# Security and Emotion: Sentiment Analysis of Security Discussions on GitHub

Replication of a paper done by Daniel Pletea, Bogdan Vasilescu, Alexander Serebrenik for the MSR 2014 challenge

Rasheed Elsaleh for CSC666
Dept. of Computer Science
Northern Kentucky University
Highland Heights, KY, USA

***Abstract*— Application security is becoming increasingly prevalent during software and especially web application development. Consequently, countermeasures are continuously being discussed and built into applications, with the goal of reducing the risk that unauthorized code will be able to access, steal, modify, or delete sensitive data. This paper is a replication of the work done by Daniel Pletea et al [1] where they gauged the presence and atmosphere surrounding security-related discussions on GitHub, as mined from discussions around commits and pull requests. First, security related comments accounted for approximately 4.3% of all comments and about 11% of all discussions were security related discussions. Second, it was found that more negative emotions are expressed in security related discussions than in other discussions. These findings confirm the importance of addressing security related issues in applications and providing the proper training to do so. It also expresses the need to test applications thoroughly for security vulnerabilities in order to reduce the frustration of developers.**

***Keywords—Security; GitHub; sentiment analysis; mining challenge; paper replication***

## I. INTRODUCTION

Application security is becoming increasingly prevalent during software and especially web application development. Security vulnerabilites have higher financial implications than traditional bugs and may lead to disclosure of sensitive, confidential, or personally identifiable data as well as legal ramifications [1] .

Consequently, countermeasures are continuously being discussed and built into applications, with the goal of reducing the risk that unauthorized code will be able to access, steal, modify, or delete sensitive data. Moreover, despite their potentially huge impact, security concerns are often given only a side thought. This is happening because programmers are typically trained to write code that implements the required functionality without considering its security aspects [1].

In this paper we gauge the sentiment surrounding security related discussions on GitHub through a dataset provided for the MSR 2014 challenge [2], the dataset used for this paper differs slightly from the original dataset, several updates were made to the dataset since its first release. GitHub is the largest code host in the world, with more than 5M developers collaborating across 10M repositories. GitHub offers support for distributed version control (Git) and pull-based development, and allows developers to comment on commits or pull requests [1].

The study is divided into to parts. First, the identification of security related discussions amoung commit and pull request comments. Comments are identified as security related if they contain one or more preselected security words. Second, expressions of emotion are explored for all discussions and security related discussions are then compared to non security related discussions. It was found that security related discussions account for approximately 11% of all discussions in the GitHub dataset, and that they have a more negative tone than other discussions. These findings confirm the importance of properly training developers to avoid security mistakes during development, and the need to test applications thoroughly for security vulnerabilities in order to reduce frustration and improve the overall atmosphere surrounding a project.

The rest of this paper is organized as follows. Section 2 discusses the related work from the previous paper. Section 3 discusses the methodology. Section 4 contains results. Section 5 contains a comparison between this paper the original paper it was based on. Finally, Section 6 contains the conclusion.

## II. METHODOLOGY

### A. Dataset

The MSR 2014 dataset used for this paper is an updated version of the MSR 2014 mining challenge dataset [2], which contains more data than what was used in the original paper. It was available as a MySQL or MongoDB data dump, MongoDB was chosen for this paper. MongoDB is a NoSQL database, a MongoDB database contains collections (tables), a

collection contains documents (rows) and a document contains fields (columns).

There were 60,845 commit comments and 93,198 pull request comments, each were located in a separate collection. The dataset includes data from the top-10 starred software projects for the top programming languages on Github, which gives 90 projects and their forks [2].

The term "discussion" is used to denote the comments belonging to a commit or a pull request. Both comments and discussions were analyzed.

### B. Identification of Security Related Comments and Discussions

A *keywords-based* approach was used to identify security comments. The keywords were taken directly from the original paper, the keywords were constructed in two ways. First, by selecting well known security words such as *security, ssl, encryption, authentication, authorization, encrypt, de- crypt, audit, integrity, repudiation, confidentiality, privacy, ldap, dsa*. Next, the authors selected keywords from the top 25 co-occuring tags on stack overflow. Finally, a Porter stemming was performed on the list resulting with the following keywords: *access policy, access role, access-policy, access-role, accesspolicy, accessrole, aes, audit, authentic, authority, authoriz, biometric, black list, black-list, blacklist, blacklist, cbc, certificate, checksum, cipher, clearance, confidentiality, cookie, crc, credential, crypt, csrf, decode, defensive programming, defensive-programming, delegation, denial of ser- vice, denial-of-service, diffie-hellman, dmz, dotfuscator, dsa, ecdsa, encode, escrow, exploit, firewall, forge, forgery, gss api, gss-api, gssapi, hack, hash, hmac, honey pot, honeypot, honeypot, inject, integrity, kerberos, ldap, login, mal- ware, md5, nonce, nss, oauth, obfuscat, open auth, openauth, openauth, openid, owasp, password, pbkdf2, pgp, phishing, pki, privacy, private key, private-key, privatekey, privilege, public key, public-key, publickey, rbac, rc4, repudiation, rfc 2898, rfc-2898, rfc2898, rijndael, rootkit, rsa, salt, saml, sanitiz, secur, sha, shell code, shell-code, shellcode, shib- boleth, signature, signed, signing, sing sign-on, single sign on, single-sign-on, smart assembly, smart-assembly, smar-tassembly, snif, spam, spnego, spoofing, spyware, ssl, sso, steganography, tampering, trojan, trust, violat, virus, white list, white-list, whitelist, x509, xss.*

A comment was labled as a security related comment if it contained a full three-letter keyword (e.g. sha or ssl) or a substring of any other keyword. The approach taken to apply the labels to the documents in MongoDB was to query the database in a Python script for each keyword, the result contained the *commit_id* for all documents where the keyword was found in the body field. Next, a loop iterated through the results and an update command was called to add a new boolean field named "security" to each document returned in the query. The find query (1) and the update command (2) are below:

```
r = db.COLLECTION.find({"body": {'$regex' : "%s"%(w), '$options': 'i'}}, {'commit_id': 1, 'body': 1})
```
*(1)*

```
db. COLLECTION.update({"_id": comment.get('_id')}, {"$set": {"security": True}})
```
*(2)*

### C. Sentiment Analysis

The sentiment analyis was performed using the Natural Language Text Processing (NLTK) tool [3]. NLTK accepts HTTP POST requests containing the text to be analyzed, the text can be a maximum of 80,000 characters long. The response to each requsest is a JSON object containing two attributes a label and a probability. The label will be *pos* if the text is determined to be positive, *neg* if the text is negative, or *neutral* if the text is neither pos nor neg. The probability attribute contains the probability for each label, neg and pos will add up to 1 while neutral is standalone. If neutral is greater then 0.5 the label is neutral, otherwise, the label is either pos or neg, whichever has the greater probability. The tool trained its classifiers on both twitter sentiment as well as movie reviews datasets.

The approach taken to perform the sentiment analysis of the GitHub commit and pull request comments begins by quering the database (3) for all comments for a given collection. For each document in the query results, the database is checked for an exisiting sentiment, if one exists an API call is not made to limit the number of API calls. If a sentiment does not exist an API call is made with the comment as the text field of the request truncated at 80,000 characters to meat the API's limit. The result returned by NLTK is stored in the database by running an update command (4) which creates a new sentiment field with the entire JSON object returned by NLTK as its value.

The NLTK tool's API has a daily limit of 1000 requests per IP, to overcome this limitation a VPN service was used, after receiving a response code of 503 from the API the Python script would terminate, the IP address for the VPN service was then changed manually and the script run again. This was repeated until all comments were analyzed resulting in approximately 24 hours of processing time.

```
res = db.COLLECTION.find({}, {"body":1})
```
*(3)*

```
db. COLLECTION.update({"_id": id}, {"$set": {"sentiment": r.json()}})
```
*(4)*

## III. Questions and Results

### A. How many comments and discussions are security related?

The numbers for the security comments and discussions are found in Table I. It can be seen that the number of security related comments is around 4% of all comments for both commits and pull requests. In addition, the number of security related discussions for both commits and pull requests is around 11%. These percentages are similar to the percentages

found in the original paper even though the overall number of comments was different. These percentages are big enough to derive relevant conclusions from the MSR 2014 Challenge Dataset.

TABLE I. NUMBER OF SECURITY-RELATED COMMENTS AND DISCUSSIONS RESULTS

| Type | | Comments | Discussions |
|---|---|---|---|
| ***Commits*** | Security | 2895 (4.76%) | 1798 (9.78%) |
| | Total | 60845 | 18380 |
| ***Pull Requests*** | Security | 3753 (4.03%) | 1638 (13.15%) |
| | Total | 93198 | 12457 |

*B. Are the security comments or discussions different (sentiment-wise) than the rest of the comments or discussions?*

*1) Analysis*

Table II and Table III show the results for the sentiment analysis on both types of discussions (commits and pull requests). For discussions, the fraction of negative discussions is higher for security related discussions than non-security related discussions 65.85% vs. 48.56% for commits and ??? for discussions. Similar results are observed for individual comments, the fraction of negative comments is higher for security related comments than for non-security related comments: 59.97% vs. 47.83% for commits and ??? for pull requests, this shows that aggregating comments into discussions makes little difference.

The difference in percentages is due to the different approaches taken to calculate them. Sentiment analysis for all comments was done using the NLTK tool. The percentages for comments were calculated directly by counting the different sentiment labels. Querying the database retuned the counts for each type. Example of a query for all negative comments that are non-security related is seen in (5) and an example of a query for all negative comments that are security related in seen in (6).

```
db.COLLECTION.find({$and: [{'security': {$exists: false}},
{'sentiment.label': 'neg'}]}).count()
```

*(5)*

```
db.commit_comments.find({$and: [{'security': true},
{'sentiment.label': 'pos'}]}).count()
```

*(6)*

The same approach would not work for discussions because discussions contain multiple comments with different sentiment labels. In order to determine the sentiment of a discussion a special query is constructed (7), this query aggregates comments by commit or pull request and the values of the security and sentiment labels fields are put into their own arrays. The sentiment of a discussion is determined by the count of the label types for each result. Similarly to how the NLTK tool performs the selection, the discussion is neutral if the count of neutral labels is greater than or equal to 0.5 otherwise it is not considered. For negative, if the number of negative labels is greater than or equal to the number of positive labels the discussion is negative, this means all ties are considered negative. Otherwise, the discussion is positive.

```
r = db.COLLECTION.aggregate([{"$group": {"_id":
"$commit_id", "security": {"$push": "$security"},
"sentiment":{"$push": "$sentiment.label"}}}])
```

*(7)*

TABLE II. STATISTICS FOR SENTIMENT ANALYSIS ON COMMITS

| Type | | Positive | Negative | Neutral |
|---|---|---|---|---|
| ***Discussions*** | Security | 11.57% | 65.85% | 22.58% |
| | Rest | 19.61% | 48.56% | 31.84% |
| ***Comments*** | Security | 18.41% | 59.97% | 21.52% |
| | Rest | 26.14% | 47.83% | 26.03% |

TABLE III. STATISTICS FOR SENTIMENT ANALYSIS ON PULL REQUESTS

| Type | | Positive | Negative | Neutral |
|---|---|---|---|---|
| ***Discussions*** | Security | | | |
| | Rest | | | |
| ***Comments*** | Security | | | |
| | Rest | | | |

*2) Wilcoxon test*

The original paper performed a Wilcoxon test but the representation of the data used for the test was not clear and was not replicated. The Wilcoxon test performed here was on two binary vectors with the length of the total number of comments. The vectors contained 1s for the number of neg comments and zero for the remaining count. The first vector was for security related comments and the second vector was for non-security related comments. The results showed a very small p-value of < 2.2e-16, which is good enough to reject the null hypothesis (the number of security related comments and non-security related comments are similar) and accept the alternative hypothesis that they are not similar.

## IV. COMPARISON

The original paper was submitted to the MSR 2014 Mining Challenge [4], per the challenge's requirements, the paper must only be 4 pages long, this resulted in a lot of details being left out of the paper. Therefore, the approaches taken to calculate the results might be different than the approaches taken in the original paper. In addtion, there are several other differences described in the sections below.

### A. Dataset

The dataset linked from the challenge's website no longer exists, after further research it was found the the data had been updated and moved to a new location [2]. The updated dataset contained more comments than the one used in the original paper. It contained 60,845 commit comments vs. 60,658 in the original paper and 93,198 pull requests vs. 54,892 in the original paper.

The dataset was available in the form of a database dump in two flavors MySQL and MongoDB. The original paper did not specify which database was chosen. For this paper MongoDB was chosen.

### B. Security comments and discussions count

The original paper constructed a list of keywords to be used in identifying security comments and discussions. A custom list and keywords from Stack Overflow tags were used, there was no explanation as to how the words were collected from Stack Overflow. However, the complete list of keywords was provided and it was used in this paper. The approach used for counting the number of comments and discussions is the same as the original paper. Count percentages were similar for both comments and discussions in both papers.

### C. Sentiment Analysis

To perform the sentiment analysis both papers used the Natural Language Text Processing tool (NLTK) [3]. The tool itself has changed since the time it was used for the original paper. The main differences are the character length for the submitted text has increased to 80,000 characters, the tool's classifiers were trained on more data including twitter data, and the number of daily API calls allowed per IP decreased from 5000 requests to 1000 requests creating a major challenge. The original paper used an Amazon Web Services EC2 virtual machine with multiple IPs to over come the API call limit. The analysis was completed in approximately 16 hours. For this paper a VPN service was used but IPs were changed manually every time the limit was reached, this significantly increased the overall time taken to complete the analysis.

For the approach, the original paper did not specify the method taken to calculate the percentages of sentiment types (Negative, Neutral, Positive), particularly for the discussions, which contain multiple comments, with possibly different sentiment types. The resulting percentages in this paper were slightly different than the original paper, which is expected given that the dataset is slightly different and the sentiment analysis tool was trained on different data. However, the final conclusion that security comments and discussions were more negative than the rest of the data was the same.

Finally, the original paper performed a couple of tests to formalize the difference between the security related and non-security related comments and discussions, however, they did not explain how these tests were performed and what the data representation was for these tests. Therefore, these tests were not successfully replicated. Instead, a Wilcoxon test was performed on two binary vectors with the length of the total number of comments. The vectors contained 1s for the number of neg comments and zero for the remaining count. The first vector was for security related comments and the second vector was for non-security related comments. The results showed a very small p-value of < 2.2e-16, which is good enough to reject the null hypothesis (the number of security related comments and non-security related comments are similar) and accept the alternative hypothesis that they are not similar. Future work will be to find the proper representation for the dat in order to perform the tests.

## V. Case Study

In the original paper the authors conducted a case study to manually verify the accuracey of the overall analysis. Their approach was to select 30 security related commit discussions. Based on their security scores (the number of keywords found in each discussion), they randomly selected 10 discussions from the top 10%, 10 discussions from the middle 10% and 10 discussions from the bottom 10% of all security related discussions. The 30 discussions were then shuffled and analyzed (both their relevance as security related discussions as well as the dominant emotion as compared to the NLTK results) without knowing their security scores and their sentiment analysis results.

Their analysis showed a significant number of false positivtes (discussions mislabeled as security-related) among the middle tier: 6/10 false positives and lower tier: 7/10 false positives. In most cases, the discussions had been labeled as security-related due to a single keyword being present in them. They also observed a mixture of agreement and disagreement between the emotion labels computed by NLTK and the ones resulted from manual review.

For this paper manual analysis was done for several randomly selected commit discussions, 5 of them are documented in Table IV.

First, random discussions were selected from the results of the query in (8). The commit id was used in (9) to retreive the security identified comments and sentiment results. Second, a python script was used to count the number of security words in a discussion by iterating through all comments for that particular discussion. Finally, the comments and sentiment results were manually analyzed one by one to determine their correctness.

```
db.commit_comments.find({security: true}, {"commit_id": 1}).skip(Math.random() * 10)
```
*(8)*

```
db.commit_comments.find({commit_id: "45f2b3ba67e2fb81754cbfce19b743a4d6d1108f"}, {"body":1, "security": 1, "sentiment.label": 1}).pretty()
```
*(9)*

The analysis yielded a number of observations. First, the majority of the analyzed comments were false positives (identified as a security related comment when they were not security related). Many of the discussions were labeled as

security related due to a single keyword present in a comment. In addition, many comments were identified as security related due the use of words common in both security related and non-security related contexts. An example of the latter is the word hack, the word appears in many comments referring to a code hack not a security hack. One comment was identified as security related due the word hack appearing in the name of an IRC channel (#hackercounsel). Another example of a false positive is a comment that contained a confidentiality agreement statement:

TABLE IV. CASE STUDY RESULTS

| Sec. relevance | Discussion (commit ID) | # sec. keywords | Sec. relevance((human) | Sentiment (tool) result | Sentiment(human) |
|---|---|---|---|---|---|
| High | 1f67d07c60c37e60052db37fc03d42af482c2d03 | 4 | No | neg | neg |
| High | 9d16fe6283667396094d49559a37fc672c06252c | 5 | No | neg | neutral |
| Low | dec541f7e56506342394e466fa6e9d9805dd77fb | 1 | No | neutral | neutral |
| Medium | 915b77339711ec1278ac06ec80d206133bdb427a | 2 | No | neg | neutral |
| Medium | dc83072878ed9636c8158a014bd9fa4acc1ccce3 | 2 | Yes | neg | neutral |

"This message is intended only for the designated recipient(s). It may\r\ncontain confidential or proprietary information"

These issues do not come as a surprise given the approach taken to identify security related comments using a keyword based approach only. To gain acceptable results a classification method would be required.

Second, the analysis of the sentiment results from NLTK showed fewer false positives than the keyword based identification approach. The analysis showed a mixture of agreement and disagreement between the emotion labels computed, mostly leaning towards disagreement. Many of the negative lables were considered neutral. The analysis done here was on the overall discussion and not per comment.

The sentiment results can also be explained by the difference in the type of data used by NLTK to train the classifiers. For more accurent results, classifiers should be trained on related commit and pull request comments.

## VI. CONCLUSIONS

This paper replicated the original work done by Daniel Pletea et al in [1], which consisted of mining emotions from secuirity related comments and discussions from commits and pull requests on GitHub, provided by a dataset from the MSR 2014 Mining Challenge. It was found that security related commets account for approximately 4.3% of all comments and 11% of all discussions. These findings confirm the anecdotal evidence that implementing application security and finding security problems in code can often lead to frustration and anger among developers, and is a source of tension to the overall project atmosphere.

As mentioned in the original paper the final results indicating more negativity in security related comments and discussions should be taken with a grain of salt. First, the results could be affected by the unbalanced number of sample sizes between security related and non-security related comments and discussions. Second, the keywords based approach to identify security related comments is weak and can generate many false positives which was observed manually. Third, sentiment analyis was performed using the NLTK tool, the results showed that the number of negative security related comments and discussions is higher than non-security related comments and discussions. The limitations in API calls per day prevented the completion of the analysis in a timely manner. The tool also rejected a few long comments containing code and configuration.

Future work should include the use of multiple datasets. A classification technique to identify the secuity related comments instead of keywords approach used in this paper. Sentiment analysis classification algorithms should be used directly and trained on relavant data instead of only using the NLTK tool.

## REFERENCES


[1] Daniel Pletea, Bogdan Vasilescu, Alexander Serebrenik, Security and Emotion: Sentiment Analysis of Security Discussions on GitHub, MSR, 2014

. *(references)*

[2] MSR 2014 Mining Challenge Dataset, http://ghtorrent.org/msr14, accessed March 2017.

[3] Python NLTK Sentiment Analysis API, http://text-processing.com/, accessed March 2017 .

[4] MSR 2014 Mining Challenge, http://2014.msrconf.org/challenge.php, accessed March 2017.

# Appendix A