An Exploratory Analysis of Security and Vulnerability related Discussions from GitHub and StackOverflow

Sahrima Jannat Oishwee
Dept of Computer Science
University of Saskatchewan
Saskatoon, Canada
sao107@usask.ca

***Abstract*—Software developers are using a wide range of social collaboration platforms for software development. They share their learning, ask queries, and provide solutions. Therefore, social collaboration platforms are one of the best places to understand the trends of software developers' interests. GitHub and StackOverflow (SO) are the two most popular social collaboration platforms for developers. GitHub provides web hosting for software development and version control using Git. Where more than 73 million development projects model the future of the software. Additionally, Stack Overflow (SO) is a question-and-answer website for 14 million registered professional and enthusiast programmers. In this field of software development, security is a prime concept. The developers want to provide more secure software for the users. In this study, we investigate the security and vulnerability-related trending topics among the developers. The goal of this paper is to explore and analyze the commonly discussed security and vulnerability-related topics by the developers on GitHub and SO. Through these analyses, the software researchers and novice developers will get insight into security and vulnerability-related issues so that they can address those issues while building their software. So that, it will be easier to make a secure software system for the users. Additionally, the researchers can further evaluate these mostly discussed terms to extend their knowledge on these security and vulnerability-related topics. In this study, we have found the commonly discussed topics and the most prominent terms for each of the topics by topic modeling. Lastly, we performed qualitative analysis to understand the contexts of the topics, validate the findings from topic modeling and provide relevant names to those topics.**

***Index Terms*—Security, Vulnerability, GitHub comments and discussions , SO question and answer**

I. Introduction

Security is getting increasingly prevalent during software and particularly web application development [1]. Security vulnerabilities are more concerning than conventional bugs as those may lead to the revelation of touchy, confidential, or personally identifiable information. Developers continuously try to resolve those vulnerabilities and to provide less vulnerable software. GitHub[1] is the largest collaborative source code hosting site built on top of the Git version control system and in 2021 [2]. SO[2] has over 14 million registered users and has received over 21 million questions and 31 million answers. This is a platform where developers discuss their issues, comment, questions, and answers about the concerning factors of the projects. As these two are the largest collaborative platform for the developers, these contain a good amount of vulnerability-related issues, discussion, questions, and answers. From these, we can have an overview that what are the concerning security and vulnerability topics for the developers. In the previous work, there is little light on the commonly discussed security and vulnerability-related topics by the developers. Our motive is to bring some light to those topics.

[1]https://github.com/
[2]https://stackoverflow.com/

The goal of this paper is to:

- Explore and analyze the commonly discussed security and vulnerability-related topics.

With this knowledge, both the novice developers and researchers understand the view of the developers on security-related issues. Therefore, when a novice developer will develop software, he/she will know what are the security and vulnerability issues that he/she should keep in mind for software development. So that, he/she can take address those issues while developing the software. It will help to provide more secure software and the amount of re-work will also reduce. Because, after release, if the software is facing more security issues, the developers need to redo the work to address the issues. Changes in the system can create a lot of dependency issues, fixing those is also time-consuming in some cases. So, if these issues were already addressed there will be less amount of rework after the release. Additionally, the researchers can understand the trend of security-related issues that will help them in their future research.

Contributions of this study:

- This paper provides commonly discussed security and vulnerability-related topics by topic modeling.
- With manual analysis, provide the topic name and address relevant contexts that have been discussed under those topics.

The rest of the paper is organized as follows: Section II provides the methodology of this study, where it is divided into the subsection of research questions, data collections, and data analysis. The results are discussed in Section III and Section IV gives a discussion on the results. Section V provides the threats to the validity of this study. The related works are discussed in Section VI. Lastly, In Section VII is consists of the conclusion with the future work of the study.

II. Methodology

A. Research Question:

The goal of this study is to explore and analyze the trending security and vulnerability topics discussed by the developers on GitHub and SO. To address this goal we pose the following research questions:

- **RQ 1:** What are the common security and vulnerability-related topics discussed by the developers?

With this research question, we find the commonly discussed security and vulnerability-related topics that are discussed on GitHub and SO. Additionally, we have an comparison between the GitHub and SO discussion. Lastly, we find the patterns in the most prominent terms under each topic. With this information, we have an insight on the most concerning security and vulnerability-related topics and their contexts and what are the commonly discussed terms by the developers under each topic.

B. Study Design:

To perform this study, We examined a publicly available dataset. This dataset consists of security-related discussions on 15 different programming languages from GitHub and SO. The details about this dataset are presented in the data collection section. We focused on the discussions that developers conduct during security and vulnerability issues for their systems. To find the commonly discussed topics, we performed two topic modeling: Latent Dirichlet Allocation (LDA), Latent Semantic Analysis (LSA). The rationale behind choosing these two methods:

- LAD and LSA are the most commonly used in text mining [3].
- As a classifier, these topic modeling methods provide pretty decent accuracy on average. At the high level of abstraction, the accuracy rates were 84% for both LSA and LDA [4] respectively.

Dirichlet Allocation (LDA): LDA is an unsupervised machine learning technique which is used to recognize the latent topic structure of textual documents [5]. It is also used in the information retrieval field, document modeling and classification [6]. It uses probabilistic text modeling approaches in machine learning. It uses Bayes estimation instead of maximum likelihood estimation. LDA assumes each document in a corpus which is a random mixture over latent topics. Next, each of the latent topic is characterized by a distribution over words [7]. These latent topics can be generated from a collection of documents where the proportion of each topic in each document is different. There are two parts in LDA: words that belong to a document and the probability of words belonging to a topic.

At first, LDA goes through each document and randomly assigns each word in the document to one of the *k* topics (*k* is the number of topics user wants to know). It goes through each word w of each document d, and computes two probability :

- **p(topic *t* | document *d*):** It is the proportion of words in document d that are assigned to topic *t*. It tries to capture that for a given document d how many words belong to the topic *t*, excluding the current word. If a lot of words from d belong to *t*, it is more probable that the word w belongs to *t*. Equation for this:

$$p(t|d) = \frac{(words\ in\ d\ with\ t)\ +\ alpha}{(words\ in\ d\ with\ any\ t) + (\ k\ \ *alpha)} \tag{1}$$

The *alpha* represents the document-topic density.

- **p(word w | topic t):** This term refers the proportion of assignments to topic t over all the documents coming from word *w*. It calculates that how many documents are in topic t because of word w.

The documents are represented as a mixture of topics by LDA and a topic is a mixture of words. In LDA all the documents having w will be more strongly associated with *t*, if a word w has a high probability of being in the topic. Similarly, the documents which contain the w will be having a very low probability of being in *t*, if w is not very probable to be in *t*. It is because the rest of the words in document d will belong to some other topic and in consequence d will have a higher probability for that topic. Therefore, it won't be bringing many such documents to *t*, even if w gets added to *t*. Then, it updates the probability for the word w belonging to topic t. The equation follows:

$$p(w|t) = p(t|d) * p(w|t) \tag{2}$$

LDA provides accurate scores for each word concerning each topic after it performs multiple iterations over all the documents and for each topic. From this accurate score, we get the prominent topics.

Latent Semantic Analysis (LSA): LSA is one of the foundational strategies utilized in topic modeling. This unsupervised approach has pillars:

- The distributional hypothesis, which states that words with comparable meanings seem often together.
- Singular Value Decomposition (SVD) is a mathematical approach.

At first it takes a matrix of documents and terms and tries to decompose it into separate matrices : A document- topic matrix and A topic-term matrix. For calculating the matrices it calculates it uses Term frequency-Inverse document frequency.

The Term frequency-Inverse document frequency assigns a weight for term j in file i with the following formula:

$$W_{i,j} = t * f_{i,j} * log\frac{N}{df_j} \tag{3}$$

Where, $W_{i,j}$ is Term frequency-Inverse document frequency, $t * f_{i,j}$ is quantity of the occurrences term in documents, N is total quantity of documents and df_j is quantity of documents containing words. Then it performs the dimensionality Reduction with SVD. This technique follows linear algebra that factorizes any matrix M into the product of three separate matrices:

$$M = U * S * V \tag{4}$$

In which S is a diagonal matrix of the singular values of M.

With those document vectors and term vectors, follow cosine similarity to evaluate: the similarity of various documents, the similarity of various words, and the similarity of terms (or "queries") and documents (which will become beneficial in statistics retrieval. With this process, it performs the topic modeling.

LDA and LSA both have different topic modeling approach. So, that we choose these two approaches. With LDA we have ten prominent topics for GitHub and SO separately. Then we perform LSA on those datasets and compare them with the LDA topics. After that, we provide the name of each topic manually. Next, we perform a qualitative analysis of the data. From that, we have more insight on what are the relevant discussions under those topics. Finally, we have come up with the most discussed terms under each prominent topic. Figure: 1 presents all the phases of this study in a nutshell.

C. Data Collection

In this study, we analyzed a publicly available dataset [8] which was used to find the security challenges of the various programming languages. The dataset has security-related discussions for different programming languages. Below, we discuss the data collection process of the prior work.

For SO, the dataset consists of programming language-specific data, based on the tags of each post. All the appropriate SO tag was manually identified. The complete tag list is presented in the prior work's appendix[3]. Subsequently, only security-related discussions were leveled as relevant and filtered as the dataset for that study. The dataset has security-related question-answer discussions for 15 different programming languages considering all discussions on SO up to May 1st, 2020. For GitHub, the data was collected from the different programming language repositories which have at least 100 stars. Then they filtered the programming language datasets to security-related discussions, considering data up to July 1st, 2020.

[3]https://github.com/RolandCroft/Language-Security-Challenges/blob/master/Appendix/SOLangsTop50.txt

We use a sample dataset from this large entire dataset. This sample dataset is available in GitHub[4]. In the SO dataset there are 386 security-related questions and answers. Each row of this dataset has postID, the title of the post, question, answer, creation time. All the questions and answers were reported between July 2008 to July 2020. For example: the first row of this dataset consists of a question and answer with the postID: 40471439, the tag of the post is *"blockchain diffie-hellman encryption javascript"*, the title of the post is *"js diffi hellman ssl blockchain auth"*, and the creation date is 2016-11-07. In the GitHub dataset, there are also 386 security-related questions and answers from different repositories. Each row of this data set has postID, link of that post, tags, title, question, and answer. Such as: The first question and answer have the postID: 4636 with link: https://api.github.com/repos/microsoft/AL/issues/4636, and the tag of this post is *"issue"*. For both the dataset the questions and answers were lemmatized.

D. Data Analysis

We perform our data analysis in two steps: Topic Modeling and Qualitative analysis. Below we discuss the analysis:

1) Topic Modeling: The analysis from the topic modeling for SO and GitHub dataset follows:

LDA for SO and GitHub: We performed LDA for both the dataset separately. We got ten topics for each of these. Table: I represents the prominent ten topics for SO security and vulnerability discussion and Table: II represents the prominent ten topics for GitHub vulnerability discussion. From Table: I and Table: II we found that the topics from the SO dataset are more meaning full than the GitHub dataset. We get meaningful topics relevant to security and vulnerability for the SO dataset. On the other hand, we get very sparse and less meaningful vulnerability and security-related topics for GitHub. To have more validity to this analysis we performed LSA on each of the datasets.

LSA for SO and GitHub: We performed LSA to validate the resultant topics from LDA. Table: III and Table: IV represents the resultant topics from LSA. LSA also shows a similar pattern to LDA for both the datasets. For SO the topics were meaningful and similar to LDA topics but for GitHub, the topics were sparse and less meaningful.

To have better visualization of these topics we created a visualization graph of the LDA topics for both of the datasets. Fig. 2 and Fig. 3 represent the visualization for SO topics and GitHub topics respectively. For SO data the Fig. 2 shows that the topics are distinct from each other. There is almost no overlapping property. We can see topic 3, topic 8, and topic 10, topic 9 have little overlapping property. In Fig. 3, we see that

[4]https://github.com/RolandCroft/Language-Security-Challenges/tree/master/Appendix/dataset$_{samples}$

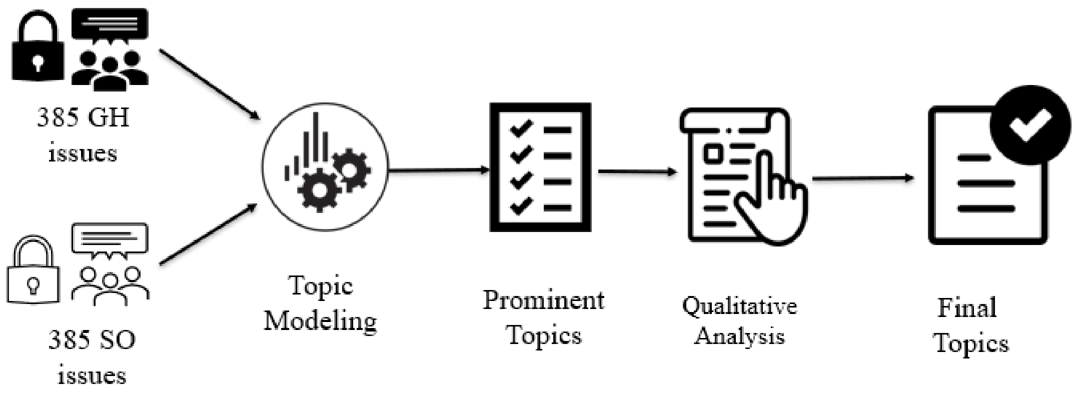

Fig. 1. Study Design

the topics are not very distinct from each other and having a lots of overlapping property. For example: Topic-7, topic-8, topic-9, and topic-10 are overlapping with each other and it is very difficult to distinguish among topic-1, topic-3, and topic-4. With this analysis, we decide that we will perform the qualitative analysis on the SO dataset only as it provides more meaningful and distinct topics.

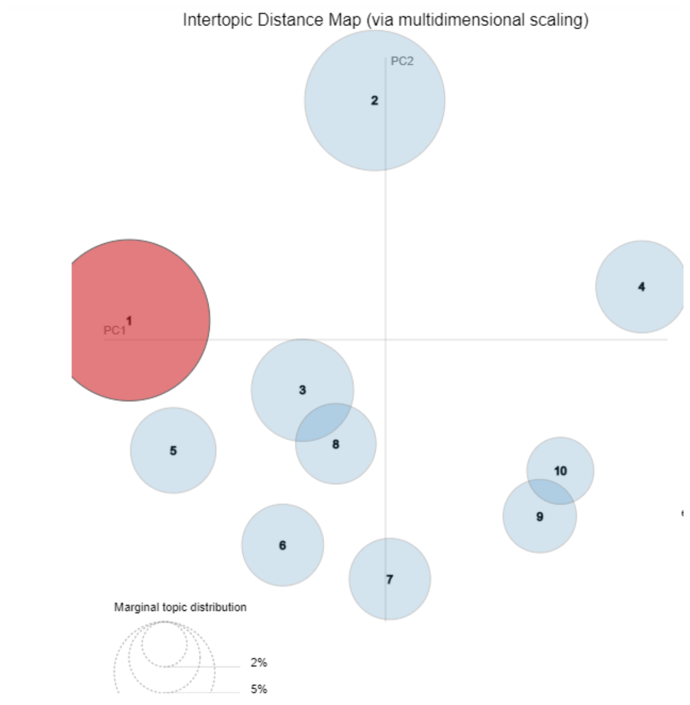

Fig. 2. LDA visualization of SO Topic

2) Qualitative Analysis: In this phase we provide the name of the topics and to understand the topics in detail, we perform qualitative analysis on the question-answer of the

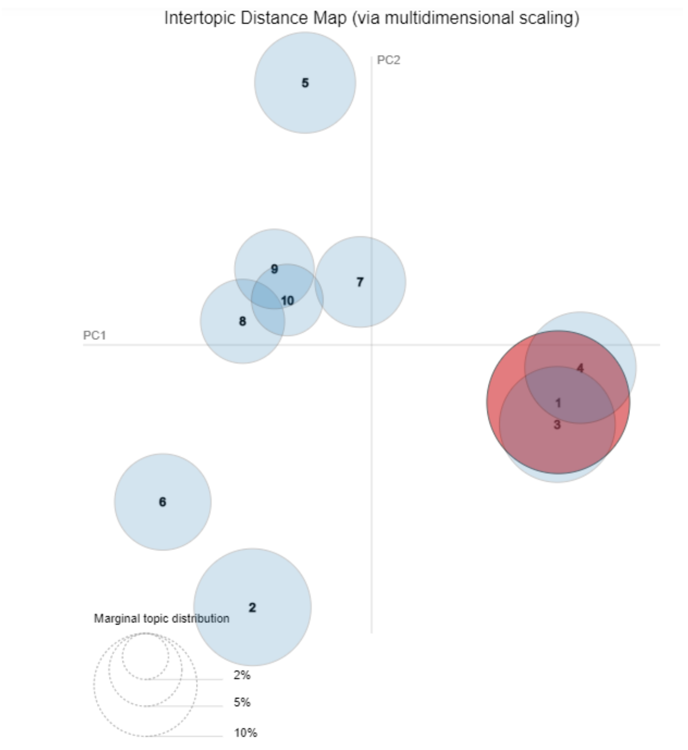

Fig. 3. LDA visualization of GitHub Topic

SO dataset.

Manual Naming: We provide naming to the topics from the weight of each term in the topics. For example: from the first row of Table: I, we see that the term *"password"* has the highest weight of 0.206 and *"login"* has the second highest weight of 0.168. So, these two are the most dominant terms in topic-1. Additionally, we notice that other terms are related to the user authentication, authorized login or login

TABLE I
LATENT DIRICHLET ALLOCATION (LDA) TOPIC FOR SO DATA

(0, ’0.206*”password” + 0.168*”login” +0.152*”authentication” +’0.119*”integrity” + 0.109*”username” +0.090*”scripting” +’0.075*”security” + 0.103*”credential” + ’ 0.086*”openid”+ ’0.034*”encrypt” + 0.029*”malicious” +0.014*”ssh”’+ 0.013 ”sign”+’0.003*”Cookie”),
(1, ’0.191*”key” + 0.151*”Public” + 0.077*”Private”+ 0.074* ”sign” + ”0.072*”certificate” + 0.063*”signature” + 0.046* ”openssl”+ 0.045*”salt” +’0.043*”signing” + 0.037* ”encode” +0.031*”scripting”’),
(2, ’0.226*”encrypt” + 0.119*”safe” + 0.103*”crypt”+ 0.095* ”security” + ”0.085*”salt” + 0.054*”attack” + 0.043* ”thread”+ 0.042*”protect” + ”0.033*”cookie” + 0.027 ”honeypot”’),
(3, ’0.226*”security” + 0.119*”certificate” +0.103*”script” + 0.095*”validate” + ”0.085*”sign” + 0.054*”exploit” + 0.043*”attack”+ 0.042*”transport” + ”0.033*”bypass” + 0.027*”trust”’),
(4,’0.321*”authorise” + 0.207*”authentication” +0.191 ”security” + ”0.093*”crypt” + 0.063*” openid ” +0.015 ”control” + 0.014*”integrity” + ”0.009*”day” + 0.001 ”password” +0.001*”username”’),
(5, ’0.278*”vulnerability” + 0.157*”inject” +0.132*”attack” + 0.081*”safe” + ”0.051*”security” + 0.039*”injection” + 0.039*”leak”+ 0.036*”SQL” + ”0.033*”XSS” + 0.015*” unsafe”’),
(6, ’0.278*”tunneling” + 0.157*”teft” + 0.132*”confidential”+ ’ 0.081*”penetration” +’0.051*”vector” +0.039*”asset” + ’0.039*”leak” + 0.036*”account” + ’0.033*”table”+ 0.015 ”unsafe”’),
(7, ’0.161*”hash” + 0.140*”availability” + 0.067*”scrpting” + ’ ’0.054*”encrypt” + 0.054*”hack” + 0.050*”sanitize”+ 0.049*”XSS” + ’ ’0.041*”crack” + 0.036*”checksum” + 0.036*”security”’),
(8, ’0.177*”signing” + 0.134*”forge” + 0.099*”captcha” + 0.076*”trust” + ”0.064*”privilege” + 0.062*”hijacking” +0.052*”spam” +’ 0.050*”insecure” + 0.036*”account” + 0.033*”asset”’),
(9, ’0.301*”sign” + 0.267*”validate” + 0.139*”OWASP” +’0.091*”information” + 0.046*”sensitive”+ 0.037*”table” +’0.017*”Disclosure” + ’0.008*”steal” + 0.006*”integrity” + 0.006*”test”’)]

credential related. Again, if we look at the fifth row of Table: I we can see that this topic-5 have similar kind of terms to topic-1. Therefore, we combine topic-1 and topic-5 together and provide the name ”Login Password”. With this process we provide name to the topics and lastly, we get seven distinct security and vulnerability-related topics from SO dataset. The details of this analysis is presented in Table: V.

After this manual naming, we try to understand what are types of discussions are happening under these topics. For that, we analyze the question-answers from the SO dataset. As all the question-answers were lemmatized, it was very difficult to understand the contexts. We come up with a solution for this. The dataset provides postID of each question-answers,

TABLE II
LATENT DIRICHLET ALLOCATION (LDA) TOPIC FOR GITHUB DATA

[(0, ’0.094*”use” + 0.082*”secur” + 0.070*”need” + 0.069*”think” + 0.064*”make” + ’ ’0.046*”want” + 0.042*”implement” + 0.031*”encrypt” + 0.029* ”user” + ’ ’0.023 *”packag”’),
(1, ’0.075*”run” + 0.069*”command” + 0.061*”file” + 0.056*”build” + ’0.045*”config” + 0.039*”use” + 0.034*”fix” + 0.033*”int” + 0.028*”default” ’ ’+ 0.026*”bufferlen”’),
(2, ’0.109*”user” + 0.090*”login” + 0.090*”password” + 0.073*”authent” + ’0.057*”token” + 0.053*”valid” + 0.052*”field” + 0.052*”request” + ’ ’0.032*”session” + 0.031*”access”’),
(3, ’0.083*”error” + 0.078*”use” + 0.047*”method” + 0.042* ”version” + ’ ’0.031*”object” + 0.031*”datum” + 0.028*”reproduc” + 0.023*”valu” + ’ ’0.021*”decrypt” + 0.020*”file”’),
(4, ’0.073*”encod” + 0.061*”use” + 0.056*”string” + 0.056*”function” +’ ’0.055*”type” + 0.051*”valu” + 0.048*”code” + 0.043*”ssl” + 0.031*”return” + ’ ’0.030*”version”’),
(5, ’ 0.112*”fix” + 0.065*”thank” + 0.059*”request” + 0.059*”pull” + ’ ’0.055*”email” + 0.047*”plea” + 0.043*”author” + 0.032*”new” + ’ ’0.029*”commit” + 0.029*”sign”’),
(6, ’0.093*”vulner” + 0.052*”report” + 0.045*”close” + 0.041*”site” + ’ ’0.033*”self” + 0.029*”avail” + 0.029*”codebuild” + 0.029*”day” + ’ ’0.029*”tab” + 0.026*”server”’),
(7, ’0.040*”code” + 0.038*”sign” + 0.038*”link” + 0.036*”protect” + 0.034*”bug” ’ ’+ 0.033*”run” + 0.032*”step” + 0.031*”key” + 0.031 ”warn” + 0.026*”window”’),
(8, ’0.111*”work” + 0.108*”test” + 0.077*”add” + 0.066*”log” + 0.053*”ad” +’ ’0.041*”option” + 0.034*”look” + 0.028*”thank” + 0.028*”check” +’ ’0.026*”case”’),
(9, ’0.104*”use” + 0.074*”key” + 0.063*”connect” + 0.058*”server” + ’ ’0.043*”decrypt” + 0.038*”encrypt” + 0.035*”support” + 0.033*”file” +’ ’0.031*”host” + 0.030*”client”’)]

TABLE III
LATENT SEMANTIC ANALYSIS (LSA) TOPIC FOR SO DATA

[(0, ’-0.521*”security” + -0.406*”authentication” + -0.336*” password”+ ’-0.275*”login” + -0.216*”encrypt” + -0.185*” scripting” + -0.183*”authorise” ’+ -0.182*”username” + -0.172*”validate” + -0.167*”sign”’),
(1, ’-0.552*”key” + 0.310*”password” + -0.309*”public” + 0.309 ”authentication” ’+ -0.292*”encrypt” + 0.268*”login” + -0.249* ”private”+ 0.212*”username” + ’-0.156*”security” + -0.134 ”safe”’)
(2, ’-0.651*”security” + 0.412*”key” + 0.368*”password” + 0.243 ”username” + ’0.208*”public” + 0.195*”private” + 0.168 ”authentication” + 0.112*”sign” + ’ -0.099*”attack” + -0.091 ”validate”’),
(3, ’-0.527*”encrypt” + -0.326*”password” + -0.299*”scripting” + ”0.299*”authorise” + 0.295*”authentication” + -0.239*”validate ”+0.205*”key” + 0.201*”attack” + -0.150*”hash” + -0.142*”salt”’),
(4, ’-0.567*”validate” + -0.529*”sign” + -0.385*”scripting” + ”0.277*”authentication” + - 0.180*”encrypt” + -0.177*” signature”+ ”0.136*”authorise” +- 0.131*”security” + 0.107*”key” + 0.097*”signing”’)]

TABLE IV
LATENT SEMANTIC ANALYSIS (LSA) TOPIC FOR GITHUB DATA

[(0, ’-0.634*”use” + -0.318*”work” + -0.197*”need” + -0.196*”user” + ’ ’-0.192*”think” + -0.157*”make” + -0.152*”secur” + -0.144* ”implement” + ’ ’-0.134*”key” + -0.121*”run”’),
(1, ’0.400*”request” + 0.342*”fix” + 0.270*”thank” + 0.260*”pull” + 0.222*”plea” ’ ’+ 0.214*”test” + 0.202*”add” + -0.189*”need” + -0.187*”think” + ’ ’-0.156*”implement”’),
(2, ’0.285*”login” + -0.269*”think” + -0.266*”need” + 0.251*”user” + ’ ’0.239*”server” + -0.236*”make” + 0.230*”authent” + -0.230* ”fix” + ’ ’-0.178*”add” + -0.177*”secur”’),
(3, ’0.335*”user” + 0.266*”login” + -0.242*”valu” + -0.242*”type” + ’ ’0.237*”authent” + -0.219*”error” + -0.205*”string” + 0.176* ”think” + ’ ’-0.174*”use” + -0.171*”encod”’),
(4, ’-0.300*”file” + 0.261*”server” + 0.245*”valu” + -0.224* ”version” + ’ ’0.221*”request” + -0.216*”run” + 0.197*”type” + 0.189*”string” + ’ ’0.180*”return” + 0.170*”valid”’)]

so we pulled the questions-answers through this postID from SO. Then we performed the qualitative analysis on this.

Login Password: Under these topics, we study the question-answers and try to find the pattern of the context of the discussion. We provide a few examples below to understand this analysis in detail:

- *” I am working with an application that allows the **user to login** using NTLM and a form login simultaneously. I mean that a user can log using their windows credentials (using jespa), but it is possible to logout and use a formulary that checks your user/password in a database. It works fine if you work with IE8 (build bigger than 8.0.6001.18702), **IE9**, Firefox or Chrome, but when you get to work with **IE7** (at least with 7.0.5730.13) and IE8 (build 8.0.6001.18702), it doesn’t work.”*

In this part of the discussion, the developer face some version related issues to develop user login for his/her application. So, he/she asks for the solution.

- *”How do I check if the **username and password** are correct in this method before returning c?”*

For this above question, the developer is asking for development-related help for username and password. From several question-answers, we found this pattern that, developers ask different development-related issues under this topic.

Additionally, we found that developers talk about different techniques to make the user login more secure. A few examples are given below:

- *”My question is about using **blowfish** to hash your passwords for storage.”*
- *”How to hash long password with **blowfish?**”*
- *”Posting **plain text** password in url is dangerous.”*
- *”I am wondering now, what are my options to prevent the password being sent as **plain text**?”*
- *”Does **Keycloak** support basic Authentication (Authorization header that contains the word Basic word followed by a space and a base64-encoded string username:password) and if so how I can configure realm and client settings for it ? I want to secure my rest api with Keycloak and support also basic Authentication as an option.”*

From the above list of part of the question-answers, we can see that developers are talking about different techniques to make the user login more secure. For example, Blowfish, Plain text, Keycloak, etc.

Public-Private Key: For the Public-private key, we have found a pattern that developers are taking about the digital signatures to make their system more secure and save their software from middle attackers. Some examples are discussed as follows:

- *”I’m using a PKCS11 smartcard at work, and would like to use jarsigner to **sign** jar files using the certificate on my card. I finally got jarsigner to see the card, but it reports that it isn’t a valid entry and doesn’t have a **private key**.”*
- *”I thought of learning Strong Name , created one example CacheUI project - CacheUI.dll , created strong name by using sn.exe ”C:test.snk” and added this test.snk into Project - Properties - **Signing** - sign the assembly - choose a strong name key file. Also generated the public token key. I understand that strong name consists of the assembly’s identity—its simple text name, version number, and culture information (if provided)—plus a **public key** and a digital **signature**.How can i Reference to Strong-Named Assembly? What is delay sign and how it’shelpful for developers ?”*

In both of the discussions they are talking about the digital signature.

Encryption: Below we provide some examples for the encryption related question-answers:

- *”I need to **encrypt** a 32-bit integer to a 32-bit integer and also i can **decrypt** it, Dose any one know how to do it?”*

- *"I'm looking at the source code for the .net membership provider, and it sqlmembershipprovider.cs there are calls to* ***EncryptPassword*** *and* ***DecryptPassword*** *but I don't see the method anywhere in the source. What algorithm are they using? Isn't the source for that released also?"*

- *" I also know that this* ***encryption key*** *should be stored somewhere.The problem is that I don't want to store it in my* ***database****, because it could be accessible for hackers. Where could I also store my keys safely? "*

From the above examples, we can see that the context of every question-answer is different. In the first example, the developer is asking for help for encrypting 32-bit integers, in the second example, the developer is asking for some particular algorithm and lastly, another developer is asking for help to store the encryption key anywhere secure but database. Therefore, we could not find any pattern in the context for the topic Encryption. We notice similar behavior for the topic of Hashing as well.

Security Certificate: We have also performed the qualitative analysis for the topic security certificate. Examples are discussed bellow:

- *"I'm fairly new to* ***HTTPS/SSL/TLS*** *and I'm a bit confused over what exactly the clients are supposed to present when authenticating with certificates. I'm writing a Java client that needs to do a simple POST of data to a particular URL. That part works fine, the only problem is it's supposed to be done over HTTPS. The HTTPS part is fairly easy to handle (either with HTTPclient or using Java's built-in HTTPS support), but I'm stuck on authenticating with client certificates."*

- *"I am using Java 6 and am trying to create an* ***HttpsURLConnection*** *against a remote server, using a client certificate. The server is using an selfsigned root certificate, and requires that a password-protected client certificate is presented. I've added the server root certificate and the client certificate to a default java keystore which I found in /System/Library/Frameworks/JavaVM.framework/Versions/1.6 .0/Home/lib/security/cacerts (OSX 10.5). The name of the keystore file seems to suggest that the client certificate is not supposed to go in there?"*

From the abode example, we can see both of the discussions are related to provide the client certificate for HTTPS/SSL/TLS servers.

Vulnerability: We have also noticed that the developers show interest in particular type of vulnerabilities. For example:

- *The correct way to avoid* ***SQL injection attacks****, no matter which database you use, is to separate the data from SQL, so that data stays data and will never be interpreted as commands by the SQL parser. It is possible to create SQL statement with correctly formatted data parts, but if you don't fully understand the details, you should always use prepared statements and parameterized queries.These are SQL statements that are sent to and parsed by the database server separately from any parameters. This way it is impossible for an attacker to inject malicious SQL."*

- *"How do I prevent* ***XSS (cross-site scripting)*** *using just HTML and PHP? I've seen numerous other posts on this topic but I have not found an article that clear and concisely states how to actually prevent XSS."*

- *I have been reading some articles on memory leaks in Android and watched this interesting video from Google I/O on the subject. Still, I don't fully understand the concept, and especially when it is safe or dangerous to user inner classes inside an Activity. When exactly is it* ***leak*** *safe to use (anonymous) inner classes? "*

In these discussions, it is clearly visible that developers talk about different types of vulnerabilities: SQL injection, XSS, and Leak. This summery of this qualitative analysis is presented in Table: VI.

After this qualitative analysis, we analyze which topic is covering how many terms in this SO security and vulnerability-related question-answers. We notice that some topics cover almost half of the security and vulnerability-related terms, some of the topics cover a little part of the terms. The analysis is reported in Fig: 4

III. Result

RQ1: What are the common security and vulnerability-related topics discussed by the developers?

After the topic modeling and qualitative analysis, we found seven topics from the SO security and vulnerability-related discussions. At first from LDA topic modeling we get ten topics, after qualitative analysis, we provide a name to those topics and merge the relevant topics. Table: V represents the resultant topics after manual naming and merging. From topic-1 and topic-5 we get the topic related to Login password. From topic-2 we get the topic Public-private key. we named topic-3, topic-8, topic-4, and topic-6 as Encryption, Hashing, Security Certificate, Vulnerability respectively. Lastly, there were some sparse topics: topic-7, topic-9, and topic-10. We named them all together as Others. Therefore, we find that developers discuss the topics: Login password, Public-private key, Encryption, Hashing, Security certificate, and vulnerability on SO platform.

After qualitative analysis, from Table: VI we see that for the topic Login password, developers talk mostly on two contexts: Development problem and Techniques to make the user login more secure. In the Public-private key, developers are more concerned about digital signature. For the topic Encryption and Hashing, they mostly do the general discussions. The topic Security certificate covers client certificate for HTTPS/SSL/TLS server. Finally, the developers commonly talk about SQL injection, XSS, and Leak under the topic Vulnerability.

Lastly, the most dominant topic is Login password as it covers 44.6% terms. The topics Public-private key, Encryption, Hashing, and Scripting cover 14.20%, 10.0%, 7.10% and 7.10% of terms whereas other topics cover 11.60% of terms all together. This result is summarized in Fig. 4

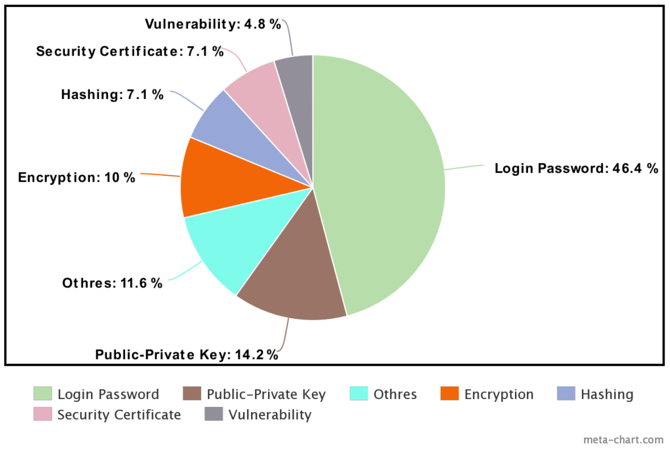

Fig. 4. SO data: Token/Term covered by Topics

IV. Discussion

In this section, we discuss our findings and their reasoning. From RQ1, we got that on the SO platform the developers ask or discuss Login Password, Public-Private kye, Encryption, Hashing, Security Certificate, and Vulnerability. From the qualitative analysis, we found that for all of these topics they discuss mostly the development issues. But we could not find any discussions related that the developers are facing a security breach in their system. There can be two reasons: firstly, we only investigated 386 security-related question-answers from SO. Secondly, the developers may not want to share the security status of their software publically. But, to make this assumption valid, we need to work with a large data set of security-related question-answers.

From the result, the novice developers can have an idea that what are the common security concern for developing secured software. For example: under the topic, Public-private key developers talk about digital signature. Digital signatures use the public key cryptography technique. It provides higher security to ensure authenticity and non-repudiation of the software. So, when a developer is developing new software, they should consider the digital signature to make secure software. Again, under the Security certificate topic, developers discuss the HTTPS/SSL/TLS server. Therefore, when developers work with these servers, they should provide the certificate to the authenticated clients. Otherwise, malicious users may attack their servers. Then again, for making the password more secure, developers can take the Blowfish and Keycloak techniques into their considerations.

As all these findings are on a dataset of 386 question-answers on SO, for understanding the topics and their contexts we need to work with a large dataset. With that, we can provide a very detailed insight on the security and vulnerability-related topics to novice developers and researchers.

V. Threats to Validity

Below, we identify threats to the validity of our study.

Internal validity: In this study, we analyze the security-related posts in general. We did not consider the other variables: the system used by the developer, the language used by the developer, etc. Though these technical details may have an influence on this analysis. Developers using different technical facilities may discuss different types of security issues. We did not consider this kind of variable.

Construct Validity: We only use two types of topic modeling. There can be some bias in the resultant topics. To avoid this bias, we use such two topic modeling approaches, which have different techniques to find the topics. So, the chance of bias is less. Only one author performed the qualitative analysis. The names of the topics were given by manual coding. So, there can be some unintentional human error. In the future, we can have a security expert to verify this qualitative analysis and understand the topic's contexts better.

External Validity We used the dataset of only 386 question-answers. Additionally, those data were under the tag of different programming languages. There are other discussions under different tags which can have also vulnerability and security-related topics. Those vulnerability or security-related discussions were not analyzed in this study. As a result, we may be left with some commonly discussed topics related to vulnerability and security. In the provided dataset, all comments, discussions, questions, and answers were stemmed. Additionally, there was no consistency in the discussions. So, it was very hard to find the context from this dataset. Though it has provided distinct topics via topic modeling, manual verification is quite complex since the data is already lemmatized. To avoid this problem, for manual analysis we pulled the question-answers from SO by the postID.

VI. Related Work

In this section, we highlight existing research that focused on security and vulnerability, security and vulnerability-related discussions, trending topics, and discussion analyze of different social media and collaboration platforms.

There are some studies on security-related trending topics on social platforms. Shin et al. [9] conducted an empirical

TABLE V
PROMINENT TOPICS AND TAXONOMIES FOR STACKOVERFLOW

Topic Name	Relevant Terms	Topic Number
Login Password	Password, Password, Login, Authentication, Integrity, Username, Scripting, Security, Credential, OpenID, Encrypt, Malicious, ssh, Sign, Cookie, Authorise, Security, Crypt, Control, Day	Topic-1, 5
Public-Private Key	Key, Public, Private, Sign, Certification, Signature, Openssl, Salt, Signing, Encode, Scripting	Topic-2
Encryption	Encrypt, Safe, Encode, Crypt, Security, Scripting, Salt, Attack, Thread, Protect, Cookie, Honeypot	Topic-3
Hashing	Hash, Availability, Scripting, Salt. Encrypt, Hack, Sanitize, XSS. Crack, Checksum, Security.	Topic-8
Security Certificate	Security, Certificate, Script, Validate, Sign, Exploit, Attack, Transport, Bypass, Trust.	Topic-4
Vulnerability	Injection , Attack, Safe, Security, Injection, Leak, SQL, XSS, Unsafe	Topic-6
Others	Signig, Forge, Violate, Captcha, Trust, Privilege, Hijacking, Spam, Insecure, Account, Asset, OWASP, Information, Sensitive, Table, Disclosure, Steal, Integrity, Test, Tunneling, Confidential,Theft, Penetration, Vector	Topic-7, 9, 10

TABLE VI
TOPIC'S CONTEXT OF DISCUSSION

Topic Name	Context of Discussion
Login Password	Development issues, Techniques to make more secure user login
Public-Private Key	Digital signature
Encryption	General discussion
Hashing	General discussion
Security Certificate	Client certificate for HTTPS/SSL /TLS servers
Vulnerability	SQL injection, XSS, and Leak

study for early detection of Cybersecurity events with new and re-emerging words. In this paper, they have proposed an event detection system called W2E (word to event), that can quickly identify critical security events by monitoring new and emerging security-related words on Twitter. This approach helped them to detect new threats as early as possible. They have found that Twitter is the first source and sometimes the only source to discuss security events. Additionally, this study showed that their W2E has high performance with a simple algorithm. In another paper, Horawalavithana et al. [10], presented a comparison of user-generated content related to security vulnerabilities on Reddit and Twitter, and GitHub. They have used two neural network setups for this study. Their analysis showed that while more security vulnerabilities are discussed on Twitter, relevant conversations go viral earlier on Reddit. 95% of conversations with CVE details were found in Twitter whereas only 0.5% was present in Reddit and 4.5% was common in both platforms. They have also shown that these two social media platforms can be used to accurately predict activity on GitHub. In both the studies they find security-related trending or re-emerging topics but these were not specifically related to the developers' interests on the security and vulnerability-related issues.

Next, there are some researches where researchers are more focused on the social platforms used by the developers to identify their interest in discussions of security and vulnerability-related issues. In a very recent study [8], the authors performed a large-scale study to find the taxonomy related to the security challenges of different programming languages. They performed qualitative and quantitative analysis on the developer's discussion from GitHub and StackOverflow for 15 programming languages. Lastly, they proposed 18 security challenge taxonomy under six topics. The reported taxonomy of security challenges can assist both practitioners and researchers in better understanding and traversing the secure development landscape. This study was more focused on the security challenges of programming languages, whereas we investigated security and vulnerability-related issues in general for software development. Pletea et al. [1] analyzed the ambiance around security-related discussions on GitHub. They used the MSR14 data set and analyzed the sentiment with Natural Language Text Processing (NLTK) tool. They had two findings: Security-related discussions are 10% of all GitHub discussions and there are more negative emotions are expressed in security-related discussions than in other discussions. Lastly, they performed a case study to gain more insight into their findings. They only provide insight on that security-related discussions have a negative ambiance, they did not discuss any commonly discussed topics. Johri and Bansal used Latent Dirichlet Allocation (LDA) topic modeling algorithm to find the trends in technologies and programming languages from Stack Overflow data [11]. They found the dominant topics and trends from that textual data, for example, popular technology or language, the impact of technology, technology trends over time, the relationship of a technology or language, etc. Though this study provided the trends and topics related to developers' interests, it was not something specific to security and vulnerability. An empirical study was performed by Rahman et al. to explore the security challenges faced by developers [12]. They used a complete Stack Overflow dataset which is publicly available on Stack Exchange Data Dump and performed topic modeling to find the security-related challenges. In this study, they only provided security-related topics but they did not perform any qualitative analysis to understand the context of those topics.

Lastly, we study some related works to understand which social platforms should be studied to find the developers' interests and how to study those. Hata Hideaki et al. [13] performed an explanatory study on the discussion of GitHub and found the early adoption factors on the discussion feature.

Their findings were the step towards data-informed guidance for using GitHub Discussions and opening avenues for future work on GitHub. As GitHub discussions is a new feature, we can not get much insight from only the discussions of GitHub. Gousios et al. [2] investigated the usability of GitHub data in large-scale and quantitative research. The authors observed that obtaining data from GitHub is not trivial, the data may not be suitable for all types of research, and improper use can lead to biased results. As we are performing a specific type of investigation and only focused on the security and vulnerability-related data, this improper bias was not an issue for us. Barua et al. [14] performed an analysis on the topics in StackOverflow to find that what are the developers talking about. The topics resultant from SO data were very distinct and meaning full. Lee et al. [15] performed an empirical study on the developer's interest from multiple social collaborator platforms (GitHub and StackOverflow). They found that the developers have a similar interest in both of the platforms and 39% of GitHub repositories and StackOverflow questions are the same. With this insight, we studied GitHub and SO datasets for our studies.

Although the earlier works investigated the security challenges faced by developers, the re-emerging security topics, and security-related taxonomies of different programming languages, there are very few lights on the security-related topics that are discussed by the developers. Most of the studies did not provide the context of discussions of the topics. In this paper, we addressed the security-related topics in which developers are highly interested, additionally, we investigated the context of those topics to get a deeper understanding.

VII. CONCLUSION

In this study, we work with a small dataset for SO and GitHub. After topic modeling, we find that SO data is providing more distinct topics whereas the GitHub data is providing very sparse topics. So, for the qualitative analysis, we work with the SO data. We find the commonly discussed security and vulnerability-related topics by the developers. The topics are Login password, Public-Private key, Encryption, Hashing, Security certificate, Vulnerability. With only 386 question-answers we can not generalize this study.

In the future, we will perform this study with a large dataset and generalize our findings. Additionally, we will perform the analysis on both GitHub and SO datasets. We will find the similarity and dissimilarities among the discussed security and vulnerability-related topics on these two platforms.

REFERENCES

[1] D. Pletea, B. Vasilescu, and A. Serebrenik, "Security and emotion: sentiment analysis of security discussions on github," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 348–351.

[2] G. Gousios and D. Spinellis, "Mining software engineering data from github," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2017, pp. 501–502.

[3] R. Albalawi, T. H. Yeap, and M. Benyoucef, "Using topic modeling methods for short-text data: A comparative analysis," *Frontiers in Artificial Intelligence*, vol. 3, p. 42, 2020.

[4] L. H. Anaya, *Comparing Latent Dirichlet Allocation and Latent Semantic Analysis as Classifiers.* ERIC, 2011.

[5] J. C. Campbell, A. Hindle, and E. Stroulia, "Latent dirichlet allocation: extracting topics from software engineering data," in *The art and science of analyzing software data*. Elsevier, 2015, pp. 139–159.

[6] S. H. Mohammed and S. Al-augby, "Lsa & lda topic modeling classification: Comparison study on e-books," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 1, pp. 353–362, 2020.

[7] C. M. Intisar, Y. Watanobe, M. Poudel, and S. Bhalla, "Classification of programming problems based on topic modeling," in *Proceedings of the 2019 7th International Conference on Information and Education Technology*, 2019, pp. 275–283.

[8] 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," *arXiv preprint arXiv:2107.13723*, 2021.

[9] H. Shin, W. Shim, J. Moon, J. W. Seo, S. Lee, and Y. H. Hwang, "Cybersecurity event detection with new and re-emerging words," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, 2020, pp. 665–678.

[10] S. Horawalavithana, A. Bhattacharjee, R. Liu, N. Choudhury, L. O. Hall, and A. Iamnitchi, "Mentions of security vulnerabilities on reddit, twitter and github," in *IEEE/WIC/ACM International Conference on Web Intelligence*, 2019, pp. 200–207.

[11] V. Johri and S. Bansal, "Identifying trends in technologies and programming languages using topic modeling," in *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*. IEEE, 2018, pp. 391–396.

[12] M. S. Rahman, "An empirical case study on stack overflow to explore developers' security challenges," 2016.

[13] H. Hata, N. Novielli, S. Baltes, R. G. Kula, and C. Treude, "Github discussions: An exploratory study of early adoption," *arXiv preprint arXiv:2102.05230*, 2021.

[14] 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, no. 3, pp. 619–654, 2014.

[15] R. K.-W. Lee and D. Lo, "Github and stack overflow: Analyzing developer interests across multiple social collaborative platforms," in *International Conference on Social Informatics*. Springer, 2017, pp. 245–256.