

Vulnerability detection using BERT based LLM model with transparency obligation practice towards trustworthy AI

Jean Haurogné^a, Nihala Basheer^{b,*}, Shareeful Islam^b

^a Institut National des Sciences Appliquées, 20 Avenue des Buttes de Coesmes, 35700, Rennes, France

^b School of Computing and Information Science, Anglia Ruskin University, East Road, CB1 1PT Cambridge, United Kingdom

ARTICLE INFO

Keywords:

Large language model
Transparency obligation
Vulnerability
Explainability
BERT model
EU AI Act

ABSTRACT

Vulnerabilities in the source code are one of the main causes of potential threats in software-intensive systems. There are a large number of vulnerabilities published each day, and effective vulnerability detection is critical to identifying and mitigating these vulnerabilities. AI has emerged as a promising solution to enhance vulnerability detection, offering the ability to analyse vast amounts of data and identify patterns indicative of potential threats. However, AI-based methods often face several challenges, specifically when dealing with large datasets and understanding the specific context of the problem. Large Language Model (LLM) is now widely considered to tackle more complex tasks and handle large datasets, which also exhibits limitations in terms of explaining the model outcome and existing works focus on providing overview of explainability and transparency. This research introduces a novel transparency obligation practice for vulnerability detection using BERT based LLMs. We address the black-box nature of LLMs by employing XAI techniques, unique combination of SHAP, LIME, heat map. We propose an architecture that combines the BERT model with transparency obligation practices, which ensures the assurance of transparency throughout the entire LLM life cycle. An experiment is performed with a large source code dataset to demonstrate the applicability of the proposed approach. The result shows higher accuracy of 91.8 % for the vulnerability detection and model explainability outcome is highly influenced by “vulnerable”, “function”, “mysql_tmpdir_list”, “strmov” tokens using both SHAP and LIME framework. Heatmap of attention weights, highlights the local token interactions that aid in understanding the model’s decision points.

1. Introduction

In the realm of software development, vulnerability detection is critical for secure software development. The recent report by Open-Source Security and Risk Analysis (OSSRA) reveals that about 73 percent of the analysed commercial codebases, including the integration of open-source components, pose high-risk vulnerabilities that have increased in the past couple of years. Furthermore, the report shows that 91 % of codebases have components that are at least ten versions outdated, and almost half of them contain components with no updates in two years. AI-enabled systems have been categorised as a solution for effective vulnerability detection (Harzevili et al., 2023; Risse & Böhme, 2023). The capabilities of the AI model enable us to discover new threats relating to vulnerabilities for overall security enhancement. However, AI-enabled solutions exhibit limitations when dealing with large datasets, specifically understanding the context of the data, and labelling the data, which could be very time-consuming (Yang et al., 2023). This, in

turn, can lead to delayed detection and mitigation of critical vulnerabilities.

In this context, the Large Language Model (LLM) can be seen as a more effective solution to understand the context of the dataset, specifically by linking tokens in the source code-based vulnerability dataset (Giner-Miguel et al., 2024). LLM is an advanced AI model that is used to learn and produce natural language by using deep learning and copious amounts of text data. The rise in use of LLM is due to their capability of understanding natural language, data applicability to a wide range of tasks, and performing language-related tasks more effectively, thereby increasing efficiency and productivity in the given tasks (Nam et al., 2024). One of the significant drawbacks of using LLMs, especially in cases such as classification, is explaining the model’s decision-making (Luo & Specia, 2024). Due to the intricate nature of the models, both AI and LLMs are commonly regarded as black boxes. Such opacity can negatively affect the development of trustworthy AI solutions, especially when it is applied in important use cases such as

* Corresponding author at: School of Computing and Information Science, Anglia Ruskin University, East Road, CB1 1PT Cambridge, United Kingdom.

E-mail addresses: jean.haurogne@insa-rennes.fr (J. Haurogné), nihala.basheer@aru.ac.uk (N. Basheer), shareeful.islam@aru.ac.uk (S. Islam).

vulnerability identification. Transparency obligations in this context can effectively support explaining model decision-making processes and making sure that the decision-making processes of the model are accurate and consistent. This is particularly important in vulnerability detection, where understanding the rationale behind a model's decision can lead to more informed and effective security measures. The recent introduction of the EU AI Act also emphasises the transparency obligations practiced through the working mechanisms, data use, and decision-making processes of the AI model and general-purpose AI model (Outeda, 2024).

In this context, the novel contribution of this research is towards the development of trustworthy AI solutions and proposes an advanced vulnerability detection approach using the LLM model, which ensures transparency obligation practice throughout the entire life cycle of the model. This research makes three main contributions.

- Firstly, we adopt a BERT (Bidirectional Encoder Representations from Transformers) based LLM architecture for vulnerability detection. The pre-trained BERT model is fine-tuned using a specific dataset to enhance its capability for detecting vulnerabilities. The model's performance is evaluated by comparing its predictions with actual labels. BERT is chosen due to its bidirectional training approach, which considers the context from both the left and right of each word in a sentence (Devlin et al., 2018). This enables BERT to achieve a deeper understanding of language nuances and context. Its pre-training on enormous amounts of data makes it highly versatile and effective across a wide range of applications.
- Secondly, transparency obligation practice is uniquely considered from three distinct dimensions, including data, model outcome, and explainability (Key Issue 5: Transparency Obligations - EU AI Act, n. d.). This provides for a systematic explanation and comprehension of the LLM models and their assessment by all user groups. In particular, the explainability of the model decision is performed using the widely used SHAP and LIME frameworks, which provides transparency for the decision-making of the LLM model. Additionally, visual explanation can also enhance the explanation of the model's decision-making through multiple instances of the dataset and illustrate the model's decision points (Maehigashi et al., 2023). In this context, this work uses heat maps to provide further visual elaboration of model decision-making. Both the feature and visual-based explanation mechanisms have their specific contributions, which aid in the practice of the transparency obligation.
- Thirdly, to validate the proposed approach, we conducted an experiment, aimed at rigorously testing our BERT fine-tuned model for vulnerability detection and the assurance of the transparency obligation practice. We fine-tuned our pre-trained BERT model using the DiverseVul Dataset, which contains several instances of C/C++ source code (Chen et al., 2023a). The results of the experiment achieved an accuracy of 91.8 % for vulnerability detection. Additionally, transparency obligations were successfully implemented throughout the entire lifecycle of the LLM model to ensure clarity and interpretability of the model's decision-making.

2. Literature review

LLM-based solutions are recently gaining attention in various domains of cyber security, specifically for the purpose of vulnerability detection (Mathews et al., 2024). This section provides an overview of the recent work and highlights advancements that are relevant to the proposed work.

2.1. LLM based vulnerability detection

LLM-based approaches such as GPT is widely considered in several works. Mathews et al. (2024) explored the efficacy of GPT-4 for vulnerability detection in Android applications, demonstrating that

LLMs can correctly flag insecure apps in 91.67 % of cases in the Ghera benchmark. They introduced the LLB Analyzer, aimed at simplifying the scanning process for Android projects. The study also highlighted the importance of sanitizing data to prevent semantic clues from influencing the model's reasoning. Another work by Gao et al. (2023) aims to introduce VulBench, a comprehensive benchmark that gathers high-quality data from various CTF challenges and real-world applications. The study reveals that several LLMs outperform traditional deep learning approaches to vulnerability detection. Ullah et al. (2023) designed a framework to emulate the step-by-step reasoning process of a human security expert using LLMs, to efficiently detect vulnerabilities in source code. They created a synthetic dataset based on a subset of the MITRE 2022 top 25 most dangerous vulnerabilities. The results indicate that step-by-step reasoning significantly enhances the LLM's ability to detect vulnerabilities, achieving an F1 score of 0.70 and precision of 0.72. Wang et al. (2021) presented CodeT5, a unified pre-trained encoder-decoder Transformer model that better leverages the code semantics conveyed from the developer-assigned identifiers. The study explains that compared to encoder-only and decoder-only models that respectively favour understanding and generation tasks, encoder-decoder models can well support both types of tasks, as shown by CodeT5, which significantly outperforms prior methods on both code-related understanding and generation tasks. Zhou et al. (2024) explored the effectiveness of GPT-3.5 and GPT-4 in vulnerability detection tasks using various prompt designs to enhance model performance. They demonstrated that GPT-4 outperformed CodeBERT by 34.8 % in terms of accuracy, showing that with appropriate prompts, GPT-3.5 and GPT-4 show competitive performance against state-of-the-art models like CodeBERT. Steenhoek et al. (2024) evaluated eleven LLMs' capabilities in vulnerability detection, focusing on prompting methods. Their analysis highlighted significant errors in code understanding, with models frequently misidentifying bug types and failing to distinguish between buggy and fixed versions. The study underscores the need for further refinement in applying LLMs for precise vulnerability detection.

2.2. BERT model for vulnerability detection

Another dimension of work that adopts the BERT-based architecture for vulnerability detection. Shestov et al. (2024) adapted the LLM WizardCoder for vulnerability detection through additional fine-tuning. The team used the LORA method, adapted WizardCoder for classification tasks, and addressed the inefficiency of training by packing multiple functions into each training sequence, reducing padding. The WizardCoder model allows an improvement in ROC AUC from 0.66 to 0.69 in contrast to the existing ContraBERT. Karlsen et al. (2024) adapted LLMs for log analysis for security, focussing on unsupervised anomaly detection. The system architecture for log analysis is composed of three modules: an LLM, an LSTM autoencoder, and Self-organizing maps. The results highlight that GPT-2 and GPT-Neo outperform all other LLMs. Omar (2023) introduced VulDetect, a transformer-based vulnerability detection framework achieved through the fine-tuning of GPT-2 and by exploring the technique of knowledge distillation. GPT-2 is consistently performing the best among CodeBERT and LSTM. Thus, the VulDetect system harnesses the performance of LLMs for automated vulnerability detection with a classification accuracy of up to 92.59 on the SARD dataset. Chen et al. (2023a) proposed a new open vulnerability dataset for C/C++, DiverseVul. According to the study, an encoder-decoder architecture might have an advantage over a decoder-encoder only architecture. They observe that the best LLM achieves an F1 score of 47.2, showing that LLMs are better able to make use of large datasets than GNNs. The study shows that pretraining on code-specific tasks offers large improvements and is more important than the model size. However, the performance of all models on unseen projects decreases significantly to only 9.4 % on unseen projects, illustrating that these models have poor generalization performance. Feng et al. (2020)

present CodeBERT, a bimodal pre-trained model that captures the semantic connection between natural and programming languages with two objectives, i.e., masked language modelling and replacement of token detection. CodeBERT is pre-trained in six programming languages. Results show that CodeBERT performs better than the RoBERTa model using codes only.

2.3. Explainability in the context of LLM models

Finally, there are some works that focus on the explainability of the LLM specifically. Luo and Specia (2024) provided an in-depth review of existing explainability methods, focussing on pre-trained transformer-based models. Their work explores innovative approaches to model editing, control generation, and enhancement to improve the practical application of LLMs. Another work by Zhao et al. (2023) contributed a comprehensive survey on the explainability of LLMs, classifying various XAI techniques into local and global analyses based on two training paradigms: traditional fine-tuning and prompting-based. Thereby, the team explores how these explanations can be used to debug and enhance LLM performance. A subsequent investigation by Zhao et al. (2024) explores the internal mechanics of LLMs using mechanistic interpretability techniques to map knowledge architecture, probe representation to understand knowledge embedding, and investigate training dynamics to clarify phenomena like memorization and grokking. Heyen et al. (2024) explored how DeBERTaV3 model size impacts the quality of explanations provided by LIME, finding that larger models perform better but do not yield more plausible explanations. It calls for improved faithfulness metrics and task-specific evaluation methods, encouraging further research and collaboration in LLM explainability. The study by Maehigashi et al. (2023) adds to the body of research on how humans and AI interact by showing that AI attention heatmaps and AI pointing to these heatmaps make it much easier for people to accept AI suggestions, especially when the tasks are complicated. It offers valuable insights into how visual explanations improve interpretability and trust, emphasizing the need for clear communication in AI systems to foster better user experience and collaboration.

In summary, all these works mentioned above are important contributions towards the adoption of LLMs for vulnerability detection. However, there is a lack of focus on explaining the model decision and the overall transparency obligation practice. In terms of explainability in the context of LLM, existing works only focus on overview of properties relating to explainability and lack of implementation techniques. Lack of transparency obligation practice could lead to inaccurate and biased model outcomes thereby limiting users trust in using AI-based solutions. Additionally, the recent AI regulatory framework, the EU AI Act, also imports the transparency obligation practices for the deployment of trustworthy LLM models. However, ensuring transparency is challenging due to the intricate and complex nature of the LLM models as well as the dataset being utilized. This paper aims to tackle these key limitations using a novel approach by considering several transparency obligation practices throughout the entire life cycle of the LLM model.

3. Transparency obligation practice

The transparency obligation refers to the requirement for organizations developing and deploying LLM models to provide clear and comprehensible documentation of the functioning of these systems, their decision-making, and data processing (Liao & Vaughan, 2023). This obligation makes AI models accountable, fosters trust, and provides users with an explanation and an opportunity to object to the decisions made by those systems. Additionally, it is also particularly important for LLM models to understand the context of the data they process in order to make accurate and reliable decisions. In the context of the European Union's AI Act launched by the European Commission, the transparency requirements represent a key precondition that guarantees the

corresponding LLM models safety, compliance with the law, and sustainability (EU AI Act: First Regulation on Artificial Intelligence, 2023). This implies that there is a need for organizations to disclose information regarding the algorithms being deployed and their inner workings, particularly when the LLM model is involved in making decisions in an automated manner. As illustrated in Fig. 1, we have primarily assessed transparency obligations across three dimensions for this research.

- **Data Transparency:** This dimension involves providing information on how the data was gathered and analysed; this encompasses the specifics of the sources of the data. Data transparency is particularly critical to LLM-based solutions since the quality and source of data impact the effectiveness and precision of such systems (Zhang, 2024). For LLMs, data transparency includes explaining the method used to collect data and preprocess it. This also entails outlining the techniques to be employed for data collection so that the data is collected in an efficient and rigorous manner. However, when dealing with extremely large datasets, LLMs face the challenge of increased computational power and longer processing times (Chen et al., 2023a). These factors highlight the importance of proper strategies for data management and sufficient support for the proper functioning of LLMs to handle large volumes of data and to learn from them for enhanced model performance and accuracy. It also requires the description of preprocessing activities, including data cleaning aimed at removing noise or errors and data creation in circumstances whereby augmented instances similar to what is likely to be given to the model in the training phase are generated (Joshi & Bhardwaj, 2018). These preprocessing steps are crucial in enhancing the quality and relevance of the data, as they affect the LLM's performance.
- **Model Outcome Transparency:** The concept of model outcome transparency in relation to LLM models means providing clear documentation of the outcomes generated by these models. This transparency is particularly important if the stakeholders or the users of the LLM are to understand and accept the decision or the prediction made by the models (Laoutaris, 2018). It concerns offering interpretations of the results produced by the LLM to ensure that stakeholders understand the predictions or decisions made by the models. Moreover, the use of performance metrics including accuracy, precision, recall, and F1 score is crucial in determining the dependability of the generated output by the LLM (Brown, 2024). For accountability, it is vital to document and report all the results of the model when changes were made during training, testing, and deployment of the model (Raji et al., 2020). In addition, the explicit identification and reporting of bias in the LLM's outputs through a fairness audit prevents the systems from generating decisions that negatively impact some groups more than others.
- **Explainability:** Explainability in the context of LLM is the ability to make the decisions of the LLM models easily understandable to the users or clients. It includes the means of communicating the decision-making and the results of LLM systems to the intended receiver in a manner that is comprehensible and credible to them (Basheer et al., 2024). This is important as it allows for governance and auditing of the model and also creates trust as the user is able to understand why the decision was made. One of the most popular frameworks for enhancing explainability is SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) (Kumarakulasinghe et al., 2020; Li, 2022). SHAP values indicate the importance of each feature, which demonstrates how each contributes to improving or degrading the model's performance. Moreover, SHAP provides feature importance that describes the relevance of each feature for the whole dataset and local importance that describes the influence of each feature for a particular instance. In terms of vulnerability detection, SHAP can pinpoint which parts of the code are considered vulnerable; thus, users and developers can rely on the model and its output. On the contrary, LIME works by

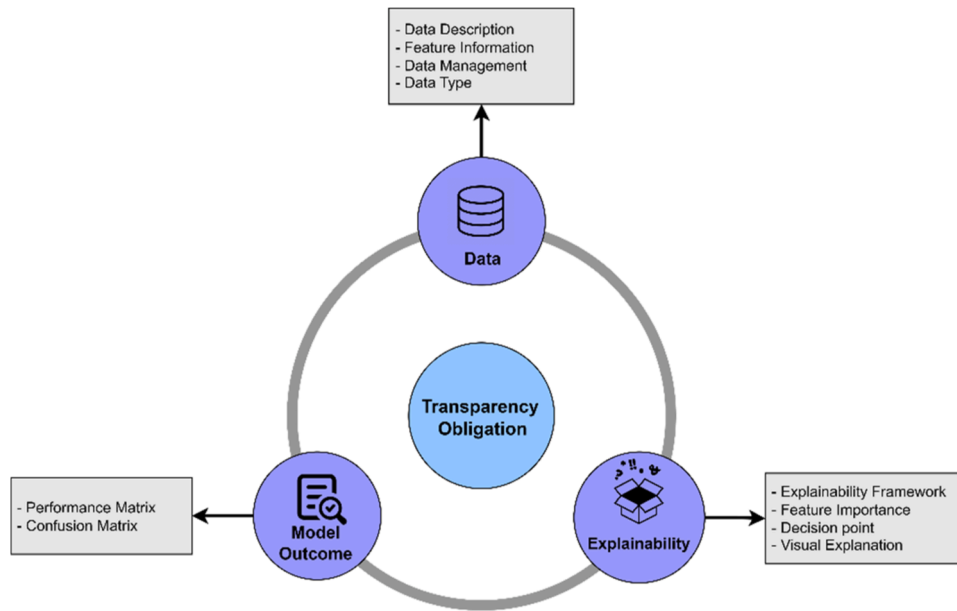


Fig. 1. Transparency obligation dimensions.

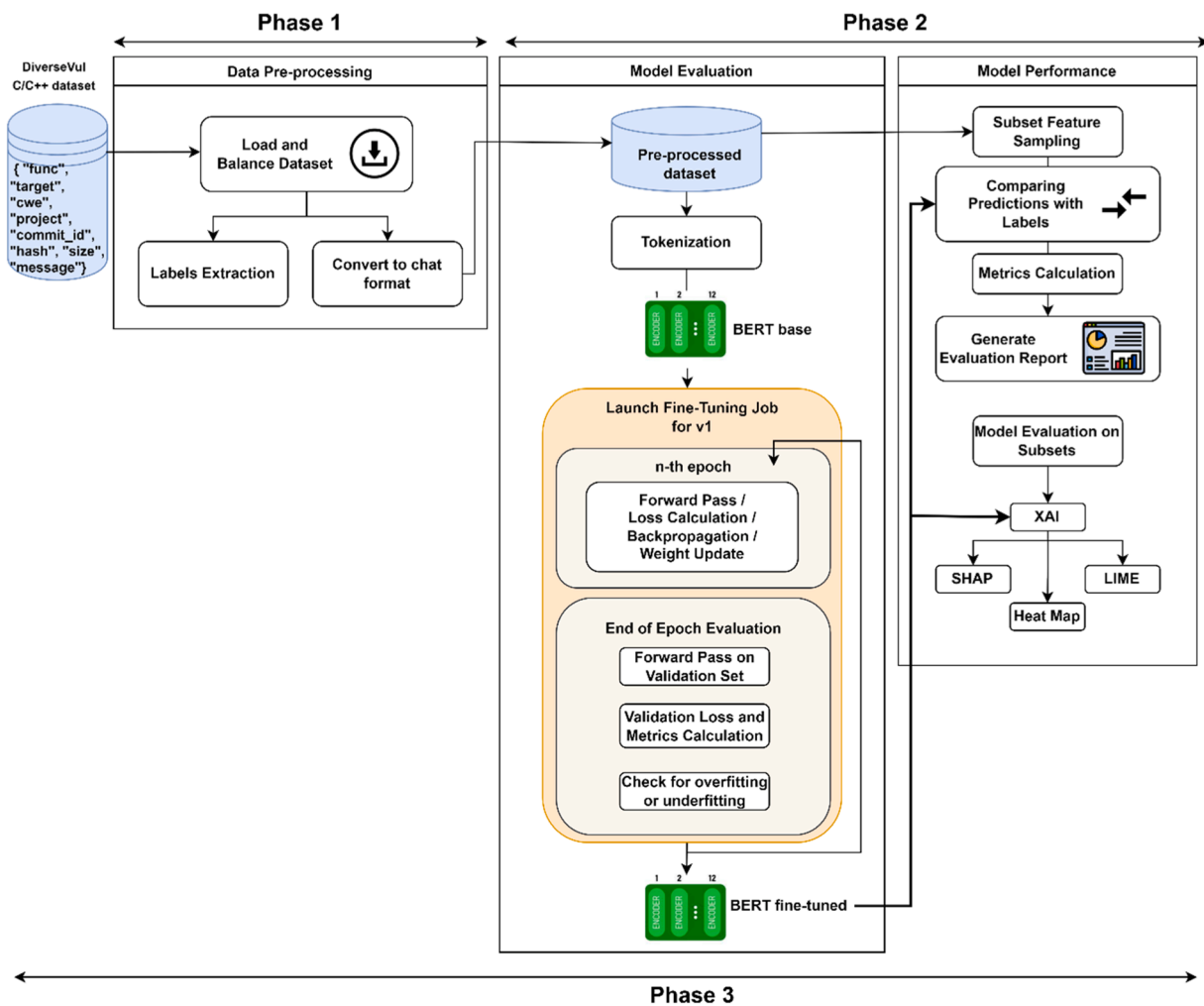


Fig. 2. Bert based architecture for vulnerability detection with transparency obligation.

perturbing the input data and fitting a simpler interpretable model locally to explain a particular prediction. LIME generates explanations by observing how changes to individual features affect the output, which helps users understand how the model makes decisions at the instance level. Additionally, the proposed approach considers heatmaps, which provide a visual representation of the contribution of each feature to the model's output. Heatmaps can highlight patterns and regions of the input that the model focuses on, offering a holistic view of the model's decision-making process (Muddamsetty et al., 2021). This combination of SHAP, LIME, and heatmaps enhances transparency and interpretability, building trust in the model's decisions.

4. The proposed architecture

The proposed approach presented in this section aims to enhance vulnerability detection using the LLM model along with the assurance of the transparency obligation practice. For this purpose, we have adopted the BERT model. The reason for considering BERT is that it sets itself apart from other large language models through its bidirectional training approach, which takes into account the context from both the left and right sides of each word in a sentence (Devlin et al., 2018). This makes it possible for BERT to gain a deeper understanding of the language as well as its context. In addition, due to the large amount of data that it was trained on, it is highly flexible and performs well on a wide variety of tasks.

The proposed architecture, as shown in Fig. 2, consists of three sequential phases: data pre-processing, model evaluation, and transparency obligation. The final phase of Fig. 2 covers all aspects of transparency obligations, as illustrated in Fig. 1. Each of these phases consists of several steps that describe the specifics of the techniques performed in the given phase. Data pre-processing is the first and one of the most critical stages in the architecture. This phase helps in preparing the raw data for the analysis and modelling phases by purging it. It comprises processes like data cleansing, data transformation, and data organisation to improve data quality and meet the requirements of the next phases. The second phase is the model evaluation, which starts with the tokenization of the pre-processed data to be fed into the pre-trained BERT model. Following this is the fine-tuning process, where the data is fed to the model in small portions called batches. This batching allows the model's parameters to be optimized after each batch, enhancing its overall performance. This interleaved process helps to get the model properly fine-tuned for the specifics of the vulnerability detection task. The final phase is the transparency obligation implementation, which encompasses three dimensions: data, the outcome of the model, and its explainability, as described in Section 3. This phase also ensures that the origins and quality of the data are clearly documented, the results produced by the model are comprehensible, and the decision-making process of the model is explainable.

4.1. Phase 1: data pre-processing

The first process of the architecture is data pre-processing, which enhances the quality of the raw data by making it usable through cleaning and formatting. This step also involves the derivation of labels from the raw data, which is important, especially in the case of supervised learning. Data also needs to be pre-processed into the needed format (for example, chat format for language models) to match the process of model training. Additionally, pre-processing entails techniques for handling class imbalances, such as under sampling, to ensure that the model can learn effectively from all classes (Bach et al., 2019). In the case of an imbalance in the classes, the performance of the model often deteriorates as the majority is more frequently represented in the training data. As a result, the model may achieve high accuracy overall, but it will have low precision and recall for the minority class. This can be especially disastrous when the primary objective of the classification

is to isolate the minority class, as in the case of fraud or disease detection. To balance the classes, methods like under sampling are used (Mohammed et al., 2020). In cases of under sampling, the number of examples from the majority class is decreased to the number of instances from the minority class. This involves taking a small sample of the majority class and adding it to the dataset with the samples of the minority class. By pre-processing the data to address class imbalance and converting it into a suitable format, we ensure that the model learns effectively and performs well across all classes.

4.2. Phase 2: model evaluation

The original data is digested and transformed into a list of dictionaries containing the text and labels as fields. All these texts and labels are combined into a new data frame, which is transformed into a Hugging Face dataset. The texts are truncated and padded to ensure a consistent input size. The tokenized dataset is then split into training and test sets with a ratio of 80–20 split, where the test set is further divided into a validation set and a test set with a 50–50 split. This pipeline processes the text data for modelling by balancing the data, pre-processing, tokenizing, and splitting the data into train, validation, and test sets for model testing. Subsequently the model was tuned.

4.2.1. Model fine-tuning

Fine tuning typically starts with a model that has been trained on a large dataset, often in a general domain, such as image recognition or language understanding. The key steps in fine-tuning include adjusting the final layers of the model to make them more relevant to the specific task, while freezing the earlier layers to retain the learnt features that are generally applicable across tasks. Suppose we have transferred the knowledge from our model 'm1' to our model 'm2'. It is important that the learning of 'm2' on tasks specific to 't2' does not cause it to forget the knowledge transferred from 'm1'. To avoid this issue, we prevent any modification of the weights of the first layers of neurons. Only the last layer(s) can be modified to specifically learn how to solve the task 't2'. During the learning phase of 'm2', the weights of the unfrozen layers are updated according to the Adam Optimizer method (Kingma & Ba, 2014). This method regulates the learning rate based on the first and second moments of the gradients for each parameter. It is a crucial point because if the learning rate is set remarkably high, then the model can go out of the basin of attraction and the algorithm will never converge to a minimum; similarly, if it is set too low, then convergence will be slowed down. The learning rate is lowered during fine-tuning to make smaller changes to the model so as not to overfit on the new, smaller dataset.

During the fine-tuning phase, it is important to pay attention to hyperparameters. We have already discussed the learning rate. Another hyperparameter is the number of epochs. An epoch represents a complete cycle of passing all the training data through the model. During an epoch, the model adjusts its parameters to reduce the error between its outputs and the expected outputs. If a few epochs are performed, there is a risk of underfitting: the model does not have enough time to learn the new data. If many epochs are performed, there is a risk of overfitting: the model memorizes the specifics of the training data and does not generalize well (Ying, 2019). To determine the number of epochs, we monitor the model's performance on the valuation set and train it until no improvement on the set is noticed.

Firstly, we use BERT (Devlin et al., 2018) as the starting point for our fine-tuning. "Google-bert/bert-base-uncased" is one of the most common versions of BERT (Bidirectional Encoder Representations from Transformers) created by Google (Google-bert/Bert-base-uncased · Hugging Face, 2001). BERT stands out from other transformers due to its ability to understand the context in which it trains the transformers bidirectionally. This model is essentially trained to have extremely high competency in terms of natural language processing, as it was trained on a large dataset. The general configuration for BERT-base entails 12 layers, 768 hidden sizes, and 12 attention heads; it has 110 million

parameters (Google-bert/Bert-base-uncased · Hugging Face, 2001). Applying this model to sequence classification involves putting a classification layer on top of BERT's architecture, and this makes it easier to classify sequences into a given set of classes. This modification makes use of BERT's deep understanding of language for accomplishing specific classification processes. BERT was used for the experiment owing to the fact that it has numerous benefits when it comes to NLP. It is especially efficient in two-way context dependency and translates the meaning of the given sequence of words as a whole, not word by word. BERT is trained on a large dataset; hence, it masters the patterns, syntax, and semantics of the language. Due to its ability to adjust with a relatively small set of data pertaining to a specific task (Devlin et al., 2018), it is applicable to multiple NLP problems such as text categorization and named entity identification.

4.2.2. Model performance evaluation

After fine-tuning, the model is evaluated using the test set, which was not applied during the fine-tuning. This is useful in evaluating the extent to which the model can generalize to new, unseen data. There is a set of measures frequently used to assess the model's quality, especially in classification tasks, such as accuracy, precision, recall, and F1 score.

- Accuracy represents the ratio of correctly predicted instances to the total instances:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Instances}} \quad (1)$$

- Precision represents the ratio of true positive predictions to the total predicted positives:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (2)$$

- Recall represents the ratio of true positive predictions to the total actual positives:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (3)$$

- The F1-score is the harmonic mean of precision and recall:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

Additionally, the utilization of confusion matrices will aid in the visualization of the model's performance across different classes. Confusion matrices are used to classify the true positive, false positive, true negative, and false negative predictions to provide details regarding where the model works best and where it has issues. This type of information is useful in refining the model and correcting areas that require enhancement in order to arrive at a reliable and more accurate predictive model.

4.3. Phase 3: transparency obligation

To achieve the transparency obligation within the proposed approach, we ensure that the decision-making process of the model is explainable and understandable by users. For this research, only the three dimensions mentioned in Section 3 are taken into consideration.

- The first step of the phase is the transparency of the data, which ensures clear and comprehensive detail on how data is gathered, processed, and used, which ensures that the information is accurate and available. Data transparency also provides information about the datasets, which includes their size, attributes, formats, sources, and limitations, thereby enhancing stakeholders' informed decisions. This transparency builds trust among the users as it enhances informed decision-making. It helps to avoid data misuse and allows stakeholders to check and comprehend the data utilized.

- The performance evaluation metrics such as accuracy, precision, recall, F1 score, as well as the confusion matrix as discussed in Section 4.2.1, enhance the understandability of model outcomes by providing a detail about the model's performance. They enable the stakeholders to determine the accuracy of the model, the proportion of successful positive predictions, the ability of the model to correctly classify positive cases and also provide the strengths and the weaknesses of the model. The above metrics provide a detailed analysis of the areas that are well captured by the model and the areas that require enhancement hence promoting trust, transparency, and informed decision-making.
- Finally, to increase the explainability of the model, we have utilized the SHAP and LIME frameworks and heat map. It improves the transparency of model outcome by offering specific and comprehensive means of interpreting the effect of every input variable to the stakeholders and improve their confidence in the model's outputs. Both SHAP and LIME framework allows to further enhance the explainability of the model.

The SHAP value for a feature j in a prediction is calculated using the formula as shown in Eq. (5):

$$\phi_j = \frac{\sum_{S \subseteq N \setminus \{j\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [f(S \cup \{j\}) - f(S)]}{|N|!} \quad (5)$$

where N is the set of all features, S is a subset of features without j , $f(S)$ is the model output without feature j , and $f(S \cup \{j\})$ is the model output with feature j . This formula iteratively computes the contribution of feature j by comparing the change in prediction with and without the feature across all possible combinations of other features.

Notably, the SHAP provides both global and local importance making the interpretation of the model easier. Global importance, in general, indicates the significance of each feature throughout the analysis of the dataset to see how well the chosen features force the model in practice (Basheer, Pranggono et al., 2024). In contrast, local importance centers on the individual predictions and provides an interpretation of why each feature is relevant to a specific instance to include details about how the decision of the model was made (Basheer, Pranggono et al., 2024).

In the case of LIME, it involves the development of a simpler model, $(g(x))$, which approximates the behaviour of a complex model in the local environment of an instance. This is achieved by minimizing the objective function as shown in Eq. (6).

$$\xi(x) = \underset{g \in G}{\text{argmin}} \mathcal{L}(f, g, \pi_x) + \Omega(g) \quad (6)$$

where $\mathcal{L}(f, g, \pi_x)$ measures the difference between the predictions of the complex model f and the simpler model g , with π_x assigning more weight to points closer to the instance being explained. The term $\Omega(g)$ ensures that the simpler model remains interpretable by penalizing complexity, and G represents the class of interpretable models, such as linear regression. This approach enables LIME to come up with explanations that are locally accurate of the original model, yet simple for the stakeholders to comprehend.

We also employ a heat map as the feature importance plot to show the significance of each feature on the developed model's prediction (Maehigashi et al., 2023). These heatmaps give the stakeholder a clear feel of the parts of the input that are most concentrated by the model and thus which patterns or features are considered important. Such an approach, supplemented by numerical descriptions, helps to get the most complete vision of the nature of the model's interpretability.

5. Experiment

This section presents the detailed about the experiment and results obtained from the experiment to demonstrate the applicability of the proposed approach. Hence the main aims of the experiment are:

- To evaluate the effectiveness of BERT model in vulnerability detection
- To implement the transparency obligation practice through the entire LLM life cycle

5.1. Dataset description

The experiment is conducted using DiverseVul dataset (Chen et al., 2023b), which consists of a large set of C/C++ source codes. The dataset is derived from CVEFixes dataset, and it contains the respective CVEs described in the publications up to August 27, 2022.

Key features of the DiverseVul dataset:

- The dataset comprises of 18,945 vulnerable functions spanning over 155 CWEs and 311,547 non-vulnerable functions derived from 7514 commits of several projects.
- Each vulnerability is described by detailed characteristics such as function snippet, target status, CWE identifier, project name, commit ID, hash, size, and descriptive message.
- The dataset contains C/C++ source code instances whose tokens are characterized in detail, including function pointers, macros, and buffer manipulation functions like `strcpy` and `sprintf`, which are often associated with security vulnerabilities.
- The dataset was collected by crawling security issue websites and extracting both vulnerability-fixing commits and corresponding source code from various projects.

5.2. Phase 1: data preprocessing

Data pre-processing is the initial process of the architecture proposed. It encompasses cleaning of data and converting it into a format that is easily usable while meeting the quality standards for analysis and modelling. This makes it necessary to balance the classes in the DiverseVul dataset in order to avoid the training of a model that is biased to the over-representing class. It has a total of 349,437 instances where 18,945 are vulnerable and the rest is non-vulnerable. Fig. 3 shows a significant imbalance between the number of vulnerable and non-vulnerable instances in the dataset with two categories as shown below. This necessitates the implementation of under-sampling method to handle imbalanced data to develop effective and unbiased LLM models.

Fig. 4 illustrates the results after the implementation of the under-sampling technique to balance the dataset, where in the beginning, there was a significant difference between the numbers of vulnerable and non-vulnerable instances. As a result, the number of instances in both the vulnerable and non-vulnerable groups is equal i.e. 18,945 instances, thereby reducing the skewness of the majority class discovered

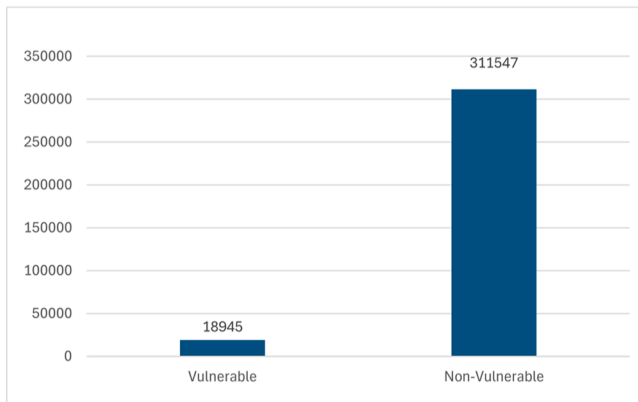


Fig. 3. Class distribution before under-sampling.

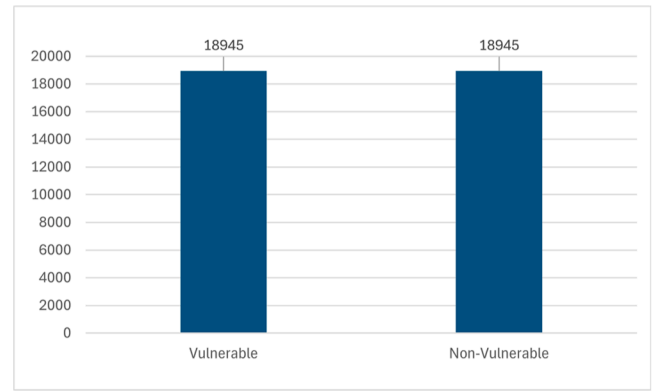


Fig. 4. Class distribution after under-sampling.

earlier. This balanced dataset is now more suitable for LLM models that require an equitable representation of both vulnerable and non-vulnerable instances.

5.4. Phase 2: model performance evaluation

After the pre-processing of the data, the next phase is the performance evaluation of the BERT model. At this stage data splitting is required into the training and the testing sets. Such differentiation is necessary in order to prove the effectiveness of the BERT model under realistic conditions. Once the training set has been acquired, one can proceed with the training of the model. Thus, the fine-tuned BERT model was trained through ten training epochs to enhance the key metrics of the model. Initially, the model exhibited a training loss of 61.84 % and a validation loss of 56.51 % during the first epoch, with an accuracy of 67.27 %, as shown in Table 1. Based on these early measures, we could see that the model was starting to learn well from the data provided. By the fifth epoch, significant improvements were observed: the training loss dropped to 30.03 %, and the validation loss decreased to 27.45 %. The accuracy rose to 78.56 %, and performance metrics such as the F1-Score, precision, recall, and ROC AUC improved to 78.91 %, 82.72 %, 75.44 %, and 81.25 %, respectively. In the last epoch, the model exhibited its best performance with an accuracy of 92.19 % with a primary training loss of 12.14 % and a validation loss of 14.04 %. This demonstrates the overall improvement throughout the training process.

Following the training phase, the accuracy of the developed model was tested on the test set in order to evaluate its generalization ability. The findings of the evaluation phase, shown in Table 2, attest to the effectiveness of the model. The accuracy reached 91.89 %, indicating that the model correctly classified the vast majority of instances. The F1-Score, which balances precision and recall, was 87.90 %, highlighting the model's strong performance in handling both false positives and false negatives. The model also exhibited a precision of 91.99 %, signifying that the proportion of correctly predicted positive instances was very high. Additionally, the recall was 84.15 %, confirming that the model effectively captured a large portion of the true positive instances. Lastly, the ROC AUC of 91.46 % demonstrates that the model has strong overall discriminatory power, performing well across different classification thresholds. From these outcomes, it could be observed that the fine-tuned BERT model not only achieved good results during training but also demonstrated high efficacy in unseen data evaluation, (Fig. 5).

The confusion matrix represented in Fig. 6 illustrates the performance of the fine-tuned BERT model in identifying vulnerable and non-vulnerable code. The model was able to accurately flag 1116 vulnerable codes (true positives) and 2366 non-vulnerable codes (true negatives). However, it classified 97 samples of non-vulnerable code as vulnerable (false positives) and another 210 samples with vulnerable code as non-vulnerable (false negatives). This suggests that though the model has high accuracy in identifying the vulnerable and non-vulnerable

Table 1
Training phase metrics.

Phase	Epoch	Training Loss	Validation Loss	Accuracy	F1- Score	Precision	Recall	ROC AUC
Training	1	0.618397	0.565131	0.672740	0.681341	0.712800	0.652541	0.689293
	2	0.522058	0.486810	0.705315	0.711654	0.744286	0.681764	0.716598
	3	0.448365	0.423881	0.731733	0.737993	0.774412	0.704846	0.749337
	4	0.393824	0.331696	0.757498	0.761657	0.798145	0.728360	0.779905
	5	0.300338	0.274585	0.785616	0.789133	0.827183	0.754429	0.812591
	6	0.232790	0.214687	0.820131	0.818662	0.854518	0.785613	0.842395
	7	0.166873	0.157415	0.849875	0.848472	0.885842	0.814127	0.867583
	8	0.118243	0.139777	0.875998	0.876952	0.916685	0.840521	0.896347
	9	0.118614	0.136972	0.905352	0.894504	0.931131	0.860649	0.925125
	10	0.121389	0.140472	0.921903	0.894655	0.931067	0.860984	0.931029

Table 2
Evaluation phase metrics.

Phase	Accuracy	F1-Score	Precision	Recall	ROC AUC
Evaluation	0.918903	0.879013	0.919922	0.841587	0.914673

instances, as highlighted by the high true positive and negative rates.

5.5. Phase 3: transparency obligation

As stated in [Section 3](#), the proposed transparency obligation is considered three dimensions, and this section provides the implementation of these dimensions. Note that, data dimension is implemented during phase 1, while model outcome and explainability dimensions are implemented in the phase 2 of the proposed approach.

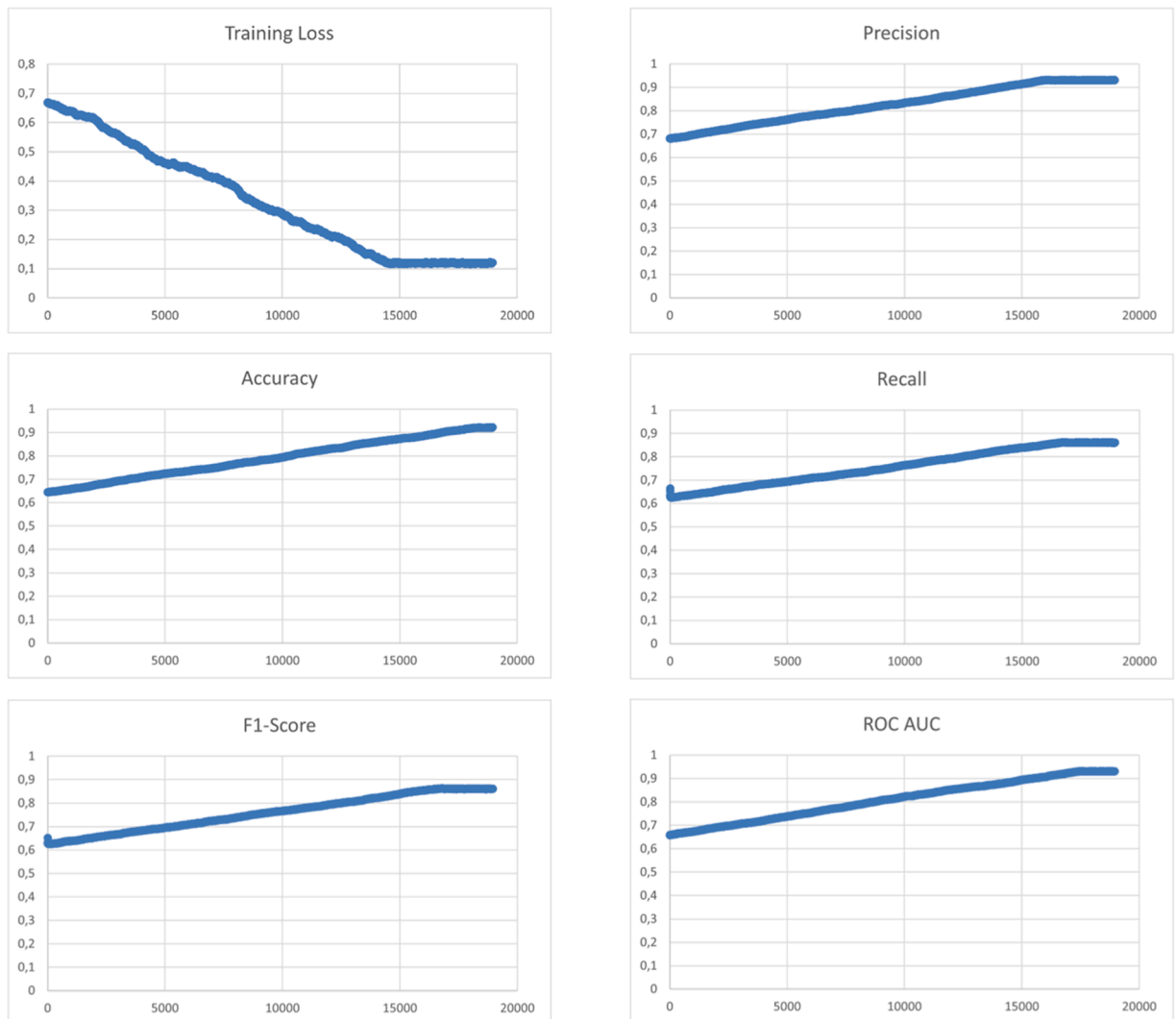


Fig. 5. Evolution of performance metrics for fine-tuned BERT model.

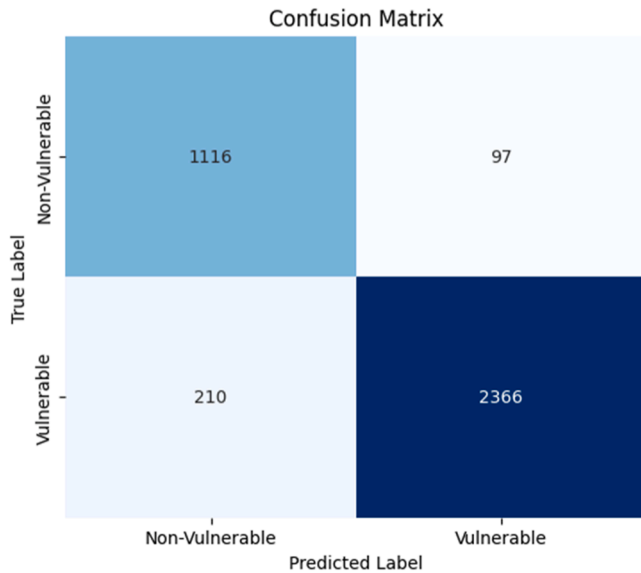


Fig. 6. Confusion matrix for fine-tuned BERT model.

- **Data:** Data transparency is a vital dimension as it provides the users or stakeholders with the required information about the origin and type of the dataset, along with details about its attributes and the number of the instances. Section 5.1 offers broad descriptive information about the dataset by describing the several types of features, their distribution within the dataset, and the existence of any possible biases. The dataset contains 18,945 vulnerable functions that cover 155 distinct CWEs, as well as 311,547 non-vulnerable functions collected from 7514 commits across many projects. This information can help stakeholders assess the reliability and trustworthiness of the LLM model and its outcomes. Additionally, it also ensures the suitability of the chosen dataset for the intended application, thereby ensuring its accountability in the model result.
- **Model Outcome:** We made use of the BERT model for the training purpose of the proposed approach due to its ability to understand the context of the data from both left and right side of a given sentence. From the data in Table 2, it can be seen that the accuracy of the proposed model is equal to 0.693851. The evaluation loss was 0.599968, with the ROC AUC of 0.691847. Based on these results it can be concluded that while the model is capable of dealing with unseen data, however, it performs poorly when it comes to generalizing. The relatively modest improvements in accuracy and other metrics indicate that the model requires significant enhancement before it can be depended upon for reliable vulnerability detection. From Fig. 6, we get the results of the confusion matrix. The model correctly identified 1504 cases of vulnerable code and 1125 cases of non-vulnerable code. But it marked 724 samples of non-vulnerable code as vulnerable code and 436 samples of vulnerable code as non-vulnerable code. This means that while the model has good accuracy in identifying the vulnerabilities, there are many misclassifications made by the model.
- **Explainability:** SHAP is a method used to explain the predictions of machine learning models. It is based on Shapley Values from cooperative game theory (Lundberg & Lee, 2017). It aims to understand the impact of each feature on the model's predictions. Thus, it assigns a contribution value to each feature. Each feature can either contribute to increase or decrease the model's predictions. We have also considered LIME as another technique for explaining our model outputs. SHAP offers another coherent, theoretically founded approach to the explanation of global and local model predictions, whereas LIME gives qualitative explanations by approximating complex models locally in addition to single predictions. Moreover, a

heat map has also been incorporated to improve the visualization of the decision-making process of the model.

When working with LLMs we employ the model-agnostic version of SHAP: Kernel SHAP, as it can be applied to any machine learning model regardless of its internal working. Kernel SHAP uses a linear regression model to approximate Shapley values. Model agnostic SHAP methods consist in perturbing the input features and observing the changes in the output. SHAP can provide local explanations to show the impact of each feature on individual predictions. This involves defining all possible combinations of the features and measuring the marginal contribution of a specific feature in these interactions. It can also give a global explanation to express the importance of the features within a dataset. In this case it will accommodate the average of the absolute Shapley Values of the feature across all instances. This aids in identifying which features are important for making the model's decisions.

In Fig. 7, we observe that the word “vulnerable” strongly contributes to the model predicting that the code is vulnerable. This is not necessarily a desirable behaviour because attackers could then deceive the detection system by using words like “safe” or others. Some contributions calculated by the model are perplexing, and even with simpler examples than those extracted from the DiverseVul dataset, it is difficult to understand why certain words contribute positively or negatively to a given prediction. Indeed, SHAP sometimes provides contributions that are not very intuitive.

Fig. 8 represents the contributions of each token to the prediction: “the code is not vulnerable”. Consistent with what was stated for Fig. 7, the word “vulnerable” contributes to decreasing the probability that the code is not vulnerable. Not only can focusing on the meaning of words be very misleading, but even if it were informative, the model treats the token “vulnerable” the same way whether it is preceded by nothing or by “not” (“not vulnerable”). Its interpretation is then incorrect. The results of our experiment show that the model is overly sensitive to the meanings of words, particularly those used as function names or variable names. These tend to have a predominant contribution compared to the rest of the words used in the code. This can be problematic. Indeed, in a situation where a function is named ‘not vulnerable,’ the model could be wrong. In this context, where the model provides incorrect information, to enhance our system from these kinds of false predications, we suggest the use of a large and diverse dataset so that the model does not over-emphasize such correlation data. When using this enriched dataset to fine-tune the model, it generalizes to different coding scenarios, particularly mitigating misdirection from token patterns such as “not vulnerable.” Also, the human-in-the-loop approach provides a way to verify doubtful predictions, increasing reliability.

Fig. 9 represents the LIME explanations for the influence of various features on the model's prediction. Features are displayed as bars, where the direction (left for negative, right for positive) indicates whether they push the model's decision towards one class or another, and the length signifies the magnitude of their impact. Positive weights are the features that help the predicted class, negative weights are the features that hinder it. From the graph, the “max” feature shifts the model's prediction predominantly to Class1, meaning that it has a strong positive impact. On the other hand, the “mysql_tmpdir_list” feature is strongly negative, meaning that predictions will be moved away from Class 1 in favor of Class 0. This graphical representation proves to be useful in interpreting the decision-making process of most machine learning models to simplify, debug, as well as improve on them. It also improves interpretability, which is a necessary component in assessing the accuracy of model results.

Fig. 10 depicts the heatmap for attention weights of the tokens in a source code snippet in terms of how one token contributes or receives an attention from another token. Most of the attention is mainly aligned along the diagonal, which means that tokens pay much attention to themselves or their neighbours, which suggests that the context is local. Bright spots off the diagonal suggest significant interactions between

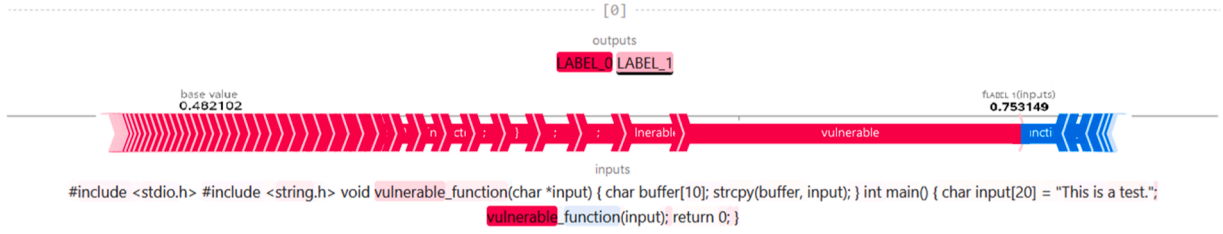


Fig. 7. Token contributions to vulnerability predictions.

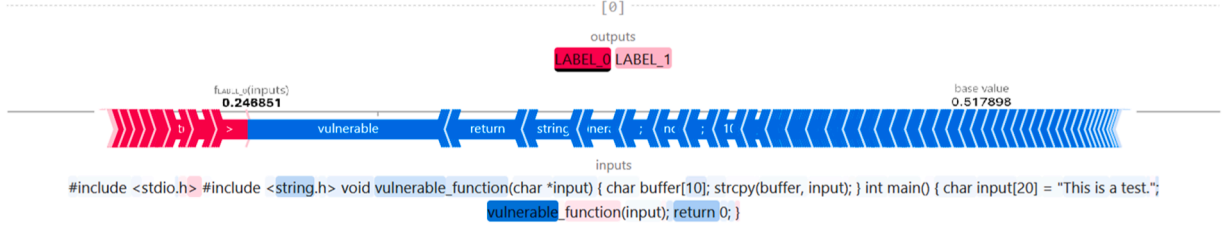


Fig. 8. Token contributions to non-vulnerability predictions.

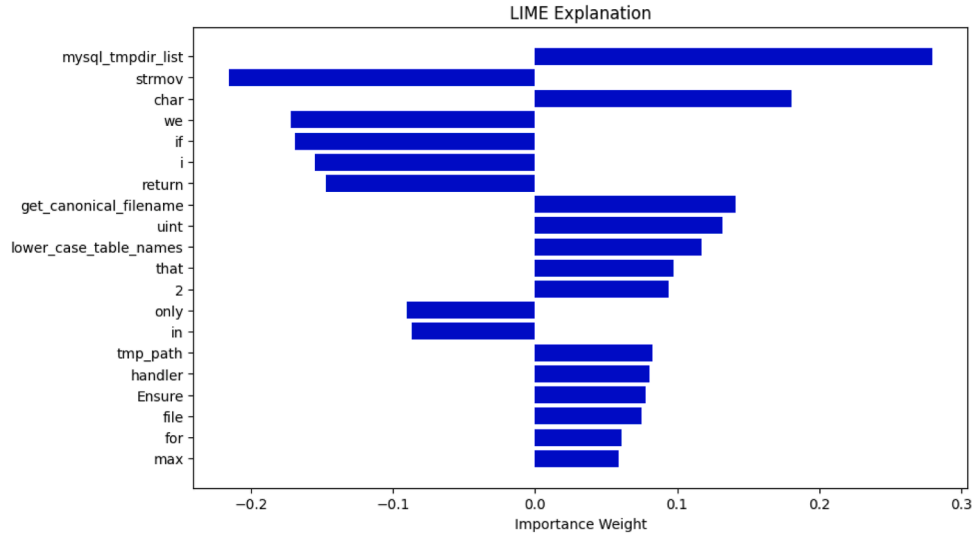


Fig. 9. Feature importance using LIME.

distant tokens, potentially identifying security-sensitive patterns or pivotal points in the code logic. Darker areas signify minimal attention, indicating less relevance to the task at hand. Such visualizations help in pinpointing key areas of focus in the model's decision-making process, useful for debugging or enhancing model interpretability, especially in tasks like security analysis where understanding context and token relationships is crucial.

6. Discussion

In software development, detecting vulnerabilities in source code is vital for the safe production of the end product. However, the size of the source code is considerably large, making it challenging to train models to detect these vulnerabilities. One aspect that makes it difficult is the highly diversified and complex coding patterns in use, requiring sophisticated tools to perform the analysis effectively to detect threats. This is where LLMs exhibit high potential because of their efficient feature of processing context between tokens in the input. By training LLMs to read and analyse enormous codebases, they can easily identify various patterns in the code indicative of security vulnerabilities.

6.1. Suitability of LLM based approach for vulnerability detection

The adoption of LLM, specifically the BERT model, in the proposed approach has a prospect that other architectures of machine learning had failed to offer. They are particularly effective in contextual understanding, whereas traditional methods mostly use simple pattern matching or analyse code line by line. Besides, they are pre-trained on a vast amount of data: documentation, code snippets, and various programming languages. So that they can learn a wide variety of coding patterns, enabling them to generalise better. In addition to this, LLMs also present the option to be fine-tuned, which simply means that they can be trained to identify particular vulnerabilities. Finally, they have an opportunity to keep on updating the new vulnerabilities that are likely to arise hence being relevant on security threats.

This work contributes to advancing the vulnerability detection approaches. It utilizes state-of-the-art LLMs, which represent the latest advancement in natural language processing. The models enable a more sophisticated understanding and detection of vulnerabilities within code. Such an advanced detection system is more robust than traditional detection systems based on pattern matching.

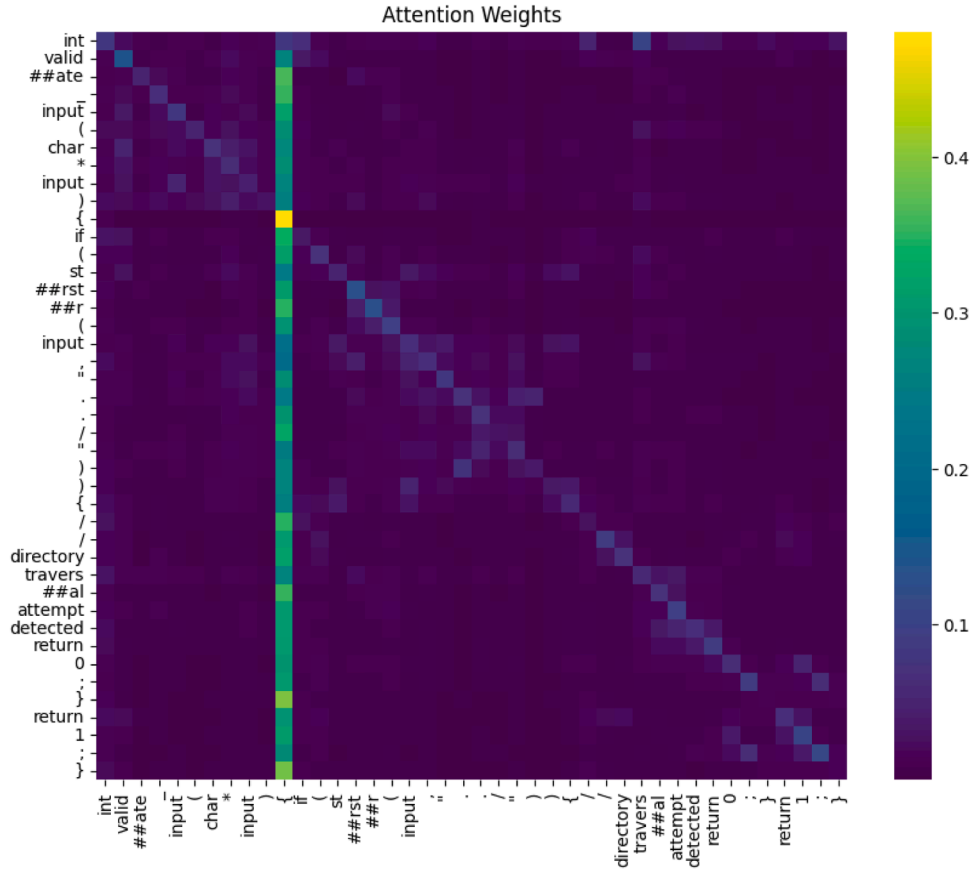


Fig. 10. Heatmap of attention weights.

Our work specifically utilizes BERT, a sophisticated LLM known for its particularly good natural language understanding capabilities. A distinct feature of our approach is the emphasis on explainability. We prioritize not only the detection of vulnerabilities but also the transparency of the model's decision-making process. Unlike traditional methods that may treat code as a whole, our use of SHAP allows analysis into the granular details, understanding how each token influences the model's predictions.

6.2. Adoption of transparency obligation practice towards trustworthy AI

During the training phase, a snippet of code, whether vulnerable or not, is provided to our model, and we then adjust the weights by comparing the model's prediction with reality. However, the code snippets given to the model are long, and our fine-tuned model seems to struggle with identifying which part of the code is responsible for the vulnerability, when there is one. The training could be optimized by precisely detailing which lines are the source of the security flaw. However, according to current knowledge, only humans could perform this very time-consuming task. We also encountered an issue with the context window size of the BERT model. The model accepts up to 512 tokens as input, but a considerable number of code snippets exceeded this maximum size. Therefore, if the vulnerability were located after the first 512 tokens, it was not provided to the model, and the learning process could not occur but was also disrupted.

As detailed previously, we chose to explain our model's decision-making using SHAP, LIME, and heatmaps for attention weights. Explaining source code is extremely challenging because, unlike sentiment analysis on natural language, where individual words often carry meaning, the components of programming languages do not inherently indicate the presence of a vulnerability. Instead, it is the structure, articulation, and sequence of these elements that can introduce

vulnerabilities. A key challenge arises from the probabilistic nature of our model: if a word is frequently associated with vulnerabilities, the model may predict a flaw every time it encounters that word, even if it doesn't signal an actual issue.

The benefits of SHAP include transparency, and trustworthiness of AI models, enabling users to understand the decision-making process and identify the most influential features driving the predictions. Additionally, SHAP explanations can help detect biases, improve model performance, and facilitate feature engineering by highlighting which features have the most significant impact on the model's output. In addition, LIME complements SHAP by providing localized explanations for specific predictions. It creates interpretable approximations of complex model decisions, allowing us to gain a deeper understanding of individual instances of code and how certain tokens affect predictions in those cases. Finally, heatmaps for attention weights give us a visual representation of where the model focuses when making decisions, showing how certain tokens or sequences influence the prediction. This visual layer helps clarify the model's internal attention mechanisms, making the decision-making process more transparent and interpretable.

6.3. Work in context

This section compares the outcome of experiment with the relevant existing works. The overall result from our experiment shows approximately 91.89 % accuracy with a precision of 92.00 %, a recall of 84.16 %, an F1 score of 85.85 %, and a ROC AUC of 91.47 % in vulnerability detection. The work by [Chen et al. \(2023b\)](#) achieved an accuracy of 91.68 %, a precision of 46.02 %, a recall of 28.22 %, and an F1-score of 34.98 % using RoBERTa model which is based on BERT model with DiverseVul dataset. Another work by [Shestov et al. \(2024\)](#) for vulnerability detection also achieves a ROC AUC score of 69.00 % and an F1

score of 71.00 % using BERT based WizardCoder approach using three different datasets CVEfixes, a Manually Curated Dataset and VCMATCH. Comparing between our work and A. Shestov et al. we obtain better ROC AUC and F1-score. Steenhoek et al. (2024) contribution relating vulnerability detection using prompting method achieved an accuracy score of 56 % using the SVEN source code dataset which was collected from CVE records reporting real-world vulnerabilities from the Cross-Vul, BigVul, and VUDENC datasets. Yang et al. (2023) in the context of vulnerability detection using source code achieved an F1 score of 92 %, a precision of 93 %, and a recall of 91 % using the BigVul dataset by concatenating embeddings from an instruction-tuned LLM with a Graph Neural Network for classification. Their prompting approach incorporates different aspects of vulnerability explanation to enhance the detection accuracy of a pre-trained LLM. However, this work did not consider model fine-tuning, instead using prompting to provide the model with more information. Zhou et al. (2024) explored the use of vulnerability-fixing commit datasets. To extract vulnerable functions, they analysed software versions prior to vulnerability-fixing commits and marked functions with patched lines as vulnerable. Other functions in files altered by the commit were labelled as non-vulnerable. They evaluated GPT-4 which achieved an accuracy of 75.5 %, a precision of 73.7 %, a recall of 79.3 %, and an F1 score of 76.4 % using code examples from CWE types. In comparing all these works, we have obtained a slightly higher accuracy due to performing the experiment with more iterations which allows the model to undergo deeper learning. Moreover, the model accuracy also relies on the context of the datasets, model types and related algorithm.

In the context of explainability, we have combined both feature and visual analysis using three distinct techniques including SHAP, LIME and heat map, which provide a novel way for transparency obligation practice. There are several existing works that focus on presenting the explainability in the context of LLM. The work by Zhao et al. (2023) provide analysis of explainability considering both local and global important of features without any experimental result. Our work implements SHAP and LIME method through an experiment and result, where SHAP provides global feature importance across the dataset, while LIME offers local explanations for individual predictions. Our results also complement with Zhao et al. (2024)'s exploration of mechanistic interpretability. We observed that the model's sensitivity to tokens like vulnerable can lead to misinterpretations, like their findings on phenomena like memorization and grokking. Both works highlight the need for deeper investigation into token processing to ensure accurate interpretations of models. The unintuitive results from SHAP in our study align with Heyen et al. (2024), who found that large language models do not necessarily yield more plausible explanations. Both studies call for better faithfulness metrics and task-specific evaluation to improve the quality of model explanations. Regarding visual explanations considering heat map, the work by Machigashi et al. (2023) only provide different techniques to implement heat maps for neural network model which can enhance users' trust. In comparing to our work, we have implemented heat maps using attention weight of tokens for understanding key decision points of the model. Therefore, our contribution is beyond the existing practices of explainability by not only utilizing three mechanisms but also implementing them accordingly towards transparency obligation practice. In summary the proposed architecture combines both LLM and transparency components to ensure the developed model follows effective transparency obligation practices.

7. Conclusion

Vulnerability detection is critical and prerequisite for development of secure software intensive system. However, this task is challenging due to the large number of vulnerabilities and their evolving nature specifically source code related vulnerability. Recently, several works focused on adopting LLM model for the vulnerability detection but lack

of consideration on explaining the model decision and overall transparency obligation practice. This paper presents an approach that adopt BERT model for LLM based vulnerability detection and ensures the transparency obligation practice throughout the life cycle of LLM Model. The transparency obligation is considered from three dimensions including data, model outcome and explainability which allows to document the reasoning behind model predictions enabling stakeholders to understand and trust the process. The novelty of this research lies in the implementation of the transparency obligations practices, with a special focus on the explainability dimension. Three different mechanisms, namely SHAP, LIME, and the use of heat maps for visual representation of the predictions made by the model, are employed to explain the rationale for the prediction. It also facilitates the identification of potential biases and error, enhancing overall system transparency. By leveraging the SHAP and LIME framework and heat map using attention weight, specific to the context of LLMs, we tried to understand the features influencing the model's predictions. Our contribution extends beyond current explainability practices by integrating three distinct mechanisms and applying them in alignment with transparency obligation practices. The results obtained are consistent with the known functions that can potentially cause vulnerabilities. To achieve better results and enhance the quality of the explanations provided, it might be beneficial to develop methods for identifying more complex structures in a code snippet. Unlike traditional models that read text sequentially, BERT reads text in both directions simultaneously. This bidirectional approach allows BERT to understand the context of each token in relation to its surrounding words, providing a deeper understanding of the text. In future work, we plan to extend this work with data augmentation techniques such as synthetic data generation or oversampling of minority classes which could significantly help combat overfitting and enhance the model's ability to generalize across diverse datasets. Additionally, it is also necessary to optimize the decision thresholds used to classify code segments for more effective balance between recall and precision. We would also like to utilize diverse types of datasets in cybersecurity domain such as log analysis and malware as part of our future research.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the European Union's Horizon Europe Project, CyberSecDome—An innovative Virtual Reality-based intrusion detection, incident investigation, and response approach for enhancing the resilience, security, privacy, and accountability of complex and heterogeneous digital systems and infrastructures, funded under grant agreement No. 101120779.

Author statement

Jean Haurogné (JH) and Nihala Basheer (NB) : Conceptualization, Methodology, Experiment-BERT model (JH), Experiment-Transparency obligation (NB)

Shareeful Islam (SI): Background and Research Challenge, Funding Acquisition

Jean Haurogné, Nihala Basheer, Shareeful Islam: Writing- Original Draft Preparation, Review

All authors have read and agreed to the published version of the manuscript.

Data availability

Data will be made available on request.

References

- Bach, M., Werner, A., & Palt, M. (2019). The proposal of undersampling method for learning from imbalanced datasets. *Procedia Computer Science*, 159, 125–134. <https://doi.org/10.1016/j.procs.2019.09.167>
- Basheer, N., Islam, S., Alwaheidi, M. K. S., & Papastergiou, S. (2024). Adoption of deep-learning models for managing threat in API calls with transparency obligation practice for overall resilience. *Sensors*, 24(15), 4859. <https://doi.org/10.3390/s24154859>
- Basheer, N., Pranggono, B., Islam, S., Papastergiou, S., & Mouratidis, H. (2024). Enhancing malware detection through machine learning using XAI with SHAP framework. *IFIP advances in information and communication technology* (pp. 316–329). https://doi.org/10.1007/978-3-031-63211-2_24
- Brown, N. B. (2024). *Enhancing trust in LLMs: algorithms for comparing and interpreting LLMs*. Cornell University. <https://doi.org/10.48550/arxiv.2406.01943>. arXiv.
- Chen, X., Li, L., Chang, L., Huang, Y., Zhao, Y., Zhang, Y., & Li, D. (2023a). *Challenges and contributing factors in the utilization of large language models (LLMs)*. Cornell University. <https://doi.org/10.48550/arxiv.2310.13343>. arXiv.
- Chen, Y., Ding, Z., Alowain, L., Chen, X., & Wagner, D. (2023b). DiverseVul: A new vulnerable source code dataset for deep learning based vulnerability detection. ArXiv. <https://github.com/wagner-group/diversevul>.
- Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). *BERT: pre-training of deep bidirectional transformers for language understanding*. Cornell University. <https://doi.org/10.48550/arxiv.1810.04805>. arXiv.
- EU AI Act: first regulation on artificial intelligence. (2023, August 6). Topics | European Parliament. <https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence>.
- Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. (2020). CodeBERT: a pre-trained model for programming and natural languages. Cornell University. <https://doi.org/10.48550/arxiv.2002.08155>. arXiv.
- Gao, Z., Wang, H., Zhou, Y., Zhu, W., & Zhang, C. (2023). *How far have we gone in vulnerability detection using large language models*. Cornell University. <https://doi.org/10.48550/arxiv.2311.12420>. arXiv.
- Giner-Miguel, J., Gómez, A., & Cabot, J. (2024). *Using large language models to enrich the documentation of datasets for machine learning*. Cornell University. <https://doi.org/10.48550/arxiv.2404.15320>. arXiv.
- google-bert/bert-base-uncased · Hugging Face. (2001, May 3). <https://huggingface.co/google-bert/bert-base-uncased>.
- Harzevili, N. S., Belle, A. B., Wang, J., Wang, S., Ming, Z., Jiang, & Nagappan, N. (2023). *A survey on automated software vulnerability detection using machine learning and deep learning*. Cornell University. <https://doi.org/10.48550/arxiv.2306.11673>. arXiv.
- Heyen, H., Widdicombe, A., Siegel, N. Y., Perez-Ortiz, M., & Treleaven, P. (2024). *The effect of model size on llm post-hoc explainability via lime*. Cornell University. <https://doi.org/10.48550/arxiv.2405.05348>. arXiv.
- Joshi, M., & Bhardwaj, P. (2018). Impact of data transparency: Scientific publications. *Perspectives in Clinical Research*, 9(1), 31. https://doi.org/10.4103/picr.picr_104_17
- Karlsen, E., Luo, X., Zincer-Heywood, N., & Heywood, M. (2024). Large language models and unsupervised feature learning: implications for log analysis. *Annals of Telecommunications*. <https://doi.org/10.1007/s12243-024-01028-2>
- Key issue 5: Transparency obligations - EU AI Act. (n.d.). <https://www.euaiact.com/key-issue/5>.
- Kingma, D. P., & Ba, J. L. (2014). *Adam: a method for stochastic optimization*. Cornell University. <https://doi.org/10.48550/arxiv.1412.6980>. arXiv.
- Kumarakulasinghe, N. B., Blomberg, T., Liu, J., Leao, A. S., & Papapetrou, P. (2020). Evaluating local interpretable model-agnostic explanations on clinical machine learning classification models. In *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)* (pp. 7–12). <https://doi.org/10.1109/cbms49503.2020.00009>
- Laoutaris, N. (2018). Data transparency: Concerns and prospects [point of view]. *Proceedings of the IEEE*, 106(11), 1867–1871. <https://doi.org/10.1109/jproc.2018.2872313>
- Li, Z. (2022). Extracting spatial effects from machine learning model using local interpretation method: An example of SHAP and XGBoost. *Computers Environment and Urban Systems*, 96, Article 101845. <https://doi.org/10.1016/j.compenvurbsys.2022.101845>
- Liao, Q. V., & Vaughan, J. W. (2023). *AI transparency in the age of LLMs: a human-centered research roadmap*. Cornell University. <https://doi.org/10.48550/arxiv.2306.01941>. arXiv.
- Lundberg, S. M., & Lee, S. (2017). *A unified approach to interpreting model predictions*. Cornell University. <https://doi.org/10.48550/arxiv.1705.07874>. arXiv.
- Luo, H., & Specia, L. (2024). *From understanding to utilization: a survey on explainability for large language models*. Cornell University. <https://doi.org/10.48550/arxiv.2401.12874>. arXiv.
- Maehigashi, A., Fukuchi, Y., & Yamada, S. (2023). Experimental investigation of human acceptance of AI suggestions with heatmap and pointing-based XAI. In *HAI '23: Proceedings of the 11th International Conference on Human-Agent Interaction* (pp. 291–298). <https://doi.org/10.1145/3623809.3623834>
- Mathews, N. S., Brus, Y., Aafer, Y., Nagappan, M., & McIntosh, S. (2024). *LLBzpeky: leveraging large language models for vulnerability detection*. Cornell University. <https://doi.org/10.48550/arxiv.2401.01269>. arXiv.
- Muddamsetty, S. M., Jahromi, M. N. S., & Moeslund, T. B. (2021). Expert level evaluations for explainable AI (XAI) methods in the medical domain. In *Lecture notes in computer science* (pp. 35–46). https://doi.org/10.1007/978-3-030-68796-0_3
- Mohammed, R., Rawashdeh, J., & Abdullah, M. (2020). Machine learning with oversampling and undersampling techniques: Overview study and experimental results. In *2020 11th International Conference on Information and Communication Systems (ICICS)*. <https://doi.org/10.1109/icics49469.2020.239556>
- Nam, D., Macvean, A., Hellendoorn, V., Vasilescu, B., & Myers, B. (2024). Using an LLM to help with code understanding. ArXiv. <https://doi.org/10.1145/3597503.3639187>.
- Omar, M. (2023). *Detecting software vulnerabilities using language models*. Cornell University. <https://doi.org/10.48550/arxiv.2302.11773>. arXiv.
- Outeda, C. C. (2024). The EU's AI Act: A framework for collaborative governance. *Internet of Things*, Article 101291. <https://doi.org/10.1016/j.iot.2024.101291>
- Raji, I. D., Smart, A., White, R. N., Mitchell, M., Gebru, T., Hutchinson, B., Smith-Loud, J., Theron, D., & Barnes, P. (2020). *Closing the ai accountability gap: defining an end-to-end framework for internal algorithmic auditing*. Cornell University. <https://doi.org/10.48550/arxiv.2001.00973>. arXiv.
- Risse, N., & Böhm, M. (2023). *Limits of machine learning for automatic vulnerability detection*. Cornell University. <https://doi.org/10.48550/arxiv.2306.17193>. arXiv.
- Shestov, A., Cheshkov, A., Levichev, R., Mussabayev, R., Zadorozhny, P., Maslov, E., Vadim, C., & Bulychiev, E. (2024). *Finetuning large language models for vulnerability detection*. Cornell University. <https://doi.org/10.48550/arxiv.2401.17010>. arXiv.
- Steenhoeck, Benjamin, et al. "A comprehensive study of the capabilities of large language models for vulnerability detection." *arXiv preprint*, 2024, <https://doi.org/10.48550/arxiv.2403.17218>.
- Ullah, S., Coskun, A., Morari, A., & Pujar, S. (2023). *Step-by-step vulnerability detection using large language models*. https://www.bu.edu/peaclab/files/2023/08/USENIX_23_Poster.pdf.
- Wang, Y., Wang, W., Joty, S., & Hoi, S. C. (2021). CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. <https://doi.org/10.18653/v1/2021.emnlp-main.685>
- Yang, D., Kommineni, A., Alshehri, M., Mohanty, N., Modi, V., Gratch, J., & Narayanan, S. (2023). Context unlocks emotions: Text-based emotion classification dataset auditing with large language models. In *11th International Conference on Affective Computing and Intelligent Interaction (ACII)*. <https://doi.org/10.1109/acii59096.2023.10388131>
- Ying, X. (2019). An overview of overfitting and its solutions. *Journal of Physics Conference Series*, 1168, Article 022022. <https://doi.org/10.1088/1742-6596/1168/2/022022>
- Zhang, Y. (2024). *Unveiling LLM mechanisms through neural ODEs and control theory*. Cornell University. <https://doi.org/10.48550/arxiv.2406.16985>. arXiv.
- Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., Wang, S., Yin, D., & Du, M. (2023). *Explainability for large language models: a survey*. Cornell University. <https://doi.org/10.48550/arxiv.2309.01029>. arXiv.
- Zhao, H., Yang, F., Lakkaraju, H., & Du, M. (2024). *Opening the black box of large language models: two views on holistic interpretability*. Cornell University. <https://doi.org/10.48550/arxiv.2402.10688>. arXiv.
- Zhou, X., Zhang, T., & Lo, D. (2024). Large language model for vulnerability detection: Emerging results and future directions. arXiv. <https://doi.org/10.48550/arxiv.2401.15468>.