Analysis of Mobile Security Discussion Topics On Stack Overflow

ANA ARAUJO, University of Hawaii at Manoa, USA
JAKE IMANAKA, University of Hawaii at Manoa, USA
TIMOTHY HUO, University of Hawaii at Manoa, USA

Investigation of mobile security-related issues and trends is required, since this information may serve as a guide for security practitioners, educators, and researchers that have an interest in mobile development. Thus, we will undertake an extensive investigation into Stack Overflow questions regarding mobile security in order to answer the stated demand. We will use Stack Overflow, a well-known online question-and-answer platform, as software engineers often use it to connect, work together, and exchange knowledge. One of the many distinct themes among the numerous questions posted on Stack Overflow is mobile security. In this study, we will aim to answer the questions of what mobile security-related topics do developers face, how trends in mobile security topics change over time, and which mobile security software has the most difficult questions to answer on Stack Overflow.

Additional Key Words and Phrases: Stack Overflow, topic modeling, security, software engineering, mobile development

ACM Reference Format:
Ana Araujo, Jake Imanaka, and Timothy Huo. 2023. Analysis of Mobile Security Discussion Topics On Stack Overflow. 1, 1 (January 2023), 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Mobile application development is one of the fields with the fastest growth rates—faster than web development, according to Barua et al. Nevertheless, studies that look at the security issues that developers confront do not focus on mobile platforms. Since Android and iPhone development account for 71% and 28% of the market share [25], respectively, they will be the main topics of our study. Without the billions of lines of code that support this digital environment, we could not have imagined a connected world where millions of people use mobile devices for banking, shopping, staying connected, and self-education. In this complex programming structure, it is very difficult to avoid opening gaps in the code, paving the way for vulnerability. With the number of cyber-attacks increasing, it is more important than ever to focus on mobile app security standards. Mobile application developers must identify all security vulnerabilities and verify that the application is free from malware and virus attacks.

To accomplish this, developers frequently turn to question-and-answer websites like Stack Overflow to get assistance from their peers regarding the technical difficulties they encounter. In this project, we propose examining the postings' content to identify the main subjects of developer debate, their trends, and the relationships between them. Understanding these topics could be helpful to a wide range of people, including educators (who could use the results as a guide for what to cover when teaching students how to develop mobile applications securely), security researchers (who could use our replication package and the discussion topics results as representations of

Authors' addresses: Ana Araujo, acoa@hawaii.edu, University of Hawaii at Manoa, Honolulu, Hawaii, USA; Jake Imanaka, jimanaka@hawaii.edu, University of Hawaii at Manoa, Honolulu, Hawaii, USA; Timothy Huo, thuo@hawaii.edu, University of Hawaii at Manoa, Honolulu, Hawaii, USA.

XXXX-XXXX/2023/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

real-world mobile security issues), and developers of all levels (for the development of better automated security vulnerability detection tools). Additionally, users can utilize our method to find posts with related material so they can read every question and response that pertains to their problem.

Previous research subjects have covered challenges with mobile development [15], refactoring trends and topics [9], and the typical security questions developers ask [27]. Similar techniques were employed in all of them, which included mining Stack Overflow posts to produce a data set; topic modeling using Latent Dirichlet Allocation (LDA), a well-liked statistical model for searching and grouping data to identify trends and topics [2], [9]; carrying out various statistical analyses on the dataset; and establishing the context of Stack Overflow discussion topics discovered using LDA to generate topic clusters for further analysis.

2 MOTIVATIONS

While mobile development is a subject that is constantly changing and improving, there are still many security issues. The overwhelming variety of methods for building mobile applications may expose end users to potential security issues. The open discussions between developers on Stack Overflow are an indication of the demand for mobile security investigation. Mobile security posts on Stack Overflow have been surging in popularity, and researchers have brought attention to the need for more investigations in this area. There are no studies dedicated solely to analyzing mobile development security topics, despite the number of publications that aim to understand trends in Stack Overflow threads. Besides that, according to Yang at el, mobile security posts rank on the higher end in terms of the level of difficulty and average time span of 12.54 days for an answer to be submitted and accepted. Investigation into mobile security-related issues and trends is therefore necessary because this data could act as a beneficial resource for security researchers, educators, and developers.

For our analysis, we use Stack Overflow as a dataset source. It is one of the most well-known websites for software information, where software developers discuss software development and maintenance, ask questions about a variety of topics, and seek assistance on technical issues. The site saves the queries and responses, which search engines can access. As a result, the website serves as a knowledge base for different development needs. Understanding and analyzing Stack Overflow posts could help us gain important knowledge about the issues with mobile security. Stack Overflow has been the subject of several recent investigations. Barua et al. conducted a large empirical study on Stack Overflow to analyze the topics and trends of what developers talk about.

3 GOALS AND RESEARCH QUESTIONS

3.1 Goals

Investigating security-related mobile trends and issues can help security researchers and developers by providing them with useful information. In this work, we thoroughly examine Stack Overflow's security-related questions to address the demand above. Understanding developers' common problems and/or misunderstandings with relation to security-related concerns in iOS and Android is the main objective of our current study. We use data from Stack Overflow, which is frequently used by developers to identify and address technical problems and concerns in the software development process, to do a mixed-method analysis on the data in order to look into this issue.

3.2 RQ1: How do trends in security topics change over time?

As technologies evolve, so too do their related security vulnerabilities and topics. To keep up with the constantly changing security landscape, it is important to analyze how it has changed in the past. This research question seeks to examine how the volume of mobile security questions, and their topics, has changed over time on Stack Overflow. Identifying these patterns may give developers ideas of how security issues can change in the future as new technologies are introduced.

3.3 RQ2: What security challenges do mobile developers face?

Many different frameworks, languages, libraries, and other tools are available to developers to meet today's demand in the mobile application market. However, the overwhelming variety of methods for building mobile applications may expose end-users to potential vulnerabilities and other security issues. This research question allows us to understand the security challenges mobile developers encounter, and discover patterns and groupings amongst the different security topics. For example, determining the most common and popular cybersecurity topics among mobile software on Stack Overflow. These results can help researchers and practitioners pinpoint major security problems and explore the causes of these security challenges and methods to mitigate them.

3.4 RQ3: Which mobile softwares have the most popular and difficult security questions on Stack Overflow?

The objective of this question is to comprehend the kinds of concerns developers have about mobile security that is difficult to respond to. Since developers frequently discuss different aspects of mobile security on Stack Overflow, we think it is critical that readers comprehend these complex issues and perhaps help with solutions. This RQ draws in part on the findings of the first RQ, which sought to comprehend the community's mobile security challenges. We will examine the questions for which there are no answers, questions for which there are no accepted responses, and the median time it takes the Stack Overflow community to come up with a satisfactory response to a question.

4 BACKGROUND AND RELATED WORKS

Six relevant papers were reviewed and their relevant information is summarized below. We define three categories of relevant information as it pertains to our research: mobile security, Stack Overflow, and Data Collection and Analysis Methods. (More may be added in the future)

4.1 Mobile Security

Mobile devices and applications have become integrated into our daily lives including dealing with monetary transactions and storing sensitive information. This correlation however attracts hackers and raises the probability of security threats and invading users' privacy. An indication of the demand for mobile security investigation are the open discussions between developers on Stack Overflow. Mobile security post on Stack Overflow has been surging in popularity, and researchers have brought attention to the need for more investigations in this area [2, 9, 11].

Barua, Thomas, and Hassan [2] found that mobile platforms surpass web development in popularity, and presented concerns about mobile security. The authors noted there is limited academic research today surrounding mobile applications and believe having additional security measurements and information could increase developers' confidence in the security and reliability of applications.

Yang, Lo, Xia, Wan, and Sun [27] categorized Mobile security as one of the five main categories covered in the research. Amongst the topics corresponding to the five categories, app access in mobile security posts ranked on the higher end in terms of the level of difficulty and average time span of 12.54 days for an answer to be submitted and accepted. The authors brought up the recent popularity growth in mobile security, showing the need to investigate more into mobile security trends and challenges.

Fischer et. al. [11] found that 15.4% of all Android applications (1.3 million total) contained security-related code snippets from Stack Overflow and 98% of those had at least one insecure code snippet. This result shows the current security state in android development and raises questions if the same security-related code issues occur in other mobile development software.

Although Linares-Vasquez et al.'s [15] work does not focus on the security of mobile applications, our work could still benefit from their methodology. For example, [6] used tags such as android and iOS to filter threads on Stack Overflow that are related to mobile. In our case, we could also include tags such as 'security' only to see security-related threads.

4.2 Stack Overflow

Stack Overflow (SO) is one of the most popular question-answer-based community discussion platforms for developers to discuss and share programming knowledge. Users on the platform can ask questions in the form of discussion threads and other users are able to respond to further the discussion or post a possible solution. User responses can be 'upvoted' if other users agree, or 'downvoted' if they disagree. The original poster of a thread also has the ability to accept a suggested answer if they believe a user's response provides a sufficient solution.

Due to its popularity, accessibility, and wide range of topics, Stack Overflow has been widely used in past academic research. Example past research endeavors that used Stack Overflow are Barua et al.'s [2] research to analyze software development discussion topics, Yang et al.'s [27] research to analyze software security discussions, Peruma et. al.'s [9] research to analyze software refactoring trends, Fischer et al.'s [11] research on the negative impact of Stack Overflow on Android application security, Rahman's [17] research on the security challenges software developers face, and Linares-Vásquez et al.'s [15] research into mobile development issues.

As Stack Overflow has proven to be a rich, and acceptable, source of data to analyze trends in software development, we will also use data from Stack Overflow to answer our research questions about mobile software security.

4.3 Data Collection and Analysis Methods

The mixed-method approach of doing quantitative and quantitative steps seems to be the consensus for analyzing discussion topics on Stack Overflow. More specifically, the qualitative steps of querying Stack Overflow for data, preprocessing the data using NLP techniques, and performing topic modeling using LDA, and the quantitative step of manually analyzing the resulting topics to assign context and cluster topics is popular among similar research projects. This study is dissimilar from these studies as it is an investigation of mobile security specifically; however, these methods may prove useful as we are also investigating discussion topics on Stack Overflow.

5 METHODOLOGY

5.1 Dataset

The dataset for this study was based on Stack Overflow posts relating to mobile security. Our approach was to query posts by tags and titles using Stack Exchange's Data Explorer.

5.1.1 Data Explorer. Stack Exchange's Data Explorer[1] is the official source for collecting Stack Overflow data. It allows users to compose and run SQL queries in the Stack Exchange network such as Stack Overflow, Server Fault, and more. Data Explorer provides a diagram and list of the Stack Overflow schema to query specific information. The results from a query can be saved or downloaded as a CSV file.

5.1.2 Queries. In total, three queries were used to construct the final dataset: a query to extract mobile security questions, a query to extract every answer from the questions returned in the first query, and a query to extract only accepted answers for the questions returned in the first query [1]. The extracted dataset ranges from the launch of Stack Overflow in September 2008 to the time when the dataset was collected in October 2022. The dataset that will be used consists of 5786 questions, and 4886 answers — 2371 of those are accepted answers.

[1]https://data.stackexchange.com/

Table 1. Literature Review Summary

Study	Approach	Finding
Barua et al. [2]	Topic modeling with LDA on a data dump of SO.	Provides a method for analyzing discussion topics on SO and provides an approxamation of the wants and needs of developers.
Peruma et. al.'s [9]	Mixed approach of quantitative and qualitative methods. Quantitative methods included traditional statistical analysis and LDA.	Shows that SO is a popular resource for developers to find help for refactoring challenges and, while statically typed languages see the most activity, dynamically typed languages are increasing in popularity.
Fischer et al.'s [11]	Mined and analyzed code snipped from SO, filtered for security-related snippets, and classified snippets as 'insecure' or 'secure' using ML (SVM).	Found that 15.4% of all 1.4 million Android applications on the Play Store contained security-related code snippets from SO, and 97.6of those contained insecure code snippets. Although it is still uncertain whether SO should be considered harmful, it still contributes a significant amount to production code in Android apps.
Linares-Vásquez et al.'s [15]	Topic modeling with Gibbs sampling implementation of LDA on a filtered dataset by mobile technologies tag of SO.	Found majority of developers who contributed to an acceptable answer on Stack Overflow prefer to work in only one mobile platform. Also, identified specific topics that are more prone to getting accepted answers than others.
Rahan [17]	Topic modeling with LDA on a filtered dataset by security and other related tags of SO.	Provided an overview of security related topics that developers ask on SO.
Yang et al. [27]	Topic modeling with LDA on an extracted dataset using heuristcs of SO.	Investigated and categorized security-related questions into mainly five categories, web security, mobile security, cryptography, software security and system security.

Each of the queries consist of the same filter or SQL WHERE clauses when querying posts by tags and title. The tags chosen for these queries consist of keywords including ios, android, mobile, and security. The reasoning behind the inclusion of iOS and Android tags—specific mobile operating systems—is that some people may use the tags "android" or "ios" when they refer to a framework, library, or application that is only compatible with one type of mobile operating system, such as the development environment, Android-studio, which is exclusive to the Android operating system. Moreover, we take into account answers in our study in addition to question posts for a few different reasons. The first thing to note is that the answers make up a sizable percentage of the posts' data. Second, one of our research objectives is to find which software has the most difficult security questions on Stack Overflow, meaning that we must examine the answers and upvotes.

5.1.3 Tags and Title. Querying Stack Overflow posts by tags and title were chosen instead of the body to reduce false positives on our dataset. On Data Explorer, the posts' Tags type is a string that consists of tags wrapped in '<>' as separators to distinguish themselves. The string format allowed us to manipulate the query to form a more accurate result. For all of our queries, we selected specific columns of the Posts table, including Id, PostTypeId, AcceptedAnswerId, ParentId, CreationDate, Title, Tags, AnswerCount, CommentCount, Body. The condition for querying tags was that the post must have '%security%' plus '%<android%' or '%<ios%' anywhere in the Tags value. The percentage sign(%) surrounding the tag indicates a wildcard search, where the searched phrase could appear anywhere in a string on the percentage sign side. The less-than sign(<) indicates that the tag must start

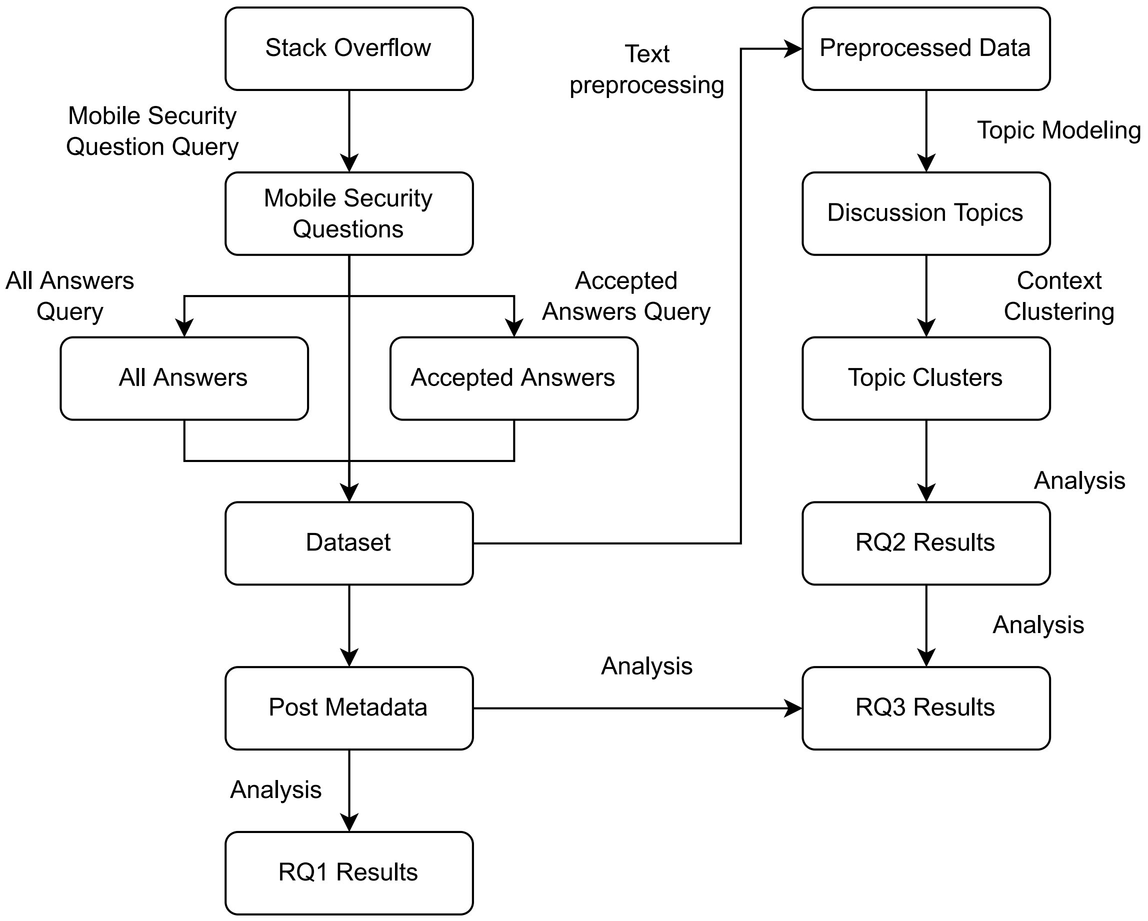

Fig. 1. Methodology Flow Chart

with the word 'android' or 'ios' to exclude false positives because tags distinguish themselves with '<>'. For example, when searching for posts for Android and iOS without the less-than sign, posts with tags such as 'kiosk' appear because they have the keyword 'ios' in the word. These posts may not be completely associated with mobile security. By keeping the tag search open at the end of the word for '%<android%' and '%<ios%', the query can capture tags such as 'android-security'. Searching for '%security%' and only the '%<android>%' tag would exclude posts with only the 'android-security' tag which has over 600 associated posts.

The condition for querying titles is similar, where the post Tags value contains the word '%security%' anywhere and looks if the Title contains the word '% security %' plus '% android %' or '% ios %'. The spacing between the title searches helps exclude any false positive titles similar to less-than sign for the tag query.

5.2 Preprocessing

After the data is available through the process of querying posts on StackOverflow, we move to the preprocessing step. Data preprocessing is necessary before continuing to remove inaccuracies, inconsistencies, and unusable sections of data. Similar to previous research on StackOverflow topic modeling, our preprocessing steps included noise removal, expanding contractions, tokenization, lowercasing, and stop word removal [2, 9, 27]. All preprocessing steps were performed using the Python programming language with various natural language processing packages explained in the steps below.

5.2.1 Noise Removal. When it comes to Natural Language Processing (NLP), one of the first things we consider is noise removal to improve text analysis. Noise can be eliminated in many ways. It can be done by eliminating punctuation, special characters, digits, URLs, HTML formatting, domain-specific keywords, source code, headers, white space, single characters, and other elements that are all included in this [6].

5.2.2 Expanding Contractions. Eliminating contractions helps to normalize language since they are words or combinations of words that are abbreviated by omitting letters and substituting them with apostrophes. In our situation, we will experience contractions. There are a variety of ways to expand contractions, but one of the simplest is to compile a list of contractions and their two-word equivalents. Below, the term "you're" has been expanded to "you are" in the example, which needs to be dealt with before subsequent normalization. Example: dictionary = "you're": "you are". In order to do that, the Contractions[2][19] python package will be utilized. It is a python library used during text preprocessing to expand shortened words with apostrophes into normalized text.

5.2.3 Tokenization. Tokenization is also critical for text analysis. It enables models to comprehend lengthy phrases by breaking them up into more manageable chunks[6]. Tokenization helps readers understand the text's meaning by looking at the word order in the text. For example, it can turn the string "xcode is not functioning" into the words "xcode", "is", "not", and "working." For this task, the NTLK[3][22] package will be used.

5.2.4 Lowercasing. One of the easiest and most efficient forms of text preprocessing is to lowercase ALL the text. It can considerably improve the consistency of expected output. The sparsity problem is resolved by lowercasing when the same words with various cases map to the same lowercase form, as shown in the following example: "Mobile", "mobile", and "MOBILE", all map to mobile "mobile".

5.2.5 Stop word Removal. A group of frequently used words in a language are called stop words. The words "in," "the," "of," and others are examples of stop words in English. The idea behind it is that by eliminating words with little meaning from the text, we may concentrate on the keywords instead. Furthermore, stop word lists can be made specifically for a website, such as Stack Overflow, or taken from pre-existing sets. During the text analysis stage, we intend to start by looking at the frequency of words. The stop words list will contain words with a high frequency that won't be useful for our analysis. Moreover, as programming languages also have their own words as part of the syntax – commonly known as reserved words, such as "print" and "None" for Python, it is necessary to remove those as well [9]. Pre-existing stop word lists were provided by the NTLK[2][22] python package.

5.3 RQ1: How do Trends in Mobile Security Topics Change Over Time?

This research question aims to find what types of mobile security topics and issues developers face over time by conducting a quantitative study on the extracted dataset. This research question is composed of two sub-research questions examining different attributes of Stack Overflow posts over time. RQ1.1 investigates the volume of mobile security questions and if the posts have an associate accepted answer to them. RQ1.2 investigates the growth of the common tags developers used for mobile security. The python package pandas[4] was used to visualize the results.

[2]https://pypi.org/project/contractions/
[3]https://www.nltk.org/
[4]https://pandas.pydata.org/

5.3.1 RQ1.1: How has the growth of mobile security posts changed over time? The purpose of this sub-research question is to examine the volume of mobile security posts. Also, to map and analyze questions with accepted and unaccepted answers. To examine the growth of mobile security posts, we will filter the posts by the year it was created from the extracted dataset. Then, we break down the set into three groups, total questions, questions without an accepted answer, and questions with an accepted answer.

5.3.2 RQ1.2: What tags do developers use to label posts as mobile security over time? The purpose of this sub-research question was to examine common tags developers use to label posts and investigate the usage of the tags over time. We first count the number of occurrences of each tag used in the extracted dataset. Then we find the top 5 appeared tags and for each tag, we will gather all the associated posts and compare their growth over time. Next, we get a sample size of all the distinct tags in the extracted dataset and perform a manual cross examination between the three authors to label those tags into specific categories.

5.4 RQ2: What Security Challenges do Mobile Developers Face?

This research question aims to find the main security challenge topics on Stack Overflow. We investigate the tags used by Stack Overflow to categorize each question as a starting point for measuring the primary conversation themes. However, these tags have a number of shortcomings:

(1) When a user posts a question, they choose the tags, which might result in incorrect tags.
(2) Since tags may only be added to question postings, a significant portion of the content remains untagged.
(3) Tags might not be able to convey anything more specific than a broad understanding of the subject post.

For this reason, we use an n-gram method to examine the typical words developers use to describe their issues or challenges. We then conducted a topic modeling study to pinpoint the major subjects connected to mobile security questions. The python packages used to perform this step include Matplotlib[5], Wordcloud[6], Genism[7], pyLDAvis[8], Joblib[9], and Scikit-learn[10] [4, 7, 21, 29]. Finally, we carefully examine a significant sample of questions from each topic manually determine the context and significance of the underlying topics that were discovered. But before topic modeling and n-gram analysis, the data is subjected to a number of preprocessing operations as mentioned above, including the expansion of word contractions, removal of URLs, stop words, etc.

Moreover, we introduce our own collection of stop words, which is specific to StackOverflow, beside the standard set of stop words provided by NLTK. Words such as "Thanks" and "Help" are specific Stack Overflow stop words. For our topic modeling analysis, we use the Latent Dirichlet Allocation (LDA) algorithms, as they have been shown to be very effective in other research papers [9]. LDA groups similar words together into clusters, where each cluster denotes a subject. The quantity of topics to be created is a necessary input for the LDA algorithm. Thus, it is necessary to find the optimal number of topics for the quantity of topics in our dataset, which can be achieved by visualization and manual analysis.

5.5 RQ3: Which Mobile Security Topics Have the Most Difficult Security Questions on Stack Overflow?

This research question aims to comprehend how difficult each mobile security discussion topic is to answer by conducting a quantitative study based on the extracted Stack Overflow mobile security dataset [1], and the discussion topic results from RQ2. We first group all security questions by the topic modeling results, then

[5]https://matplotlib.org/
[6]https://amueller.github.io/word_cloud/
[7]https://radimrehurek.com/gensim/
[8]https://pyldavis.readthedocs.io/en/latest/
[9]https://joblib.readthedocs.io/en/latest/
[10]https://scikit-learn.org

measure each questions' level of difficulty by analyzing various metrics about the questions and the questions' answers. In total, we use 4 metrics to measure the difficulty of each question:

(1) AnswerCount = # of answers
(2) AAnswerCount = # of accepted answers
(3) ΔT_{A} = time taken for the first answer
(4) ΔT_{AA} = time taken for poster to accept an answer

AnswerCount, AAnswerCount, and CommentCount will be determined by examining the 'AnswerCount', 'AcceptedAnswerId', and 'CommentCount' fields for each question (Q) respectively. AnswerCount and Comment-Count will be the exact values from the dataset, while AAnswerCount will range from 0 - 1 as there can only be one accepted answer.

$$AnswerCount = Q.AnswerCount$$

$$AAnswerCount = \left\{ \begin{array}{ll} 1, & \text{if } Q.AcceptedAnswerId \neq NULL \\ 0, & \text{if } Q.AcceptedAnswerId = NULL \end{array} \right\}$$

The time metrics ΔT_{A} and ΔT_{AA} will be calculated by taking the difference of the questions' 'CreationDate' and the answers'/accepted answer's (A/AA) 'CreationDate'.

$$\Delta T_{\mathrm{A}} = \text{Q.CreationDate} - \text{A.CreationDate}$$

$$\Delta T_{\mathrm{AA}} = \text{Q.CreationDate} - \text{AA.CreationDate}$$

5.6 Results Analysis

Results from our methods will be analyzed using a mixed-method approach. RQ1, RQ2, and RQ3 will each have quantitative analysis steps. The quantitative analysis steps for each RQ are as follows: RQ1–the visualization and analysis of trend data, RQ2–the topic model analysis, and RQ3–the analysis of topic metrics. Additionally, RQ1 and RQ2 also include qualitative analysis steps including manual analysis of question tags and discussion topics. More details of the results are provided in section 6.

6 RESULTS

6.1 RQ1: How do Trends in Mobile Security Topics Change Over Time?

6.1.1 RQ1.1: How has the growth of mobile security posts changed over time? The extracted dataset consists of 5,786 questions that were broken into three categories, questions with accepted answers, questions with answers, but not accepted by the author, and questions with no answers. There were 2,371 (40.98%) questions with accepted answers, 2,187 (37.8%) questions with answers but not accepted, and 1,228 (21.22%) with no answers.

Figure 2 shows a yearly visualization of mobile security questions and answers. The red bar represents all questions, including questions consisting of accepted answers, answers not accepted, and no answers. The count of mobile security answers corresponds to the creation date of the question and not the date of the answer. The green bar represents questions with accepted answers, the orange bar represents questions with answers not accepted, and the blue bar represents no answers to the question. The number of no-answer questions continues to increase in percentage corresponding to all questions per year. Also, the number of accepted answers was consistently above the number of answers that were not-accepted until 2016. After 2016, the relationship between accepted and not accepted answers was the opposite, where there were more not-accepted answers than accepted answers.

Figure 3 shows the year-over-year growth rate of mobile security posts of the 4 categories, all questions, with accepted answers, with no accepted answers, and with no answers. The figure starts from 2010 because it compares the number of questions from the previous year to the current year. The four categories show similar

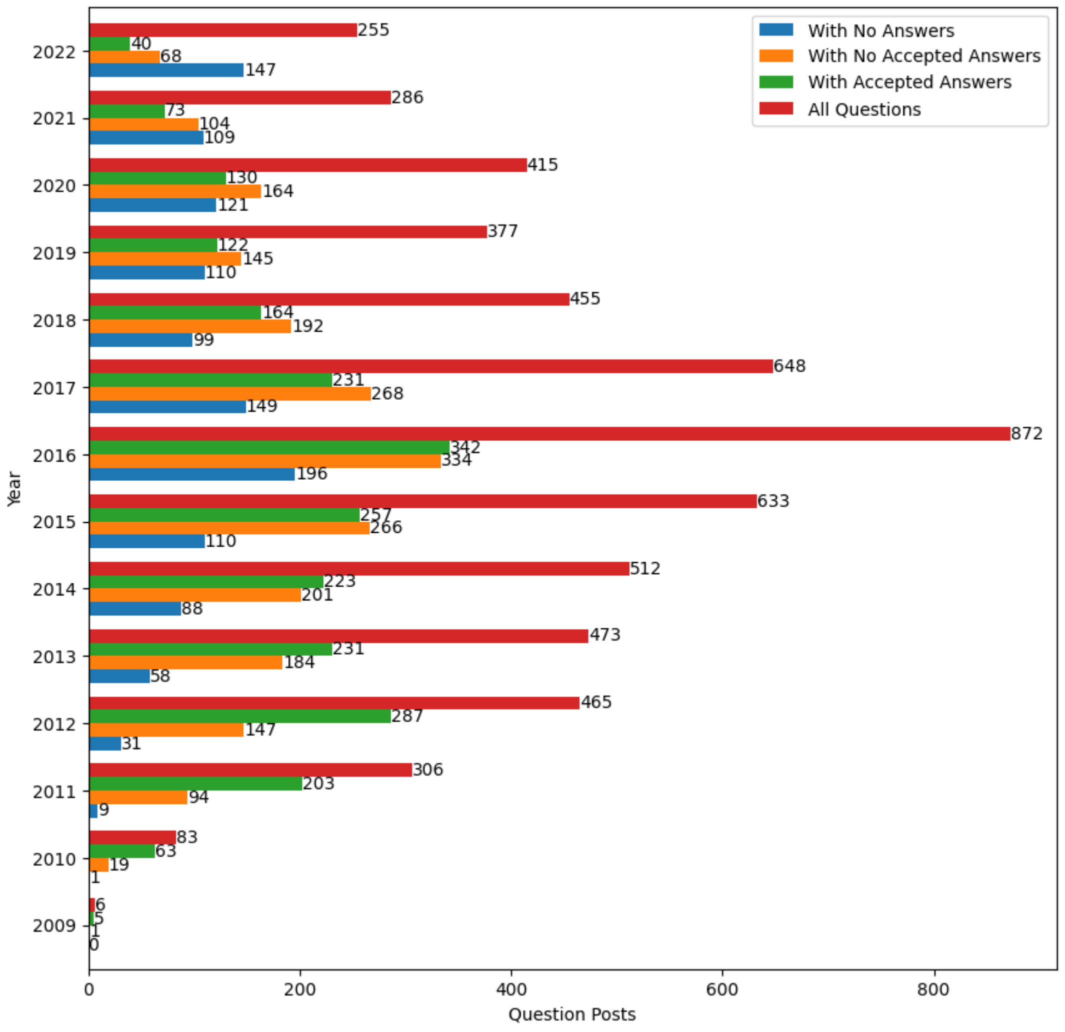

Fig. 2. Count of mobile security questions, accept answers, answers and no answers, per year

growth throughout the years with the no answers categories slightly above the others. Both figures show an increase in growth from 2009 to 2011, as the number of questions grew exponentially.

6.1.2 RQ1.2: What tags do developers use to label posts as mobile security over time? Next, we looked at the individual tags developers used to label mobile security. In our extracted dataset, there were a total of 1,796 unique tags. Figure 4 shows a yearly visualization of the five most used tags. We calculated the number of occurrences tags that appeared in each post and selected the five most used tags Android, iOS, Security, Android-Security, and Java. Similar to figure 2, from 2015 to 2016, there was a spike in the number of posts. Figure 4 narrows down the type of posts, as only the Android, iOS, and Android-Security tags rose while Security and Java continued on a steady trend. Also, Android-Security was used as a tag until 2015, representing the increase in the popularity

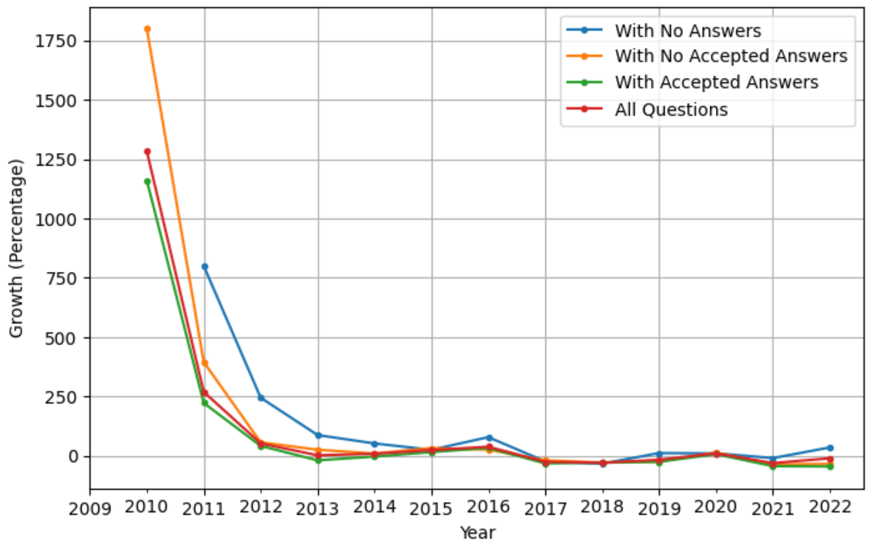

Fig. 3. Year-over-year growth of mobile security posts

of Android security development. However, like figure 2, the number of questions for each of the selected tags began to drastically decrease after 2016. This observation does not directly align with the increase in popularity of mobile development and would require more research to come to an accurate conclusion.

To determine the volume and types of categories the tags relate to, we manually examined and labeled 413 distinct tags contributing to a confidence level of 99% and a confidence interval of 5%. There were 10 categories that included framework/library/API, language, tool, security concept, security features, operating system, in-app feature, software engineering concepts, hardware, and other. We did a cross-examination between the there authors and labeled the tags in their respective categories. If there was a disagreement on a labeled tag, the majority would take priority and would be discussed between the authors.

Figure 5 shows the percentage of categorized tags used by developers in our extracted dataset. The top category Framework/Library/API had 101 labeled tags and made up almost a quarter (24.46%) of related tags in the dataset. Examples of tags in this category include mobile frameworks (e.g., cordova) and security libraries (e.g., android-jetpack-security). The second most category is Security Concepts with 89 (21.55%) labeled tags, examples include keychain, passwords, fingerprint, digital-signature. The difference between the Security Concept and Security Feature categories is Security Concepts focus on security terms and vocabularies. While Security Feature with 40 (9.69%) labeled tags, focusing on security-specific implementation (e.g., transport layer security, ssl-certificate, and rest-security). The Tool category consists of IDE (e.g., xcode, eclipse), databases (e.g., sqlite, firebase), and others that developer use. The operating system mainly consists of different android and ios versions. The Software Engineering Concept category looked at programming implementations (e.g., sockets, cookies) and terms (e.g., api, rest), making up 36 (8.7%) labeled tags. The Other category also had 36 (8.7%) labeled tags, these tags were broad (e.g., service, random) or did not relate to many other tags or categories. The next two categories focus on user interaction, In-App Features with 21(5.08) labeled tags and Hardware with 15(3.63) labeled tags. Examples of In-App Features include mobile apps (e.g., google-play, facebook) and features (e.g., in-app-purchase). Examples of Hardware include screen-capture and camera. The category with the least amount of labeled tags

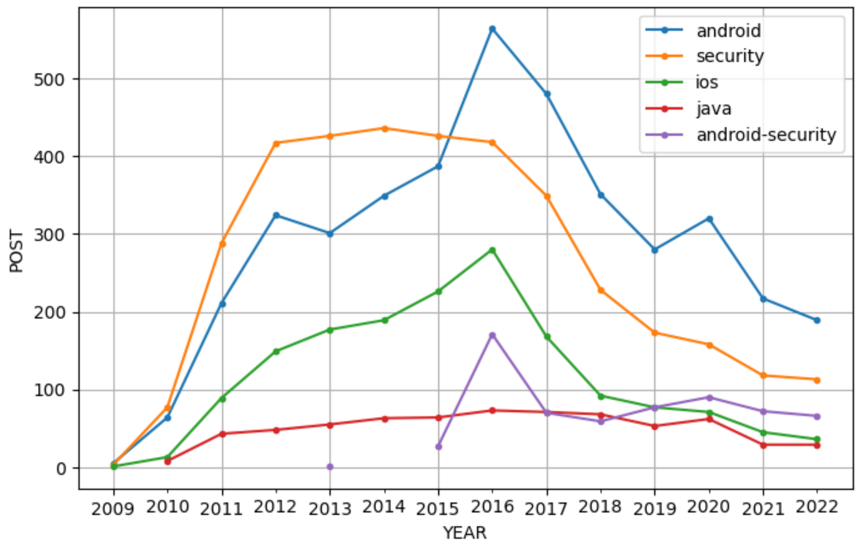

Fig. 4. Yearly growth of popular mobile security tags from the extracted dataset

with 14 (3.4%) labeled tags is Language, which includes common coding languages (e.g., python, c#), mobile coding languages (e.g., swift), and database languages (e.g., mysql).

6.1.3 ***RQ1 Conclusion****.* Our trend analysis showed a period of growth in the number of overall questions from 2009 to 2011, coupled with a spike in 2016. Tag analysis during these periods also supported this result showing an initial rapid growth from 2009 to 2011 with a spike in 2016. Overall, of the 5 most popular mobile security tags, the Android, iOS, and Android-Security tags grew while the Security (after the initial growth period) and Java tags stayed relatively stable until 2016. The number of questions and the growth of tags begins to decrease after the 2016 spike, which does not correlate with the increase in popularity of mobile development. Manual review of the tags also resulted in 10 specific tag categories: framework/library/API, language, tool, security concept, security features, operating system, in-app feature, software engineering concepts, hardware, and other.

6.2 RQ2: What Security Challenges do Mobile Developers Face?

6.2.1 RQ2.1: What are the common words developers use while discussing mobile security? In Table 2, we list the top ten bigrams that appear the most in question posts. The ‘google play’ (also known as ‘play store’) App Distribution Service is shown in this table to be crucial to mobile security. The bigrams ‘public key’ and ‘private key’ are more examples that highlight how crucial encryption is to security operations. The bigrams ‘username password’ and ‘firebase database’ highlight a common reason why developers request support with the security of their database and authentication. We further interpret unigrams in the following question and offer examples of posts from which they were taken.

6.2.2 RQ2.2: What concerns do developers have about mobile security? Our LDA study shows that the best model produces seven topics that are related to Stack Overflow mobile security questions: General Security, Secured Communications, Database, App Distribution Service, Encryption, Permissions, and File-Specific. According to our findings, General Security is the subject that comes up the most often (3,125 questions, or 53.89%). We display the distribution for each dataset topic and display a subset of words (unigrams) connected to each subject in

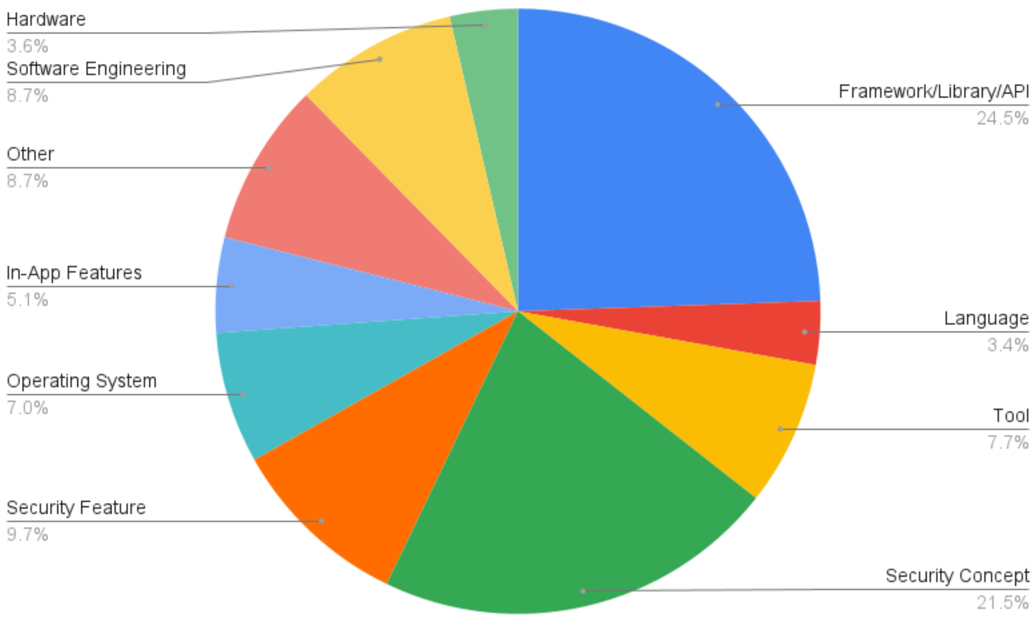

Fig. 5. Percentage of categorized areas of tags

Table 2. Top ten bigrams for question postings that are used frequently.

Bigram	Count	Percentage
google play	339	1.35%
public key	241	0.96%
private key	228	0.91%
web service	219	0.87%
api key	177	0.70%
username password	168	0.67%
play store	124	0.49%
firebase database	121	0.48%
server side	118	0.47%
access token	111	0.44%
Others	23,328	92.67%

Table 3. The name of a topic can be determined by looking at the words that appear in it, but the topic's meaning cannot be fully understood by doing this and the issues developers face on mobile security are not indicated by these topic names alone. As a result, we manually analyze a selection of questions that are statistically significant. A sample of 343 postings was created, and three authors independently examined them. The dataset was then shared for final evaluation once the writers had annotated it with the topic's justification. Finally, we are able to

Table 3. LDA topics with a sample set of their relevant words and their frequency of recurrence.

Topic	Count	Percentage	Unigrams
General Security	3125	53.89%	application, android, server, device, secure, api, token, client, access, user.
Secured Communications	845	14.57%	certificate, error, ssl, http, transport, load, ats, ios, https, network.
Database	673	11.61%	database, firebase, rule, user, write, username, firestore, users, read, db.
App Distribution Service	394	6.79%	play, google, vulnerability, cordova, warning, version, implementation, apk, unsafe, console.
Encryption	329	5.67%	key, keystore, encryption, string, private, public, encrypt, decrypt, rsa, aes.
Permissions	253	4.36%	permission, activity, spring, denied, infoplist, intent, realtime, admin, manifest, javalangsecurityexception.
File-Specific	180	3.10%	file, line, folder, directory, images, uri, path, gradle, pdf, config.

provide a summary of our findings, the main problem(s), and typical examples from Stack Overflow posts in the form of quotes, to help put topics in context.

6.2.3 ***General Security.*** Mobile security consists of various issues developers face, and many of these issues are unique to a specific need or broad concepts that can have multiple interpretations. The general security topic covers these types of issues that may not fit in a specific topic. Thus, the questions being asked in this topic vary greatly, and many of them overlap with other topics below. More specifically, in this topic some of the questions with higher score are related to App Transport Security (e.g. “How do I load an HTTP URL with App Transport Security enabled in iOS 9?”, [13]), screen capture (e.g. “How to prevent Screen Capture in Android?”, [3]), Swift (e.g. “How to obscure a UITextField password?”, [14]), and among others.

The general security topic is a problem for developers since they rely on the community for assistance in enhancing the security of the products they create. Finding methods to make hacking more difficult, for example, is a challenge that developers frequently encounter in achieving this goal. Our analysis of example postings revealed a number of recurring problems, such as Quote 1, where developers need to protect the assets, source code, and APK (Android Package Kit) file of the program by avoiding reverse engineering. When users are not protected, hackers may occasionally inject harmful code with the explicit purpose of stealing personal information.

How to avoid reverse engineering of an APK file

“How can I completely prevent the reverse engineering of an Android APK? How can I protect all the app’s resources, assets, and source code so that hackers can’t hack the APK file in any way? And is there a way to make hacking tougher or even impossible? What more can I do to protect the source code in my APK file?”

Quote 1: An example of a question on general security is soliciting assistance on how to protect source code in APK files and prevent reverse engineering [24].

Having a solid understanding of how to store access tokens and secrets in Android is the next issue we discovered in our research on sample posts (Quote 2). When it comes to encryption, the developer must either save the key to a file or require the user to repeatedly enter the decryption password, both of which might lead to security issues. Furthermore, employing tokens has several advantages. The password is kept secret even if a token is obtained, and tokens have a duration, expire after a predetermined period of time, and are subject to cancellation if it is believed that they have been hacked.

How to securely store access tokens and secrets in Android?

"I am going to use oAuth to fetch mail and contacts from Google. I don't want to ask the user each time to log in to obtain an access token and secret. From what I understand, I need to store them with my application, either in a database or SharedPreferences. But I am a bit worried about the security aspects of that. I read that you can encrypt and decrypt the tokens, but it is easy for an attacker to just decompile your apk and classes and get the encryption key."

Quote 2: An example of a question on general security is soliciting assistance to securely store access tokens and secret [28].

We now identify ten terms that are used frequently in the general security topic (see Table 3), such as 'device,' 'api,' 'token,' 'client,' 'access,' and 'user,' highlighting the fact that developers seek assistance in order to improve the security of the application's code and guarantee users' safety. After all, the user expects the software to uphold its commitments for data protection, just as the developer must be aware of the risks involved in doing its task. In conclusion, this topic shows that programmers mostly seek assistance in boosting the overall security of their products. Although it seems like many developers are aware of the potential repercussions of insufficiently secured apps, many don't know how various methods of enhancing end-user security work or which ones are the most efficient.

6.2.4 ***Secured Communications****.* The secured communications topic is among the top issues for developers, accounting for over 14% of all mobile security-related topics. One of the frequently encountered issues (such as Quote 3) is how to use NSURLConnection to connect with SSL for an untrusted certificate. SSL keeps internet connections secure by ensuring that any data exchanged among users and domains or between two systems is incomprehensible. Data is encrypted in transit using encryption techniques to obstruct spying by hackers.

How to use NSURLConnection to connect with SSL for an untrusted cert?

"I have a simple code to connect to a SSL webpage, however, it gives an: untrusted server certificate. Is there a way to set it to accept connections anyway (just like in a browser you can press accept) or a way to bypass it?"

Quote 3: An example of a question on communication security is soliciting assistance to use NSURLConnection to connect with SSL [10].

As we continue, we find certain terms connected to communication security when we look at the terms used frequently in this area (see Table 3), such as 'certificate,' 'error,' 'ssl,' 'transport,' 'ats,' 'https,' 'network', and others. And from there, we may draw the conclusion that programmers have trouble loading URL requests using class methods like NSURLConnection and identifying the problems to avoid errors like the untrusted server certificate.

6.2.5 ***Database****.* The database topic is a concern for developers as they go to the community for assistance with the various user authentication methods. Making one field in the database unique so that there is no trouble in the authentication process when the user registers is one of the common concerns we found in our analysis of example posts (such as Quote 4).

Firebase android: make username unique

"I need to implement a register process with 3 fields: email, username, password. Since Firebase is not providing an easy way to manage username, I decided to use only the email/password registration and save some additional data. But what I want to do is to make the username unique and to check it before creating an account."
Quote 4: An example of a question on database is asking help to make a field unique for the authentication process [12].

The next problem identified in our research on sample posts (Quote 5), which is also related to Firebase, is understanding how to write and read data for the user after authentication. The majority of programs include user authentication as one of their core features. The developer must take into account factors like password encryption, user registration, session control, and data security, among others. This capability is difficult to develop as a result of all these requirements. Thankfully, Firebase can help with that. The first step in adopting Firebase authentication is to enable the application's required login methods. For this example, only authenticated users will be able to read and write their own data.

Firebase - How to write/read data per user after authentication

"How to add user specific data after authentication. Can't it be seamless like when we don't have any restriction on read/write on a user basis, because I could easily read/write data. And is there any Firebase web view to visualize the database or see JSON data, where I can see/modify the data to/from an Android device?"
Quote 5: An example of a question on database soliciting assistance to allow the authenticated user to write and read his data [16].

In Table 3, which lists the terms that are often used in this topic, we discover various terms linked to databases, such as 'firebase,' 'rule,' 'user,' 'write,' 'username,' 'firestore,' 'read,' and among others. Thus, it follows that Firebase is a very well-liked user authentication tool in mobile development. Even though Firebase offers straightforward methods for user authentication and is a token-based authentication system that allows for simple integration with most platforms, developers still struggle to understand the login options it offers.

6.2.6 ***App Distribution Service.*** The app distribution service is a concern for developers as they ask for assistance from the public to launch and maintain their application in those digital stores. Some prevalent issues in our examination of sample posts (such as Quote 6) include how to avoid security alerts from the Google Play store. In this case, the implementation of onReceivedSslError ignores all SSL certificate validation errors, making the app vulnerable to man-in-the-middle attacks.

WebView: avoid security alert from Google Play upon implementation of onReceivedSslError

"I have a link which will open in WebView. The problem is it cannot be open until I override onReceivedSslError. I am getting a security alert from Google Play: unsafe implementation of the WebViewClient.onReceivedSslError handler. Is there any way I can open the page in WebView and avoid security alerts?"
Quote 6: An example of a question on app distribution service soliciting assistance to avoid security alerts from Google Play [5].

The following issue was also connected to unsafe implementation, more precisely of the HostanmeVerifier (Quote 7), but this one may possibly involve third-party libraries, as the responder suggested. Generally speaking, a skilled developer will take the codes from the third-party libraries and adequately test them. Codes that are unsafe will be found during testing. Once the code has been removed, the developer will look for replacements that won't compromise the security of their software. The use of secure codes will be ensured by a library that takes security seriously. Additionally, there must be clear documentation.

App is using an unsafe implementation of the HostnameVerifier

"Recently one of my apps got a security alert from Google Play: Your app is using an unsafe implementation of the HostnameVerifier. And refer a link to Google Play Help Center article for details regarding fixing and deadline of vulnerability. Anyone can explain with example about, what changes should I do to fix this warning."

Quote 7: An example of a question on digital distribution stores soliciting assistance on how to fix the unsafe implementation of the HostnameVerifier [20].

Looking at the terms used frequently in this topic (see Table 3), we find some words related to digital distribution stores, such as 'vulnerability,' 'cordova,' 'warning,' 'implementation,' 'unsafe,' and among others. And with that, we can reach the conclusion that developers find it challenging to handle unsafe implementation messages, in large part because they are frequently brought on by the use of unsafe third-party libraries. Given how much time these codes save developers, it is impossible to ignore the fact that they cannot completely cease using them. However, it's critical to be aware of any library issues that can let attackers execute malicious code to bring down the system.

6.2.7 ***Encryption.*** Encryption is a very important topic, despite being part of only 5.67% of the discussions about mobile security. Understanding how to implement common encryption algorithms – whether it is RSA, AES or others - is a common difficulty faced by developers. Some prevalent issues in our examination of sample posts (such as Quote 8) include generating a private key from a string in order to encode data with encryption algorithms. Data is transformed into an encrypted form by developers and only when it has been converted to the original form, it can be read. If there is a secret decryption key, it is possible to perform this translation and an attacker can easily obtain the data if the data is not encrypted properly. The creators of the programs typically employ highly robust encryption. The implementation of the data is not, however, the real issue. It has to do with how the developers handle key management.

Java: How can I generate PrivateKey from a string?

"I am trying to encode a message with SHA-1 and RSA, but I have no experience with security except some base information about RSA. I have been given a private key as String. I have managed to write the following code block to do the job, but I am not sure if I am doing the job securely and correctly. I am not an expert but putting my private key as String in code is not secure, I guess. Can anyone guide me?"

Quote 8: An example of a question on encryption soliciting assistance with the encoding of a message using SHA-1 and RSA [18].

From Table 3, we find some terms related to encryption, such as 'key,' 'keystore,' 'string,' 'private,' 'public,' 'encrypt,' 'decrypt,' 'rsa,' 'aes.' We can therefore deduce that even the most powerful data encryption methods won't protect the app if the keys are not secured. The encryption keys should never be kept in a file or database that is not safe and that may be accessed by numerous people. Hackers go the other way where they do not target the encryption itself but the keys. They can translate the data without any big problems after obtaining the key.

6.2.8 ***Permissions.*** Permissions topic is a concern for developers as they need help to improve the security of the programs they create. Developers frequently struggle with using permissions effectively and knowing what to do in the event of a security exception. Some prevalent issues (such as Quote 9) include understanding which permission to include in the manifest file for specific cases, such as calling an activity from another application to display notification. Moreover, an obstacle to securing mobile apps is that apps can be overly generic in the permissions they ask for, and users may not think to question this. Furthermore, it is also typical for developers to disregard the context of the acceptance request, using these opportunities to request that users access resources that are entirely irrelevant to the context. For instance, is it really necessary for a weather app on a phone to have access to the camera or microphone? In addition to having a poor usability standard, some applications request resources like the photo gallery and microphone before they are even required by the application, which might lead to a dangerous destination for information.

Android: java.lang.SecurityException: Permission Denial: start Intent

"I have created an application containing GWVectraNotifier activity which is called from other applications to display Notification. In the Notification dialog, there will be a 'show' button, where the activity will be started. To check the

functionality, I started the GWVectraNotifier activity from K9Mail application on checkmail event trigger. But I can't understand what permissions to include in my manifest file to access MessageList of k9Mail."
Quote 9: An example of a question on permissions soliciting assistance on what kind of permission to include in the manifest file [23].

The next problem identified (Quote 10) is understanding which android permissions are dangerous, and which are normal. Although user consent is required for an application to access a location, photo gallery, etc., protecting users' privacy is also a top priority for developers. An app's attack surface is widened when too much is allowed. The likelihood that a user may share potentially sensitive data inadvertently increases the more non-essential information a program obtains. In order to protect user privacy, it is crucial to comprehend how permissions can be exploited.

Android permissions: How can I learn which are dangerous vs normal?

"Android defines a set of permissions that third-party apps can request. Permissions are categorized by sensitivity; most permissions are either normal or dangerous. Normal permissions are granted automatically; dangerous permissions are presented to the user when the app is installed. How can I tell whether it is a normal permission or a dangerous permission? Is there a list of dangerous permissions and normal permissions?"
Quote 10: An example of a question on permissions soliciting clarification on which permissions are dangerous [8].

We now identify some terms connected to permissions, when we look at the terms used frequently in this area (see Table 3), such as 'activity,' 'spring,' 'denied,' 'infoplist,' 'intent,' 'manifest,' and others. And hence, we may draw the observation that this issue shows that although some developers are aware of the usage of permission but are unsure of what kind of permission should be applied to their context, others are informed but struggle to resolve security exceptions linked to it.

6.2.9 ***File-Specific***. The file-specific topic is part of only 3.10% of the discussions about mobile security on StackOverflow but is also a concern for developers. Understanding how to manage files is a common difficulty faced by developers. Some prevalent issues in our examination of sample posts (such as Quote 11) include knowing a secure format to store information, such as game preferences and saved games. The truth is that regardless of the storage technique used, the stored data can typically be compromised if a hacker is persistent enough and possesses the necessary skill set. It all comes down to the practical uses for the app and the time and effort invested in protecting the data.

Storing game preferences and saved games in a secure format

"I know that data can be stored into .plist file, .xml or .json, and even in a database. But all of that is non-encrypted plain text. What is considered as a secure format? And what else methods/classes/techniques can be used to store sensitive data?"
Quote 11: An example of a question on storing information in a secure file format [26].

Now, looking at the terms used frequently in this topic (see Table 3), we find some popular terms, such as 'folder,' 'images,' 'uri,' 'path,' 'gradle,' 'config', and among others. And therefore, we can draw the conclusion that this topic shows that developers have problems with the files used in mobile development. It's important to keep in mind that many of the questions on this topic also include the other topics stated above, such as permission, where the manifest file is used to add permission for the app. However, we must recognize that some developers could find it difficult to make programs secure through the usage of their files at some moment in the development process.

6.2.10 ***RQ2 Conclusion***. Performing topic modeling with LDA and manually labeling a statistically significant subset of questions resulted in 7 different security topics: General Security (53.89%), Secured Communications (14.57%), databases (11.61%), App Distribution Services (6.79%), Encryption (5.67%), Permissions (4.36%), and

File-Specific (3.10%). Common unigrams and bigrams were also examined to help give context to each topic. Example bigrams/unigrams can be seen in table 2 and 3.

6.3 RQ3: Which Mobile Security Topics Have the Most Difficult Questions on Stack Overflow?

Table 4 shows each topic's 'hardness' metric from the topic modeling results performed during RQ2.

6.3.1 ***Average Answers per Question.*** The Average Answers metric measures the average amount of answers per question for a given topic. Topics with higher scores are perceived as less difficult, while topics with lower scores are perceived as more difficult. The topics rated from best to worst are General Security (1.315), Permissions (1.285), Secured Communications (1.259), Encryption (1.207), Databases (1.135), App Distribution Service (1.056), and File Specific (1.050). Notably, the App Distribution Service and File Specific topics are the two worst scoring topics for this metric and are nearly tied with only a 0.006 difference between them.

6.3.2 ***Average Accepted Answers per Question.*** The Average Accepted metric measures the average amount of accepted answers per question for a given topic. Similar the the Average Answers metric, topics with a higher score are perceived as less difficult, while topics with lower scores are perceived as more difficult. The topics rated from best to worst are General Security (0.434), Databases (0.429), Permissions (0.407), Encryption (0.405), Secured Communications (0.386), File Specific (0.339), and App Distribution Service (0.279). This metric sees a shift in nearly all positions compared to the Average Answers per Question metric; however, File Specific and App Distribution Service are still the bottom two scorers. App Distribution Service's and File Specific's are noticeably different unlike the previous metric with a difference of about 0.06.

6.3.3 ***Average First Answer Time.*** The Average First Answer Time metric measures the average time taken for the first answer to question to be posted. Topics with low Average First Answer Time scores are considered less difficult, and topics with high scores are considered to be more difficult. The topics from best to worst scores are App Distribution Service (26.59), File Specific (35.25), General Security (35.32), Databases (44.57), Secured Communications (47.03), Permissions (53.33), and Encryption (62.65). Completely opposite to the previous metrics, App Distribution Service and File Specific achieved the two highest score for this metric. General Security stays near the top while Encryption and Permissions score the lowest.

6.3.4 ***Average Accepted Answer Time.*** The Average Accepted Answer Time metric measures the average time taken for the original poster of a question to mark an answer as 'accepted'. Similar to Average First Answer Time, topics with low scores are considered to be less difficult while topics with high scores are considered to be more difficult. The topics listed from best to worst scores are Databases (6.173), File Specific (21.60), General Security (22.68), Secured Communications (25.46), Encryption (30.24), Permissions (33.79), and App Distribution Service(38.89). For this metric, App Distribution Service is again on the bottom while Databases clearly has the best score with a difference of about 15 Days to the next closest topic.

6.3.5 ***RQ3 Conclusion.*** To determine which topic was the most difficult, questions from the dataset were categorized into specific topics and measured against four metrics–AVG answers, AVG accepted answers, AVG first answer time, and AVG accepted answer time. From looking at these metrics, we believe that the App Distribution Service topic is the most difficult as it achieves the worst scores, or ties for the worst score, in 3 out of 4 metrics. The App Distribution Service topic scores the worst in AVG answers, AVG accepted answers, and AVG accepted answer time, while also oddly achieving the top score in AVG first answer time.

Table 4. Difficulty Metrics Summary

Topic	AVG AnswerCount	AVG AAnswerCount	AVG ΔT_A	AVG ΔT_{AA}
General Security	1.315367	0.433750	35.320996	22.681986
Databases	1.135417	0.428571	44.565205	6.172950
Secured Communications	1.259215	0.386445	47.029933	25.456147
File Specific	1.050000	0.338889	35.247605	21.602213
Encryption	1.207317	0.405488	62.650562	30.243870
Permissions	1.284585	0.407115	53.332987	33.793748
App Distribution Service	1.056266	0.278772	26.591125	38.887821

7 THREATS TO VALIDITY

7.1 Construct Validity

For Research Questions 1 and 3, categorizing tags into areas and identifying topics from sample sets of posts were done manually. Bias may have been introduced when doing the manual human process, however, the three authors performed a cross-examination on categorizing tags and discussed and identified general topics from the identified general topic dataset from the LDA model.

7.2 Internal Validity

Internal Validity To obtain an extracted mobile security dataset from Stack Overflow, we relied on keywords to query posts such as security, ios, and android. Other mobile and synonyms of the selected keywords were not searched for, which could have excluded potential mobile security posts from our dataset. However, when searching for tags that contain the keywords, the query was adjusted to gather tags that had ios or android in the beginning and security anywhere within the tag with ios and android dominating the mobile market. In addition, the identified general topic dataset from the LDA model assigned all but four questions to a specific topic resulting in a total of 5,782 out of 5,786 questions having assigned topics. This discrepancy could be caused by the LDA model and will need to be investigated further.

7.3 External Validity

Although Stack Overflow is one of the most popular question-answer-based community platforms, the results may not be an overall representation of mobile security. There are other platforms such as android/ios forum threads, GitHub, and more that could provide different insights not captured by our dataset. However, the results from this study can be replicated to draw a more generalized conclusion about mobile security issues that developers face.

8 DISCUSSION

The findings from this study provide an analysis of the evolution of trends, a list of focused mobile security discussion topics, and a measurement of each topics' difficulty metrics for mobile security questions on Stack Overflow. This can be applied to new, and experienced, mobile developers, security researchers, and educators.

8.1 Software Developer Takeaways

8.1.1 ***Security focus during software development.*** The discussion topics presented in this study are generally more related to back-end development rather than issues in front-facing features. Encryption, Database Rules, Secure Communications, permissions, and App Distribution Service, for example, are back-end features that the end-user does not interact with. Thus, it is recommended that developers first focus on securing back-end features like their database and data encryption, then focus on front-end features.

8.1.2 ***Automated Tools.*** As seen in the RQ3 results, the app distribution topic is the most difficult in terms of the average number of answers, the number of accepted answers, and the time for posters to accept answers. From our manual review of a statistically significant subset of questions categorized under the App Distribution Service topic, it is clear that most questions are due to errors thrown by an automated security auditing tool when trying to upload apps to an application distribution platform (e.g. Google Play Store). Thus, rather than relying on built-in vulnerability detection tools on the Google Play Store or the Apple App Store, we recommend developers use separate detection tools throughout their development process to catch vulnerabilities before deployment. If developers find currently available detection tools to be unsatisfactory, we then also call for the development of better automated security vulnerability detection tools.

8.2 Security Research Takeaways

8.2.1 ***Research Focus Points.*** Similar to the software developer takeaways, our results also provide areas of focus for security researchers. The list of mobile security topics was developed using Stack Overflow and is thus a fair representation of what real developers are discussing about in terms of mobile security. Due to this, we recommend that future security research related to mobile software development and practices take into account the discussion topics provided as representations of real-world mobile security issues.

8.2.2 ***Future Research Avenues.*** The results from this research provide new unanswered questions like what influenced the specific patterns seen in the RQ1 results and why certain security topics are more difficult than others. To assist in any future research, a replication package including our dataset and analysis scripts is provided. More information is provided in the Future Works section below.

8.3 Educator Takeaways

8.3.1 ***Education focus Points.*** Again, similar to the previous takeaways, educators may also use our provided list of mobile security topics. Educators may use these security topics as a guide for what to cover when teaching students how to securely develop a mobile application as the topics represent real-world issues encountered by mobile developers of all levels. Teaching students these topics can help them create more secure applications–increasing securing as a whole.

9 SUMMARY

In our research, we carried out a comprehensive examination of Stack Overflow questions about mobile security and performed mixed-method analysis on the data. To be more precise, we collected data from Stack Overflow, preprocessed it using NLP methods, applied the data to topic modeling using LDA, and manually analyzed the obtained topics to give them meaning. Tags were used to understand the change over time in trends related to mobile security. The tags exclusively related to mobile development (such as Android, iOS, and Android-Security) had their highest peak in the year of 2016, while tags partially related to it (such as Security and Java) stayed relatively stable throughout the years as they are used in other fields. Besides that, one important point to note is that in 2016, there were a large number of questions, and afterwards, the number of questions decreased significantly. This phenomenon could be attributed to the popularity of mobile security in 2016 and questions getting a high number of upvotes, resulting in a lack of need to ask questions since they would be duplicates. LDA was used in order to figure out the security challenges mobile developers face. The best model produced seven topics, which were General Security, Secured Communications, Database, App Distribution Service, Encryption, Permissions, and File-Specific. We found that programmers primarily seek assistance to make their applications more secure. Programmers don't appear to understand the various methods for enhancing security for users, despite being aware of the negative effects of apps with weak security. Furthermore, security exceptions are challenging for developers to fix in large part because they are frequently brought on by the use of unsafe third-party libraries. Now, to find the topics that have the most difficult security questions, we used five metrics. Based on an examination of the metrics employed, we think that the app distribution service issue is the most challenging because it receives the lowest scores.

10 FUTURE WORK

For our future work, a possible research question is related to the change in volume of mobile security questions during and after the 2016 spike. It seems counterintuitive that there are fewer questions about mobile security today given that we think the industry of mobile development is expanding. Researchers may therefore gain new insight into mobile security by questioning what occurred in 2016 to generate the rise and why there was a decline in the growth of mobile security queries after 2016. The App Distribution Service topic was the most challenging, which raises concerns about whether the errors and warnings from automated security detection features offer enough details for developers to comprehend and resolve the problem. It will take more investigation to determine why this topic is the most challenging. Resolving the problem can give developers a clearer idea of why their application has been flagged for security flaws and what they can do to fix it.

REFERENCES

[1] Araujo A., Huo T., and Imanaka J. 2022. Stack Overflow Mobile Security [Replication Package]. https://github.com/iSQARE-Lab/StackOverflow_MobileSecurity Replication package.

[2] Barua A., Thomas S., and Hassan A. 2012. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering* 19, 3 (2012), 619–654. https://doi.org/10.1007/s10664-012-9231-y

[3] Anad. 2015. *How to prevent Screen Capture in Android.* Retrieved November 30, 2022 from https://stackoverflow.com/questions/28606689/how-to-prevent-screen-capture-in-android

[4] Mabey B. 2018. *Welcome to pyLDAvis's documentation! — pyLDAvis 2.1.2 documentation.* Retrieved October 29, 2022 from https://pyldavis.readthedocs.io/en/latest/

[5] captaindroid. 2016. *WebView: how to avoid security alert from Google Play upon implementation of onReceivedSslError.* Retrieved November 30, 2022 from https://stackoverflow.com/questions/36050741/webview-how-to-avoid-security-alert-from-google-play-upon-implementation-of-onr

[6] Jurafsky D and Martin J.H. 2009. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Pearson Education.

[7] Joblib Developers. 2008. Joblib: running Python functions as pipeline jobs — joblib 0.14.1.dev0 documentation. Retrieved October 29, 2022 from https://joblib.readthedocs.io/en/latest/

[8] D.W. 2011. *Android permissions: How can I learn which are dangerous vs normal?* Retrieved November 30, 2022 from https://stackoverflow.com/questions/7339743/android-permissions-how-can-i-learn-which-are-dangerous-vs-normal

[9] AlOmar E.A., Peruma A., Mkaouer M.W., Newman C.D., Ouni A., and Kessentini M. 2021. How do I refactor this? an empirical study on refactoring trends and topics in stack overflow. *Empirical Software Engineering* 27, 1 (2021). https://doi.org/10.1007/s10664-021-10045-x

[10] erotsppa. 2009. *How to use NSURLConnection to connect with SSL for an untrusted cert?* Retrieved November 30, 2022 from https://stackoverflow.com/questions/933331/how-to-use-nsurlconnection-to-connect-with-ssl-for-an-untrusted-cert

[11] Fischer F., Böttinger K., Xiao H.g, Stransky C., Acar Y., Backes M., and Fahl S. 2017. Stack Overflow Considered Harmful? The Impact of CopyPaste on Android Application Security. In *2017 IEEE Symposium on Security and Privacy (SP)*. 121–136. https://doi.org/10.1109/SP.2017.31

[12] FloGz. 2016. *Firebase android : make username unique.* Retrieved November 30, 2022 from https://stackoverflow.com/questions/35243492/firebase-android-make-username-unique

[13] Mathieson G. 2015. *How do I load an HTTP URL with App Transport Security enabled in iOS 9? [duplicate].* Retrieved November 30, 2022 from https://stackoverflow.com/questions/30731785/how-do-i-load-an-http-url-with-app-transport-security-enabled-in-ios-9

[14] hard_017. 2011. *Obscure a UITextField password.* Retrieved November 30, 2022 from https://stackoverflow.com/questions/6578824/obscure-a-uitextfield-password

[15] Linares-Vasquez M., Dit B., and Poshyvanyk D. 2013. An exploratory analysis of mobile development issues using stack overflow. *2013 10th Working Conference on Mining Software Repositories (MSR)*, 93–96. https://doi.org/10.1109/msr.2013.6624014

[16] Mithun. 2016. *Firebase - How to write/read data per user after authentication.* Retrieved November 30, 2022 from https://stackoverflow.com/questions/35822249/firebase-how-to-write-read-data-per-user-after-authentication

[17] Rahman M.S. 2016. *An Empirical Case Study on StackOverflow to Explore Developers' Security Challenges.* Master's thesis. Kansas State University, Manhattan, Kansas.

[18] Ertaş O. 2015. *Java: How can I generate PrivateKey from a string?* Retrieved November 30, 2022 from https://stackoverflow.com/questions/34454531/java-how-can-i-generate-privatekey-from-a-string

[19] Kooten P. 2016. *contractions 0.1.73.* Retrieved December 11, 2022 from https://pypi.org/project/contractions

[20] Patel P. 2016. *Google Play Security Alert - Your app is using an unsafe implementation of the HostnameVerifier.* Retrieved November 30, 2022 from https://stackoverflow.com/questions/40928435/google-play-security-alert-your-app-is-using-an-unsafe-implementation-of-the-h

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[22] Bird S., Edward L., and Ewan K. 2009. *Natural Language Processing with Python.*

[23] EP S. 2010. *Android: java.lang.SecurityException: Permission Denial: start Intent.* Retrieved November 30, 2022 from https://stackoverflow.com/questions/4162447/android-java-lang-securityexception-permission-denial-start-intent

[24] sachin003. 2012. *How to avoid reverse engineering of an APK file.* Retrieved November 30, 2022 from https://stackoverflow.com/questions/13854425/how-to-avoid-reverse-engineering-of-an-apk-file

[25] Statista. 2022. *Mobile operating systems' market share worldwide from 1st quarter 2009 to 4th quarter 2022.* Retrieved December 9, 2022 from https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/#:~:text=Android%20maintained%20its%20position%20as,the%20mobile%20operating%20system%20market

[26] Whirlwind. 2014. *Storing game preferences and saved games in a secure format.* Retrieved November 30, 2022 from https://stackoverflow.com/questions/27566483/storing-game-preferences-and-saved-games-in-a-secure-format

[27] Yang X., Lo D., Xia X., and Wan Z. 2016. What Security Questions Do Developers Ask? A Large-Scale Study of Stack Overflow Posts. *Research Collection School Of Information Systems* 31, 5 (2016), 910–924. https://doi.org/10.1007/s11390-016-1672-0

[28] yeahman. 2012. *How to securely store access token and secret in Android?* Retrieved November 30, 2022 from https://stackoverflow.com/questions/10161266/how-to-securely-store-access-token-and-secret-in-android

[29] Řehůřek R. 2009. Gensim topic modelling for humans. Retrieved October 29, 2022 from https://radimrehurek.com/gensim/