

AN EMPIRICAL CASE STUDY ON STACKOVERFLOW TO EXPLORE
DEVELOPERS' SECURITY CHALLENGES

by

Muhammad Sajidur Rahman

B.Sc., Bangladesh University of Engineering & Technology, 2011

A REPORT

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2016

Approved by:

Major Professor
Eugene Y. Vasserman

Copyright

Muhammad Sajidur Rahman

2016

Abstract

The unprecedented growth of ubiquitous computing infrastructure has brought new challenges for security, privacy, and trust. New problems range from mobile apps with incomprehensible permission (trust) model to OpenSSL Heartbleed vulnerability, which disrupted the security of a large fraction of the world's web servers. As almost all of the software bugs and flaws boil down to programming errors/misalignment in requirements, we need to retrace back Software Development Life Cycle (SDLC) and supply chain to check and place security & privacy consideration and implementation plan properly.

Historically, there has been a divergent point of view between security teams and developers regarding security. Security is often thought of as a “consideration” or “toll gate” within the project plan rather than being built in from the early stage of project planning, development and production cycles. We argue that security can be effectively made into everyone's business in SDLC through a broader exploration of the users and their social-cultural contexts, gaining insight into their mental models of security and privacy and usage patterns of technology, trying to see why and how security practices being satisfied or not-satisfied, then transferring those observations into new tool building and protocol/interaction design.

The overall goal in our current study is to understand the common challenges and/or misconceptions regarding security-related issues among developers. In order to investigate into this issue, we conduct a mixed-method analysis on the data obtained from Stack Overflow (SO), one of the most popular on-line QA sites for software developer community to communicate, collaborate, and share information with one another. In this study, we have adopted techniques from *mining software repositories* research paradigm and have employed topic modeling for analyzing security-related topics in SO dataset. To our knowledge, our work in SO data mining is one of the earliest systematic attempts to understand the roots of challenges, misconceptions, and deterrent factors, if any, among developers while they

try to implement security features during software development. We argue that a proper understanding of these issues is a necessary first step towards “build security in” culture in SDLC.

Table of Contents

List of Figures	vii
List of Tables	viii
Acknowledgements	viii
Dedication	ix
1 Introduction	1
1.1 State of Software Security	1
1.2 Software Security: Myths vs. Reality	2
1.3 Research Objective	3
2 Preliminaries	5
2.1 Stack Overflow	5
2.2 Security-Related Posts in Stack Overflow	6
2.3 Topic Modeling and LDA	7
3 Methodology and Study Setup	9
3.1 Data Selection	9
3.2 Data Extraction	12
3.3 Data Sanitization and Normalization	13
3.4 Topic Modeling	15
4 Case Study and Results	18
4.1 RQ1: <i>What security-related topics do developers discuss?</i>	18

4.2	RQ2: <i>What are the distinct characteristics of security-related posts on Stack Overflow?</i>	21
4.3	RQ3: <i>What are the main challenges developers face during security feature implementation?</i>	24
5	Conclusion	29
5.1	Implications	29
5.2	Related Work	30
5.3	Summary and Future Work	33
	Bibliography	34

List of Figures

2.1	Samle security-related post	6
2.2	Security-related post without ‘security’ tag	7
2.3	LDA graphical model	8
3.1	Research methodology	11
3.2	Tag-weight distribution	13
3.3	Pre-processed data	14
3.4	Sample security post	16
4.1	Primary categories of security-discussion	19
4.2	Topic visualization-1	20
4.3	Topic visualization-2	21
4.4	Median security answers	22
4.5	Security posts per year	23
4.6	Growth of security posts	24
4.7	Security community	25
4.8	Question about encryption	26
4.9	Plaintext password retrieval	27

List of Tables

1.1	Seven Popular Software Security Myths	2
3.1	Detailed Schema of a SO Post	10
3.2	A Subset of Final Candidate Tags	13
4.1	Topic Names and Related Top 10 Key Terms (Un-stemmed)	28

Acknowledgments

First I would like to express my gratitude to the Almighty for allowing me to finish my work. I cannot express enough thanks to my committee for their continued support and encouragement: Dr. Eugene Y. Vasserman, my major professor; Dr. Mitch Neilsen, my committee chair and Dr. William Hsu. I offer my sincere appreciation for the learning opportunities provided by my committee. I would like to express my heartiest thanks to Dr. Eugene for going extra miles for scrutinizing my works and making it look solid. A special thanks to Dr. Venkatesh Ranganath to provide important insights about my work.

My completion of this project could not have been accomplished without the support of my wife, Humaira. To Humaira – thank you for allowing me time away from you to research and write. You deserve a trip to Disney! Thanks to my parents as well, Mr. and Mrs. Rahman, for their love and blessings.

Last, but not the least, I would like to thank *Nadeen*, my yet-to-be-born child, who has been a continuous inspiration for my work.

Dedication

For My parents and my wife, who always believe in me.

Chapter 1

Introduction

1.1 State of Software Security

With the rapid rise of technology and ubiquitous computing, software is now an integral part of our daily lives. It affects our personal lives, the economy, society and the planet, not only through the well-known areas of communication, medical devices, and nuclear energy, but also in newly explored areas like self-driving cars, wearable devices, smart cities, and smart grids, to name a few. This unprecedented growth of ubiquitous computing infrastructure has brought new challenges for security, privacy, and trust. New problems range from mobile apps with incomprehensible permission (trust) model to the OpenSSL Heartbleed vulnerability,¹ which disrupted the security of a large fraction of the web servers. As almost all of the software bugs and flaws boil down to programming errors or misalignment in requirements, we need to retrace back software development life cycle and software supply chain to check and place security & privacy consideration and implementation plan properly.

Historically, there has been a divergent point of view held regarding security feature implementation in Software Development Life Cycle (SDLC). Information Security staff usually blame developers for putting their focus on code and time to release rather than protecting the code. On the other hand, security is often thought of as a “consideration” or “optional”

¹<https://heartbleed.com>

within the project plan rather than being built in from the early stage of project planning, development and production cycles. This disconnect between security and development has ultimately produced software development efforts that lack of contemporary understanding of technical security risks. As a result of this, software builders virtually guarantee that software they created will have way too many security weaknesses that could-and should-have been avoided. In recent ‘State of Software Security’,² it has been reported that 61.4% of applications failed OWASP Top 10 policy and 65.8% of them failed CWE/SANS Top 25 policy.

1.2 Software Security: Myths vs. Reality

Historically the notion of software security or ‘app-sec’ has been clouded with misconceptions. Sometimes software security has been viewed as firewalls or ‘perimeter security’ while some have thought of security as merely a cryptography problem. Table 1.1 has listed the popular 7 myths about software security:³

Table 1.1: *Seven Popular Software Security Myths*

Myth 1	<i>Perimeter security can secure your applications</i>
Myth 2	<i>A tool is all you need for software security</i>
Myth 3	<i>Penetration testing solves everything</i>
Myth 4	<i>Software security is a cryptography problem</i>
Myth 5	<i>Software security is only about finding bugs in your code</i>
Myth 6	<i>Software security should be solved by developers</i>
Myth 7	<i>Only high-risk applications need to be secured</i>

Ideally, software security refers to *building security* into software as it is being developed. That means arming developers with tools and training, reviewing software architecture for flaws, checking code for bugs, and performing some real security testing before release, among other things. But there’s no single silver bullet for software security problems. Current approaches have depended heavily on static analysis, dynamic analysis, bug reports etc.

²Veracode’s State of Software Security: 2016, Volume 7: <https://www.veracode.com/resources/state-of-software-security>

³<https://www.digital.com/blog/7-myths-of-software-security-best-practices/>

but overlooked the importance of understanding developers’ challenges and misconceptions centered around software security. We argue that software security is never just a technology problem, rather a people, processes and knowledge problem.

1.3 Research Objective

As software development is a “team sport”, it is required to take into consideration each player (from requirement analysts to UI/UX consultants to developers to testers to devOps groups) in terms of shared responsibility for adopting security best practices. We argue that security can be effectively made into everyone’s business in SDLC through a broader exploration of the user and her socio-cultural context, gaining insight into her mental model of security and privacy and her usage patterns of technology, trying to see why and how security practices being satisfied or not-satisfied, then transferring that observations into new tool building and protocol/interaction design. The overall goal in our current study is to understand the common challenges and/or misconceptions regarding security-privacy issues among developers. In order to investigate into this issue, we will conduct a mixed-method analysis on the data obtained in the Stack Overflow⁴ data dump.

While research studies have been done to assess security behavior of developers¹ and design security tools for them,² there has been no systematic attempt to understand the roots of challenges, misconceptions, and deterrent factors, if any, among developers while they try to implement security features during software development. We argue that a proper understanding of these issues is a necessary first step towards “build security in” culture in software development. Our research questions are as follow:

1. RQ1: *What security-related topics do developers discuss?*
2. RQ2: *What are the distinct characteristics of security-related posts on Stack Overflow?*
3. RQ3: *What are the main challenges developers face during security feature implementation?*

⁴<http://stackoverflow.com/company/about>

In order to answer our questions, we will use Latent Dirichlet Allocation (LDA), a type of Topic Modeling technique. Topic Modeling, a type of statistical modeling, allows us to discover hidden topics in a collection of documents, based on the the statistics of words in each document. LDA also helps us to explain similarity in different parts of data. Generally, LDA outputs a list of topics, topic proportion of each document, and topic share of each topic in the collection. The topic proportion of each document refers to what percentage of it belongs to each topic, while topic share measures how much topic X has been discussed as compared to other topics in the collection.

The rest of the report has been organized as follows. We elaborate the motivation of our work, describe the necessary background of Stack Overflow and introduce Latent Dirichlet Allocation in Chapter 2. Our research methodology along with data collection steps are described in Chapter 3. Chapter 4 talks about experimental results while Chapter 5 concludes after presenting discussion and related work and future work plan.

Chapter 2

Preliminaries

In this section, we first introduce Stack Overflow and the security-related posts on Stack Overflow in section [2.1](#) and [2.2](#), respectively. We then briefly introduce Latent Dirichlet Allocation (LDA), which is a classic topic model which we use to group different topics of security question posts in this report.

2.1 Stack Overflow

Stack Overflow (SO) is an online, collaborative platform for developers to post their programming questions, provide answers to the existing questions and find solution to their difficulties faced during programming. A developer needs to add tags while posting a question to help other users to find out what the question is about. If the answer provided by any user gives solution to the problem faced by the questioner, that answer can be selected by the questioner which is called the accepted answer to that question. Different members of the site can vote on questions and answers. The positive votes called the upvote and negative votes called the downvote, which shows how helpful that question/answer was for other users. The score of a question/answer is determined by the difference between the number of up/down votes. Based on the different activities of each user on Stack Overflow such as posting questions or answers, voting on them, posting comments, etc. their reputation score

increases which help them build their reputation on SO website. Greater the reputation values, more the capabilities for a member on SO like deletion of questions/answers, closing questions, etc. Since its inception on 2008, SO has gained gradual popularity in developer community. At the time of writing this report, SO has 13 million question type posts and 20 million answer type posts for the questions. 72% questions in SO have been answered. As of Oct. 2016, SO has an user base of 6.3 millions.

Due to the high volume of data and collaborative nature of problem solving, SO has been an active research area for communities like CSCW, CHI, KDD, MSR, to name a few. There have been a number of studies on SO dataset. Some studies categorize the questions on SO,³ identify design features,⁴ and analyze the textual contents of posts.⁵ Studies have been made for tag analysis and recommendation,⁶ response time⁷ while others have analyzed the topics and trends of different software development areas in Stack Overflow.⁸⁻¹⁰

2.2 Security-Related Posts in Stack Overflow

Stack Overflow contains millions of posts which cover a wide range of topics, such as general programming languages, web development, mobile development and security-related topics. According to SO, security related posts are more related to *application security and attacks against software*,¹ which is completely aligned with our current study.

Why is char[] preferred over String for passwords in Java?

▲ In Swing, the password field has a `getPassword()` (returns `char[]`) method instead of the usual `getText()` (returns `String`) method. Similarly, I have come across a suggestion not to use `String` to handle passwords.

▼ Why does `String` pose a threat to security when it comes to passwords? It feels inconvenient to use `char[]`.

★ 632

java string security passwords char

share improve this question

edited Sep 1 '15 at 7:59

Ripon Al Wasim 16.8k ● 22 ● 87 ● 121

asked Jan 16 '12 at 14:20

Ahamed 11.4k ● 9 ● 23 ● 54

asked 4 years ago
viewed 214523 times
active 3 days ago

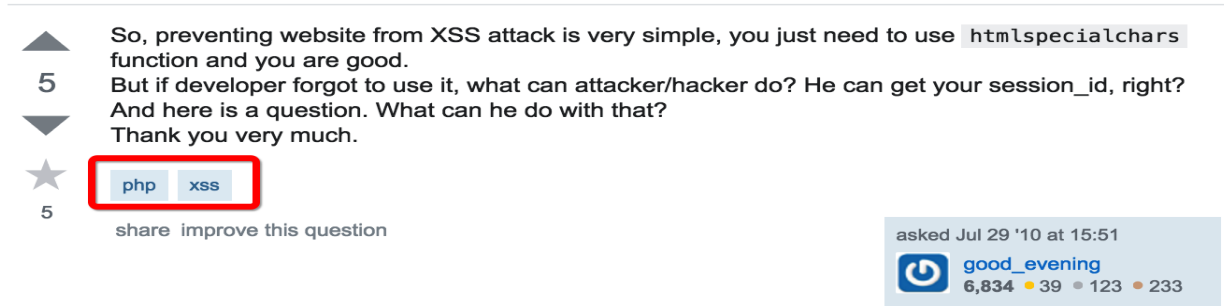
UPCOMING EVENTS

🔥 2016 Community I ends in 8 days

Figure 2.1: An example of a security related post on Stack Overflow tagged with ‘security’

¹<http://stackoverflow.com/tags/security/info>

How XSS attack really works?



▲ So, preventing website from XSS attack is very simple, you just need to use `htmlspecialchars` function and you are good.
5 But if developer forgot to use it, what can attacker/hacker do? He can get your `session_id`, right?
▼ And here is a question. What can he do with that?
★ Thank you very much.
5 php xss
share improve this question


asked Jul 29 '10 at 15:51
 good_evening
6,834 ● 39 ● 123 ● 233

Figure 2.2: A security related post without a ‘security’ tag

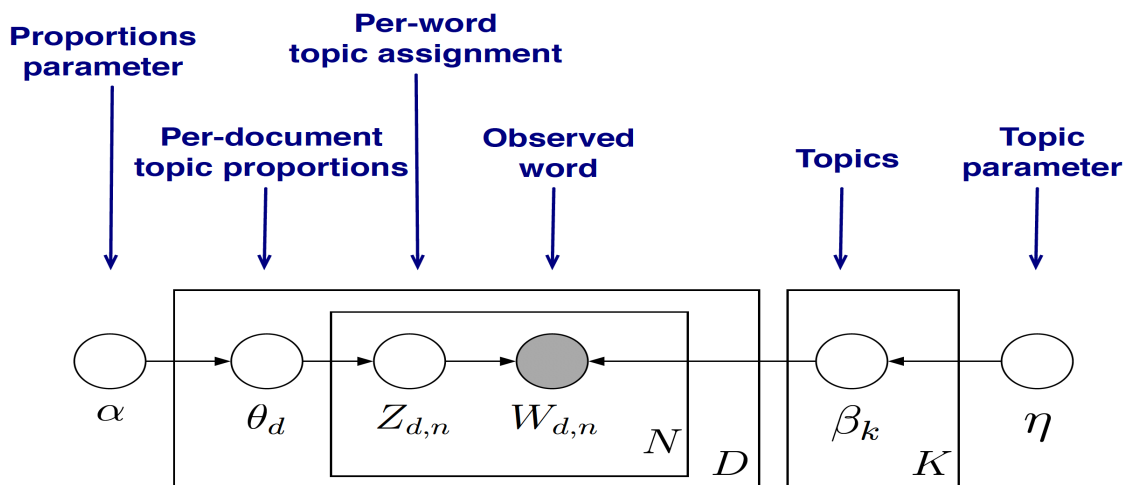
Fig. 2.1 presents a top-rated security question on Stack Overflow. The title of the post is “Why is `char[]` preferred over `String` for passwords in Java?” The question had been tagged with 5 tags: “java”, “string”, “security”, “passwords” and “char”. Between the title and the tags is the body of the post, describing the question in detail. Also, there are several metadata visible in the left and right margin of the post, such as the favorite count, upvotes, view counts, post date and edit date.

Although it seems logical and natural that the security related posts are generally tagged with “security”, there are exceptions of this general rule. For example, Fig. 2.2 shows a security-related post whose tags do not contain “security”. Therefore we cannot determine security-related posts by simply checking whether the posts contain the tag of “security”, since the extracted posts will not be sufficient and satisfactory. To address this limitation, in this paper, we first design two heuristics (which are elaborated in Chapter 3) to extract security-related tags, and then extract security-related posts according to the extracted tags.

2.3 Topic Modeling and LDA

Topic modeling is a suite of algorithms which help to discover and annotate large corpus of documents with thematic information. The algorithms are statistical methods that analyze the words of the original texts to discover the themes that run through them, how those

themes are connected to each other, and how they change over time. In this study, we are using one of the popular topic modeling technique called Latent Dirichlet Allocation (LDA). LDA is a generative model for randomly generating observable data, given some hidden parameters.¹¹ LDA represents documents as mixtures of topics that spit out words with certain probabilities. LDA assigns probability distribution of topics over words in the corpus and probability distribution of documents over the discovered topics. In LDA, each document may be related to a mixture of various topics. LDA creates topics when it finds set of words that tend to co-occur frequently in the documents of the corpus. The probabilistic graphical model of LDA is depicted in Fig 2.3.¹²



- Nodes are random variables; edges indicate dependence.
- Shaded nodes are observed; unshaded nodes are hidden.
- Plates indicate replicated variables.

Figure 2.3: Graphical model for LDA

Chapter 3

Methodology and Study Setup

In this chapter, we describe the details of the methodology of our empirical experiments. We first present the details of data collection strategy in Subsection 3.1, and then we elaborate our experimental approach in Subsection 3.2. Our research methodology is based on topic modeling, using the discovered topics as approximations of the discussion topics in which we are interested. Our methodology consists of the following steps (see Fig. 3.1). First, we extract and pre-process the posts from the Stack Overflow dataset. Second, we apply the topic modeling technique to the extracted and pre-processed posts. Finally, we analyze the discovered topics by defining metrics on their usage and inspecting metadata and the posts both quantitatively and qualitatively.

3.1 Data Selection

To conduct a comprehensive empirical study, we use a complete Stack Overflow dataset which is publicly available on Stack Exchange Data Dump.¹ Our Stack Overflow dataset contains a total of 31,977,259 posts, spanning from July 2008 to August 2016. In the dataset, there are 12,245,441 (38%) question posts and 19,731,818 (62%) answer posts. For each post, it includes a title, body and several metadata elements. Table 3.1 shows a detailed schema of

¹<https://archive.org/details/stackexchange>

a post.²

Table 3.1: *Detailed Schema of a SO Post*

Name	Description
Id	ID of the post
PostTypeId	1 is a question, 2 is an answer. Answers will have a ParentId field populated to link back to the question post.
AcceptedAnswerId	For questions, this points to the Post.Id of the officially accepted answer. This isn't necessarily the highest-voted answer, but the one the questioner accepted.
ParentID	ID of the corresponding question post for the answer post (optional and appears only when PostTypeId = 2)
CreationDate	Creation date of the post
Score	Average score by the viewers for the post
ViewCount	Total number of views for the post (optional and appears only when PostTypeId = 1)
Body	Body of the post
OwnerUserId	ID of the post owner (optional). If OwnerUserId = 1, that's the community user, meaning it's a wiki question or answer.
OwnerDisplayName	Username of the post owner (optional)
LastEditorUserId	ID of the person who last edited the post
LastEditorDisplayName	Username of the person who last edited the post
LastEditDate	Date when the post is last edited
LastActivityDate	Date when the status of the post is last changed
Title	Title of the post (optional)
Tags	Tags of the post (optional and appears only when PostTypeId = 1)
AnswerCount	Number of answers for the post (optional and appears only when PostTypeId = 1)
CommentCount	Number of comments for the post
FavoriteCount	Number of people who like the post (optional and appears only when PostTypeId = 1)
ClosedDate	If the question was closed for any reason (subjective, off-topic, etc) then the ClosedDate will be populated (optional)

As we have discussed before that the posts in SO can be related to any topics, so it is required to devise a method to filter the data-set to extract *security-related* posts. One layman approach would be to collect all question-type posts tagged with 'security'. This approach can provide an overview of major discussion topics. But our analysis found that some posts have been mislabeled with improper tags. Questioners might choose in-appropriate tags with

²<http://meta.stackexchange.com/questions/2677/database-schema-documentation-for-the-public-data-dump-and-sede>

questions as they are not sure about the correct topic of discussion and how to phrase the problem properly.

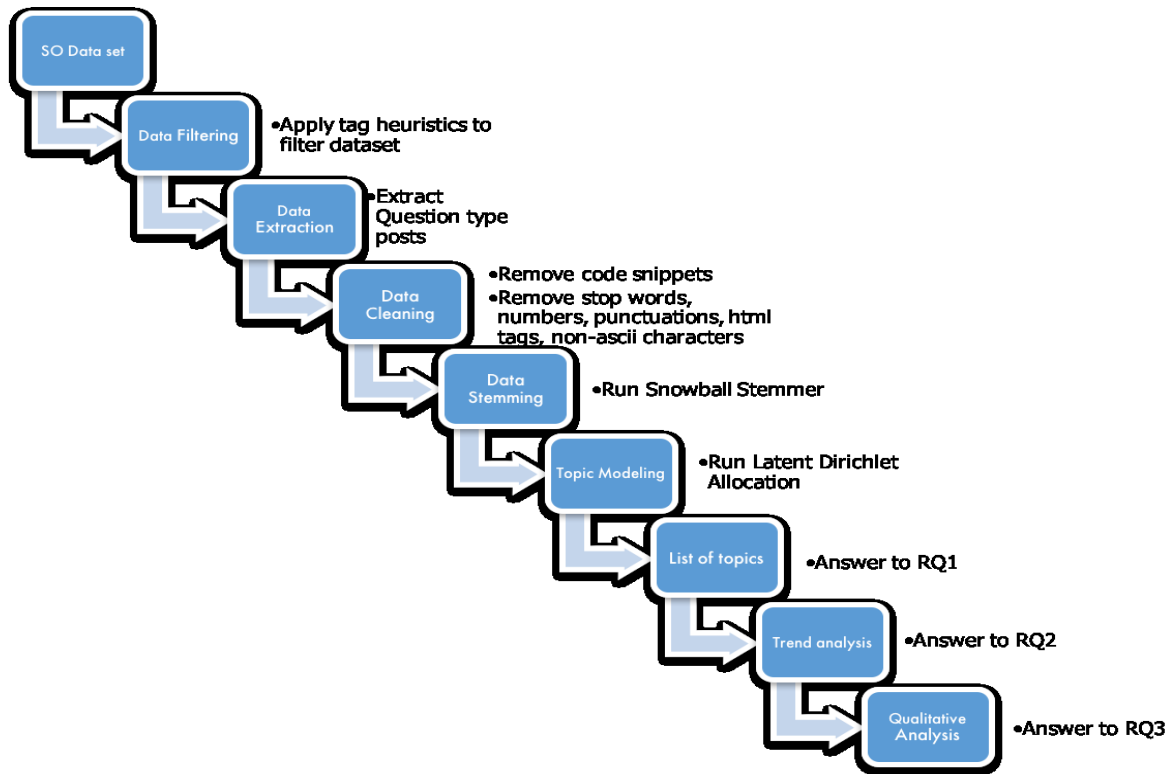


Figure 3.1: *An overview of our research methodology*

In order to solve this problem, we count the number of all questions in our SO data-set tagged with ‘security’, and find 38,313 questions. To select a representative subset, we devise the following *tag-heuristics* in our study:

1. We make a list of all other tags which have been associated with ‘security’ tags in SO posts. We call them ‘candidate tags’. In our study, we have found 3770 candidate tags.
2. For each candidate tag t , we calculate two values:
 - Let the number of times t co-appeared with ‘security’ = a and total the number of security posts = b . *Relative Tag Frequency*,

$$(RTF) = \frac{a}{b}$$

- Let the total number of questions tagged with $t = c$ and the total number of SO question posts = d . *Tag Significance Score*,

$$TSS = \lg \frac{d}{c}$$

- From RTF and TSS, we now calculate weight of a tag t as following

$$Tag - Weight = RTF * TSS$$

3. The tag weights follow a normal distribution with mean value of 0.097, median of 0.0555229 and standard deviation of 0.100169571. Fig. 3.2 shows that the distribution has a positive skew. In order to choose a cut-off value for the tag-weights, we have used OWASP Top 10³ as a dictionary of most discussed software vulnerability. After choosing different combination of a and *tag-weight*, we have finally found that with a threshold value for $a \geq 100$ and *tag - weight* ≥ 0.07 , the resulted 40 final tag-set contains tags related with OWASP Top 10. We have used this tag-set to filter our data-set in order to generate the final corpus for the analysis. Table 3.1 shows a subset of the final tags along with their tag-weights.

3.2 Data Extraction

In the final step, we traverse the security-related dataset again to find the question posts whose tags contain at least one of the tags in the tag set. The total number of such posts is 20,220 when we use the final candidate tag-set. For our analysis, we primarily use the title and the body of these extracted question posts. For each extracted post, we maintain a record of its metadata, which includes a timestamp i.e. *creation-date*, *score*, *favorite-counts*, *answer-counts* and *accepted-answer* for that question, if any. We use these metadata to analyze the security posts quantitatively.

³OWASP Top Ten Cheat Sheet: https://www.owasp.org/index.php/OWASP_Top_Ten_Cheat_Sheet

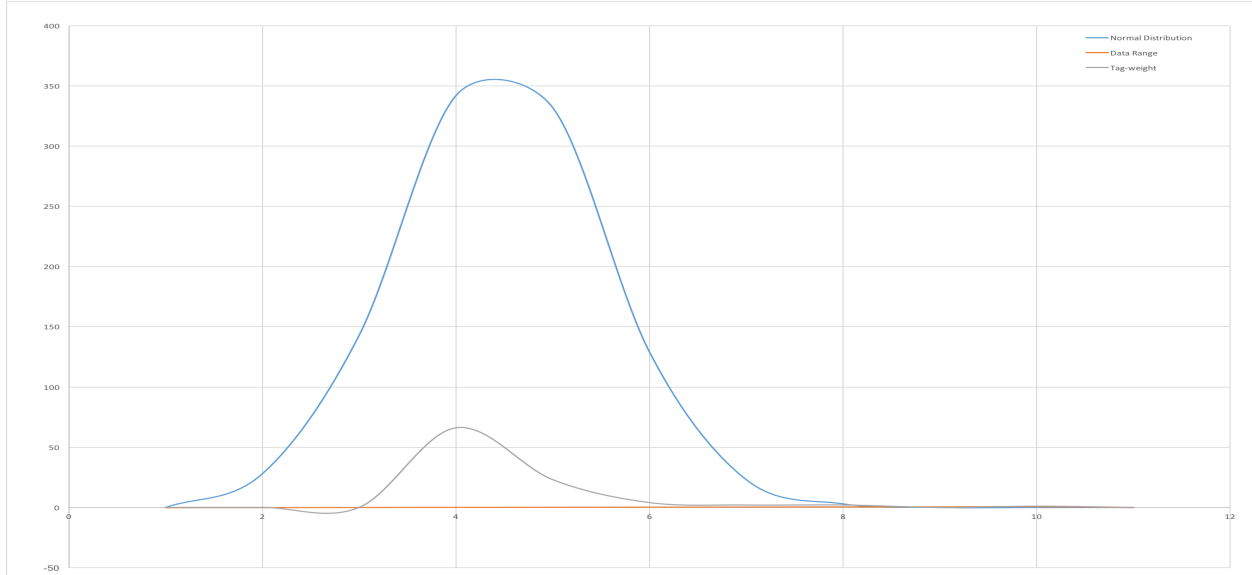


Figure 3.2: *Tag-weight distribution*

Table 3.2: *A Subset of Final Candidate Tags*

<i>Tags</i>	<i>RTF</i>	<i>TSS</i>	<i>Tag-Weight</i>
php	0.167541044	3.621969985	0.606828631
encryption	0.053271735	9.122691159	0.48598159
authentication	0.046955342	8.696727322	0.408357802
java	0.116487876	3.40012762	0.396073645
ssl	0.036306215	8.812673097	0.319954801
passwords	0.027223136	10.73858776	0.292338032
xss	0.02051523	12.00270958	0.246238345
cryptography	0.02215958	10.60855151	0.235081049
javascript	0.062668024	3.293531034	0.206399082
https	0.019941012	9.815058988	0.195722211
hash	0.018844779	9.561411951	0.180182691
spring	0.025683188	6.963672938	0.178849325
android	0.045467596	3.749041079	0.170459885
login	0.01759194	9.636598348	0.169526461

3.3 Data Sanitization and Normalization

We clean up the textual content of the extracted posts in four steps.

1. We discard any code snippets that are present in the posts (i.e., enclosed in `<code>` HTML tags), because source code syntax (e.g., if statements and for loops) introduces noise into the analysis phase. Because of the similarity in code-snippets in terms

Original: “<p>Implementing "read" and "write" security rules in Firebase In my app I want to have a "can read"- and a "can write" view. When the app starts without existing parameters ("first time user"), then 2 random hashes are created, together with the firebase app secret they are POSTed to the PHP FirebaseTokenGenerator to receive my token. Then I want to do this: The first hash ("read hash") represents an "anonymous user" entry The second hash ("write hash") represents a "key" which is a child entry of the user entry. My problem is: How do I store the write_hash the first time without my security rule ".write" preventing writing ?? Any other idea how to achieve this? Any architectural / inconsistencies? I'm using these libs: Backbone JS with backbone-firebase.js FirebaseTokenGenerator PHP Thanks in advance.</p>”

After Pre-processing:“implement read write secur rule firebas app want can read can write view app start without exist paramet first time user random hash creat togeth firebas app secret post php firebasetokengener receiv token want first hash read hash repres anonym user entri second hash write hash repres key child entri user entri problem store writehash first time without secur rule write prevent write idea achiev architectur inconsist im use lib backbon js backbonefirebasej firebasetokengener php thank advanc”

Figure 3.3: *An example of data pre-processing on SO post*

of programming language syntax and keywords, these do not help topic models to find useful topics.¹³ Unlike the previous work on source code topic analysis,¹⁴ we are focusing on user generated discussions in the SO and as most source code on Stack Overflow is only shown in small snippets, there is not enough context to allow the extraction of meaningful content from the snippets.

2. All HTML tags have been removed (e.g., <p>and), since these do not add any value in our current analysis.
3. We remove common English-language stop words such as “in”, “do”, “can”, etc. which do not help to create meaningful topics. For this purpose, we have used the ‘SMART’ stop word list.⁴ We have also removed numbers, punctuation marks and non-ASCII characters.
4. Finally, we have used Porter stemming algorithm,¹⁵ using the Snowball⁵ library, which maps words to their base form (e.g., “encrypted” and “encryption” both get mapped to “encrypt”). Fig. 3.3 shows an example post before and after pre-processing stage.

⁴<http://www.lextek.com/manuals/onix/stopwords2.html>

⁵<http://snowballstem.org/>

3.4 Topic Modeling

Topic modeling gives us a way to infer the latent structure behind a collection of documents. LDA is a probabilistic model with a corresponding generative process, which means that each document in the corpus is assumed to be generated by this process. It uses word frequencies and co-occurrences of frequencies in the document to build a model of related words. It is assumed that each topic is a distribution over words while each document is a mixture of corpus-wide topics. Each word is drawn from one of those topics. LDA creates topics when it finds set of words that tend to co-occur frequently in the documents of the corpus. The layman’s description of topic modeling is that all text is created from words contained in jars representing a certain topic. Therefore, we can mathematically discover from which jar a piece of text was assembled. However, the model has no semantic knowledge. By manually examining the keywords in a topic or jar, we can derive its meaning. For example, a topic extracted from social media containing the key words *rstats*, *bigdata*, *machinelearning*, *python*, *dataviz* might show what people are talking about data science.

Fig. 3.4 demonstrates a security question posted on Stack Overflow. The title of the question nicely summarizes the problem being asked by the questioner; confusion over hashing and encryption. The body of the post provides interesting insight into what the developer’s confusion about *hashing vs. encryption* in his own words, in terms of using them and the fundamental difference between them. Unlike the study methodology followed by Rosen et al.,⁸ we have concatenated the title along with the question body for topic modeling. Our motivation is to investigate the developer’s point of view as closely as possible and analyzing only the post titles might lose important insight of the questioner’s mental model about the problem. Thus we get a corpus containing the question titles along with a narrative of what developers ask about. The data cleaning and pre-processing part has already been discussed in Sec. 3.3.

We apply LDA to the pre-processed data using **lda** R package⁶ which uses a collapsed Gibbs sampling algorithm implementation.¹⁶ There are various parameters available as input

⁶<https://cran.r-project.org/web/packages/lda/lda.pdf>

Fundamental difference between Hashing and Encryption algorithms

I see a lot of confusion between hashes and encryption algorithms and I would like to hear some more expert advice about:

347

1. When to use hashes vs encryptions
2. What makes a hash or encryption algorithm different (from a theoretical/mathematical level) i.e. what makes hashes irreversible (without aid of a rainbow tree)

★

243 Here are some *similar* SO Questions that didn't go into as much detail as I was looking for:

[What is the difference between Obfuscation, Hashing, and Encryption?](#)
[Difference between encryption and hashing](#)

security encryption hash cryptography

share improve this question

edited Jun 9 '13 at 19:19

asked Feb 9 '11 at 17:30

Adam Lear ♦ 22.5k 9 58 91

Kenny Cason 5,356 9 27 59

Figure 3.4: *An example of a security question posted on Stack Overflow*

to **lda**. Among them, three primary parameters need to be optimized:

- K : the number of topics
- α : which dictates how many topics a document potentially has. The lower alpha, the lower the number of topics per documents
- η : scalar value of the Dirichlet hyperparameter for topic multinomials.

The number of topics K is a user-specified parameter that provides control over the granularity of the discovered topics. Larger K will produce finer-grained, more detailed topics while smaller values of K will produce coarser-grained, more general topics. There is no single value of K that is appropriate in all situations and all datasets. In our study, our goal is for topics of medium granularity, so that the topics capture the broad trends in our data-set while remaining distinct from each other as much as possible. After several experimentation, we set K to 20, which provided the characterization that we desired. Next, we set up a topic model with 20 topics, relatively diffuse priors for the topic-term distributions ($\eta = 0.02$) and document-topic distributions ($\alpha = 0.02$), and we set the collapsed Gibbs sampler to run for 5,000 iterations (slightly conservative to ensure convergence).

Output of LDA. The result of applying LDA to our pre-processed data is a set of topics, defined as distributions over the unique words in the dataset. As mentioned before, the highest-probable words in a topic are semantically related, which together reveal the nature, or concept, of the topic. For ease of readability, We examine the keywords and a random sample of 10 question posts whose dominant topic was assigned to this topic by LDA and then we manually provide a short label for each topic, for example “Mobile Security” for the topic that has top words app “android”, “ios”, “keychain”, “store” and “apk”.

Chapter 4

Case Study and Results

Thus far, we have seen that security-related posts are increasing in Stack Overflow and that the number of communities centered around security is growing as well. Now, we analyze these security-related posts in more detail to answer our aforementioned research questions.

1. RQ1: *What security-related topics do developers discuss?*
2. RQ2: *What are the distinct characteristics of security-related posts on Stack Overflow?*
3. RQ3: *What are the main challenges developers face during security feature implementation?*

4.1 RQ1: *What security-related topics do developers discuss?*

Motivation. With the first research question, we aim to showcase the topics covered by security-related questions asked on Stack Overflow. This research question can give both application security teams and developers deeper insight into the security-related questions and make them aware of different topics about security.

Approach. As mentioned in Section 3.4, we use topic model algorithm LDA to cluster the security-related questions. Generally, LDA needs a pre-defined number of topics K and

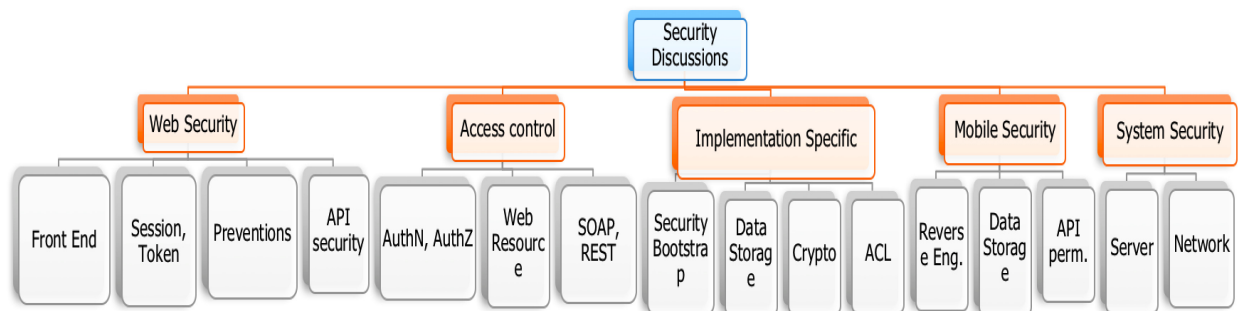


Figure 4.1: Primary categories of security-related posts in SO

it has different optimal values of K for different problems. For our study, we chose K to be 20.

Results. By using our approach, we group the security-related questions into 20 topics. Table 4.1 presents the 20 topics including the topic names and the related top 10 key terms. From Table 4, we find that the topics of the security-related questions cover a wide range. Some topics have a finer granularity while some have a coarser one. For example, “Cryptography” is a technique while “Session Management” includes several techniques to tackle the issue. In addition, we find that all the topics mainly belong to five main categories, i.e., web security, access control, implementation-specific, mobile security, and system security. Fig. 4.1 shows the five categories with the sub-categories associated with each of them. For example, discussions in web security has spanned into four sub-categories: front-end security implementation, session/token management, questions related to preventive techniques against common web vulnerabilities like *xss*, *csrf*, *sql-injection* etc. and questions asking about securing web APIs. In our analysis, we have found that web security covers more than half of all the security-related questions (60% of security posts). It indicates that web security is very popular among Stack Overflow users.

Data Interpretation. In order to better understand the topic similarity and overlap, we have used LDAvis¹⁷ library¹ for data visualization. LDAvis maps topic similarity by calculating a semantic distance between topics (via Jensen Shannon Divergence). LDAvis

¹<https://github.com/cpsievert/LDAvis>

is a powerful tool to tune up LDA model and visualize the cohesion of resulted topics. Fig. 4.2 shows the visualization of the topics. It can be seen that there is high overlapping in topics. One possible explanation might be the use of stemming as stemming causes feature reduction. Also, stemming may cause to lose potential contextual information, e.g. both ‘http’ and ‘https’ turn into ‘http’ when stemmed, although they carry completely different meaning in terms of *secured vs. unsecured connection*. In order to compare our results with *un-stemmed* documents, we run **lda** on documents which were pre-processed but un-stemmed. Fig. 4.3 shows the data visualization on the topics generated from un-stemmed corpus. Here we find that topics overlap less. In both cases, we have found that each topic share similar top keywords.

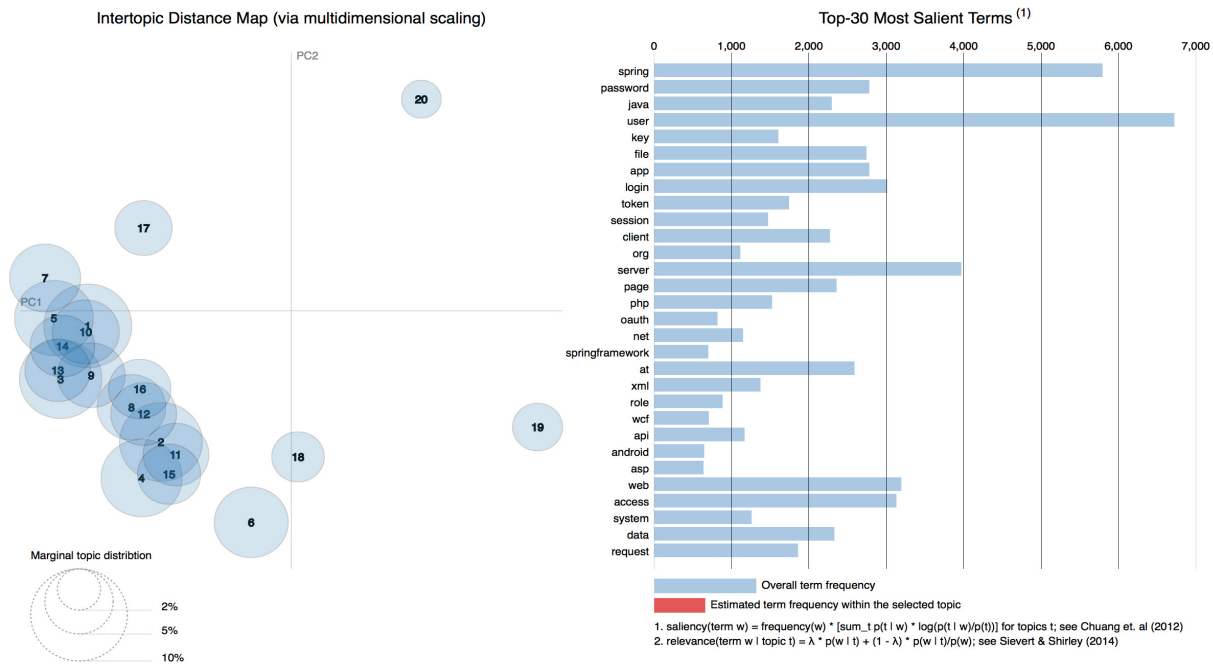


Figure 4.2: Topic visualization (stemmed data)

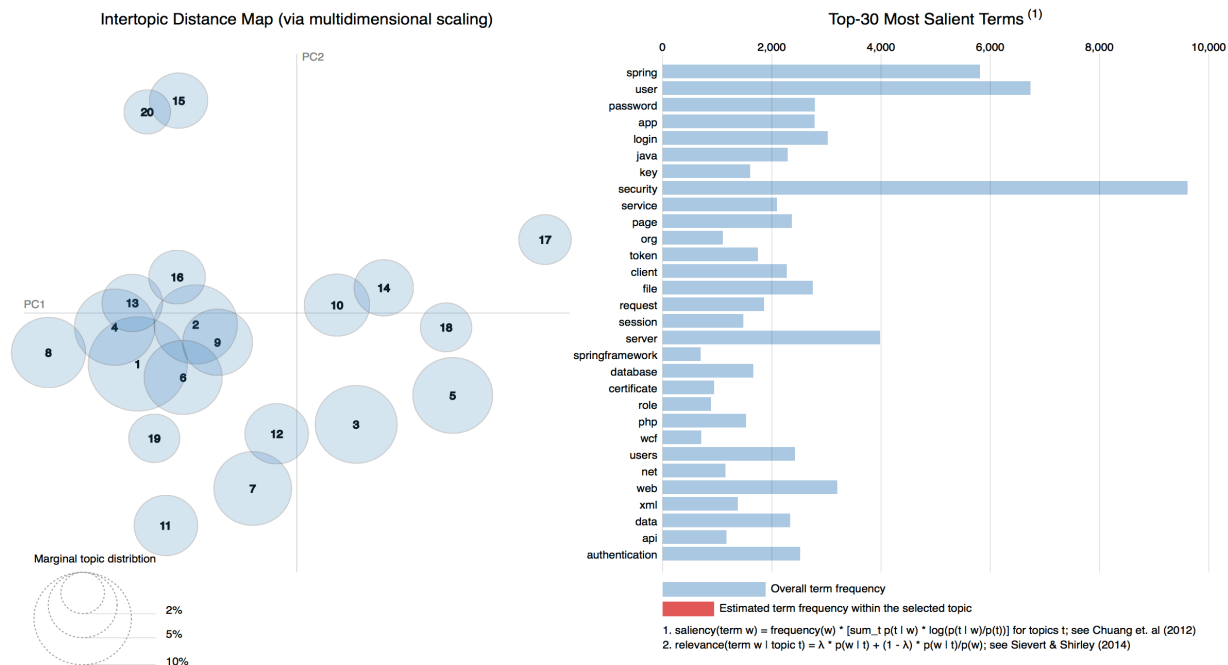


Figure 4.3: Topic visualization (un-stemmed data)

4.2 RQ2: What are the distinct characteristics of security-related posts on Stack Overflow?

Motivation. To gain an understanding of how security-related questions compare to other questions at SO.

Findings. We analyzed several *quantitative aspects* of the security-related question posts. Our findings are as follow:

- *Security posts makeup less than 1% for the SO Post.* There are 38,313 question type posts and 49,200 answer posts related to security in our dataset, which accounts for less than 1% of the total SO posts.
 - Web programming languages make up $\approx 32\%$ of security posts. The number of questions posts in different web frameworks are in descending order as follows :
php > asp.net > spring > python > symfony2 > ruby-on-rails > node.js
 - Front end technology makes up $\approx 20\%$ of security-related posts, where the tech-

nologies include but not limited to html, javascript, jquery, ajax, forms, iframe, session, cookies etc.

- General programming languages consist of $\approx 2\%$ (java, c#, c)
 - Web service security consists of $\approx 1\%$ (rest, api, wcf, web-services, jwt, token, json, curl)
 - Mobile Security makes up 7.4% (android ($\approx 1.8k$), ios (≈ 860))
- *High answer rate.* 87.85% of green questions are answered and the rate of accepted answer is 55.88%. Whereas, 88% of questions at SO are answered, i.e., received at least one answer, and 58% of the questions have an accepted answer. Average answer count is 1.78 per question for security-related posts. Fig. 4.4 shows median number of answers related with the community graph of security.

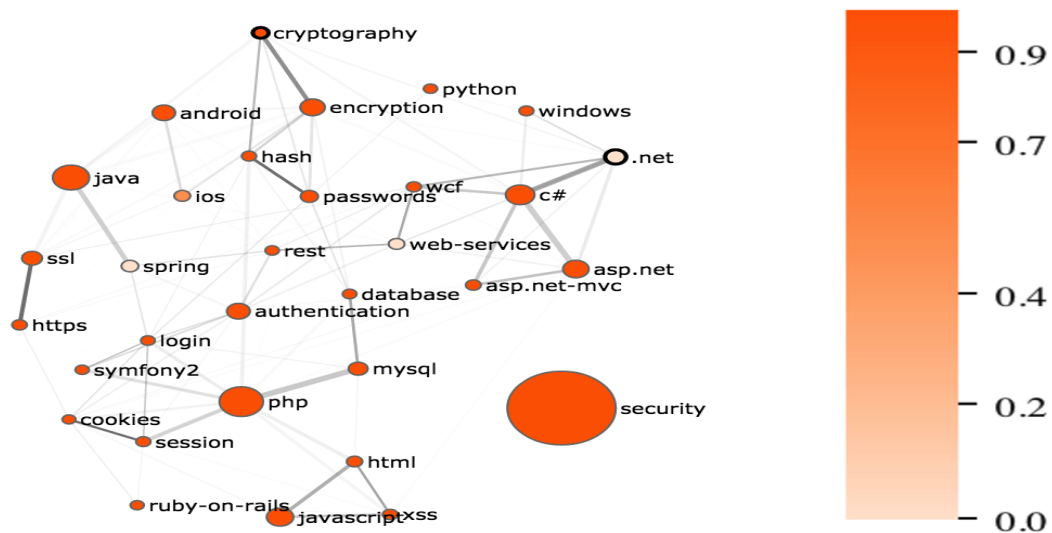


Figure 4.4: Median number of answers in security community

- *Low rate of unanswered questions.* Security-related questions have a a low rate of unanswered questions. 3.69% of the questions have neither answers nor comments while 8.52% questions have no answers, but trigger discussions.
- *Security is a trending topic in SO.* Fig. 4.5 shows the graph of number of questions

at SO, since the first one appeared in 2008. Although the number of security posts are less compared to rest of SO posts, Fig. 4.6 shows the increasing rate of security questions related with javascript, android and ios respectively.

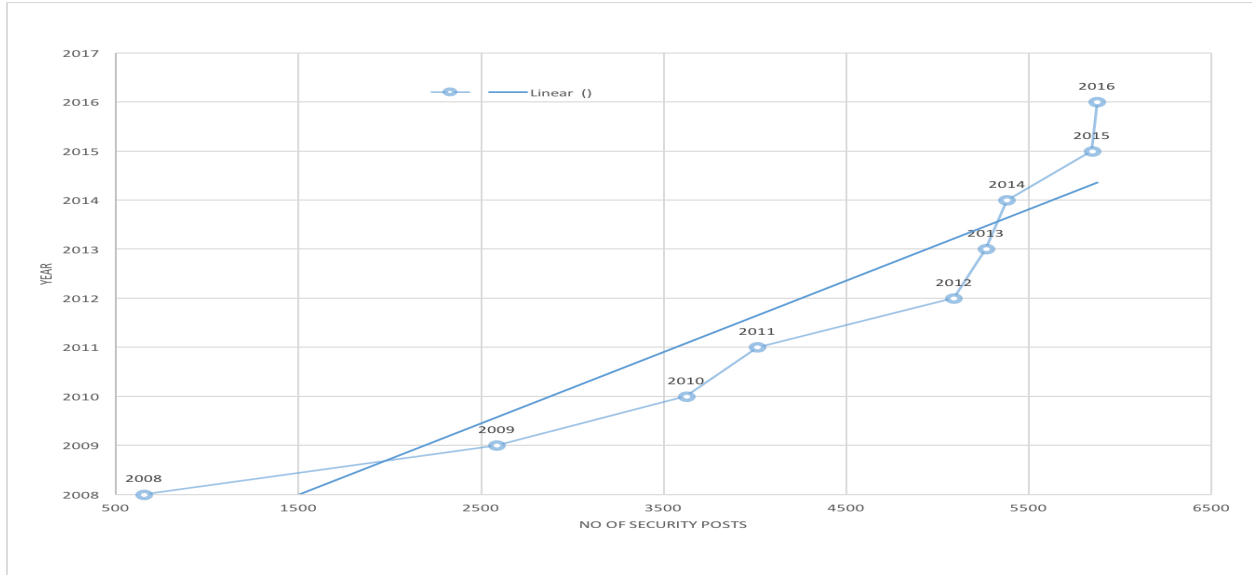


Figure 4.5: Number of ‘security’ posts per year with linear trendline

- *Security attracts a diverse community.* For detecting communities who have cross-cutting interests in security, we have used **tagoverflow**,² an online tool which provides interactive map of tags from Stack Exchange sites.³ Fig. 4.7 shows a community graph centered around *security* tag. Nodes in the graph represent tags and the area of a node is being proportional to the number of questions associated with *security* and that particular tag. The edges represent relation. Their width is related to the number of questions with both tags (e.g. with both php and mysql), while their shade – how much more often they occur than one should expect by random chance. The co-occurrence weight which is used for edge shade and strength is calculated from the *observed to expected ratio*.

²Piotr Migdał, Marta Czarnocka-Cieciura, TagOverflow (2015), <https://github.com/stared/tagoverflow>

³<http://stackexchange.com/sites>



Figure 4.6: *Number of security-related posts per month (from top, clockwise, left-to-right: javascript, android, ios, php posts tagged with ‘security’)*

4.3 RQ3: *What are the main challenges developers face during security feature implementation?*

In order to answer this question, we randomly selected 150 questions from our dataset and analyzed them qualitatively. Our findings are as follow:

Security is confusing and hard to implement. Developers face challenge in while working with security features either for hard-to-understand concepts or the confusing APIs available to implement security functionality. For example, the following question was asked by a

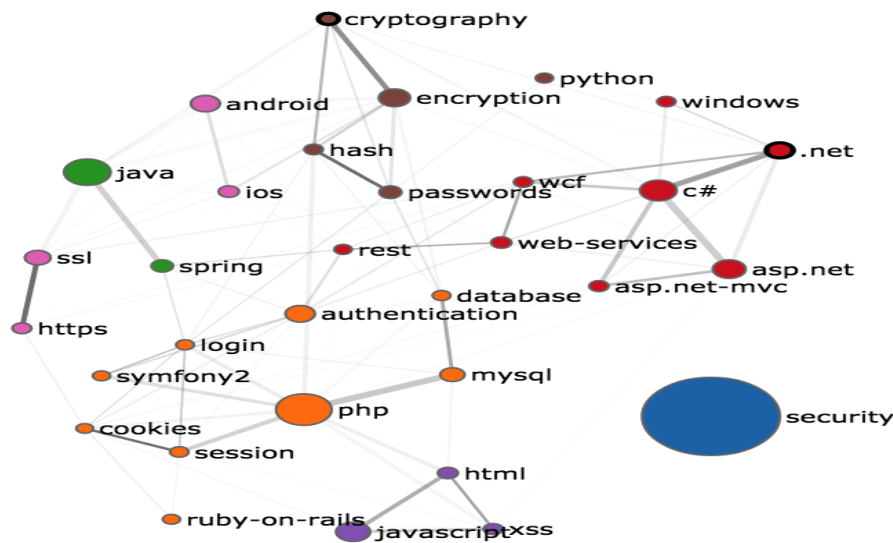


Figure 4.7: Communities involved in ‘security’ discussion

developer:⁴

When a user registers I clean the password with
`mysql_real_escape_string` as follow:
`password = clean($_POST['password']);`
 Before adding it into database I use:
`$hashedpassword = sha1('abcdef'.$password);` and save it into `mySQL`.
 My question is should I clean it or am I protected that the password is
 hashed before adding it into the DB?

At first glance, it seems the question is related to the secure password storage in the database. However, the accepted answer below points out that the questioner used *mysql-real-escape-string* function in a completely misunderstood context where using this function might break the system later, thus showing that this was the main point of confusion for the user.

⁴<http://stackoverflow.com/questions/7274242/should-i-mysql-real-escape-string-the-password-entered-in-the-registration-form>

```
Well, there is one major misunderstanding. mysql_real_escape_string()
does not clean anything. It has nothing to do with security at all.
This function is used just to escape delimiters and nothing more.
It can help you to put string data into SQL query - that's all...
```

Cryptography is difficult to implement. Developers often talk about the difficulty they face when implementing encryption/decryption. Their difficulties range from understanding difference between symmetric and asymmetric encryption, hashing and encryption, encoding and encryption to choosing proper library to implement encryption/decryption feature. For example, the following post shown in Fig. 4.8 simply asked about providing a sample code to encrypt/decrypt a string in C# programming language. The question has more than 20 answers discussing various libraries for encryption and their pros and cons. Interestingly, the post is still active although the question was posted eight years ago and still doesn't have an accepted answer.

Encrypt and decrypt a string

Can someone give me the code to encrypt and decrypt a string in C#?

asked 8 years ago
viewed 460620 times
active 17 days ago

446 votes
282 stars

edited Mar 22 at 16:51 by CodeCaster (78.2k ● 10 ● 86 ● 141)
asked Oct 14 '08 at 17:07 by NotDan (13.5k ● 28 ● 95 ● 146)

2 Check this link codeproject.com/KB/recipes/Encrypt_an_string.aspx – Dimi Dec 17 '10 at 19:28

2 Needed something simple... this link worked for me saipanyam.net/2010/03/encrypt-query-strings.html – MrM Apr 18 '11 at 20:31

4 I would HIGHLY recommend dropping 3DES and using AES-GCM. AES-GCM is NOT found in .NET 4.5 crypto libs and IS different from 'usual AES' (=AES-CBC mode usually). AES-GCM is far better than 'usual' AES for cryptographic reason I won't go into. So jbtule has the best answer below under this Bouncy Castle AES-GCM subsection. If you don't believe us, at least trust the experts at the NSA (NSA Suite B @ nsa.gov/ia/programs/suiteb_cryptography/index.shtml: The Galois/Counter Mode (GCM) is the preferred AES mode.) – DeepSpace101 Jan 25 '13 at 9:45

1 @Sid Personally I'd prefer AES-CBC + HMAC-SHA2 over AES-GCM for most situations. GCM fails catastrophically if you ever reuse a nonce. – CodesInChaos Feb 27 '13 at 20:08

2 @Sid Nonce reuse is a bad idea, yes. But I've seen it happen, even with competent programmers/cryptographers. If that happens, GCM breaks down totally, whereas CBC+HMAC only develops some minor weaknesses. With an SSL like protocol GCM is fine, but I'm not comfortable with it as the standard "encrypt&authenticate" API. – CodesInChaos Feb 27 '13 at 21:54

show 5 more comments

22 Answers
active oldest votes

UPCOMING EVENTS
2016 Community ends in 6 days

BLOG
How We Make M Overflow: 2016 E
Stack Overflow P Don't Care If Brei

Want a job?
Platform Engineering Remote Opportunity w Datapipe Jersey City, REMOTE
java ruby-on-rails
Software Engineer, Us Pure Storage, Inc.

Figure 4.8: A post showing a developer's question about encryption

Balance between functionality and security. Often times, developers have faced the dilemma of prioritizing between functionality and securing a software from between. Developers face this dilemma in terms of meeting deadline, satisfying client even to the extent of providing *security backdoor* in the software.

How should I ethically approach user password storage for later plaintext retrieval?

▲ 1289 As I continue to build more and more websites and web applications I am often asked to store user's passwords in a way that they can be retrieved if/when the user has an issue (either to email a forgotten password link, walk them through over the phone, etc.) When I can I fight bitterly against this practice and I do a lot of 'extra' programming to make password resets and administrative assistance possible without storing their actual password. asked 6 years ago viewed 57261 times active 2 months ago

★ 401 When I can't fight it (or can't win) then I always encode the password in some way so that it at least isn't stored as plaintext in the database—though I am aware that if my DB gets hacked that it won't take much for the culprit to crack the passwords as well—so that makes me uncomfortable.

UPCOMING EVENTS
2016 Community

Figure 4.9: A post showing a developer's ethical dilemma for plaintext password retrieval

Table 4.1: *Topic Names and Related Top 10 Key Terms (Un-stemmed)*

<i>Topic name</i>	<i>Top LDA words</i>
Server/Network(System Sec)	<i>ip server secure site security address port users web</i>
System Security	<i>file program code command run system windows process linux</i>
Spring security(Implementation)	<i>spring method security custom class filter authentication object controller exception</i>
Front-end (Web Sec)	<i>php input injection xss javascript form html code sql jquery</i>
Java EE (Implementation)	<i>spring xml security configuration web application boot tomcat java login</i>
RESTful services (Web Sec)	<i>token api oauth client rest server authentication request authorization access</i>
Authentication (Access Control)	<i>login page user redirect session form logout url security log</i>
Password storage (System Security)	<i>passsword hash username database salt user email store encrypted md</i>
File System (System Security)	<i>file image content php upload folder iframe chrome htaccess site website directory</i>
Web Service (Web Security)	<i>service wcf client soap ws message web sts binding webservice security certificate</i>
Authorization (Access Control)	<i>role user admin access group permissions based model authorization member</i>
Windows Web Framework (Implementation)	<i>net asp windows iis application account web server sql</i>
Mobile Security	<i>app android device ios card phone payment keychain store apk cordova</i>
Java (Implementation)	<i>applet java exception policy assembly code flash system error security</i>
Cryptography (System Security)	<i>key certificate private public encryption ssl aes tls signature rsa</i>
Session management (Web Security)	<i>cookie request session https csrf http ajax post domain response cors</i>
Spring Framework (Implementation)	<i>org springframework java beans factory jar release lang support abstractbeanfactory</i>
Grails Framework (Implementation)	<i>grails plugin saml groovy security core mod error apache idp cas</i>
Firebase platform (Implementation)	<i>firebase rules database table acl rule data delete user</i>
Lower level security (System Security)	<i>string buffer byte random length memory number overflow attack array shellcode</i>

Chapter 5

Conclusion

5.1 Implications

For Researchers. Our large-scale empirical study provides an overall view about security-related topics that developers ask about during SDLC. Our qualitative analysis also shows that security misconceptions and lack-of ‘usable’ security library are key deterrent factors for developers to implement security features properly. Domain-specific automated question answering tool and security bootstrap framework would be an interesting research topic that we encourage future research to focus on.

For Educators. Teaching secure coding practices and educating developers about potential vulnerabilities can go a long way for the longevity of software’s functionality and dependability. The benefits of fixing code earlier in the software development lifecycle (SDLC) – a byproduct of both a strong security education program and finely tuned testing tools – are well researched. Security educators and trainers can potentially use our findings for better planning and prioritizing their training materials. SQL injections, for example, have been prevalent in applications for the past 15 plus years – and they still manage to evade developers and code testers. Security educators can emphasize on web security suite to produce training materials, leveraging gamification, visualization techniques to help developers learn security concepts and secure coding practices with manageable learning curve.

For Practitioners. One vital part of embedding security into the development culture is to learn how the development team in an organization works and use that to determine how and how secure coding practices can be better implemented. Project managers can utilize the findings from this study to prepare security checklist and requirements for the developers and allocate adequate time and resources for security consideration from early stages of SDLC. Also, application security teams can better collaborate and communicate with developers once they understand the challenges of developers and their struggle to implement security features during SDLC. In a short, our findings can help different stake holders involved with SDLC to start an effective risk-communication about software security.

5.2 Related Work

In this section, we briefly review related studies. We first review some previous text mining and topic-modeling studies based on Stack Overflow. Next, we describe studies which concentrate on developers' security perception in software engineering.

Text Mining and Topic Modeling. Allamanis et al.¹⁸ used topic modeling to identify the relation among question concepts and types in Stack Overflow. Barua et al.⁵ conducted a large empirical study on all the posts on Stack Overflow. They used LDA to analyze the topics and trends of what developers talk about. Bajaj et al.⁹ conducted a study on web development related posts on Stack Overflow. They concluded several points about common challenges and misconceptions among web developers. Rosen et al.⁸ narrowed down the research scale by specifically studying mobile-related questions on Stack Overflow. They also applied LDA to the dataset to investigate the topics mobile developers are interested in. Linares-Vasquez et al.¹⁰ performed an exploratory analysis of mobile development issues using Stack Overflow. They employed topic model to extract the main discussion topics from more than 400K mobile-development related questions. Beyer et al.¹⁹ manually analyzed 450 Android-related posts on Stack Overflow and found that the most common question types are “How” and “What”. They also found the dependencies between question types and problem categories. Nadi et al.²⁰ performed an empirical investigation into the obstacles Java

developers face with cryptography APIs, through triangulating data including top 100 Java cryptography related questions on Stack Overflow. They identified nine main topics related to cryptography and the results suggest that developers do face difficulties using cryptography. Pletea et al.²¹ conducted a sentiment analysis on GitHub to gauge the presence and atmosphere surrounding security-related discussions on GitHub. Vasilescu et al.²² reported the activity correlation of askers and developers in Stack Overflow and GitHub. Stevens et al.²³ analyzed SO data to find developers concerns and usage pattern of android permission model. Sinha et al.²⁴ proposed a heuristic approach to prevent API secret key leakage in source code repositories like GitHub.

The work closest to ours is the work by Yang et al.²⁵ which ran a topic modeling on SO data-set to analyze security-related discussion on SO. In contrast, a) our work is based on a more fine grained question set that we obtained by applying our tag-heuristics to get a wider coverage of security-related posts at SO. We used 40 keywords whereas the previous work used 7 keywords for filtering data. b) Unlike previous work, we have adopted mixed methods to analyze the data which helped us to understand the dynamics of developers in question asking and problem-solving. Qualitative study also helped in exploring rich descriptions of complex phenomena and illuminating the experience and interpretation of events by actors with widely differing stakes and roles which was not readily available by merely analyzing data-set through topic modeling.

Software Security during SDLC. With the ever changing landscape of security vulnerabilities, privacy leaks, increasing number of exploits, along with the dynamic nature of today's software development life cycle, we need to ask how we can bridge the gap between people and organization wanting to be more secure, and actually being more secure. Organizations need to embrace built-in security practice both in software lifecycle and custom and COTS software supply chain, from a *socio-technical system design/engineering perspective*. As software development is a "team sport", we need to take into consideration each player (from requirement analysts to UI/UX consultants to developers to testers to devOps groups) in terms of shared responsibility for adopting security best practices. A robust software engineering process should consider security as an important part of development and integrate

security requirements early stages of SDLC.²⁶ Historically, there has been a divergent point of view held regarding security between security teams and developers. Security personnel usually blame developers for putting their focus on code and time to release rather than protecting the code. On the other hand, security is often thought of as a “consideration” or “toll gate” within the project plan rather than being built in from the early stage of project planning, development and production cycles.²⁷ Several software development life cycles (SDLC) have been proposed, in order to integrate security in various phases of software development. Microsoft adapted a tighter security integration model of waterfall SDLC which is known as security development lifecycle (SDL).²⁸ SANS also introduced a new methodology called Scalable and Agile Lifecycle Security for Applications (SALSA), advocating for better security education opportunity for developers, integrating security best practices into existing SDLC strategically, improving code review and analysis, to name a few. Besides this advocacy, research has been conducted to understand perceptions of both security and development teams about application security maturity. Survey has found that 71% of developers and 51% of application security teams feel security is not adequately addressed during the SDLC. 60% security practitioners believe security is addressed ideally in the design and development phase, but in reality it is addressed in the launch and post-launch phase, according to 51% developers.²⁹ Study has shown that there exists a conceptual gap³⁰ between developers’ understanding of security and their attitudes regarding their personal responsibility and practices for software security. The researchers have also found a no-questions-asked attitude in the team collaboration during different phases of SDLC. Oliveira et al.³¹ have argued that software security is a blind spot in a developer’s heuristic-based problem-solving approaches. They have shown that security is neither a priority, nor a typical point of concern in developer’s mindset while coding. They have, although, shown that developers pay attention once they are primed about a design/coding flaws on the spot. Balebako et al.¹ have presented a relationship between security and privacy behaviors of app developers and company characteristics (e.g., company size, having a CPO, etc.). They have demonstrated that developers know and care more about security tools than privacy policies and tools, mostly because of obscure language of privacy policies and unclear user

data collection disclosure of third-party libraries.

5.3 Summary and Future Work

In this research work, we conducted a large-scale empirical study by specifically investigating security-related questions on Stack Overflow, in-order to gauge the security perception of developers and to understand the challenge and misconceptions they face during SDLC. We first used two heuristics to extract security-related posts from Stack Overflow. And then we used topic model, LDA to cluster different security-related questions based on their texts. After obtaining different topics of security-related questions, we used their metadata to make various analysis. We found that security-related questions on Stack Overflow cover a wide range of topics. These topics mainly belong to five main categories, i.e., web security, access control, implementation specific, cryptography, and system security. And among them most questions are about web security. We have also showed qualitatively that security is hard to implement due to hard-to-grasp concepts and difficult-to-manage configurations for bootstrapping security in SDLC. For our future work, we intend to run a fine grained text mining in different developer communities in SO to explore security maturity in different languages/frameworks.

Bibliography

- [1] Rebecca Balebako, Abigail Marsh, Jialiu Lin, Jason Hong, and Lorrie Faith Cranor. The privacy and security behaviors of smartphone app developers. *Proceedings 2014 Workshop on Usable Security*, 2014.
- [2] Jim Witschey, Olga Zielinska, Allaire Welk, Emerson Murphy-Hill, Chris Mayhorn, and Thomas Zimmermann. Quantifying developers' adoption of security tools. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015*, 2015.
- [3] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? *Proceeding of the 33rd international conference on Software engineering - ICSE '11*, 2011.
- [4] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design lessons from the fastest q&a site in the west. *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, 2011.
- [5] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering*, 19(3):619–654, 2014.
- [6] Clayton Stanley and Michael D. Byrne. Predicting tags for stackoverflow posts. In *Proceedings of International Conference on Cognitive Modeling, ICCM'13*, 2013.
- [7] Vasudev Bhat, Adheesh Gokhale, Ravi Jadhav, Jagat Pudipeddi, and Leman Akoglu. Min(e)d your tags: Analysis of question response time in stackoverflow. *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, 2014.

- [8] Christoffer Rosen and Emad Shihab. What are mobile developers asking about? a large scale study using stack overflow. *Empirical Software Engineering*, 21(3):1192–1223, 2016.
- [9] Kartik Bajaj, Karthik Pattabiraman, and Ali Mesbah. Mining questions asked by web developers. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 112–121, 2014.
- [10] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk. An exploratory analysis of mobile development issues using stack overflow. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR 2013, pages 93–96, May 2013.
- [11] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.
- [12] David Blei. Probabilistic topic models. *Proceedings of the 17th ACM SIGKDD International Conference Tutorials on - KDD '11 Tutorials*, 2011.
- [13] Stephen W. Thomas. Mining software repositories using topic models. *Proceeding of the 33rd international conference on Software engineering - ICSE '11*, 2011.
- [14] Adrian Kuhn, Stéphane Ducasse, and Tudor Gírba. Semantic clustering: Identifying topics in source code. *Inf. Softw. Technol.*, 49(3):230–243, March 2007.
- [15] M. F. Porter. Snowball: A language for stemming algorithms, Oct 2001. URL <http://snowball.tartarus.org/texts/introduction.html>.
- [16] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*, 2008.
- [17] Carson Sievert and Kenneth E Shirley. Ldavis: A method for visualizing and interpreting

- topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, pages 63–70, 2014.
- [18] Miltiadis Allamanis and Charles Sutton. Why, when, and what: Analyzing stack overflow questions by topic, type, and code. *2013 10th Working Conference on Mining Software Repositories (MSR)*, 2013.
- [19] Stefanie Beyer and Martin Pinzger. A manual categorization of android app development issues on stack overflow. *2014 IEEE International Conference on Software Maintenance and Evolution*, 2014.
- [20] Sarah Nadi, Stefan Krüger, Mira Mezini, and Eric Bodden. Jumping through hoops: Why do java developers struggle with cryptography apis? In *Proceedings of the 38th International Conference on Software Engineering, ICSE '16*, pages 935–946, 2016.
- [21] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. Security and emotion: sentiment analysis of security discussions on github. *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, 2014.
- [22] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Stackoverflow and github: Associations between software development and crowdsourced knowledge. *2013 International Conference on Social Computing*, 2013.
- [23] Ryan Stevens, Jonathan Ganz, Vladimir Filkov, Premkumar Devanbu, and Hao Chen. Asking for (and about) permissions used by android apps. *2013 10th Working Conference on Mining Software Repositories (MSR)*, 2013.
- [24] Vibha Singhal Sinha, Diptikalyan Saha, Pankaj Dhoolia, Rohan Padhye, and Senthil Mani. Detecting and mitigating secret-key leaks in source code repositories. *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 2015.
- [25] Xin-Li Yang, David Lo, Xin Xia, Zhi-Yuan Wan, and Jian-Ling Sun. What security questions do developers ask? a large-scale study of stack overflow posts. *J. Comput. Sci. Technol. Journal of Computer Science and Technology*, 31(5):910–924, 2016.

- [26] K.r. Van Wyk and G. McGraw. Bridging the gap between software development and information security. *IEEE Security and Privacy Magazine*, 3(5):75–79, 2005.
- [27] Dave Shackelford. Integrating security into development, no pain required, Sep 2011. URL <https://www.sans.org/reading-room/whitepapers/analyst/integrating-security-development-pain-required-35060>.
- [28] Microsoft security development lifecycle (SDL): Process guidance, 2013. URL <https://msdn.microsoft.com/en-us/library/windows/desktop/84aed186-1d75-4366-8e61-8d258746bopq.aspx>.
- [29] 2012 application security gap study: A survey of it security & developers, 2012. URL http://www.ponemon.org/local/upload/file/2012_application_security_gap_final.pdf.
- [30] J. Xie, H. R. Lipford, and B. Chu. Why do programmers make security errors? In *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 161–164, Sept 2011.
- [31] Daniela Oliveira, Marissa Rosenthal, Nicole Morin, Kuo-Chuan Yeh, Justin Cappos, and Yanyan Zhuang. It’s the psychology stupid: How heuristics explain software vulnerabilities and how priming can illuminate developer’s blind spots. In *Proceedings of the 30th Annual Computer Security Applications Conference, ACSAC ’14*, pages 296–305. ACM, 2014.