Exploring Technical Debt in Security Questions on Stack Overflow

Joshua Aldrich Edbert*, Sahrima Jannat Oishwee*, Shubhashis Karmakar*, Zadia Codabux*, Roberto Verdecchia†
*Department of Computer Science
University of Saskatchewan
Email: {joshua.edbert, sahrima.oishwee, shubhashis.k}@usask.ca, zadiacodabux@ieee.org
†Department of Information Engineering
University of Florence
Email: roberto.verdecchia@unifi.it

***Abstract*—Background: Software security is crucial to ensure that the users are protected from undesirable consequences such as malware attacks which can result in loss of data and, subsequently, financial loss. Technical Debt (TD) is a metaphor incurred by suboptimal decisions resulting in long-term consequences such as increased defects and vulnerabilities if not managed. Although previous studies have studied the relationship between security and TD, examining their intersection in developers' discussion on Stack Overflow (SO) is still unexplored. Aims: This study investigates the characteristics of security-related TD questions on SO. More specifically, we explore the prevalence of TD in security-related queries, identify the security tags most prone to TD, and investigate which user groups are more aware of TD. Method: We mined 117,233 security-related questions on SO and used a deep-learning approach to identify 45,078 security-related TD questions. Subsequently, we conducted quantitative and qualitative analyses of the collected security-related TD questions, including sentiment analysis. Results: Our analysis revealed that 38% of the security questions on SO are security-related TD questions. The most recurrent tags among the security-related TD questions emerged as "security" and "encryption." The latter typically have a neutral sentiment, are lengthier, and are posed by users with higher reputation scores. Conclusions: Our findings reveal that developers implicitly discuss TD, suggesting developers have a potential knowledge gap regarding the TD metaphor in the security domain. Moreover, we identified the most common security topics mentioned in TD-related posts, providing valuable insights for developers and researchers to assist developers in prioritizing security concerns in order to minimize TD and enhance software security.**

***Index Terms*—Technical Debt, Security Vulnerability, Stack Overflow, Crowdsourcing**

I. Introduction

Security is a crucial component in software development [1] and is concerned with the ability of applications to withstand malicious attacks brought on by exploiting flaws in the software [2]. Neglecting security in software development will result in severe consequences for users and companies, such as financial loss, personal data compromise, confidentiality breaches, damaged reputation, and delays in software development efforts [3]–[6]. In addition, security threats in software development have become more complex due to growing advancements in the Internet of Things [7]. Hence, ensuring secure software is crucial to software development [8], [9].

Technical Debt (TD) is a metaphor used to describe suboptimal artifacts created as a result of design and implementation choices that, while they may achieve short-term objectives, may cause issues during the maintenance and evolution phases of a software project [10]. TD can be incurred during any phase of the software development lifecycle but primarily impacts software maintenance [11]. Inadequate TD management can result in higher costs, poor product quality, and a slowdown in the long-term success of software development [12]. As a result, TD is also acknowledged as a critical problem in software development [13].

Identifying security threats during the software development lifecycle is crucial for secure development [14]. Hence, mechanisms to help developers detect security risks before software releases are important [15], [16]. For instance, TD has been used to highlight security vulnerabilities in software products [17]. Suboptimal security implementation practices can weaken a system disastrously, and simple coding mistakes or design issues can lead to exploitable vulnerabilities [18], [19]. Incurring TD due to suboptimal security practices can make it harder to maintain or update the software, increasing the risk of future attacks and further compounding the consequences of a breach [17]. Therefore, studying the intersection between TD and security in developers' discussions is essential to understanding and preventing suboptimal security practices.

Previous studies have explored the connection between TD and security vulnerability [17], [20]–[22] and examined the security-related posts on Stack Overflow (SO) [3]. Additionally, studies have examined the TD-related posts on SO [23], [24]. However, to the best of our knowledge, no studies have focused explicitly on security-related questions with TD on SO. SO is a crowdsourcing platform where users exchange information about programming tasks with over 24 million questions and 20 million users[1]. It allows practitioners to examine how users request support and exchange expertise about technical issues [24]. SO is a helpful resource for

978-1-6654-5223-6/23/$31.00 ©2023 IEEE

[1]https://stackexchange.com/sites#traffic

understanding real-world viewpoints on various software engineering problems [25], [26].

In this study, we investigated security-related questions on SO. The security-related questions with indications of suboptimal security practices will be referred to as Security-related TD Questions (STDQs). The security questions without indications of suboptimal security practices will be referred to as non-STDQs. More specifically, we quantitatively and qualitatively investigated various traits of STDQs and non-STDQs, including the security tags that are more recurrent in STDQs and non-STDQs, the sentiment, popularity score, length, security topics, question types of STDQs and non-STDQs, and finally, the SO account profile of users asking STDQs and non-STDQs. Note that security "tag" and "topic" are different terms used in this study. Tags are SO tags attached to the questions by the users, while topics emerged from our manual coding process of the questions (described in Section III-E).

This study offers a thorough analysis of STDQs to help better comprehend the relationship between TD and software security. Our key contributions include the following:

1) A qualitative analysis of question types and security topics of STDQs to better inform the community which specific question types and security topics are more prevalent in STDQs.
2) An exploration of the most recurrent security tags in STDQs to support researchers in identifying the areas of security most susceptible to TD.
3) An analysis of the question length, sentiment, popularity, and time needed for answers of STDQs to understand the complexity, popularity, and emotion toward STDQs.
4) An investigation of the SO account profile characteristics of the users asking STDQs to understand which group of people most frequently pose STDQs and need more understanding of TD resolution in secure development.
5) A comprehensive replication package[2] of the study.

The rest of this paper is structured as follows: Section II presents related works. Section III describes our study's methodology. Section IV and Section V describe and discuss the findings of our experiments, respectively. Section VI describes the implications of our findings. Threats to this study's validity are listed in Section VII. Section VIII concludes the study.

II. Related Work

General Stack Overflow Studies. Barua et al. [25] examined the main topics and trends of general SO discussions using Latent Dirichlet Allocation. Although our study and theirs analyzed SO discussions, our study did not investigate general SO discussions but investigated STDQs specifically. Their study found that the topic of interest among developers varies. Over time, web development, mobile applications, Git, and MySQL have gained the most traction. Rosen et al. [27] analyzed the topics related to mobile development in SO discussions instead of TD and security. Their study revealed that questions like app distribution, mobile APIs, and data management are frequently asked in SO. Haque et al. [28] also examined the topics of SO discussions related to Docker instead of security and TD. According to their findings, most developers use SO to post questions about various Docker-related subjects, such as framework development and application deployment.

Stack Overflow Studies on Technical Debt. Kozanidis et al. [24] analyzed the different characteristics of TD-related questions on SO. This study is very similar since we also analyzed the characteristics of SO questions, including sentiment and question length. However, we analyzed STDQs instead of TD-related questions. In addition, they investigated whether machine learning can be used to detect and classify TD questions on SO automatically. We used their dataset for training our classification model to detect STDQs. Their study showed that architecture debt is the most discussed debt on SO, most TD questions have a slight sense of urgency and neutral sentiment, the question length varies across different debt types, and machine learning can identify TD questions but not classify them across different debt kinds. The study by Alfayez et al. [23] extracted and examined 578 TD-related queries using a dataset derived from three Stack Exchange Q&A websites, including SO. Although their study and ours investigated TD-related questions on SO, they did not specifically investigate STDQs. In addition, most of the characteristics we investigated differed from theirs, except for the median time to receive an answer. Their findings showed that there are 14 categories in which TD-related questions can be categorized, 636 different tags are used in the acquired set of TD-related questions, and some TD-related categories have a shortage of accepted answers and a longer median time to receive an accepted answer than others. Gama et al. [29] manually evaluated a sample of SO discussions to determine how developers identify TD in their software projects. While their study and ours investigated SO, we aimed to investigate STDQs, not TD identification, in software projects. They found that SO users frequently discuss TD identification and reported 29 low-level indicators for detecting TD items in code, infrastructure, architecture, and test.

Stack Overflow Studies on Software Security. Yang et al. [3] described a large-scale study on security-related questions on SO. Their study investigated security-related topics and trends. Their study and ours investigated security-related questions on SO, but ours specifically investigated STDQs. The authors used the SO tagging system to extract security-related questions from SO, which we used in our study to collect the security-related questions of SO. They reported the top five main security-related topics and the top eight challenging security-related topics. Croft et al. [30] examined SO and GitHub discussions. They investigated the security issues faced by programmers using different programming languages by leveraging Latent Dirichlet Analysis. The difficulties and features of the security issues vary greatly

[2]https://doi.org/10.5281/zenodo.7888440

depending on the programming languages and data sources. Lopez et al. [31] studied how security knowledge and secure practices were produced and shared among practitioners on SO and how developers conversed about security in SO. Their results showed developers actively conversed on the website to address security issues, promote knowledge, exchange knowledge, and help one another.

Security and Technical Debt Studies Siavvas et al. [17] assessed TD's ability to detect security issues in software products by analyzing a large code repository with static analysis tools. They found a statistically significant positive correlation between TD and the vulnerability densities of the examined software products. Siavvas et al. [21] evaluated TD indicators in predicting software security risks at the project and class levels by developing various machine learning models. Their conclusions imply that TD indications have the potential to serve as security indicators. Izurieta et al. [20] developed a method to examine TD-related security flaws using the Common Weakness Enumeration and the Common Weakness Scoring System. While our study also investigated the intersection between TD and security, we specifically focused on SO discussion, an aspect not explored by previous studies.

Summary: Compared to previous studies, which investigated either TD or security vulnerability discussions in SO separately, our study specifically investigated STDQs. We explored the user characteristics such as the SO account age, reputation score, profile views, and earned badges. Our study also identified security topics more prone to TD.

III. Methodology

This section describes how this study was conducted. We describe the objective, research questions, data collection, and analysis. Figure 1 depicts the phases of the methodology.

A. Goal

This goal of the study is described using the Goal-Question-Metric technique [32] as follows:

Purpose: To investigate
Issue: the characteristics of
Object: security-related technical debt questions
Viewpoint: from the software engineering researchers perspective

B. Research Questions

Based on our goal, we derive the following Research Questions (RQs):

RQ_1 To what extent do developers indicate suboptimal security practices in security-related questions?

Rationale: This research question seeks to understand how frequently suboptimal security practices are indicated in security-related questions on SO. Understanding this will help us gain insights into the prevalence of incurring suboptimal security practices in security questions. This research question contributes to the broader understanding of how TD and security are intertwined from the SO discussion perspective.

RQ_2 Which security tags are more recurrent in security-related TD questions?

Rationale: By exploring which security-related tags on SO contain the most indications of suboptimal security practices, researchers can identify the areas of security most susceptible to TD. Identifying these areas can help software developers prioritize their resources and efforts to prevent and mitigate TD in the identified areas, reducing the risk of security issues.

RQ_3 What are the different characteristics of security-related TD questions?

Rationale: By examining the sentiment of STDQs, the study can gain insight into the attitudes and emotions associated with the STDQs [33]. The question score and length of a SO question are indicators of the importance and complexity of the issue being discussed in the question [24]. Lastly, we identify the tags that require greater community attention to determine whether some questions are more challenging to answer than others [34].

RQ_4 What are the characteristics of the user profiles asking security-related TD questions?

Rationale: The SO community is known for its diverse users with varying experiences and expertise. Understanding the characteristics of users who ask STDQs, including reputation score, profile age, and badges earned, as previously done by Konstantinos et al. [35] can help researchers identify patterns and characteristics of users associated with STDQs. This can help inform which users are more likely to address suboptimal security practices in security questions.

C. Extracting Security Questions

In this study, we collected questions and user information from SO by downloading the publicly available SO dump provided by the Stack Exchange Data Dump[3]. Our questions and user dataset spans from 2008 until 2022 and comprises 23,020,127 questions and 19,307,021 users of SO. Each question includes a title and body.

Since the collected SO questions could be related to any topic, developing a filtering technique to identify the questions related to security was necessary. One strategy would be to use the tags of SO questions and compile all questions with the tag "security" [3]. However, the study by Yang et al. [3] revealed that several security questions on SO are not tagged with "security" but tagged with other security-related terms.

To overcome this impediment, we followed the guideline of Yang et al. [3] to extract the SO questions related to security. The SO questions tagged with "security," "sql-injection," "passwords," "encryption," "xss," "websecurity," "csrf," "password-protection," or "cryptography" will be extracted. Cross-Site Scripting ("xss"), Cross-Site Request Forgery ("csrf"), and "sql-injection" are related to web vulnerability, while "cryptography" and "encryption" are related to cryptography techniques.

[3]https://archive.org/details/stackexchange

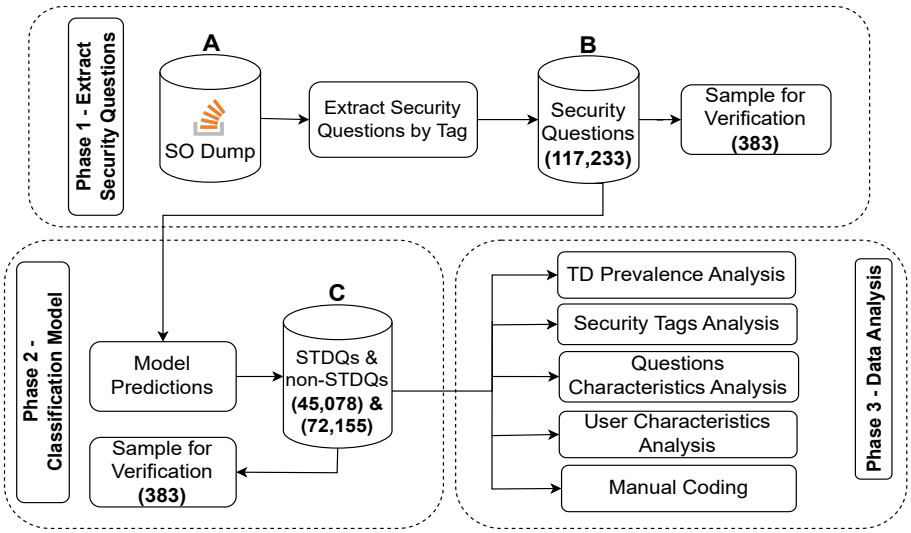

Fig. 1: Methodology Overview

The study of Yang et al. [3] independently validated the exclusivity and representativeness of those nine security tags using a two-step heuristic procedure. They started by searching for questions with the tag "security" in them. In the second step, they extracted the tags from those questions tagged with "security" and referred to them as candidate tags. Finally, they went through each candidate tag and filtered them out. This selection process resulted in a well-curated set of nine security tags, where the tags were exclusive and representative of the security questions on SO. Using the nine security tags, we extracted 117,233 security-related questions. We used the title and body of the questions for our analysis.

Next, we manually examined a sample of 117,233 security questions to assess if the strategy of automated filtering of the questions with the tags led to a high-quality result. In particular, we sampled 383 security questions (with a 95% confidence level and 5% margin of error), and two authors independently reviewed and assessed each question to determine whether they were related to security. All disagreements among raters were solved by discussing the questions marked differently by both raters. We used Cohen's Kappa [36] to measure inter-rater reliability. The result showed strong agreement among raters (Cohen's Kappa = 0.89). Our final verification results showed that of the 383 sampled security questions, 370 (97%) were related to security. We concluded that using the tags by Yang et al. [3] is suitable for our study.

D. Classification Model

After extracting security-related questions, this subsection describes our strategy to identify STDQs. This phase consisted of three main steps, namely (i) preprocessing the extracted security-related questions from the previous section, (ii) training a binary classification model to detect STDQs, and (iii) using the trained classification model to detect STDQs in our preprocessed security-related questions.

First, we preprocessed our extracted security questions from Section III-C. This preprocessing step was necessary to clean the extracted security questions. We followed the guidelines by Kozanidis et al. [24] to preprocess the extracted security questions. We tokenized every question, converted it to lowercase, removed stopwords, and removed all punctuation from the security questions. We also removed any HTML or markdown elements and replaced any instances of source code with a code tag, as these do not include information of lexical or semantic value. We deleted references to "technical debt," other TD-related keywords suggested by Kozanidis et al. [24], and "SonarQube" from the text to prevent the model from being over-fit during the training phase [24].

Next, we trained a binary classification model by following the guidelines by Kozanidis et al. [24] to detect STDQs. We used their dataset to train our model. Our binary classification model was built using Python's simple transformers library[4]. We used the pre-trained RoBERTa [37] model as it is considered the state-of-the-art model for binary text classification [38], [39]. With the training dataset prepared by Kozanidis et al. [24], we used 10-fold cross-validation to train and test the classification model. We used the standard metrics for evaluating our classification model [24], [40], namely precision, recall, and F1 score. With the test set, our model detected STDQs with an F1 score of 0.85.

Finally, we predicted each prepossessed security question with our trained classifier to identify STDQs. To verify the performance of our classification model, we took a sample with a 95% confidence level and a 5% margin of error. Two authors manually verified the output results of the model prediction. By following a labeling guideline of previously published work [24], two authors independently reviewed 384 predicted security questions and assessed each question to determine whether it was related to TD. Any disagreements

[4]https://simpletransformers.ai/

among raters were solved by discussing the questions marked differently by both raters. We had a strong agreement (Cohen's Kappa = 0.81). Our final verification, matched with the model's prediction results, showed that the model achieved an F1 score of 0.75, comparable with the results of the binary classification model of Kozanidis et al. [24].

E. Data Analysis

We analyzed the questions (both STDQs and non-STDQs) using quantitative and qualitative methods to answer our research questions. For the quantitative analysis, we investigated which of the nine security tags by Yang et al. [3] was present in the questions and recorded their frequency. We also examined how many words each question contained to determine whether indications of suboptimal security practices affect a question's length. We considered the post score of a question and the time it took for a question to have an accepted answer. The post score on SO indicates the usefulness of a question as perceived by the community. A higher score means that the post is considered more helpful by the community. The Valence Aware Dictionary and sEntiment Reasoner (VADER) tool were used to analyze the sentiment. VADER is a vocabulary and rule-based sentiment analysis tool and measures the amount of positive or negative emotion and the intensity of emotion in a text [41]. It is readily available for use on unlabeled text data and is included in Python's NLTK[5] package. We employed VADER to analyze the sentiment of STDQs and non-STDQs since it is a state-of-the-art sentiment analysis tool that performs well in SO setting [42]. Both relative sentiment components (negative, neutral, and positive) and compound sentiment scores were used to examine the sentiment expressed in our security questions. According to VADER's GitHub[6] site, each word's valence score is added, modified following the guidelines, and normalized to fall between -1 (the most severe negative) and +1 to get the compound score (the most extreme positive). Lastly, we analyzed the user characteristics regarding their reputation score, the years of their account age, the number of profile views, and the badges achieved.

For the qualitative analysis, we took a sample of 396 questions (95% confidence level and 5% margin of error) and manually coded the questions. This supplementary coding procedure aims to understand better the question types and security topics. The *question types* is defined as the nature of the questions and can be categorized into the following: debugging, conceptual understanding, implementation, code review, best practices, and learning resources. Table II defined each question type. These question types emerged as a result of the provisional coding [43] from the types reported by Allamanis et al. [44].

In addition, the *security topics* of the questions are defined as the security topics a question belongs to. This topic differs from the security tags we used to extract security-related questions. Specifically, security tags are provided as SO tags attached to the questions by the users, while security topics emerged from our manual coding process of the questions. The security topics in our security questions were identified using open coding [43]. One author coded the question types and security topics for each sampled question. Another author reviewed the final classification to ensure the accuracy of the coding procedure. Disagreements were discussed and resolved.

[5]www.nltk.org/_modules/nltk/sentiment/vader.html

[6]www.github.com/cjhutto/vaderSentiment

IV. Results

A. $\mathbf{RQ_1}$: *To what extent do developers indicate suboptimal security practices in security-related questions?*

Out of the 117,233 security-related questions we extracted, our model identified 45,078 (38%) questions as STDQs. The remaining 72,155 security questions were non-STDQs. However, these 45,078 STDQs do not explicitly mention the word "technical debt." An example of an STDQ is as follows:

> *I use gpg for encrypting a file storing my passwords in Windows. This file is an MS Excel file, which I use for convenience. Every time I want to check or update my passwords (> once per day on average), I execute the following batch script, which decodes the encrypted file and encodes the updated xlsx file again when I close the application. Obviously, this is a suboptimal solution as it creates a decrypted file, which in case of an interruption (e.g. accidentally closing the command line window or a system crash), the file remains unencrypted. Anyone with something better, e.g. using in-memory pipes or the like (in Windows)?* - SO Post ID 66626505[7]

The question asked for a script for file encryption in Windows. As we can see, although the word "technical debt" was not mentioned explicitly, this security question clearly discussed suboptimal code solutions, which indicates suboptimal security practices. The person asking the question mentioned how he tried implementing scripts but found them suboptimal.

Summary $\mathbf{RQ_1}$ (Prevalence of STDQs) 38% of security-related questions are STDQs. While addressing suboptimal security issues, developers do not explicitly use the term "technical debt" but use terminologies indicating suboptimal security practices.

B. $\mathbf{RQ_2}$: *Which security tags are more recurrent in security-related TD questions?*

Table I displays the distribution of security tags in STDQs and non-STDQs. It shows the number of questions per tag and the proportion of STDQs and non-STDQs for each tag. The security tags were ranked in order of most to least frequent number of questions. The "security" tag appears most frequently in the questions, followed by "encryption," and in third place, "cryptography." Hence, irrespective of STDQs or

[7]www.stackoverflow.com/questions/66626505

TABLE I: Distribution of STDQs and non-STDQs

Rank	STDQs			non-STDQs		
	Security Tags	**Questions**	**% per Tag**	**Security Tags**	**Questions**	**% per Tag**
1	security	24,449	44.4	security	30,532	55.6
2	encryption	12,635	35.0	encryption	23,396	65.0
3	cryptography	5,224	36.5	cryptography	9,072	63.5
4	passwords	2,898	27.7	passwords	7,555	72.3
5	xss	1,910	41.8	csrf	2,654	58.2
6	csrf	1,780	41.8	xss	2,475	58.2
7	sql-injection	1,582	43.0	sql-injection	2,100	57.0
8	password-protection	575	33.8	password-protection	1,124	66.2
9	websecurity	150	44.0	websecurity	191	56.0
	Total	51,203	39.3	Total	79,099	60.7

TABLE II: Question Types and Definitions

Question Type	Definition
Debugging	Questions for seeking help identifying and resolving errors or bugs in their code.
Conceptual Understanding	Questions for comprehending a concept's inner workings or principles.
Implementation	Questions for requesting guidance on implementing code features or algorithms.
Code Review	Questions for requesting guidance to assess code quality, readability, and maintainability.
Best Practices	Questions for recommendations on adhering to software development best practices.
Learning Resources	Questions refer to learning resources to acquire knowledge in a particular subject.

non-STDQs, the top three tags assigned to the questions are "security," "encryption," and "cryptography."

The security tags with the highest proportion of STDQs are "security," "websecurity," and "sql-injection" (44.4%, 44.0%, and 43.0%, respectively). At the same time, "passwords," "password-protection," and "encryption" have a lower proportion of STDQs (27.7%, 33.8%, and 35.0%, respectively). Therefore, the "security," "websecurity," and "sql-injection" tags have a higher proportion of STDQs.

Another finding is that the ranking of security tags based on the number of questions for STDQs and non-STDQs is generally identical (we discussed this further in Section V), except for the tags "xss" and "csrf." These two tags have switched positions in the STDQs and non-STDQs, with "xss" ranking higher in the STDQs and "csrf" ranking higher in the non-STDQs.

Summary $\mathbf{RQ_2}$ (Security Tags Distribution) The most frequent security tags in STDQs and non-STDQs are "security," "encryption," and "cryptography." The security questions tagged with "security," "websecurity," and "sql-injection" are more likely to indicate suboptimal security practices. The rank of security tags between STDQs and non-STDQs is, in most cases, comparable, with the exception of "xss" and "csrf" tags.

C. $\mathbf{RQ_3}$: *What are the different characteristics of security-related TD questions?*

The characteristics of the STDQs and non-STDQs are compared in Table III. The table lists the features (e.g., question length, compound score, and post score), mean and median scores across STDQs and non-STDQs.

Our study's results show that the median question length of STDQs is 112 words. Conversely, the non-STDQs are characterized by shorter median question lengths with 67 words. This indicates that STDQs are lengthier than non-STDQs.

TABLE III: Comparative Analysis of Different Aspects Between TD and non-TD Security-Related Questions

	STDQs		non-STDQs	
	Mean	**Median**	**Mean**	**Median**
Question Length (Words)	132.39	112.00	79.29	67.00
Negative Score	0.04	0.03	0.04	0.03
Positive Score	0.10	0.09	0.09	0.08
Neutral Score	0.86	0.86	0.86	0.87
Compound Score	0.43	0.69	0.26	0.40
Post Score	3.47	1.00	2.44	1.00
Time to Answer (Hours)	378.22	1.19	391.03	1.07

The median neutral sentiment scores were 0.86 and 0.87 for the STDQs and non-STDQs, respectively. Positive sentiment in both questions has lower scores (median sentiment score = 0.09), while negative sentiment is even lower (median sentiment score = 0.03). The median compound score (the normalized sum of positive, neutral, and negative scores) for STDQs is 0.69, higher than the median for non-STDQs (0.40). Overall, the sentiment in STDQs and non-STDQs are comparable, indicating almost no difference in sentiment from users when asking security questions regardless of the presence of suboptimal security practices indications.

The mean post score (to indicate the usefulness of a SO question post) for STDQs is 3.47, and the median score is 1.00. In contrast, non-STDQs have a mean post score of 2.44 and a median score 1.00. This shows that the median post score of STDQs and non-STDQs is the same, indicating the same attention from SO users.

Finally, the median time to answer STDQs is 1.19 hours,

while the median time needed to answer non-STDQs is 1.07 hours. According to these findings, the time for a question to have an accepted answer is comparable for the two questions.

Summary $\mathbf{RQ_3}$ (Different Aspects of Questions) Compared to non-STDQs, STDQs require more words. The median compound score for STDQs is higher compared to non-STDQs. STDQs have a slightly higher mean post score and higher median time to answer.

D. $\mathbf{RQ_4}$*: What are the user profiles asking security-related TD questions?*

TABLE IV: Comparative Analysis of User Profiles Asking TD and non-TD Security-Related Questions

	STDQs		non-STDQs	
	Mean	**Median**	**Mean**	**Median**
User Profile Age	9	10	8	9
Reputation Score	4377	469	2597	193
Profile Views	478	59	276	36
Badges	Silver	Bronze	Silver	Bronze

The results for RQ_4 are presented in Table IV. It compares the user profiles of those who asked STDQs and non-STDQs, showing the mean and median values for account age, reputation score, profile views, and each group's most common badge type.

The median account age for asking STDQs is ten years old. In contrast, those who asked non-STDQs had a slightly lower median account age of 9 years. Users who asked STDQs had a higher median reputation score, with 469 scores for STDQs and 193 scores for non-STDQs. Similar patterns emerged in profile views, with STDQs users reporting greater median (59 views) profile views than non-STDQs users (36 views). Bronze is the most prevalent (median) badge type for both categories.

Summary $\mathbf{RQ_4}$ (User Profiles) Compared to non-STDQs, users who ask STDQs tend to be more experienced, have higher reputation ratings, and have more profile views. Both groups have a similar distribution of badges.

E. Qualitative Analysis: Question Types

In this section, we report the results of the question types, which we described in Section III-E as the nature of STDQs and non-STDQs obtained through provisional coding [43] from the types given by Allamanis et al. [44]. The distribution of question types (defined in Table II) for STDQs and non-STDQs is shown in Table V. Since a security question can have multiple question types, we can assign multiple question-type labels to the question. However, the ranking of the question types for STDQs and non-STDQs differs. Most STDQs are of "Implementation" type. Contrarily, "Conceptual Understanding" is the most common type for non-STDQs. Other question types like "Debugging," "Best Practices," "Code Review," and "Learning Resources" are less common in both questions.

Figure 2 further summarizes our analysis of question types per security tag. Most security tags in STDQs have "Implementation" as their question types, except for "websecurity" and "passwords." In contrast, most security tags in non-STDQs have "Conceptual Understanding" as the question type, except for "passwords," "csrf," and "encryption."

Summary Question Types Implementation is the most discussed question type in STDQs. In contrast, conceptual knowledge questions are the main focus of non-STDQs. Other question types are less frequent in both STDQs and non-STDQs.

F. Qualitative Analysis: Security Topics

In this section, we report the results of the security topics described in Section III-E as the topics of STDQs and non-STDQs obtained through the open coding process. These security topics are different from the SO security tags. A comparison of the distribution of security topics for STDQs and non-STDQs within each security tag is summarized in Table VI. Due to page limitations, we only show the results for some of the security topics.

The distribution of security topics for STDQs and non-STDQs in each security tag differs, indicating that the two types of inquiries have different priorities. The emphasis on practical implementations is typically more significant in STDQs, e.g., "Prevention" techniques for "csrf," "sql-injection," and "xss." Non-STDQs focus more on conceptual elements, e.g., "Token" for "csrf," and have a more uniform distribution across topics.

Non-STDQs with the "cryptography" tag emphasize "Signature" more than STDQs, which prioritize "Algorithm." Similarly, STDQs with the "encryption" tag focus more on "File." In contrast, non-STDQs are more focused on "Algorithm."

Summary Security Topics Each security tag emphasized STDQs and non-STDQs differently. "Prevention" in "csrf," "xss," and "sql-injection" is highlighted in STDQs. "Algorithm" and "File" are given priority in STDQs for "cryptography" and "encryption."

V. Discussion

Prevalence of TD in Security-Related Questions. Previous studies investigated TD-related questions on SO [23], [24]. In our study, we specifically observed STDQs. Developers are voicing concerns about potentially poor implementations or design choices as they solve various security-related problems. Our results also reveal that developers only implicitly convey their concerns about suboptimal security practices during the software development process rather than mentioning "technical debt" explicitly when asking security-related questions.

Indications of suboptimal security practices in security questions can be attributed to the nature of the SO website and the intertwined relationship between security and TD. Developers use SO to improve their code quality and ensure best practices are being followed [45]. At the same time, previous studies [17], [20], [46]–[48] showed that the relationship between suboptimal implementations or design choices in software systems is strongly correlated with security vulnerabilities.

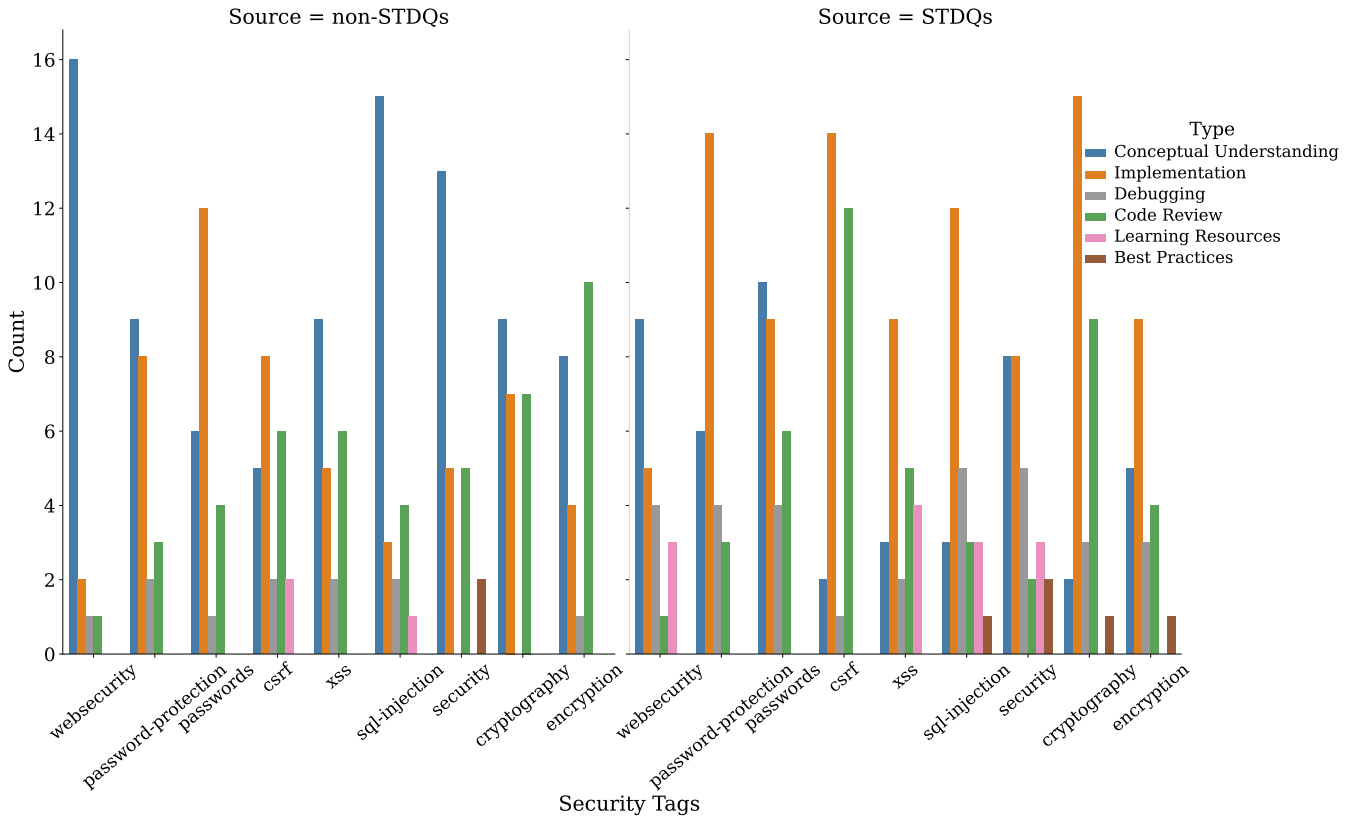

Fig. 2: Distribution of Question Types for each Security Tag

TABLE V: Comparison of Question Types between TD and non-TD Security-Related Questions

Rank	STDQs		non-STDQs	
	Type	**No. of Questions**	**Type**	**No. of Questions**
1	Implementation	95	Conceptual Understanding	90
2	Conceptual Understanding	48	Implementation	54
3	Debugging	45	Debugging	46
4	Best Practices	31	Best Practices	11
5	Code Review	13	Code Review	3
6	Learning Resources	5	Learning Resources	2
Total		237		206

The nature of the SO website and the intertwined relationship result in developers asking STDQs to gain insights and find effective solutions, albeit suboptimal security practices, which are implicitly (and potentially unconsciously) indicated in the security questions.

The absence of explicit TD mentions in all STDQs can be attributed to a lack of awareness among developers of the term TD. The latter is still unfamiliar to almost 45% developers, as suggested by Ramač et al. [49]. As per the studies by Kozanidis et al. [24] and Alfayez et al. [23], the low number (only 500) of questions on SO questions, irrespective of domain, explicitly mentioning TD, confirms that the TD metaphor is unfamiliar to many practitioners.

Security Topics in STDQs and non-STDQs. Table I displays the distribution of security tags in STDQs and non-STDQs. The ranking of security tags for STDQs and non-STDQs is almost identical. One potential reason can be attributed to the overlapping concerns between TD and security in software. Since many security-related decisions involve trade-offs between functionality, maintainability, and security, security-related issues involve overlapping concerns with TD [17], [20], [46]–[48]. This overlap contributes to the similarity in the results between STDQs and non-STDQs, as developers need to address the same topics when managing security, regardless of the presence of TD.

The “encryption” and “cryptography” security-related tags are the second and third most frequently appearing in STDQs and non-STDQs. This appears to be caused by the fact that “cryptography” and “encryption” are among the most popular security tags on SO [3], [50], regardless of whether the queries

TABLE VI: Comparison of Security Topics between TD and non-TD Security-Related Questions

Security Tags	Security Topics	No. of Questions	
		STDQs	non-STDQs
csrf	Prevention	19	8
	Token	9	13
	Concept	2	2
sql-injection	Detection	2	5
	Concept	4	8
	Prevention	22	7
	Simulation	0	3
xss	Detection	1	1
	Prevention	17	7
	Testing	0	3
	Concept	2	5
	Simulation	0	3
cryptography	Signature	6	11
	Algorithm	22	7
	Key	1	0
encryption	Algorithm	9	18
	File	14	8
	Server and Client Side	6	6
	Key	4	3
	IP Address	0	2
	Kubernetes	0	4
	Mobile	1	2

are TD or non-TD related. The complexity of cryptography principles, which have been known to present difficulties for developers attempting to understand them [51], could be another reason for their predominance. Our qualitative analysis of the security topics for "cryptography" and "encryption" tags revealed a prominent lack of understanding of developing cryptography algorithms and understanding concepts related to signatures and file encryption. Our findings align with the findings of the study of Hazhirpasand et al. [51], which also reported those security topics as the main hurdles in Cryptography for developers.

A more in-depth qualitative examination of the data revealed that web vulnerabilities associated with three security tags, namely "xss," "csrf," and "sql-injection" have significantly more STDQs. Additionally, our analysis shows that STDQs with those tags emphasize addressing preventative measures. Previous studies [52]–[54] show that implementing effective prevention strategies is challenging for developers. This difficulty makes them discuss suboptimal solutions and design choices, resulting in more STDQs.

Characteristics of Questions and Users. By quantitatively comparing various aspects of STDQs and non-STDQs, we observed that STDQs tend to use more words than non-STDQs. From a subsequent qualitative analysis, we note that STDQs mainly request guidance on implementing specific features, algorithms, or techniques within the code written by the users, thus requiring more words and details to describe the context. In contrast, non-STDQs mainly ask short, general, and open-ended questions to comprehend the inner workings or principles behind a concept, language, or technology. As a result, non-STDQs are shorter, as they do not need as many words to elaborate on the details or need to include code. We conclude that STDQs are more complex and challenging. Non-STDQs instead focus primarily on conceptual understanding, thus requiring fewer words. We observe that the median word count in our STDQs is comparable with the results of the study by Kozanidis et al. [24] and the standard length of SO questions [55]. Our analysis of STDQs length reveals a consistent pattern across numerous research efforts, highlighting the underlying complexity of STDQs.

Our quantitative analysis also shows that users asking STDQs have higher reputation scores and more profile views. This result can be attributed to the nature of STDQs, as they are more complex and challenging to comprehend for newer users. Thus, higher reputation users are more aware of incurring TD and the potential implications of such actions on security issues and are more likely to include suboptimal security issues in their questions.

VI. Implications

For researchers. The findings of our study can be used to pinpoint the security areas that are highly intertwined with TD. Knowledge of these areas allows researchers to prioritize research and analysis, especially in critical areas of the security domain. Our results reveal that web vulnerability tags ("xss," "csrf," and "sql-injection") are prominent since they are associated with a higher proportion of STDQs than most security tags. This is primarily due to a need for knowledge of prevention techniques and indicates the need for a more in-depth investigation into developers' need for knowledge of web vulnerability prevention techniques. Leaving the developers to struggle with these web vulnerability prevention techniques will cause severe consequences to clients and companies [52]–[54]. Researchers and tool developers should focus on creating more comprehensive, user-friendly tools and methodologies for implementing web vulnerability prevention techniques. These tools can guide implementing prevention techniques to minimize or eliminate TD without compromising software security.

For practitioners. Practitioners with more experience are more aware and likely to participate in discussions about suboptimal security practices. The lack of awareness from less experienced developers urges more experienced developers to help less experienced team members understand how their decisions in security may affect incurring suboptimal security practices. In order to increase the quality of the software-generated, experienced developers should mentor and direct less experienced developers in recognizing and managing suboptimal security practices by highlighting their potential implications on security issues.

For educators. Our study found a lack of explicit mention of the term "technical debt" in STDQs. This result means there is a lack of awareness among developers of the term TD in the security domain. Thus, security courses and seminars

should be structured to educate about suboptimal security practices and TD concepts. With better-structured courses and seminars, we will raise the awareness of developers of TD and any suboptimal practices in the security domain to reduce security vulnerabilities in software development.

VII. Threats to Validity

Despite our best efforts, validity threats could impact the findings of this study. Following Runeson et al. [56] classification, we consider the following factors to discuss the potential risks to this study and the relevant mitigation measures taken.

Construct Validity is regarding how well our operational measures are suited to respond to our RQs. Our datasets of security-related questions were extracted based on a set of predefined security tags by Yang et al. [3]. Since the SO tagging system is user-generated, there may be inaccuracies and inconsistencies in how posts are tagged in the used dataset. To mitigate this threat, two authors manually inspected a statistically significant sample of the security question posts, as described in Section III-C, to verify the content. The manual validation stages could be a potential threat to our study. Although we manually validated the classification model predictions, the process might have been prone to human error and bias. To mitigate this threat, two authors labeled the sampled datasets by following the guidelines prepared by Kozanidis et al. [24]. Disagreements among raters were resolved through discussion until an agreement was reached.

Internal Validity is regarding how much the "treatment," and not other factors, are responsible for the observed findings. The most relevant threat to the internal validity of our study regards the classification model used to identify STDQs. The model trained to identify STDQs is based on the RoBERTa model [37], which achieved an F1 score of 0.75. This suggests the dataset may contain false positives and negatives that could influence the outcomes. Future research can reduce this risk by enhancing model performance with more training datasets.

External Validity is regarding the scope of generalizability of our findings. In this study, the analyzed data was collected using SO. Therefore, the findings from our study might lack generalizability, as there are other platforms, such as Software Engineering Stack Exchange[8], where developers share their thoughts about security and TD. However, as also stated in the study by Kozanidis et al. [24], among the well-known group of Stack Exchange Question and Answer (Q&A) sites, SO is the most used programming Q&A website with over 24M questions and 20M registered users. Given its size and popularity, we deem SO as the best pool of data where developers discuss topics related to technical debt and security vulnerabilities in software development. Another threat to our study lies in using the dataset presented by Kozanidis et al. [24] to train our classification model. In their study, the authors documented that they deliberately constructed their automated query as encompassing as possible to filter out only the TD-related question posts on SO. Nevertheless, their findings might not reflect all TD-related queries featured on SO. Since we use their TD-related question post dataset to train our model, this threat is also cascaded to our study. We can reduce this threat in future studies by developing our own dataset.

Conclusion Validity is regarding the link between outcomes and our treatments. Our dataset of security-related questions was extracted based on a set of predefined security tags by Yang et al. [3]. However, there might be more security-related tags than those nine security tags. This threat will impact the number of extracted security-related questions, thus also impacting the results and conclusions of our study. Additionally, the model trained to identify STDQs only achieved an F1 score of 0.75. The model was imperfect, suggesting the identification of STDQs to contain false positives and negatives that could influence the results and conclusions of our study. The manual coding analyses could be another threat to our study. We inspected a statistically significant random sample from the STDQs and non-STDQs to better comprehend the security topics and question types in the dataset. Sampling bias might be present, causing the sampled STDQs and non-STDQs not to represent the whole population. Additionally, the manual coding procedure might incorporate the subjective judgment of the authors. Despite the collaborative review and editing process between the two authors, there could still be biases or inconsistencies that affect the conclusions.

[8]www.softwareengineering.stackexchange.com

VIII. Conclusions and Future Work

In this study, we explored the different characteristics of STDQs. We used a deep-learning approach to identify STDQs. The classification model revealed that 38% of security questions on SO are STDQs. The most recurrent tags among the STDQs emerged as "security" and "encryption." The latter typically have a neutral sentiment, are lengthier, and are posed by users with higher reputation scores.

For future works, we plan to extend the scope of our analysis to other platforms, such as Github or other Q&A platforms. Analyzing data from other platforms would provide a more comprehensive understanding of the relationship between TD and security. Another interesting avenue of research is to enhance our model's capability in detecting STDQs by expanding the training datasets and using more sophisticated models. Lastly, we would like to perform further correlation studies to study the relationship between STDQs and user profiles.

Acknowledgment

This research is partly supported by an NSERC Collaborative Research and Training Experience (CREATE) grant on Software Analytics at the University of Saskatchewan and the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on Telecommunications of the Future" (PE0000001 - program "RESTART").

REFERENCES

[1] S.-F. Wen and B. Katt, "Learning software security in context: An evaluation in open source software development environment," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ser. ARES '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3339252.3340336

[2] H. Assal and S. Chiasson, "Security in the software development lifecycle." in *SOUPS@ USENIX Security Symposium*, 2018, pp. 281–296.

[3] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What security questions do developers ask? a large-scale study of stack overflow posts," *Journal of Computer Science and Technology*, vol. 31, pp. 910–924, 2016.

[4] E. Moradian, "Security of e-commerce software systems," *Agent and Multi-Agent Systems in Distributed Systems-Digital Economy and E-Commerce*, pp. 95–103, 2013.

[5] R. A. Khan, S. U. Khan, H. U. Khan, and M. Ilyas, "Systematic literature review on security risks and its practices in secure software development," *ieee Access*, vol. 10, pp. 5456–5481, 2022.

[6] R. Khan, "Secure software development: a prescriptive framework," *Computer Fraud & Security*, vol. 2011, no. 8, pp. 12–20, 2011.

[7] H. Michael and L. Steve, "The security development lifecycle: Sdl: A process for developing demonstrably more secure software," 2006.

[8] D. Geer, "Are companies actually using secure development life cycles?" *Computer*, vol. 43, no. 6, pp. 12–16, 2010.

[9] I. A. Tondel, M. G. Jaatun, and P. H. Meland, "Security requirements for the rest of us: A survey," *IEEE software*, vol. 25, no. 1, pp. 20–27, 2008.

[10] P. Kruchten, R. L. Nord, and I. Ozkaya, "Technical debt: From metaphor to theory and practice," *IEEE Software*, vol. 29, no. 6, pp. 18–21, 2012.

[11] N. S. Alves, T. S. Mendes, M. G. de Mendonça, R. O. Spínola, F. Shull, and C. Seaman, "Identification and management of technical debt," *Inf. Softw. Technol.*, vol. 70, no. C, p. 100–121, feb 2016. [Online]. Available: https://doi.org/10.1016/j.infsof.2015.10.008

[12] E. Lim, N. Taksande, and C. Seaman, "A balancing act: What software practitioners have to say about technical debt," *IEEE Software*, vol. 29, no. 6, pp. 22–27, 2012.

[13] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt," *J. Syst. Softw.*, vol. 86, no. 6, p. 1498–1516, jun 2013. [Online]. Available: https://doi.org/10.1016/j.jss.2012.12.052

[14] G. McGraw, "Software security: Building security in," *Datenschutz und Datensicherheit-DuD*, vol. 36, no. 9, pp. 662–665, 2012.

[15] M. Siavvas, E. Gelenbe, D. Kehagias, and D. Tzovaras, "Static analysis-based approaches for secure software development," in *Security in Computer and Information Sciences: First International ISCIS Security Workshop 2018, Euro-CYBERSEC 2018, London, UK, February 26-27, 2018, Revised Selected Papers 1*. Springer International Publishing, 2018, pp. 142–157.

[16] M. Siavvas and E. Gelenbe, "Optimum checkpoints for programs with loops," *Simulation Modelling Practice and Theory*, vol. 97, p. 101951, 2019.

[17] M. Siavvas, D. Tsoukalas, M. Jankovic, D. Kehagias, A. Chatzigeorgiou, D. Tzovaras, N. Anicic, and E. Gelenbe, "An empirical evaluation of the relationship between technical debt and software security," in *9th International Conference on Information society and technology (ICIST)*, vol. 2019, 2019.

[18] R. Kuhn, M. Raunak, and R. Kacker, "Can reducing faults prevent vulnerabilities?" *Computer*, vol. 51, no. 7, pp. 82–85, 2018.

[19] F. Camilo, A. Meneely, and M. Nagappan, "Do bugs foreshadow vulnerabilities? a study of the chromium project," in *Conf. on MSR*, 2015, pp. 269–279.

[20] C. Izurieta, D. Rice, K. Kimball, and T. Valentien, "A position study to investigate technical debt associated with security weaknesses," in *Proceedings of the 2018 International Conference on technical debt*, 2018, pp. 138–142.

[21] M. Siavvas, D. Tsoukalas, M. Jankovic, D. Kehagias, and D. Tzovaras, "Technical debt as an indicator of software security risk: a machine learning approach for software development enterprises," *Enterprise Information Systems*, vol. 16, no. 5, p. 1824017, 2022.

[22] K. Rindell and J. Holvitie, "Security risk assessment and management as technical debt," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*. IEEE, 2019, pp. 1–8.

[23] R. Alfayez, Y. Ding, R. Winn, G. Alfayez, C. Harman, and B. Boehm, "What is asked about technical debt (td) on stack exchange question-and-answer (q&a) websites? an observational study," *Empirical Software Engineering*, vol. 28, no. 2, p. 35, 2023.

[24] N. Kozanidis, R. Verdecchia, and E. Guzman, "Asking about technical debt: Characteristics and automatic identification of technical debt questions on stack overflow," in *Proceedings of the 16th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 45–56. [Online]. Available: https://doi.org/10.1145/3544902.3546245

[25] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, pp. 619–654, 2014.

[26] C. C. Silva, M. Galster, and F. Gilson, "Topic modeling in software engineering research," *Empirical Software Engineering*, vol. 26, no. 6, p. 120, 2021.

[27] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, pp. 1192–1223, 2016.

[28] M. U. Haque, L. H. Iwaya, and M. A. Babar, "Challenges in docker development: A large-scale study using stack overflow," in *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2020, pp. 1–11.

[29] E. Gama, S. Freire, M. Mendonça, R. O. Spínola, M. Paixao, and M. I. Cortés, "Using stack overflow to assess technical debt identification on software projects," in *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, 2020, pp. 730–739.

[30] R. Croft, Y. Xie, M. Zahedi, M. A. Babar, and C. Treude, "An empirical study of developers' discussions about security challenges of different programming languages," *Empirical Software Engineering*, vol. 27, pp. 1–52, 2022.

[31] T. Lopez, T. Tun, A. Bandara, L. Mark, B. Nuseibeh, and H. Sharp, "An anatomy of security conversations in stack overflow," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 2019, pp. 31–40.

[32] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.

[33] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams engineering journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[34] A. Abdellatif, D. Costa, K. Badran, R. Abdalkareem, and E. Shihab, "Challenges in chatbot development: A study of stack overflow posts," in *Proceedings of the 17th International Conference on Mining Software Repositories*, ser. MSR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 174–185. [Online]. Available: https://doi.org/10.1145/3379597.3387472

[35] K. Stakoulas, K. Georgiou, N. Mittas, and L. Angelis, "An analysis of user profiles from covid-19 questions in stack overflow," in *25th Pan-Hellenic Conference on Informatics*, ser. PCI 2021. New York, NY, USA: Association for Computing Machinery, 2022, p. 419–424. [Online]. Available: https://doi.org/10.1145/3503823.3503900

[36] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[37] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[38] W.-C. Chang, H.-F. Yu, K. Zhong, Y. Yang, and I. S. Dhillon, "Taming pretrained transformers for extreme multi-label text classification," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 3163–3171. [Online]. Available: https://doi.org/10.1145/3394486.3403368

[39] P. Rajapaksha, R. Farahbakhsh, and N. Crespi, "Bert, xlnet or roberta: the best transfer learning model to detect clickbaits," *IEEE Access*, vol. 9, pp. 154 704–154 716, 2021.

[40] K. Rajanbabu, I. K. Veetil, V. Sowmya, E. Gopalakrishnan, and K. Soman, "Ensemble of deep transfer learning models for parkinson's disease classification," in *Soft Computing and Signal Processing: Proceedings of 3rd ICSCSP 2020, Volume 2*. Springer, 2022, pp. 135–143.

[41] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Proceedings of the international AAAI conference on web and social media*, vol. 8, no. 1, 2014, pp. 216–225.

[42] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto, "Sentiment analysis for software engineering: How far can we go?" in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 94–104.

[43] J. Saldaña, "The coding manual for qualitative researchers," *The coding manual for qualitative researchers*, pp. 1–440, 2021.

[44] M. Allamanis and C. Sutton, "Why, when, and what: analyzing stack overflow questions by topic, type, and code," in *2013 10th Working conference on mining software repositories (MSR)*. IEEE, 2013, pp. 53–56.

[45] L. An, O. Mlouki, F. Khomh, and G. Antoniol, "Stack overflow: A code laundering platform?" in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2017, pp. 283–293.

[46] R. L. Nord, I. Ozkaya, E. J. Schwartz, F. Shull, and R. Kazman, "Can knowledge of technical debt help identify software vulnerabilities?" in *CSET@ USENIX Security Symposium*, 2016.

[47] S. J. Oishwee, Z. Codabux, and N. Stakhanova, "An exploratory study on the relationship of smells and design issues with software vulnerabilities," in *Proceedings of the 1st International Workshop on Mining Software Repositories Applications for Privacy and Security*, 2022, pp. 16–20.

[48] K. Z. Sultana, Z. Codabux, and B. Williams, "Examining the relationship of code and architectural smells with software vulnerabilities," in *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2020, pp. 31–40.

[49] R. Ramač, V. Mandić, N. Taušan, N. Rios, M. G. de Mendonca Neto, C. Seaman, and R. O. Spínola, "Common causes and effects of technical debt in serbian it: Insightd survey replication," in *2020 46th euromicro conference on software engineering and advanced applications (seaa)*. IEEE, 2020, pp. 354–361.

[50] T. Lopez, T. T. Tun, A. Bandara, M. Levine, B. Nuseibeh, and H. Sharp, "An investigation of security conversations in stack overflow: Perceptions of security and community involvement," in *Proceedings of the 1st international workshop on security awareness from design to deployment*, 2018, pp. 26–32.

[51] M. Hazhirpasand, O. Nierstrasz, M. Shabani, and M. Ghafari, "Hurdles for developers in cryptography," in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021, pp. 659–663.

[52] S. Shalini and S. Usha, "Prevention of cross-site scripting attacks (xss) on web applications in the client side," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 4, p. 650, 2011.

[53] R. Kombade and B. Meshram, "Client side csrf defensive tool," *International Journal of Information and Network Security*, vol. 1, no. 3, p. 171, 2012.

[54] D. Kar and S. Panigrahi, "Prevention of sql injection attack using query transformation and hashing," in *2013 3rd IEEE International Advance Computing Conference (IACC)*. IEEE, 2013, pp. 1317–1323.

[55] F. Calefato, F. Lanubile, and N. Novielli, "How to ask for technical help? evidence-based guidelines for writing questions on stack overflow," *Information and Software Technology*, vol. 94, pp. 186–207, 2018.

[56] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, pp. 131–164, 2009.