Detecting Security Requirements in GitHub Issues -Novel Dataset and SmallBERT-based model

Polina Minina, Andrey Sadovykh, Vladimir Ivanov

Detecting Security Requirements in GitHub Issues - Novel Dataset and SmallBERT-based model

1st Polina Minina
SOFTEAM
Paris, France
ORCID: 0009-0009-7683-2237

2nd Andrey Sadovykh
SOFTEAM
Paris, France
ORCID: 0000-0003-2384-5447

3rd Vladimir Ivanov
SOFTEAM
Paris, France
ORCID: 0000-0003-3289-8188

***Abstract*—Cloud application security initiates with the analysis of security requirements in DevOps. This involves gathering, managing, and tracking requirements within integrated issue-tracking systems found in repositories like GitHub. DevOps offers advantages in cloud app development, such as accelerated deployment, improved collaboration, and enhanced reliability. In DevOps, while many security verification tools are automated, security requirements analysis often relies on manual procedures. User feedback plays a pivotal role in shaping cloud application requirements, and the industry actively seeks automation solutions to expedite development. Prior research has demonstrated the limited performance of conventional NLP models trained on established datasets, such as PROMISE, when employed in the context of GitHub Issues. Recent studies have explored the integration of deep learning, particularly leveraging modern large language models and transfer learning architectures, to address requirements engineering challenges. However, a significant issue persists - the transferability of these models. While these models excel when applied to datasets similar to those they were trained on, their performance often drastically falls when dealing with external domains.**

In our paper, we introduce an automated method for classifying requirements within issue trackers. This method utilizes a novel dataset comprising 12,000 security and non-security issues collected from open GitHub repositories. We employed a SmallBERT-based model for training and conducted a series of experiments. Our research reaffirms the challenge related to the transferability of NLP models. Simultaneously, our model yields highly promising results when applied to GitHub Issues, even in challenging scenarios involving issues from projects that were not part of the training dataset and structured requirements texts from the PROMISE dataset. In summary, our approach significantly contributes to enhancing DevOps practices within cloud applications by automating security requirements analysis.

***Index Terms*—Security requirements, GitHub Issues, NLP, Classification, Dataset, Machine Learning, Deep learning, BERT**

I. Introduction

Requirements analysis is a cornerstone of the software development lifecycle, particularly when addressing security properties [1]. It provides the foundation for design, development, and validation by defining a product's scope, characteristics, and functionality. Neglecting requirements engineering can lead to project failures, with incomplete requirements contributing to about 44% of such failures.

In recent years, the adoption of DevSecOps, especially in Cloud-based applications, has surged [2]. DevSecOps relies on automated platforms such as GitHub [3] to facilitate continuous integration, deployment, and monitoring, expediting requirement fulfilment and issue resolution. However, analysing requirements and issues often remains a manual task, despite the many automation aspects within DevSecOps. This highlights the importance of leveraging automated solutions, such as Natural Language Processing (NLP) and machine learning (ML), to analyze and classify requirements, particularly within user feedback and issue-related content [4], [5]. In this paper, we present an approach that harvests a dataset of security-related GitHub issues and utilizes an NLP model for classification, demonstrating promising precision and recall results. In future work, we aim to integrate these mechanisms into GitHub Actions to enhance DevOps practices in Cloud applications.

II. Related Work

The field of Natural Language Processing for Requirements Engineering (NLP4RE) tackles various challenges through machine learning (ML) and NLP techniques. These encompass requirements classification, named entity recognition, user story generation, and user feedback analysis.

Identifying requirement texts within specification documents is vital in requirements engineering. Research has focused on extracting and classifying requirements from specifications, with studies such as [6]–[9] addressing the classification of text segments.

Classifying functional and non-functional requirements is a common challenge in NLP4RE. This involves binary classification, often using traditional machine-learning models. Research has explored classifying requirements from software specifications, as exemplified by works such as [10]–[13]. Security, a critical non-functional requirement, is crucial for application integrity and was examined in classification tasks by [14]–[16]. Additionally, user feedback, exemplified in [17], is a valuable source for identifying security requirements, including safety aspects, though showing the general difficulty in identifying quality attributes. It is worth noting the scarcity of literature dedicated to the classification of requirements within issue trackers, exemplified by platforms like GitHub.

A. Machine Learning Models for Requirements Classification

Researchers are actively exploring diverse methods to automate requirements classification. For example, Pérez-Verdejo,

in their study [17], compared the performance of four conventional machine learning models in classifying GitHub user issues based on quality attributes, including security. The models included random forest, support vector machines (SVM), decision tree, and Naive Bayes, with SVM demonstrating the highest performance, achieving a geometric mean of up to 0.82.

Furthermore, Gang Li and Chengpeng Zheng explored multiple approaches for classifying functional and non-functional requirements, with the combination of Generative Adversarial Network (GAN) and BERT demonstrating the best performance, achieving an impressive F1 score of 0.91 [13]. Gouri Deshpande's study [18] compared the performance of two models, BERT and Random Forest, for classifying dependent requirements. BERT achieved an outstanding F1 score of 0.93, highlighting the superiority of deep learning models, which excel in capturing the semantic aspects of text often overlooked by traditional methods.

B. Datasets

A fundamental component in training and evaluating classification models involves labeled datasets containing requirements. PROMISE [19], a widely employed dataset, encompasses 625 requirement statements, including 255 functional and 370 non-functional requirements. It has been utilized in numerous studies, e.g., [11]–[13], [17], [20], with further categorization performed in [17] based on quality attributes.

Another prominent dataset is PURE [21], featuring 34,268 requirement sentences extracted from 79 openly accessible requirements documents. Various researchers have employed PURE in studies such as [22], [23], with the unique aspect that the requirements in PURE were manually labeled. Notably, PURE offers a substantially larger dataset compared to PROMISE, facilitating more extensive research and analysis.

User stories, often capturing system behavior informally, have been curated into a dataset of 1680 user stories by Fabiano Dalpiaz [24], primarily for ambiguity detection [25]. Additionally, user reviews, containing valuable requirements, bug reports, and software requests, were gathered by Eduard C. Groe, comprising 132,194 user reviews for various applications [26].

III. Approach

In this study, we opted to employ Bidirectional Encoder Representations from Transformers (BERT) as our chosen model for experimentation [27]. BERT is a neural network built upon the transformers architecture, pre-trained on a substantial corpus of data sourced from Wikipedia. Leveraging the transformers architecture, BERT excels in capturing intricate textual relationships and contextual information. It accepts word tokens as input and can be trained on raw text. Furthermore, it is noteworthy that, in classification tasks with limited dataset sizes, the BERT model has demonstrated superior performance compared to other machine learning models.

To expedite training, we utilized a streamlined variant of BERT known as SmallBERT [28]. SmallBERT is a family of extremely small BERT models [29] trained using a mixed-vocabulary distillation method. This method allows the models to be compressed to a fraction of the size of the original BERT models, while still maintaining good performance on a variety of natural language processing tasks. SmallBERT boasts fewer parameters, thus enhancing its learning efficiency.

We observed that NLP models trained on structured documents often exhibit relatively modest performance on GitHub issues, as evident in an F1 score of approximately 0.5 in [17]. Consequently, we undertook the task of assembling our dataset. To accomplish this, we collected a dataset of user issues from GitHub to facilitate model training. Specifically, we harvested user issues from the top 1,000 repositories, amassing around 3,419,446 user issues, each annotated with different labels. After harvesting user issues from GitHub, we applied filtering to isolate issues labelled as "security," resulting in 6,068 records out of the initial 3,000,000 issues. To optimize the performance of our models, we sought to balance the GitHub dataset. Thus, we selected 6,068 samples from non-security user issues with labels "bug," "feature," and "support." In addition, for the non-security dataset, we removed sentences containing the words from the cybersecurity glossary [30]. Totally, our GitHub dataset comprised 12,136 user issues and may be accessed at Zenodo [31]. The examples of security and non-security related GitHub Issues are demonstrated in Table I.

TABLE I
EXAMPLES OF USER ISSUES LABELED AS "SECURITY" AND "NON-SECURITY"

Label	Issue
Sec	Security: 4 Electron (react-devtools dep) security advisories
Sec	Discussion: vulnerabilities in Oh My Zsh (2021-11-12)
Non-Sec	Debugging yeoman generator doesn't launch on Windows
Non-Sec	Rest API needs to be consistent across all multi-bucket aggs

Dataset splitting is the process of dividing a dataset into training and testing subsets. The proportions in which the dataset is split can vary depending on the specific task and the size of the dataset. In this study, we used a relatively small dataset compared to other machine learning datasets.

Within the scope of related work, it is evident that a common practice in numerous studies is the utilization of an 80% training and 20% testing data split for requirements classification tasks. Consequently, our study adhered to this standard approach, partitioning the dataset into an 80% portion designated for training and a 20% segment allocated for testing.

In the realm of machine learning, a diverse array of metrics is available to assess model performance. Among these metrics, recall, precision, and F1 score [32] are the most widely employed for evaluating the efficacy of machine learning models in classification tasks. In this paper, we utilize the F1 score to gauge model performance. The F1 score is computed as

TABLE II
SMALLBERT MODEL PERFORMANCE ON SECURITY REQUIREMENTS CLASSIFICATION TASK FOR VARIOUS COMBINATIONS OF TRAINING AND TESTING DATASETS

Training / testing dataset	F1-score
1. PROMISE-PROMISE	0.95
2. PROMISE-GitHub	0.63
3. GitHub-GitHub	0.93
4. GitHub - external GitHub	0.82
5. GitHub-PROMISE	0.71

the harmonic mean of precision and recall, amalgamating the strengths of these two metrics for a comprehensive evaluation.

Our study encompassed a series of experiments designed to assess the performance of the SmallBERT model in the context of security requirements classification. We aimed to subject the model to diverse training and testing conditions, including the following scenarios:

1) Employing PROMISE as both the training and testing dataset.
2) Implementing cross-validation by training a model on PROMISE dataset and evaluating it on GitHub issue titles similarly to [17].
3) Utilizing GitHub issue titles as the training and testing dataset.
4) Conducting external validation by training the GitHub titles model and testing it on GitHub issues from projects not included in the training set.
5) Performing further external validation by training the GitHub titles model and assessing its performance on structured security requirements extracted from PROMISE.

IV. RESULTS

Our study involved a series of experiments encompassing different models and various combinations of training and test datasets. Detailed descriptions and the corresponding code for these experiments are accessible at [33]. Table II presents the performance of the SmallBERT model in the context of security requirements classification, considering various combinations of training and testing datasets.

SmallBERT demonstrated remarkable performance in conventional training and testing setups, particularly when the training and testing datasets belonged to the same domain. Notably, when we applied the model to the PROMISE dataset, we achieved an F1-score of 0.95. This achievement is noteworthy and aligns with previous studies that have leveraged transfer learning models for security requirements classification.

Conversely, the performance of the PROMISE-trained model, when applied to GitHub issues, yielded a more modest result, with an F1-score of 0.63. This discrepancy can be attributed to the relatively unstructured nature of GitHub issues compared to the more formally structured requirements statements within the PROMISE dataset. It serves to cross-validate the performance evaluation observed in a prior study [17].

In the traditional training and testing dataset configuration using GitHub user issues, SmallBERT once again exhibited remarkable performance, achieving an F1-score of 0.93. These results affirm our initial assumptions regarding the efficacy of transfer learning in classification tasks involving highly unstructured requirements statements. Notably, these results were obtained by testing the model on user issues not encountered during training but originating from the same projects used in the training dataset.

To assess the generalization capabilities of our trained model, we conducted an external validation. In this evaluation, the trained model was specifically tested on issues from GitHub projects that were not included in the training dataset. In this context, our model achieved a commendable F1-score of 0.82. This outcome holds significant relevance as it underscores the applicability of our model for the general classification of GitHub user issues, extending beyond the specific projects involved in the training dataset.

In our final assessment, we tested our model, which was trained on GitHub issues, on the PROMISE dataset. The achieved F1-score of 0.71, though towards the lower end of the acceptability spectrum, highlights that our model can provide practical utility even when applied to more structured and formal requirements statements.

V. CONCLUSION

This paper discusses our exploration of natural language processing (NLP) models for the classification of requirements, with a particular emphasis on identifying security-related user issues within code repositories, such as GitHub. Our primary objective was to address the limitation of models trained exclusively on requirements specification documents when tasked with issue classification. To overcome this challenge, we meticulously compiled a tailored dataset by extracting security-labeled issues from open projects on GitHub. Following a rigorous data cleaning process, we conducted a series of experiments using different training and testing dataset combinations to assess the broad applicability of our SmallBERT-based model in classifying GitHub user issues.

Our experiments affirmed the consistently high performance of BERT-like models in requirements classification tasks in accordance with prior research. Additionally, we cross-validated the subpar performance of the PROMISE-trained model on GitHub Issues, which further validated our approach of creating a specialized dataset for this specific task.

Moreover, the outcomes showcased a significant contrast in performance when deploying the trained models on issues within the same domain, meaning those from projects included in the training dataset, compared to those from external domains. These external domain issues were associated with projects entirely new and absent from the training dataset. This distinction underscores the critical consideration of a model's generalisability and transferability, ensuring dependable performance in real-world situations involving unencountered projects.

Our plans for further improvement are to experiment with more types of transfer learning models, preferably trained on the cyber-security texts. Moreover, in the training, we would like to consider the description texts of GitHub user issues, which can further improve classification. Finally, we plan to integrate our model as a plugin to GitHub Actions to integrate the security issues classification in Continuous Integration practices.

We consider our findings to have substantial relevance for the broader community. The knowledge acquired from our study offers valuable insights for tackling the issue classification challenge. In the future, we are dedicated to continued exploration of this problem, particularly within the framework of our ongoing project focused on automating requirements engineering in DevSecOps environments.

In the context of cloud applications, where scalability and agility are paramount, the ability to quickly and accurately classify user issues and security-related requirements is crucial. Our findings offer a pathway to achieving this by demonstrating the effectiveness of advanced NLP models in automating these processes. By doing so, cloud application developers can maintain a competitive edge, delivering high-quality software with minimal delay. This not only aligns with the principles of DevOps but also ensures the robustness and security of cloud applications, a critical consideration in today's digital landscape.

ACKNOWLEDGMENT

This work is partially supported by the VeriDevOps [34] project funded by the Horizon 2020 program under the grant agreement No. 957212.

REFERENCES

[1] T. Clancy, "The chaos report," *The Standish Group*, 1995.
[2] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and solutions when adopting devsecops: A systematic review," *Information and Software Technology*, vol. 141, p. 106700, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584921001543
[3] "Github," https://github.com/.
[4] C. Bishop, "Pattern recognition and machine learning errata," 2006.
[5] K. Chowdhary and K. Chowdhary, "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, 2020.
[6] S. Abualhaija, C. Arora, M. Sabetzadeh, L. C. Briand, and E. Vaz, "A machine learning-based approach for demarcating requirements in textual specifications," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019.
[7] V. Ivanov, A. Sadovykh, A. Naumchev, A. Bagnato, and K. Yakovlev, "Extracting software requirements from unstructured documents," in *International Conference on Analysis of Images, Social Networks and Texts*. Springer, 2021, pp. 17–29.
[8] J. Winkler and A. Vogelsang, "Automatic classification of requirements based on convolutional neural networks," in *2016 IEEE 24th International Requirements Engineering Conference Workshops*, 2016.
[9] J. P. Winkler and A. Vogelsang, "Using tools to assist identification of non-requirements in requirements specifications–a controlled experiment," in *Requirements Engineering: Foundation for Software Quality: 24th International Working Conference, REFSQ 2018, Utrecht, The Netherlands, March 19-22, 2018, Proceedings 24*. Springer, 2018, pp. 57–71.
[10] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "Automated classification of non-functional requirements," vol. 12. Springer, 2007, pp. 103–120.
[11] V. Subbiah, "Automatic classification and extraction of non-functional requirements from text files: a supervised learning approach," 2022.
[12] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy, "Norbert: Transfer learning for requirements classification," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 2020.
[13] G. Li, C. Zheng, M. Li, and H. Wang, "Automatic requirements classification based on graph attention network," in *IEEE Access*, 2022.
[14] M. Riaz, J. King, J. Slankas, and L. Williams, "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, 2014, pp. 183–192.
[15] K. Ameri, M. Hempel, H. Sharif, J. Lopez, and K. Perumalla, "Cybert: Cybersecurity claim classification by fine-tuning the bert language model," 2021.
[16] F. Casillo, V. Deufemia, and C. Gravino, "Detecting privacy requirements from user stories with nlp transfer learning models," 2022.
[17] J. M. Pérez-Verdejo, A. J. S. Garcia, J. O. Ocharán-Hernández, E. Mezura-Montes, and K. C. Verdín, "Requirements and github issues: An automated approach for quality requirements classification," 2021.
[18] G. Deshpande, B. Sheikhi, and S. Chakka, "Is bert the new silver bullet? - an empirical investigation of requirements dependency classification," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, 2021.
[19] J. Sayyad Shirabad and T. Menzies, "The PROMISE Repository of Software Engineering Databases." School of Information Technology and Engineering, University of Ottawa, Canada, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository
[20] M.-I. Limaylla-Lunarejo, N. Condori-Fernandez, and M. R. Luaces, "Towards an automatic requirements classification in a new spanish dataset," in *2022 IEEE 30th International Requirements Engineering Conference (RE)*, 2022.
[21] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "Pure: A dataset of public requirements documents," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 502–505.
[22] M. K. Habib, S. Wagner, and D. Graziotin, "Detecting requirements smells with deep learning: Experiences, challenges and future work," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, 2021, pp. 153–156.
[23] M. Ajagbe and L. Zhao, "Retraining a bert model for transfer learning in requirements engineering: A preliminary study," in *2022 IEEE 30th International Requirements Engineering Conference (RE)*, 2022, pp. 309–315.
[24] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Requirements classification with interpretable machine learning and dependency parsing," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, pp. 142–152.
[25] F. Dalpiaz, I. Schalk, S. Brinkkemper, F. Aydemir, and G. Lucassen, "Detecting terminological ambiguity in user stories: Tool and experimentation," 2018.
[26] E. C. Groen, S. Kopczyńska, M. P. Hauer, T. D. Krafft, and J. Doerr, "Users — the hidden software product quality experts?: A study on how app users report quality aspects in online reviews," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 80–89.
[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," vol. abs/1810.04805, 2019.
[28] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: On the importance of pre-training compact models," *arXiv preprint arXiv:1908.08962*, 2019.
[29] S. Zhao, R. Gupta, Y. Song, and D. Zhou, "Extremely small bert models from mixed-vocabulary training," *arXiv preprint arXiv:1909.11687*, 2019.
[30] R. Kissel, "Glossary of key information security terms," 2013.
[31] P. Minina, "Github user issues harvesting," Jun. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8034795
[32] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in *Advances in Information Retrieval: 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005. Proceedings 27*. Springer, 2005, pp. 345–359.
[33] P. Minina, "Github issues harvesting and model training," https://github.com/pminina01/github-security-issues.
[34] "VeriDevOps: Automated Protection and Prevention to Meet Security Requirements in DevOps," Feb. 2021, pp. 1330–1333, iSSN: 1558-1101.