International Institute of Information Technology, Hyderabad

Department of Computer Science and Information Security

System and Network Security (S22CS8.403) Paper Review

Sudipta Halder [2021202011]

Nitin Kumar [2021202020]

Professor: Dr. Ankit Gangwal

Research Paper : The Effect of Google Search on Software Security, ACM CCS 2021

March 10, 2022

Abstract

In the research paper we have tried to understand how Google Search Engine can affect the software's security that we use everyday. The search engine is used by developers to look for code online, but Google does not yet rank them based on how secure the code actually is, making the code highly vulnerable to exploits and attack and very often these codes enter into production as well which can affect a very large audience. The paper discusses and shows how this issue can be solved by doing re-ranking on the search result to provide secure and best practice code.

Keywords: Google Search Engine, vulnerable, exploits, secure, software

Acknowledgements

We would like to express our sincere gratitude to our advisor *Professor:* Dr. Ankit Gangwal for the continuous support of our research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped us in all the time of research and writing of this review paper.

Summary

Now a days, Google Search has become the way to go for the developers when they are stuck in a programming problem. A study has already shown that developers tend to prefer programming forums like Stack Overflow, Quora more as compared to official documentation or books[1]. But the problem is the results we get from Google Search or these platforms might not be secure always. It has been found that one of the top three Google Search results has about one fourth chance of resulting into insecure code[2]. Similarly, this research paper has found that one third of the code examples related to cryptography to be insecure in Stack Overflow and apart from that they were also used in numerous apps[3]. Sorting out secure examples in Stack Overflow is difficult since it does not provide any indicator. Most people end up picking the code which has been voted most without checking whether it is secure or not. In addition to it, this research has shown that people tend to use Google Search(91%) more than using the programming forums like Stack Overflow's (36%) search engine to look for answer.[4]. The reason is quite obvious though. First they do a Google Search and if any programming forum offers solution for that they visit it.

So, two online surveys were performed. In the first one, it was checked how the Google Search results were performing in terms of security. Actually, the goal was to check top ten Google Search results (t_{10}) and find the ratio of secure and insecure Stack Overflow results among them. A Stack Overflow web-page was defined to be insecure if the top accepted answer turned out to be insecure and vice-versa.

192 participants were involved in this survey 1 who were given programming assignments based on AES encryption which consisted of initializing vector (IV), cryptographic key (KEY), symmetric cipher (CIPHER)[2]. An open source dataset called TUM-CRYPTO[5] was used which consists of several secure and insecure JAVA code snippets from Stack Overflow on the topics cipher, hash, hnv, hnvor, IV, key, tls, tm.

The survey 1 result is visible in the below Figure 1. It was found from Survey 1 result that in t_{10}, t_5, t_3(where t_n: top n results in Google Search) the probability of insecure result was significantly higher than the secure result. In addition to that, it was also found that if a result ends up in t_{10} then the probability that it would end up in t_3 is much higher than ending up in the next ranks.

To tackle this problem, a security based re-ranking technique was introduced. A new type of labelling was also introduced for Stack Overflow webpages called "Secure Best Practice Examples" for KEY and IV. The Secure Best Practice Examples are different from normal Secured examples in the sense that the later may not be functionally complete even though it is secure. In that case the solution is of no use since it is not complete. In case of CIPHER, only secure and insecure categories

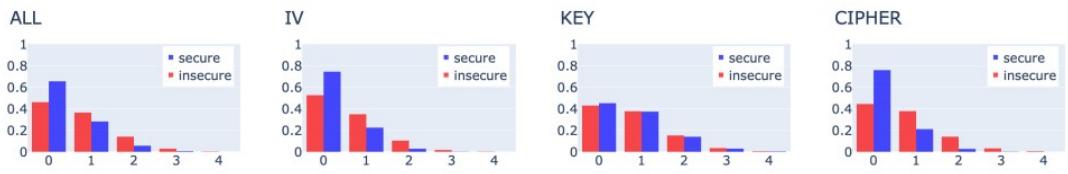

Figure 1: Binomial distribution of secure (blue) and insecure (red) Stack Overflow results over the top ten results t_{10}. X: count of secure/insecure results, Y: probability of secure/insecure.[2]

Figure 2: Binomial distribution of boosted secure best-practice (green), secure (blue), and insecure results (red) for IV, KEY, and CIPHER over t_{10} X: count of secure best-practice, secure or insecure results in t_{10} Y: probability of secure best practice/secure/insecure t_{10}.[2]

were kept because JAVA has a standard and secure way of initializing CIPHER using JAVA SDK.

A semi supervised clustering algorithm called DBSCAN was used for clustering. The datapoints (webpages here) were then sent for clustering. After successful clustering, a representative was chosen from each cluster and it's source code was reviewed manually to decide whether it was best practice or secure in case of KEY, IV where as secure or insecure in case of CIPHER.

Now, let's discuss how re-ranking was done. A parameter search was performed to find a optimal boost value b_s for secure results in range [-1.0, 1.0] to rerank the Stack Overflow webpages such that the secure best practice results get the highest priority followed by only secure results followed by insecure results with least priority[2].

From Figure 2, it is now visible that how re-ranking helped to increase the probability that secure best practice results get highest priority followed by only secure results followed by insecure results across all tasks IV, KEY, CIPHER and in search result t_{10}, t_5, t_5 where t_n = top n search results.

In study 2, total 218 participants took part and they had to search the feasible solutions online for the same task(KEY, IV, CIPHER) and write JAVA codes to execute them. They were divided into two groups. First one was control group, who were supposed to use Google's default search bar and second one was treatment group who were supposed to use the security based re-rankable search bar.

By analyzing the data from study 2, following points were noted:

- The mean security jumped from 1.6(control group) to 1.9(treatment group) which indicates that the treatment group submitted more secure code snippets[2].
- The mean functionality jumped from 2.35(control group) to 2.58(treatment group) which indicates that the treatment group submitted more functional solutions using the modified search engine[2].
- Also it was noticed that the treatment group(45.8%) faced 15.8% more secured results than the control group(30.8%)[2].
- In case of best practice results, treatment group got 36.9% whereas control group got 0.4%[2].
- In case of insecure results, treatment group got 17.3% which is 51.5% less than the control group(68.8%)[2].
- It was also noted that in the treatment group, the summation of votes per rank were significantly higher almost across all ranks(t_1 to t_{10} except t_9) as compared to the summation of votes per rank in control group. This is very useful since if it could have happened otherwise, the developers could have left the customized search engine since it would be continuously predicting less upvoted results at the top[2].

So, it is clear from the results of the study 2 that this re-ranking is giving promising results in terms of security, functional correctness and completeness. In addition to that, it is user friendly since user doesn't need to install or setup or learn anything and it continues to do it's job silently. This work is pretty significant from company's point of view also. Every tech company has to give security training to its employees and even after that the programmers mainly rely on Google Search or Stack Overflow type platforms when they are stuck. So, the security training is of no use then. This work will address the problem at the root and hence even if they search in Google they will end up with most relevant and secure results only most of the time and without any extra effort.

Critiques

Assumptions

- The Study 1 consisted of only 192 people and 218 people participated in Study 2. We argue that these numbers are not up to the mark for getting good results. Table 1 shows the number of unique queries and top ten search results(t_{10}) corresponding to each task. We can clearly see there are not much data points for a Machine Learning model to be trained properly.

Coding Tasks	**Number of**	
	Unique Queries	**Top ten search results(t_{10})**
ALL	274	3800
IV	98	1290
KEY	87	1210
CIPHER	89	1300

Table 1: Unique Queries and top ten search results corresponding to Coding Tasks

- From Figure 3 it is evident that, though developers were recruited from various countries trying to keep a healthy ratio among different countries, we still argue that more developers could have been recruited which we already mentioned in the previous point and also from more countries. Currently we are seeing people from only 5 counties(USA, Germany, China, India, Brazil) in both study 1 and study 2. The participation from more countries could have added more diversity to the data and accuracy to the AI model.

- The content of social help websites is not static, they keep changing with time as new people with different approaches come to solve the problem, but the label in the TUM-CRYPTO[5] is static and the dataset was created 3 years ago as visible from the last commit in GitHub. Along with that the TUM-CRYPTO[5] dataset is small to do any reasonable study on it for large scale generalisation like this.

		Age		
Mean = 32.03/30.94	Median = 29.5/29	Stddev = 10.34/8.59	Min = 18	Max = 74/59
		Country of Origin		
USA = 40/60	Germany = 23/19	Brazil/India = 13/18	China/Brazil = 11/9	Other = 105/112
		Gender		
Male = 171/188	Prefer not to say = 5/22	Other = 6/1		Female = 10/7
		Level of Education Achieved		
High School = 30/21	Bachelor= 80/102	Master = 44/60	PhD = 18/11	Other = 20/24
		Professional		
Yes = 133/161	No = 52/33			N/A = 7/24
		Security Background		
Yes = 56/46	No = 129/143			N/A = 7/29
		Java Years of Experience		
< 1 year = 51/61	1 to 2 years = 35/40	> 2 years = 82/80		N/A = 25/37
		Java Primary Focus of Job		
Yes = 31/49	No = 158/144			N/A = 3/25

Figure 3: Detailed data about demographics of participants for Study 1 (N = 192) and Study 2 (N = 218).[2]

- There are several security bugs or code vulnerabilities which are known to very few people (not yet public knowledge) which might have got skipped in this case. For example sudo command was vulnerable to overflow exploit (for 10 years) and was not known to the public till 2021, such cases could have got skipped.

Technical Approach

- In both Study 1 and Study 2, some programming assignments were given on the basis of AES encryption which consists of initializing a symmetric cipher (CIPHER), an initialization vector (IV), cryptographic key (KEY)[2]. This decision was most probably taken due to the presence of these 3 keywords(IV, KEY, CIPHER) in the TUM-CRYPTO[5] dataset. It consists of secure and insecure JAVA code snippets of the topics like cipher, hash, hnv, hnvor, IV, key, tls, tm. So, we can see that among all these 8 keywords only 3 keywords(KEY, IV, CIPHER) were used. But a task could have been easily set up such that all the keywords get used from TUM-CRYPTO[5] dataset. If that happened, the we could have got lot more datapoints to train the AI model properly. Also, it could have obviously increased the accuracy of the model reducing false positive and false negative cases.
- The study says they were trying to improve the top 10 result but the users were allowed to click on any link which may not include all top 10 google search results or it may also contain results beyond top 10 which might have affected the distribution of result of study 1.
- A semi supervised clustering algorithm called DBSCAN has been used here for clustering of the datapoints. We argue that a better algorithm called HDBSCAN could be used in this case. We are catering the following points to support our claim:
  - DBSACN fails in proper clustering when the data is having differing density [7].
  - HDBSCAN is faster than DBSCAN. HDBSCAN consumes about half the time as comapared to DBSCAN for about 2 lakhs datapoints [7].
- Using a semi-supervised clustering method like DBSCAN to cluster secure and best practice code is not a good choice, since the codes which have similar embedding may have completely different functionality and hence different security flaws. Code is not like the English language where the similar looking sentences are almost related to the same thing and hence have similar embeddings.
- The webpage labelling function that they have used has a problem, they have marked a webpage as insecure or secure if the top answer in that page is insecure or secure. The problem here is a web page mostly has many solutions for a given problem so labelling should not have been done solely based on the top answer in that page. The same approach was used to mark a webpage as best practice or not which should not have been done.This approach would declare the webpage insecure even if the rest of the answers are secure and vice versa.
- There are several cases when we need to search for vulnerable codes and insecure code, for those cases the security based ranking will just make the learning process difficult.
- The keyword around which they were asked to solve the problem was related to security, subconsciously giving them hints that they have to write more secure code, making the entire survey a little useless. That could be the reason that they were not able to detect the effect where with increase in interaction with their modified search engine the code became more secure and functional on the control group.

Analysis/Results

- Re ranking might affect the probability to get the desired solution to problem. When a person looks for a solution for a problem online, he may not be looking for a secure code always he may be just looking for a stub of code which he can write into his secure module or insecure module,

his priority at the moment remains to get the task done first and then add security as per his module, but if we directly give him search results which is ordered by measure of secureness of code then the code stub he may be looking for may go down the search result making it hard for him to get his desired code fastly. Just like the tradeoff between security and convenience. It may also happen that the secure code he is presented with most of it remains useless for his module and only a small part of it will be needed by him making the ranking of results fairly useless for him.

- It has not considered the case when no secure solution for a given problem exists and in this case the ranking process will just won't help in any way.
- TUM-CRYPTO[5] dataset has been used here. It consists of secure and insecure JAVA code snippets of the topics like cipher, hash, hnv, hnvor, IV, key, tls, tm. Though it is a good dataset in terms of quantity of examples but a better dataset could have been used if exists. This is because we have a lot more relevant security keywords like PRG, PRF, CCA, public-key, private-key, certificate. So, any database which covers variety of examples like this could have been better in this case. Also, TUM-CRYPTO[5] dataset is based only on JAVA. So, if we can create a dataset which contains examples from multiple coding languages that would be better.
- Only one search engine was modified in this case and that is the Goggle Search Engine. We agree this is the most used search engine throughout the world. Once we revamp this search engine in terms of security, the problem of security is eradicated almost completely. But, there might be people who uses different search engines like Bing, Yahoo, DuckDuckGo. So, these people won't be able to leverage the re-rankable search engine. Also, the survey was conducted online in participant's personal computer. So, some of them might also have used different search engines violating the rule.
- Even after manually checking each cluster, we were able to achieve a precision of 0.81. Although there were no false positives, we still think that the precision could have been made much better.

Improvements and Extension

1. As discussed in the assumptions section of critiques, we can run the same surveys with more number of people and people from more countries. This repetition of the experiment would be more relevant for Study 2 since it gives us the result how the re-rankable search engine performs in real life. So, the more number of people participates in Study 2, the more we would get to know how well the modified search is performing.

2. We can also look for better datasets than TUM-CRYPTO[5], which will contain more examples related to security and which also supports various programming languages along with JAVA. We can repeat the same experiment with that dataset and look how that performs in comparison to this present study[2].

3. As discussed in the technical approach section of the critiques, the labelling of webpages (secure best practice, secure, insecure) was done by analyzing the top answer of that webpage. This would declare a webpage insecure even if it contains some secure answers and vice versa. So, we can check at least 50% of the answers of that webpage to declare it secure best practice/secure/insecure.

4. As discussed in the technical approach section of the critiques, study 1 was done to improve the top ten Google Search results. But the participants were not explicitly instructed to click on top ten search results only. They might have clicked results beyond top ten(t_{10}). So, we should explicitly instruct users to click on top ten search results only.

5. In continuation to the above point, we can also observe that in study 2, functional codes were instructed to write in JAVA only since the dataset consisted only examples of JAVA code snippets. So, if we can look for a dataset which supports multiple programming languages, we can keep a list of programming languages from which the participants can choose to write functional codes for a particular task. This will be more flexible for the participants since not everyone has expertise in JAVA and also more and more participants could be recruited due to this flexibility. Also, the result which we will receive from it will be more relevant, diverse and feature rich and will cover more test cases than the previous one.

6. As discussed in the technical approach section of critiques, we can use HDBSCAN as the clustering algorithm instead of regular DBSCAN to improve speed and performance and check how it is performing in comparison to the present one[2].

7. As discussed in this research paper[3], we can also revamp interface of Stack Overflow to give users a visual feedback which answer is secure and which one is insecure. Figure 4 shows visual example for insecure code. Figure 5 shows visual example for secure code. In addition to this, we can also add a feature to upvote and downvote the answers based on its security as well.

8. Many cryptographic code analysis tools have been developed throughout the years like FixDroid, LGTM, CodeQL which can be installed as a plugin in different IDE's and they continue to give us security analysis (for e.g. Unsafe deserialization, Out-of-bounds Write, NULL Pointer Dereference etc.)[6] as we keep on writing code in IDE. We can use these along with our modified search while writing code. Since there is an edge case that if they are no secure results available in the internet for a particular task, we will end up with picking an insecure solution only. In

Figure 4: Example of insecure example in Stack Overflow.[3]

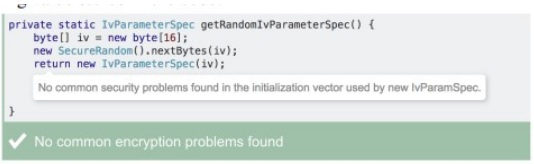

Figure 5: Example of secure example in Stack Overflow.[3]

that case, these tools can save us by showing their analysis and notifying if the code snippet is insecure.

Also, we can run some parallel experiments combining the normal search engine, modified search engine and these code analysis tools to see how they are performing against each other. Some combinations are listed below:

(a) Normal Google Search Engine
(b) Customized Google Search Engine(using security based re-ranking service)
(c) Normal Google Search Engine along with these code analysis tools
(d) Customized Google Search Engine(using security based re-ranking service) along with these code analysis tools.

9. Google or the programming forums themselves can ask for feedback from user "whether the solution on site was secure or not", using which they can develop models which can be later used by them to show secure results to users.

10. We can also classify developers in terms of experience like senior developers, junior developers, system architect etc. and see how their performance is varying. Generally, the experienced developers should perform well i.e. submit secure and functional code even with normal Google Search Engine since they are in industry for a long time. They should know these things pretty well beforehand. But in case of junior developers who don't have experience more than a year or two may not perform that well without the modified search engine. We can check whether the junior developers using modified search engine can outperform the senior developers without using the modified search engine. If that happens, it will work as a great feedback for the present study [2].

11. We can run the same experiment in a controlled environment like laboratory. Also, we can integrate this re-ranking service to other search engines like Bing, Yahoo, DuckDuckGo then few users who might use those search engines would also be able to leverage this security based ranking service. This would be really helpful because of the following reasons:

(a) Those participants who used other search engines instead of this modified search engine, whatever search they have done in web could not be recorded since recording can be done only if he uses the given search engine only. So, if the integration happens across all search engines, this problem will be eradicated. This will in turn make the resultset more complete than before.
(b) The flexibility will increase for users. Those who are on other search engines do not need to switch to Google Search for this experiment.

Conclusion

We can affect a very large audience if we implement security based re-ranking of google search results, which will lead to development of secure code at a very large scale. Very good research methodology has been adopted to highlight what are the sources of most of the vulnerabilities in softwares and also gives a solution for it. Along with that we also get to see the effect of our proposed solution on software development, the method used to conduct the study is also done very well. The selection of population for the study was done appropriately, also the information revealed to them was done in such a manner that it does not affect the result, some of their methods were debatable but still the results that came out were remarkable. Work has already started in this direction to implement a version of this or something similar to this by labelling the answer on these social help sites based on security which will affect the search result and hence improving the secure software development quality.

Bibliography

[1] Yasemin Acar et al. "You get where you're looking for: The impact of information sources on code security". In: *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2016, pp. 289–305.

[2] Felix Fischer, Yannick Stachelscheid, and Jens Grossklags. "The Effect of Google Search on Software Security: Unobtrusive Security Interventions via Content Re-ranking". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2021, pp. 3070–3084.

[3] Felix Fischer et al. "Stack Overflow Considered Harmful?" In: ().

[4] Michael Hucka and Matthew J Graham. "Software search is not a science, even among scientists: A survey of how scientists and engineers find software". In: *Journal of Systems and Software* 141 (2018), pp. 171–191.