Editors: Jens Grossklags, jens.grossklags@in.tum.de | Heather Richter Lipford, richter@uncc.edu | Jessica Staddon, jessica.staddon@gmail.com

Nudging Software Developers Toward Secure Code

Felix Fischer and Jens Grossklags | Technical University of Munich

The prevalence of insecure code is one of the main challenges security experts are trying to solve. We study behavioral patterns among developers which largely contribute to insecure software—googling and reusing code from the Web—and apply nudge theory to harness these behaviors and help developers write more secure code.

Programming is not only a highly difficult task; today it has become utterly complex. There is a vast and quickly growing amount of languages and application programming interfaces. Developers need to be flexible and willing to learn how to apply them in a very short time, and, to deal with this sometimes overwhelming task, they search online for help. Very often they find ready-to-use code examples or open source software that solves the problem at hand. The reuse of these resources provides a very efficient and effective way out. However, it becomes problematic if very popular resources provide solutions that are flawed security wise. Many solutions end up in production software used by billions of people. Some introduce critical vulnerabilities that can be exploited by attackers, for instance, to steal credentials or credit card data or to compromise a device.

We believe that we cannot keep developers from reusing content from the web as this behavior seems to be deeply rooted today. Therefore, we have opted for a different approach that harnesses this observation and tries to guide developers toward content on the web that is safe to reuse. We borrowed this idea from nudge theory, which is a concept from behavioral science and economics. It does not expect people to change their behavior but redesigns things in a way such that common behaviors lead to better outcomes. We redesigned two very fundamental resources—Google Search and Stack Overflow—such that developers can find advice that is helpful and secure.

Insecure Coding Advice on the Web

Stack Overflow is one of the most popular resources. It is a Q&A site that provides helpful advice for almost any coding problem. However, in Fischer et al. 2017,[2] we showed that Stack Overflow provides a large amount of highly vulnerable code examples. Many of them were reused in production code; 15% of apps available on Google Play contained at least one of those insecure snippets.

Even though Stack Overflow provides countless secure code examples that are safe to apply in code, we found that these were hardly reused. In Chen et al.,[5] we compared the popularity of secure and insecure code from Stack Overflow among users. We did this by relying on Stack Overflow's own voting system, which provides a community-given score for each post. Interestingly, insecure code had significantly more upvotes and was more often duplicated across discussion threads or indicated as the top answer. We also found that highly trusted Stack Overflow users—users with a particularly high reputation score—posted insecure code. In other words, all of the very meaningful indicators on Stack Overflow were pointing in the wrong direction security wise.

While Stack Overflow is part of most developers' journey through the web, they typically begin with Google Search. They type in a query and usually follow one of the top-ranked results. Depending on Google's ranking algorithm, developers end up on webpages that provide either secure or insecure advice. Therefore, we wanted to know whether top-ranked results are biased toward secure or insecure

Digital Object Identifier 10.1109/MSEC.2022.3142337
Date of current version: 21 March 2022

code and whether this has a direct effect on software security.

In an online study, we asked 192 developers to solve several programming tasks.[4] They were instructed to use Google Search to find help online. Afterward, we analyzed the distribution of secure and insecure advice among the top search results of all participants. The chance to receive at least one insecure result among the top three ranks was 23%—more than twice as high as for secure code. Developers who clicked on one of those links ended up on a Stack Overflow page that provides insecure code in the top answer of the discussion thread.

In summary, not only are Stack Overflow's own content indicators often misleading, but Google Search's ranking algorithm is too. The two fundamental web mechanics that developers rely on to find information on the Internet are inadvertently promoting insecure content.

Nudge Theory

The paternalistic way to solve this problem is to urge developers not to use Stack Overflow or even Google Search to look for help online but rather advocate for established resources that are safe. Of course, we do not expect this idea to be fruitful. Several studies explored alternatives, such as formal documentation, books, simplified programming interfaces, and code analysis tools.[1] Even though they do help in improving code security, developers still struggle to get functional solutions out of them. In this regard, the web seems to provide better options. Since functional code is the developers' primary goal, it seems unrealistic to convince developers not to use popular web resources. Behavioral science underpins this assumption: changing people's behavior is very hard! Richard Thaler—one of the inventors of nudge theory—framed it the following way: "First, never underestimate the power of inertia. Second, that power can be harnessed."

Nudge theory attempts to design around people's default behavior in a way that leads to better outcomes for the individual and society as a whole. People do not need to change; the surrounding "choice architecture" is changed. We build upon this theory and rely on the observation that developers often make the easy choice. Copying and pasting code examples from the web is as simple as it gets. By ensuring that people reuse secure examples instead of insecure ones, we can keep this level of convenience. Developers do not need to find alternatives to Google Search and Stack Overflow. We designed several nudges that help them to make safe choices. We applied the following nudges in our work.[6]

The *simplification* nudge has been applied to reduce the complexity of measures related to education, health, finance, and employment. Undue complexity reduces the benefits of measures, causes confusion, and deters participation. We implemented this nudge by moving security advice to already-existing and well-established resources that are being used by almost all developers.

Warnings are nudges that are already deployed in user communication of security issues on the web, for instance, if users visit a malicious webpage. It has been shown that warnings are much more effective if they provide *recommendations* that help people out of a potentially dangerous situation. We designed security warnings for insecure code examples on Stack Overflow. They inform developers why the examples were marked as being insecure and what risks could result from the reuse of the code. Below each warning, we provided an ordered list of recommended Stack Overflow posts that offer a very similar but secure example. In the best case, developers only have to make one additional click to find a functional and secure solution.

Reminders can have a significant impact; however, timing greatly matters. Therefore, whenever we identified a copy attempt of insecure code on Stack Overflow, we showed a reminder nudge that warns the user once more and displays recommendations.

Stack Overflow

We integrated these nudges on Stack Overflow and performed a developer study.[3] Participants were divided into two condition groups. The treatment group used a modified Stack Overflow version that applied nudging, while the control group used the original Stack Overflow. Both had to solve several security-related programming tasks where we afterward evaluated the security and functionality of the submitted solutions.

The treatment group submitted more secure solutions than the control group with statistical significance. Both groups achieved the same high level of functional solutions, which meant that our nudging interventions did not interfere with the usability of Stack Overflow. This was also a very important result since less functional solutions in the treatment group would result in developers being drawn away from the website. We were not able to isolate a specific nudge being responsible for the effects. It was rather a combination of the displayed warnings, recommendations, and reminders.

Google Search

The most effective nudge from the literature is the so-called *default* nudge.[6] It automatically preselects the most beneficial choice by default, and people only need to take action if they disagree. Popular examples are automatic enrollments in programs, including education, health, and savings.

A web search generally tries to optimize its ranking in a way that presents the user with the most relevant results. People want to immediately find the information they desire within the top ranks. It is the same for software developers. When searching for code examples, we found that they usually click on one of the top three links. Currently, there is a much higher chance to find insecure code among those results.

From Healthy Food to Secure Code

We approached this problem with an approach similar to the so-called *healthy food nudge.* It has been observed that people usually buy food that is presented at eye level in grocery stores. That means, to nudge people toward eating healthy, one should place healthy food at eye level.

We implemented this nudge in Google Search by putting relevant and secure results "at eye level." In other words, we modified the search ranking in a way that it moves secure and relevant advice to the top three ranks in the results. Developers would then be presented with a secure and relevant choice by default. Since we simultaneously down-ranked insecure results, it becomes even more unlikely that developers will click on one of them.

Ranking Signals

To rerank webpages based on security and relevance, we had to find signals first that sufficiently informed about these properties. In Fischer et al. 2019,[3] we developed a deep learning model that is able to predict whether a Java code example on Stack Overflow is insecure or not. We applied this model to determine the security signal for Stack Overflow pages that discussed questions related to Java. Further helpful tools are publicly available to obtain security signals for different programming languages. For example, LGTM performs large-scale analyses on several popular open source websites, such as GitHub, GitLab, and Bitbucket. It is able to detect the most dangerous known vulnerabilities.

To find relevant results, we tried three different approaches. First, we simply relied on Google Search to find relevant results. Since it is the most popular search engine among software developers, we expected it to perform well in this task. Second, we developed an additional method that identifies the most relevant code examples for a set of given use cases, such as encrypting a message or establishing a secure communication channel. Even though the approach was largely automated, it required manual labeling of a small sample and was also restricted to a programming language and specific use cases. Third, we relied on Stack Overflow's voting system as a signal to identify helpful examples. Both signals—security and relevance—were used to update the ranking algorithm of a custom Google Search engine.

Developer Study

We tested the updated Google Search in comparison to the original Google Search in another online study where developers had to write code to solve several programming tasks.[4] We divided the 218 participants into two groups. The control group was provided with a search bar that used original Google Search. The treatment group used the updated Google Search engine, which applied security-based reranking. Our hypothesis was that the more the treatment group used our modified search engine, the more functional and secure code they would submit in comparison to the control group.

After we evaluated the results from the study, we found that participants in the treatment group submitted more functional and secure solutions than the control group—with statistical significance—the more they used the modified search engine. This showed that the reranking had a significant positive effect on the security and functionality of the written code.

We performed an in-depth analysis of the retrieved and clicked results. We found that 83% of the results received by participants in the treatment group were secure, while 46% of the results were highly relevant to the query. In contrast, in the control group, 68% of the results were insecure. A similar distribution was also present in the clicks made by our participants. Sixty-seven percent of the clicked results were secure in the treatment group—among those 26% highly relevant—while the control group predominantly clicked on insecure results with 84% of clicks made. These results provide a much clearer picture of the causal chain: a higher usage of search engines, up-ranked relevant and secure results, clicks predominantly made on the top three results, and the reuse of code examples found on the related webpages ultimately led to more functional and secure code.

Transparency Versus Unobtrusiveness

Both interventions—on Stack Overflow and Google Search—follow the design principles given by nudge theory. They try to make it as easy and convenient for developers to engage in better security decision making. They achieve this exactly by not interfering with established behavior, such as Googling or copying and pasting code examples. They do not try to restrict any options but rather harness the status quo and lead to better outcomes.

Both approaches do not require developers to be aware of them to use them. Developers do not need to download, install, or learn how to use these methods. They do not have to cope with incomplete or unhelpful documentation or gain the advanced skills that are sometimes required to use security tools such as code analysis.

However, both approaches differ in certain aspects. Warnings and recommendations on Stack Overflow allow developers to make informed decisions on whether or not to reuse insecure code. Security-based reranking of Google Search results provides more secure options by default, without user awareness. On the one hand, the Google Search intervention leads people to stay more or less uninformed about which results are secure and which are insecure and why. On the other hand, developers do not have to pay attention to and follow security warnings, indicators, or recommendations that are often difficult to understand. Moreover, people quickly become habituated to these kinds of interventions. This happens once they disagree with a warning or find recommendations unhelpful.

With the Google Search intervention, developers do not need to evaluate whether vulnerabilities reported by code analysis tools are false positives. Moreover, there are no disruptive effects on the main programming task. The intervention remains completely invisible and does not require anything from the user. Therefore, typical human factors that need to be addressed in the field of usable security may not have any negative effects on security in this approach.

Following the defense-in-depth principle, a combined approach might provide the ideal solution. While Google Search includes code security as a signal in ranking, websites, such as Stack Overflow and GitHub inform and educate their user base about insecure content. This works best if all players are part of the game. Alternatively, a scenario that does not rely on Google and other webpages would be one where companies and institutions run our interventions internally on top of Stack Overflow and Google Search.

Based on the results of our studies, we believe that designing security interventions for developers—as well as for end users—must consider behavioral aspects. In our work, observed behavior formed the basis upon which we designed our interventions. It puts people at the center of the design and dramatically shifts responsibilities away from developers who may be laymen in security toward experts in security and beyond. This way of designing security interventions shows that there is a potential for fixing important security issues in code on a very large scale. The urgency to take action is high as the problem is otherwise much likely to worsen. ■

References

1. Y. Acar, M. Backes, S. Fahl, D. Kim, M. Mazurek, and C. Stransky, "You get where you're looking for: The impact of information sources on code security," in *Proc. 2016 IEEE Symp. Security Privacy (S&P)*, pp. 289–305, doi: 10.1109/SP.2016.25.
2. F. Fischer *et al.*, "Stack overflow considered harmful? The impact of Copy&Paste on Android application security," in *Proc. 2017 IEEE Symp. Security Privacy (S&P)*, pp. 121–136, doi: 10.1109/SP.2017.31.
3. F. Fischer *et al.*, "Stack overflow considered helpful! Deep learning security nudges towards stronger cryptography," in *Proc. 28th USENIX Security Symp. (USENIX Security)*, 2019, pp. 339–356.
4. F. Fischer, Y. Stachelscheid, and J. Grossklags, "The effect of Google search on software security: Unobtrusive security interventions via content re-ranking," in *Proc. 28th ACM Conf. Comput. Commun. Security (CCS)*, 2021, pp. 3070–3084, doi: 10.1145/3460120.3484763.
5. M. Chen, F. Fischer, N. Meng, X. Wang, and J. Grossklags, "How reliable is the crowdsourced knowledge of security implementation?" in *Proc. 41st ACM/IEEE Int. Conf. Softw. Eng. (ICSE)*, 2019, pp. 536–547, doi: 10.1109/ICSE.2019.00065.
6. C. R. Sunstein, "Nudging: A very short guide," in *The Handbook of Privacy Studies*. Amsterdam, The Netherlands: Amsterdam Univ. Press, 2018, pp. 173–180.

Felix Fischer is a Ph.D. student at the Technical University of Munich, Munich, 80333, Germany. He is also a senior researcher at Avast. His research studies include the interaction of people with information security and privacy technologies. His most recent publications focus on software engineers struggling with getting cryptography right and explore machine learning as a tool for usable security and privacy. Fischer received a Diplom in mathematics (with a focus on computer science) from Leibniz University Hannover, Germany, in 2014. Contact him at flx.fischer@tum.de.

Jens Grossklags is a professor of Cyber Trust in the Department of Informatics at the Technical University of Munich, Munich, 80333, Germany. His research and teaching activities focus on interdisciplinary challenges in the areas of security, privacy, and technology policy. Grossklags received a Ph.D. in information management and systems from the University of California, Berkeley. He is a Senior Member of IEEE. Contact him at jens.grossklags@in.tum.de.