CWE1, and CWE2) based on the extracted data. The number of CWE entries is counted, then the first and second CWE values are assigned, and cases where no CWE entries are handled. Finally, the original description column is removed, and the clean data is stored, making it ready for further processing or use. This cleaning process ensures consistency and simplifies the data structure.

The data is then organized in ascending order based on the published date of the vulnerability. Data preprocessing is done by splitting the dataset into features (X) and labels (y). The features are extracted from specific columns, and labels are derived from a range of columns. Then, the data is divided into training and test sets. The dataset will be split into 20% testing data and 80% training data. The splitting is done in a stratified manner to ensure that the class distribution in the target variable (cvssV3_baseSeverity) is preserved in both training and testing sets.

Table 1: Data Distribution

Category	Train	Test	Total
Attack Vector			
Network	27377	6829	34206
Local	8249	2045	10294
Adjacent	596	175	771
Physical	318	87	405
Attack Complexity			
Low	35381	8836	44217
High	1159	300	1459
Privileges Required			
High	3939	959	4898
Low	12036	3061	15097
None	20565	5116	25681
User Interaction			
Required	12170	2989	15159
None	24370	6147	30517
Scope			
Changed	6949	1763	8712
Unchanged	29591	7373	36964
Confidentiality			
High	20860	5213	26073
Low	8169	2089	10258
None	7511	1834	9345
Integrity			
High	17626	4460	22086
Low	8077	2020	10097
None	10837	2656	13493
Availability			
High	20607	5129	25736
Low	627	151	778
None	15306	3856	19162

Table 1 presents training, testing, and total data distribution across different CVSS metric categories, including Confidentiality Impact, Availability Impact, Integrity Impact, Attack Complexity, Attack Vector, Scope, User Interaction, and Privileges Required. For each metric, distinct subcategories are present, such as "Network" and "Local" for Attack Vector. Attack complexity shows a higher count of "low" samples compared to "high." Privileges Required and Scope also show significant imbalances, with "None" and "Unchanged" dominating, respectively. Confidentiality, integrity, and availability distributions highlight a higher count for "none" levels across categories. The distribution patterns reflect dataset diversity and potential biases relevant for predictive modeling tasks.

Table 2: CVSS Base Severity

CVSS Base Severity	Train	Test
Critical	6028	1507
High	14005	3501
Medium	15904	3977
Low	603	151

Table 2 provides the distribution of training and testing samples for CVSS Base Severity levels, which include low, medium, critical, and high. The majority of samples are concentrated in the "medium" (15,904 train, 3,977 test) and "high" (14,005 train, 3,501 test) categories, indicating a higher prevalence of medium to high-severity vulnerabilities in the dataset. The "low" severity category is significantly underrepresented, containing only 603 training and 151 testing samples. The "critical" severity is less common, with 6,028 training and 1,507 testing samples. As they are relatively rare compared to “medium” and “high”, this distribution imbalance suggests a potential challenge for machine learning models in accurately predicting "low" and "critical" severity levels.

4.3 BERT Classifier Training

A BERT-based model is trained for predicting Common Vulnerabilities and Exposures (CVSS) [4] scores using sequence classification tasks. Initially, the categorical labels (such as vulnerability severity) of the dataset are converted to numeric values as a preprocessing stage. Vulnerability descriptions are tokenized using BERT’s tokenizer. Tokenizing is a process that breaks down the text into smaller units, ensuring uniform input sizes by applying truncation or padding as necessary. For the model training, the tokenized text and corresponding numeric labels are suitably organized into a required format.

The AdamW optimizer is used to train the BERT model with a learning rate of 5e-5. To preserve pre-trained knowledge, the base BERT layers are kept frozen in the early stages, and the model adapts by training only the classification head. After each epoch, accuracy and loss are calculated to evaluate the performance of the model. During training, tokenized vulnerability descriptions and their numeric values serve as inputs, while the outputs are predicted CVSS scores or severity labels. The model is trained over several epochs, saving checkpoints for possible resumption.

Early stopping is the key component of this methodology, which halts training if there is no improvement in the model’s validation performance after a predefined number of epochs. This prevents overfitting and ensures computational efficiency. The optimal number of training epochs is determined based on validation performance, maximizing the model's generalization ability. After training, the best model is saved for future use. This process

ensures efficient training and accurate predictions for CVSS classification tasks, balancing performance, efficiency, and generalization. A gradient-based approach is applied to identify the keywords in the description that influenced the predictions, making the results more transparent.

4.4 BERT Classifier Testing

The evaluation of a BERT classifier involves assessing a pre-trained BERT model across many metrics, such as accuracy, F1 score, and balanced accuracy. Loading the model and its corresponding tokenizer is the first step in this process. The model is loaded from Hugging Face; the Bert-base-uncased tokenizer is used as a fallback. The next crucial step is preparing the test dataset, which involves reading feature data and target labels, which is done using the pandas library. The labels for the selected metric are extracted and encoded into numerical values using a LabelEncoder, ensuring the model can process them appropriately. The descriptions in the test set are tokenized using the BERT tokenizer once the data is prepared, which includes truncating and padding the text to ensure that all inputs are in the correct format for the BERT model. A PyTorch DataLoader is created to handle batching and efficient data loading, which helps manage memory and processing speed during evaluation. The model is then evaluated on the test dataset by switching it to evaluation mode and running inference on the data. The predicted labels and true labels are collected, and key performance metrics are computed. The metrics include F1 score, accuracy, and balanced accuracy. By offering a command-line interface for specifying different datasets and evaluation metrics without needing to modify the underlying code, this methodology is designed to be flexible and user-friendly. It integrates model evaluation, data preprocessing, and performance measurement into a seamless pipeline, allowing it to be applied to various tasks involving BERT-based classifiers across different datasets.

5. Evaluation

CVSS score calculation [3] method using 8 vectors of basic metrics was successfully implemented. The model was able to generate accurate vulnerability severity scores. The calculated scores were validated against known vulnerabilities to evaluate the effectiveness of the approach. Through this evaluation, it was observed how much the metric influenced the final score, providing valuable insights to risk assessment and prioritization. Overall, this model allowed for the development of a structured and reliable method for enhancing the decision-making process and assessing cyber security threats in vulnerability management.

Table 3: Performance Metrics

VECTORS	F1 SCORE	ACC.	BAL.ACC
AV	93.09	93.07	78.06
AC	97.77	98	75.14
PR	84.98	85.04	81.92
UI	95.08	95.09	94.22
S	97.24	97.25	95
C	88.15	88.18	85.96
I	88.56	88.57	87.96
A	89.89	90.2	69.22

Table 3 presents performance metrics for different vectors of base metrics used for CVSS score calculation in a BERT-based classification model, evaluated using F1 Score, Accuracy (ACC.), and Balanced Accuracy (BAL ACC.). It covers vectors of exploitability metric (e.g., AC, AV, S, UI, PR) as well as impact metric (e.g., A, I, C). The Attack Complexity (98%) shows the higher accuracy, while the lowest balanced accuracy appears in Availability Impact (69.22%), suggesting challenges in handling availability-related threats. Overall, the model performs well across most vectors, with variations in balanced accuracy indicating potential areas for improvement.

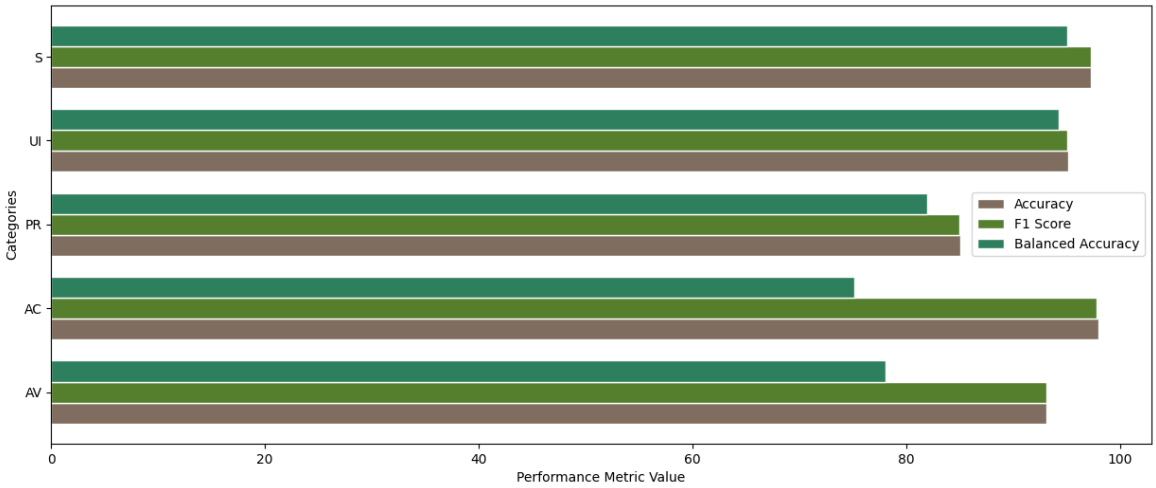

Figure 5: Exploitability Metrics Performance Evaluation

Figure 5 displays the performance measures of accuracy, F1 score, and balanced accuracy graphically. Applied to the exploitability metric, these measures span the following CVSS ranges: AC, AV, UI, PR, and S. There are three bars for every category; usually, the F1 Score and Balanced Accuracy show better values than Accuracy. Consistent high performance across all measures shows that categories including Scope and Attack Vector balance the model's forecasts. Conversely, the User Interaction and Privileges Required categories show rather lower F1 scores, suggesting possible issues with managing false classifications or class imbalances. The plot delineates the model's strengths alongside the aspects that may benefit from enhancement across all CVSS metrics.

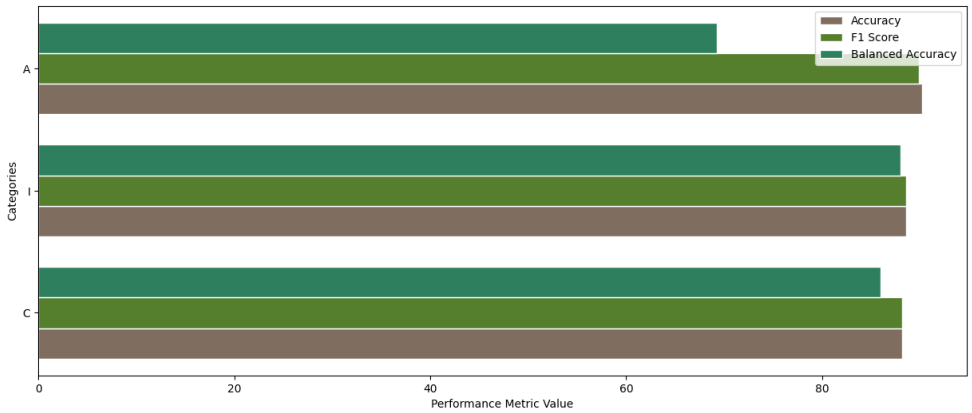

Figure 6: Impact Metrics Performance Evaluation

Fig 6 presents a comparative analysis of three performance metrics; Accuracy, F1 Score, and Balanced Accuracy across three categories of impact metrics. The x-axis represents the performance metric values, while the y-axis categorizes the three groups. Each category has three horizontal bars, each corresponding to a specific metric, as indicated by the legend. The chart reveals that categories I and C have relatively high values across all metrics, while category A has a noticeably lower Balanced Accuracy score.

6. Conclusion

In light of the large number of security vulnerabilities discovered every year, organizations should prioritize fixing them according to severity. Although CVSS is a common approach for assessing vulnerability level, it takes a lot of time and knowledge to generate the CVSS vector and score. Modern natural language processing (NLP) techniques can facilitate this aim by independently extracting the CVSS vector and score from the vulnerability description. We developed many BERT-based classifiers to assess each CVSS metric. The experiments demonstrated that the classifiers can precisely predict the CVSS vector and severity score, with their results nearly matching those of human experts. Essential terms influencing predictions were identified by a gradient analysis method, enhancing the clarity of the results. This method accelerates the process and provides clarity, as the identified terms align with the rationale typically employed by cybersecurity professionals.

The future scope of this work is to integrate interpretability into the existing fine-tuned CVSS score prediction model using Integrated Gradients, SHAP (Shapley Additive Explanations) or Attention Visualization to analyze how the model predicts the CVSS score and identify the words and categories that have the most significant impact on the prediction.

References

[1] Singla, Rishabh, et al. “Toward a Multidimensional Analysis of the National Vulnerability Database.” IEEE Access, vol. 11, 1 Jan. 2023, pp. 93354–93367, https://doi.org/10.1109/access.2023.3309850. Accessed 1 Nov. 2024.
[2] Li, Xiaodan, et al. “A Novel Approach for Software Vulnerability Classification.” 2017 Annual Reliability and Maintainability Symposium (RAMS), 2017, https://doi.org/10.1109/ram.2017.7889792
[3] “Common Vulnerability Scoring System.” FIRST — Forum of Incident Response and Security Teams, www.first.org/cvss/v4-0/.

[4] Ion Babalau, et al. Severity Prediction of Software Vulnerabilities Based on Their Text Description. 1 Dec. 2021, https://doi.org/10.1109/synasc54541.2021.00037.
[5] P. Mell, "Measuring the Common Vulnerability Scoring System Base Score Equation," 2022. https://doi.org/10.6028/NIST.IR.8409
[6] Amankwah, Richard, et al. "Evaluation of Software Vulnerability Detection Methods and Tools: A Review." International Journal of Computer Applications, vol. 169, no. 8, 17 July 2017, pp. 22–27, https://doi.org/10.5120/ijca2017914750..
[7] Tiantian-Tan, et al. "A Method for Vulnerability Database Quantitative Evaluation." Computers, Materials & Continua, vol. 61, no. 3, 2019, pp. 1129–1144, https://doi.org/10.32604/cmc.2019.06051. Accessed 25 Aug. 2021
[8] D. Gustavo Cruz, et al. "Open Source Solutions for Vulnerability Assessment: A Comparative Analysis." IEEE Access, vol. 11, 1 Jan. 2023, pp. 100234–100255, https://doi.org/10.1109/access.2023.3315595.
[9] Garg, Shivi, et al. "Analysis of Software Vulnerability Classification Based on Different Technical Parameters." Information Security Journal: A Global Perspective, vol. 28, no. 1-2, 4 Mar. 2019, pp. 1–19, https://doi.org/10.1080/19393555.2019.1628325.
[10] Imtiaz, Nasif, et al. "A Comparative Study of Vulnerability Reporting by Software Composition Analysis Tools." Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 11 Oct. 2021, arxiv.org/pdf/2108.12078.pdf, https://doi.org/10.1145/3475716.3475769.
[11] Shahid, Mustafizur R., and Hervé Debar. "CVSS-BERT: Explainable Natural Language Processing to Determine the Severity of a Computer Security Vulnerability from Its Description." IEEE Xplore, 1 Dec. 2021, ieeexplore.ieee.org/document/9680155. Accessed 12 Mar. 2024.
[12] Shah, Sahar, et al. "Fine-Tuning of Distil-BERT for Continual Learning in Text Classification: An Experimental Analysis." IEEE Access, vol. 12, 1 Jan. 2024, pp. 104964–104982, https://doi.org/10.1109/access.2024.3435537.
[13] KEKÜL, Hakan, et al. "Estimating Missing Security Vectors in NVD Database Security Reports." International Journal of Engineering and Manufacturing, vol. 12, no. 3, 8 June 2022, pp. 1–13, https://doi.org/10.5815/ijem.2022.03.01.
[14] Silva Barbon, Rafael, and Ademar Takeo Akabane. "Towards Transfer Learning Techniques—BERT, DistilBERT, BERTimbau, and DistilBERTimbau for Automatic Text Classification from Different Languages: A Case Study." Sensors, vol. 22, no. 21, 26 Oct. 2022, p. 8184, https://doi.org/10.3390/s22218184.
[15] https://nvd.nist.gov/vuln/data-feeds#JSON_FEED