How Internet Resources Might Be Helping You Develop Faster but Less Securely

Yasemin Acar, Michael Backes, and Sascha Fahl | CISPA, Saarland University
Doowon Kim and Michelle L. Mazurek | University of Maryland
Christian Stransky | CISPA, Saarland University

In an experiment, Android developers using Stack Overflow to solve common security issues produced functional—but less secure—code. Given today's time constraints and economic pressures, developers need improved official documentation that's both secure and usable.

Mobile devices in general and Android in particular are a growing market, rapidly surpassing desktops and attracting many sometimes-new developers. Security and privacy problems in mobile apps are well-documented; they are sometimes attributed to developers who are inexperienced, distracted, or overwhelmed.[1–6] For example, developers often request more permissions than are actually needed, fail to correctly use secure networking or cryptographic APIs, use insecure options for intercomponent communications (ICCs), and fail to store sensitive information in private areas.

Researchers and practitioners have speculated that one root cause for these programming errors is APIs that are too complicated or insufficiently documented, sending developers to a search engine for help to solve unfamiliar problems. These searches often lead to official API documentation, blog posts, or Q&A forums such as Stack Overflow; the security quality of content available at these resources can vary widely. Author Sascha Fahl and his colleagues, for example, interviewed Android developers whose use of pasted code snippets from Stack Overflow made their code vulnerable to man-in-the-middle (MITM) attacks.[3]

Such anecdotes set the stage for our work. Thus far, we know very little about how these security issues make their way into apps, and most of what we know remains unsubstantiated. In this article, we explore the following anecdotes' validity:

- Which information sources do Android developers use to answer security- and privacy-relevant questions?
- Does the use of Stack Overflow really lead to less secure code than the use of other resources?
- Is the official Android documentation really less usable, resulting in less functional code compared to other resources?

A Brief History of Android Insecurity

Researchers have identified several classes of common Android vulnerabilities. Perhaps the most prominent are problems with validating security certificates, which are used to establish identity at the start of secure communications. The certificate ecosystem runs on the TLS network protocol. Here's how it works. Organizations that run webservers apply for certificates from a recognized certificate authority; these are cryptographically signed assurances that associate a domain name with a specific public key. When a browser or a mobile app visits that webserver, the server presents its certificate. The client must then verify that the certificate isn't expired and matches the expected domain name, and that the server can cryptographically demonstrate knowledge of the private key (secret) associated with the public key named in the certificate. This protocol protects the client from MITM attacks, in which the client is maliciously redirected to an attacker-controlled machine rather than to the server it intended to visit. Without TLS and certificate validation, users can't be assured of secure and authentic communications with banks, webmail services, social networks, and other critical services.

Several researchers have established that TLS and certificate validation are broken in many deployed Android apps.[3,5] The most common problem is that apps accept certificates without properly validating them. This could allow an MITM attacker to present a fake certificate and steal data. Other related problems with secure data transmission, including poor or absent cryptography, have also been identified in many deployed apps.[2–6]

Other researchers have identified problems that allow apps to access too much information on a user's device. Erika Chin and her colleagues identified several deployed apps in which critical messages could be intercepted by unintended recipients, leaking private data.[1] They also identified apps in which services would accept commands from unauthorized senders, enabling unexpected startup or shutdown of services, displaying spoofed data to the user, and causing other potentially serious problems. These problems represent a failure to secure ICC.

Another well-known Android problem is that developers often request more special permissions than they need. These permissions, which guard private data as well as paid activities (such as texting or calling), are rarely considered by users, so there's little incentive for developers to choose them carefully. In fact, Adrienne Porter Felt and her colleagues found that a third of the apps they investigated overprovisioned, requesting more permissions than necessary for their operations.[4] This represents a violation of the basic security principle of *least privilege*—requesting only as many privileges as absolutely needed to run a service.

Taken together, these well-documented flaws make it clear that even well-meaning app developers frequently make security- and privacy-relevant errors that put users at risk.

How Do Real Developers Approach Security Challenges?

To understand the challenges faced during the implementation of security-critical app components, we conducted an online survey of Android developers that covered their experience, their programming habits, and the resources they use.

We collected a random sample of 50,000 email addresses for Android application developers listed in Google Play. We emailed these developers, introducing ourselves and asking them to take our online survey. A total of 295 people completed the survey between April and October 2015.

We asked participants about three security-related issues they might encounter during app development: HTTPS/TLS, encryption, and Android permissions.

Approximately half of the developers (144) said that their Android apps use HTTPS to secure network connections; fewer participants (74, or 25.1 percent) had used encryption to store content securely. (We didn't ask about Android permissions because all Android developers encounter them inherently.) In each case, at least 75 percent of the participants reported looking up information about these topics at least once. Most participants who looked up certificates or encryption reported solving these problems similarly to other programming problems. We find this particularly interesting because certificates and encryption are critical for securing connections and protecting private data, but developers don't appear to treat them differently from less security-critical problems.

Figure 1 illustrates the resources participants reported using to look up information on these three topics, compared to the information resources they used for general programming problems. In all cases, search engines and Stack Overflow were popular, confirming their status as the default resources for most problems. However, for Android permissions only, the official documentation was even more popular: 41 percent of those who had looked up the topic (91 participants), compared to 30 percent (67 participants) for search engines and 29 percent (64 participants) for Stack Overflow. One participant wrote that "[I] don't have to google. [I] go directly to Android developer resource" for authoritative information. Perhaps this is because permissions are Android specific and closely associated with the platform. These findings validate both the need to understand the resources' impact on security and privacy decisions generally, and our choice to compare Stack Overflow and the official documentation more specifically.

An Experiment to Compare Different Information Resources

Next, we used a between-subjects laboratory study to examine how information resources affect developers' security and privacy decision-making.

We recruited 54 participants who'd taken at least one

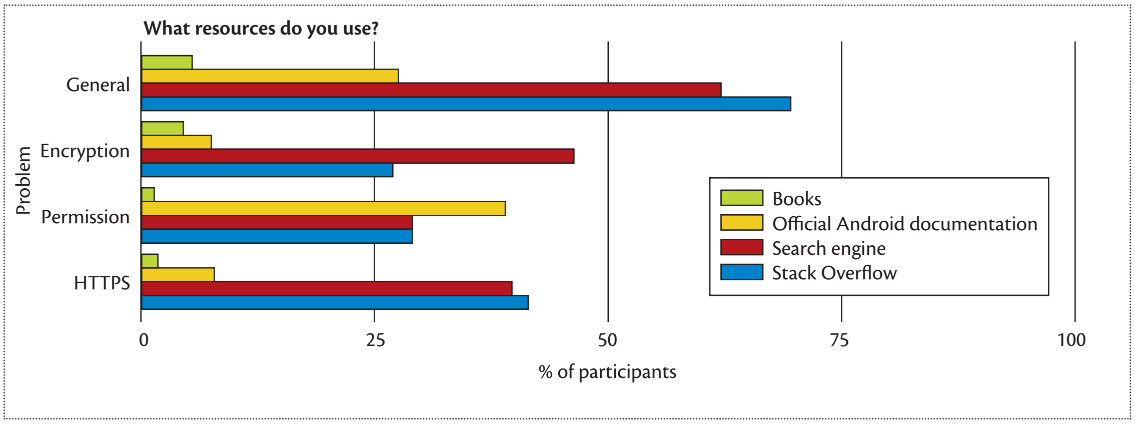

Figure 1. Resources that developers use to look up information on specific security-related issues during app development, versus more general programming problems. Search engines and Stack Overflow were confirmed as the go-to resources for most problems. For Android permissions, however, official documentation was most popular.

course in Android development or had developed professionally or as a hobby for at least one year; we also required participants to correctly answer at least three of five multiple choice questions testing basic Android development knowledge. Participants were recruited in and around Washington, DC, as well as in two university towns in Germany.

Participants were 18 to 40 years old; most were male (85 percent) and had grown up in Germany (52 percent), the US (11 percent), or India (9 percent). The majority were part- or full-time students (89 percent); eight participants were employed as Android app programmers.

The study was approved for ethical compliance by the University of Maryland Institutional Review Board.

Participants were assigned round-robin to one of four conditions:

- *official Android documentation:* allowed to access only websites within the official Android documentation (developer.android.com),
- *Stack Overflow:* allowed to access only questions and answers within Stack Overflow,
- *book:* allowed to use only two books—*Pro Android 4*[7] and *Android Security Internals*,[8] and
- *free choice:* allowed to use any web resource of their choice; also offered access to the two books used in the book condition.

Participants were provided with Android Studio, preloaded with a skeleton app, and a software Android phone emulator. The skeleton app, which was designed to reduce participants' workload and simplify the programming tasks, was introduced as a location-tracking tool that would help users track how much time they spent in various locations (at home, at work, and so on) each day.

After a brief introduction to the study and the skeleton app, participants were given four programming tasks in random order (detailed in the next section), with 20 to 30 minutes to complete each. We took care to implement baseline functionality so that the tasks could be done in any order and the remaining tasks would still work even if a previous task hadn't been successfully completed. While the short time limit impeded some participants' performance, it also simulated the pressure of writing code on tight deadlines, which many app developers face.

Security and privacy weren't mentioned during the introduction or in the directions for each task (the HTTPS task and user credential storage task do inherently imply some reference to security). We deliberately minimized security priming to account for the fact that security and privacy are generally secondary tasks compared to basic app functionality. Instead, we focused on whether developers—who in real-world scenarios might or might not be explicitly considering security—find and implement secure approaches. During the tasks, we collected the search terms used and pages visited by all participants in the non-book conditions.

After completing all tasks (or running out of time), participants were given a short exit interview about their experience. We asked whether the documentation and resources participants had access to were helpful, correct, or both. We also asked whether and how participants had considered security or privacy during each task.

Tasks with Secure and Insecure Solutions

Drawing on related work, we selected four general areas that typically result in Android security or privacy errors: making mistakes in TLS and cryptographic API handling; storing sensitive user data insecurely, such

Table 1. Four Android development tasks designed to have secure and insecure solutions.

Task	Description	Example of secure solution(s)	Example of insecure solution(s)
Secure networking	Convert an HTTP connection to HTTPS in the presence of a certificate error. The connection was to locationtracker.org (created for this study), with a certificate for the nonexistent secure.locationtracker .org. This resulted in an exception indicating a mismatch between the certificate name and domain.	Create a custom hostname verifier for this specific case, pin the mismatched certificate, or insist that the app obtain a correct certificate.	Accept the certificate regardless of the mismatched domain, or without any validation at all.
Intercomponent communication (ICC)	Modify a service within the skeleton app to make the service callable by other apps from the same developer.	Use a custom Android permission to require a matching developer signature; share one ID among all apps from the same developer.	Expose the service to any app by setting the export flag to true (the default for some ICC mechanisms).
Secure storage	Store the user's login ID and password for the app's remote server locally and persistently. The skeleton app contained empty store and load functions for the participant to fill in.	Limit access to only this app, for example, using the "private" mode for shared preferences.	Store the data as world readable in shared preferences or on the SD card.
Least privilege	Dial a hard-coded customer support telephone number by adding code to an existing but not yet functional call button.	Use the `ACTION_DIAL` mechanism, which requires no special permissions.	Use the `ACTION_CALL` mechanism, which requires an additional calling permission.

that it can be accessed by other (unauthorized) apps; using ICC in a way that violates least-privilege principles; and requesting unneeded permissions.

We designed each task so that it could be functionally completed in at least one secure and one insecure way. Table 1 describes all four tasks, with examples of each kind of solution. Before conducting the study, we verified that secure solutions for each task were available in each of the documentation resources we used. This ensured that it was possible (if not necessarily easy) for participants in all conditions to locate a correct and secure answer.

We manually scored each task for functionality: "true" if the code compiled and completed the assigned task, "false" if it didn't. We then manually scored only functional tasks for security, by labeling each participant's code with one of several solution strategies determined to be secure or insecure. Two independent coders performed scoring; conflicts were resolved by discussion to reach agreement.

We analyzed these binary functionality and security scores using a cumulative-link (logit) mixed model (CLMM) regression, which can account for multiple explanatory factors, including multiple tasks per participant. We tested several models for each regression; as is standard, we selected the model with the lowest Akaike information criterion (AIC).

Stack Overflow Helps Achieve Functional Code

In terms of functional correctness, participants allowed to use Stack Overflow or books performed best, with 67 and 66 percent achieving functional solutions, respectively. Participants restricted to the official documentation performed worst, solving 40 percent of tasks. The difference between Stack Overflow and the official Android documentation was statistically significant in our CLMM model ($p = 0.015$). These results are shown in Figure 2.

Participants' perceptions of the tasks only partially dovetailed with these results. We asked participants whether they were confident they'd gotten the right answer for each task. Results are shown in Figure 3. Participants in the free-choice condition were the most confident. This confidence rating somewhat matched our correctness rating (Cohen's kappa = 0.55).

But Stack Overflow Is Worse for Security

We found that the type of resource used had the opposite effect on security than functionality: participants restricted to Stack Overflow were the least likely to achieve secure solutions. In the Stack Overflow condition, only 51.4 percent (18 of 35) of functional solutions were graded as secure, compared to 65.5 percent (19 of 29) for free choice, 73.0 percent (27 of 37) for

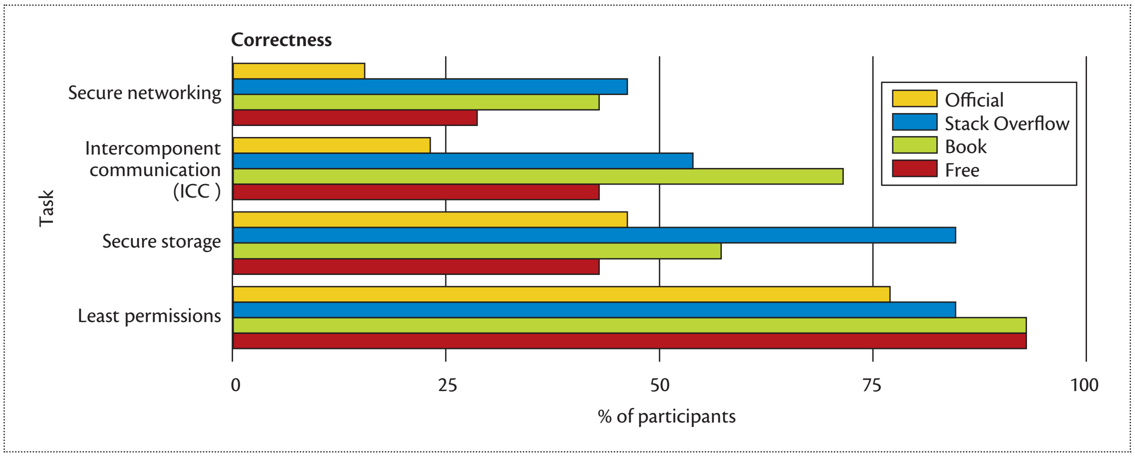

Figure 2. Percentage of participants who produced functionally correct solutions, by task and condition. Participants who used Stack Overflow or books performed best, while those restricted to official documentation performed worst.

book, and 85.7 percent (18 of 21) for official documentation. Figure 4 illustrates these results. Significantly more results in the official-documentation and book conditions were secure than in the Stack Overflow condition according our CLMM (p = 0.005 and 0.01, respectively). The difference between Stack Overflow and free choice, in which many participants elected to use Stack Overflow for most tasks, wasn't statistically significant ($p = 0.07$).

We were also interested in the extent to which participants thought about security while solving each task. We measured this in two ways: whether the participant mentioned security while thinking aloud (as directed) during the task, and whether the participant self-reported considering security for a given task during the exit interview. For both metrics, we considered all tasks, not just those that the participants functionally completed.

Most participants didn't mention security at all while thinking aloud (79 percent of all tasks). In the secure-storage task, 16 participants (30 percent) mentioned security. Of these, all seven solutions that were functional were also secure. In the secure-networking task, 20 (37 percent) mentioned security, but nine later abandoned the task as too difficult or time consuming for a study. In contrast, only five and four participants, respectively, mentioned security or privacy in the least-privilege and ICC tasks. Unsurprisingly, when prompted, more participants self-reported considering security: 60 percent of all tasks (130 tasks). Using this metric, security was most frequently considered for secure networking (80 percent), followed by ICC (70 percent) and secure storage (69 percent). Only 22 percent of participants reported considering security for the least-privilege task. We found no significant difference between conditions for either metric (Kruskal-Wallis, observed $p = 0.92$, self-report $p = 0.19$).

Professionals Are More Functional but Not More Secure than Students

Although the relatively small sample of professionals recruited makes comprehensive comparisons difficult, we examined differences in correctness and security between the professionals (14 participants employed or recently employed as programmers) and nonprofessionals (primarily university students). The professionals were randomly distributed across the conditions: five in free choice, three in Stack Overflow, two in official documentation, and four in book.

Overall, professionals were slightly more likely to produce a functional solution, with a median of 3 functionally correct tasks (mean = 2.8) compared to 2 functionally correct tasks (mean = 2.1) for nonprofessionals. We observed essentially no difference in security results: professionals' solutions were a median 67 percent secure (mean = 69 percent) compared to 67 percent for nonprofessionals (mean = 66 percent). These observations fit with the CLMM results: professional status predicted a small but significant increase in functional correctness but was excluded from the best-fitting security model.

Lookup and Search across Conditions

Participants in the Stack Overflow condition made, on average, 23 queries across the four tasks and visited 53 unique webpages, compared to 15 queries and 35 webpages for the participants in the official-documentation condition. We offer two hypotheses for these results, based on our qualitative observations. First, official-documentation participants were more likely to scroll through a table of contents or index and click on topics that seemed relevant (as opposed to doing a keyword search) than those in other conditions, presumably because the official documentation is more

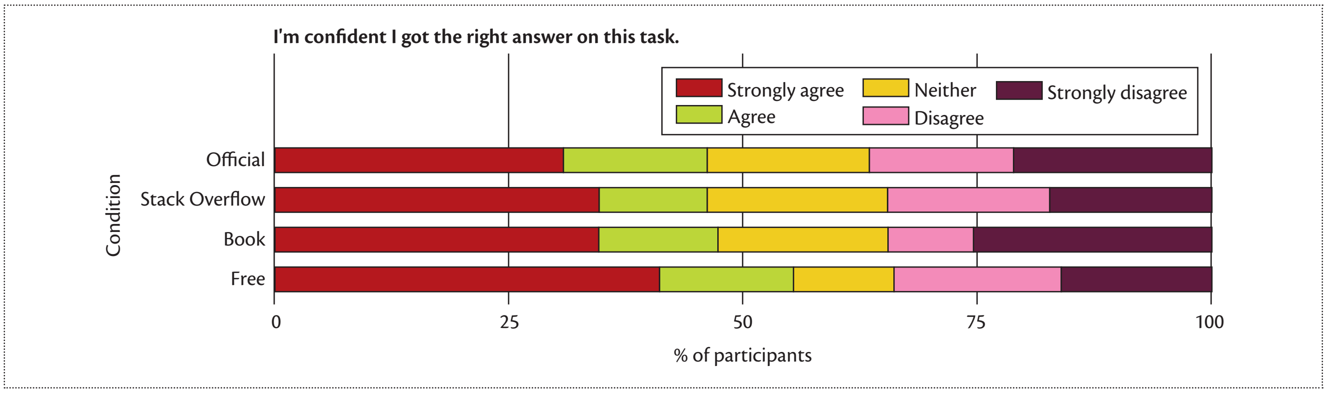

Figure 3. Participants' self-reported confidence in functional correctness, by condition. Participants in the free-choice condition were the most confident.

structured. Second, and perhaps more important, Stack Overflow participants appeared more likely to visit pages that turned out to be unhelpful and restart their searches.

Participants in the free-choice condition queried more like the Stack Overflow group, with an average of 21 queries, but visited pages more like the official group, with an average of 36 webpages. The free-choice participants started every attempt with a Google search. (Admittedly, Google was the browser's homepage.) From within their Google results, every participant selected at least one page in the official Android API documentation, and all but one visited Stack Overflow as well. A few visited blogs, and one visited an online book. These results are consistent with the online survey results reported earlier.

In terms of frequency, official documentation was most popular, representing 50 to 85 percent of all participants' non–Google search pages, except for one outlier who visited it 98 percent of the time. Most participants visited Stack Overflow, which represented 10 to 40 percent of their pages. Although free-choice participants visited more official-documentation than Stack Overflow pages, their functionality and security results more closely resembled the Stack Overflow group's than the official-documentation group's. This might be partially explained by a behavior pattern we observed: free-choice participants spent some time reading through the official documentation, but as the time limit approached, they often used content—usually a copied-and-pasted code snippet—from Stack Overflow.

We also examined the search query text that participants chose. Few participants exactly duplicated one another's queries; so to discern trends, one researcher manually coded similar terms into categories. For example, "restrict access developers," "restrict app access for same developer," and "restrict apps same developer" were categorized together. For the secure networking task, the most common queries involved hostname exceptions and HTTPS, together with just a few searches for certificates, certificate errors, and hostname verifiers. For the ICC task, the most popular searches included manifest, permissions, services, external access, and restricting access. A few more knowledgeable participants searched for intent filters, user IDs, and signatures. For secure storage, the most popular choices included storage, persistent storage, and shared preferences; for least privilege, participants most frequently searched for call and phone call, with a few searching for dial. Only four participants searched for secure or security: two in the free-choice group and one each in the Stack Overflow and official-documentation groups.

Participants' Opinions about Information Sources

We asked all participants, except those given free choice, whether they'd previously used their assigned resource. All 14 Stack Overflow participants had previously used Stack Overflow, and most (10 of 13) official-documentation participants had used official Android documentation before. However, only six of the 14 book-only participants had previously used books while programming.

Participants also rated the extent to which their assigned resource was easy to use, helpful, and correct. Results are shown in Figure 5. As might be expected, participants found free choice the easiest to use and books the hardest. In contrast, however, they were most likely to consider the books and official documentation to be correct.

We next asked participants about their assigned resource's effect on their performance. In every restricted condition, the large majority (official documentation: 92 percent, book: 93 percent, and Stack Overflow: 79 percent) said they would've performed

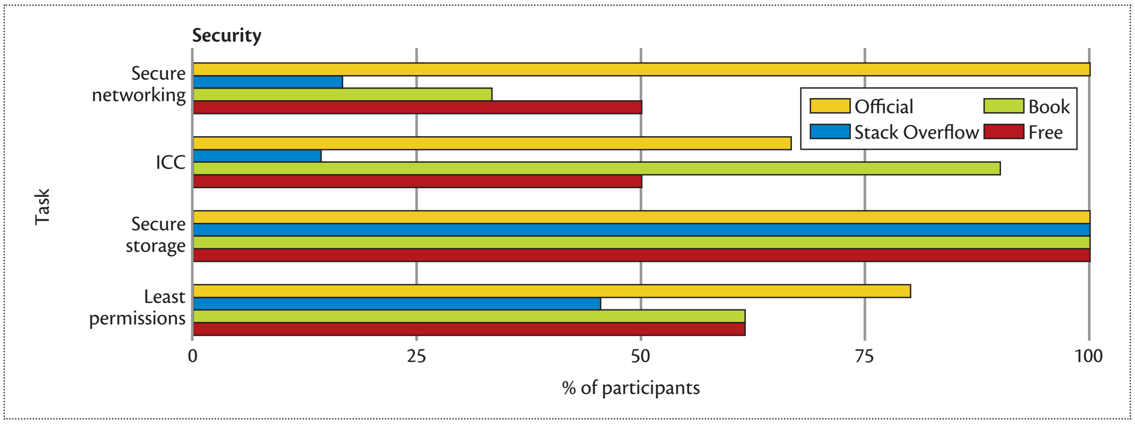

Figure 4. Resource type's effect on use of secure solutions, by task and condition. The resource used had the opposite effect on security than functionality: participants restricted to Stack Overflow were least likely to achieve secure solutions.

better if they'd been allowed to use different resources. In particular, participants restricted to the official Android documentation and books said they would've liked to have accessed Stack Overflow or search engines such as Google, so that they could directly search for their specific problems rather than reading background information. On the other hand, many Stack Overflow participants said they would've liked to have accessed the official documentation to read up on background information for their problems.

Stack Overflow seemed to help put the participants' tasks into (not necessarily correct) context. About an HTTPS problem he looked up on Stack Overflow, one participant commented that "many people seem to have the same problem." In contrast, while working on the ICC problem, the same participant said "I can't find this without official documentation. The problem is probably rare, that's why there is no solution on Stack Overflow." One participant in the books-only condition described books as "uncool" and relied heavily on AndroidStudio's autocomplete feature to make up for a lack of examples in the books; another participant mentioned the "danger that books could be outdated."

We observed that Stack Overflow (directly or accessed via a search engine) was often useful for handling errors; participants searched for the error message and, in most cases, found a functional (but not necessarily secure) answer, most often in the form of a ready-to-use code snippet. Participants were sometimes aware of this security problem (one said "I know this is a very bad idea," then used the snippet nonetheless; another said "but for a real app, you should *not* do this!," before using a code snippet) but were nevertheless happy to find easy-to-use functional code snippets. Such snippets were greatly missed in the other resources.

While working on the HTTPS connection task, a participant who was only allowed to use books complained that he was "not sure how to solve it—the book gives no example on how to implement this!"

Often, participants assigned to the official Android documentation or book conditions could correctly explain the secure concept of what they needed to do to solve a task, sometimes even writing pseudocode, but failed to write any functional code. They knew in theory what to do, but this knowledge didn't translate to writing actual code; in some cases, they didn't even know where to start writing the code in the class they were editing. In this context, one book-only participant stated, "I almost have it, but I don't know where to put the permission," while another complained that they "have to match signature, but don't know syntax."

However, we observed several participants restricted to Stack Overflow having difficulty understanding concepts. We watched several participants scroll through code snippets, one thinking aloud, "I do not understand this, and nobody explains it." For some participants in search of a conceptual explanation, being restricted to Stack Overflow was frustrating.

Examining Threads on Stack Overflow

To better contextualize our results, we examined in detail all the Stack Overflow pages (threads) visited by our Stack Overflow and free-choice participants during the tasks. All threads were independently coded by two researchers, who reached consensus on any conflicts. We classified each thread on five different attributes:

- *Task relevance.* We evaluated whether the topic of the thread was relevant to solving the study task. Irrelevant threads weren't examined further.
- *Usefulness.* Threads with no answers, or no answers that responded to the original question, were rated as not useful. Threads with answers that discussed the question and gave helpful comments, links to other

resources, or sample code were rated as useful.

- *Code snippets.* We examined all answers in each thread for ready-to-use code snippets, defined as syntactically correct code a developer could copy and paste into an app. Each code snippet was individually rated as secure or insecure relative to our study's programming tasks.
- *External links.* We noted whether answers contained links to GitHub, other code repositories, other Stack Overflow threads, or another external source. We then classified the linked content as either secure or insecure.
- *Security implications.* We determined whether any answer in the thread discussed the security implications of possible solutions. For example, if two solutions existed and one included an extra permission request, we checked whether any answer discussed a violation of the least-privilege principle.

Insecure Stack Overflow Answers Aren't Unusual

Overall, participants accessed 139 Stack Overflow threads, 41 of which we classified as on-topic. Table 2 summarizes the classification results for these 41 threads: 20 contained code snippets, and half of these contained only insecure snippets, such as instructions to use `NullHostnameVerifiers` and `NullTrustManagers`, which will accept all certificates regardless of validity. Among the 10 threads containing only insecure code snippets, only three described the security implications of using them. This creates the possibility that developers will simply copy and paste a "functional" solution that voids existing security measures, without realizing the consequences of their actions. More encouragingly, seven of the 10 threads with at least one secure code snippet contained only secure snippets.

We next investigated how threads with different properties compared in terms of popularity (measured by total upvotes for the thread). Details are shown in Table 3. Unsurprisingly, threads with code snippets were more popular than those without ($W = 319.5$, $p = 0.002$, $\alpha = 0.025$, Wilcoxon-Mann-Whitney test with Bonferroni-Holm correction [B-H]). Threads with secure snippets appeared more popular than those with insecure code snippets, but at this sample size, the difference wasn't statistically significant ($W = 73$, $p = 0.19$). On the other hand, threads that discussed security implications were slightly more popular than those that didn't ($W = 239.5$, $p = 0.03$, $\alpha = 0.05$ [B-H]).

Putting Our Results into Context

The combined results of our survey of Android market developers, our lab experiment, and our brief survey of Stack Overflow threads suggest several interesting conclusions:

- Real-world Android developers use Stack Overflow (and other Q&A communities) as a major resource for solving programming problems, including security- and privacy-relevant problems.
- The official Android API documentation doesn't provide the same degree of quickly understandable, directly applicable assistance that helps developers arrive at timely functional solutions.
- Participants given free choice of resources tended to visit both the official documentation and Stack Overflow, but their performance in both functional correctness and security was more similar to participants restricted to Stack Overflow.
- Because Stack Overflow contains many insecure answers, it's not necessarily surprising that Android developers who rely on this resource are less likely to create secure code.
- Without prompting, few developers (at least in a study environment) explicitly consider security when

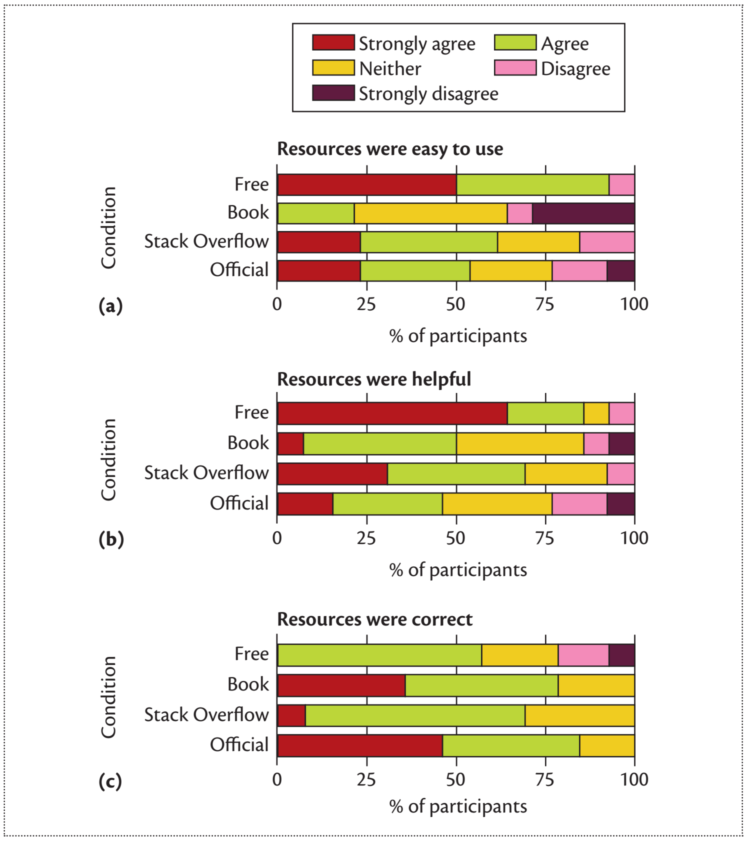

Figure 5. Participant ratings of the resources' (a) ease of use, (b) helpfulness, and (c) correctness. As expected, participants found having free choice the easiest to use and books the hardest. However, they more often considered the books and official Android documentation to be correct.

Table 2. Properties of the 41 on-topic Stack Overflow threads accessed by study participants.

Answers in the thread included	Count	%
Useful answers	35	85.4
Useless answers	6	14.6
Discussion of security implications	12	29.3
Working code examples	20	48.8
Only secure code examples	7	17.0
Only insecure code examples	10	24.4
Only insecure code examples but also discussion of security implications	3	7.3
Secure links	23	56.1
Insecure links	6	14.6
Links to GitHub	4	9.8
Links to other code repositories	1	2.4
Links to other Software Overflow threads	4	9.8
Only secure code examples and secure links	3	7.3

thinking about problems like network security or ICC. This confirms that security remains, at best, a secondary concern for many developers.

Our results confirm an important problem: official API documentation is secure but hard to use, while informal documentation such as Stack Overflow is more accessible but often leads to insecurity. Interestingly, books—the only paid resource—perform well both for security and functionality but are rarely used; in our study, only one free-choice participant used a book.

Given time constraints and economic pressures, we expect Android developers to continue using resources that help them quickly address their immediate problems. Therefore, it's critical to develop documentation and resources that combine the usefulness of forums like Stack Overflow with the security awareness of books or official API documents, and that promote security even when developers aren't directly thinking about it.

One approach might be to develop a separate programming-answers site in which experts address popular questions, perhaps initially drawn from other forums, in a security-sensitive manner. However, this seems insufficient for several reasons. For one, gaining market share for a new service could prove difficult when Stack Overflow is already serving an important developer need. In addition, creating a separate site presupposes that developers will recognize that they need a secure solution. Our results clearly show this isn't the case: many participants didn't think about security while solving the lab tasks, and most real-world developers in our survey said they don't use different resources for security-relevant problems than for other problems. In fact, Stack Exchange (Stack Overflow's parent company) has a separate information security site (security.stackexchange.com) that focuses mostly on security operations but also covers some secure programming aspects; however, a cursory examination suggests that it's significantly less popular than the more general-purpose Stack Overflow. Only two of our free-choice participants visited Stack Exchange during the study, and neither found helpful results.

Alternatively, Stack Overflow could add a mechanism for explicitly rating provided answers' security. Ideally, the security rating would factor heavily into search results and thread ordering, guiding users toward more secure solutions. This, of course, would require the community to effectively conduct a crowdsourced evaluation. Upvoting generally works well for evaluating threads, so there's hope that it could work for security as well. On the other hand, only a small pool of developers has the security expertise to effectively evaluate these threads, suggesting there'd be fewer ratings to work with and high motivation to game the system by upvoting insecure solutions. Another option might be to nudge question responders to consider discussing their provided solutions' security impact.

One more option is to rewrite official documents to be more usable. The most important objective seems to be to include more secure, functional code

Table 3. Popularity ratings (measured by three upvotes) for Stack Overflow threads containing code snippets.

	Popularity rating			
Thread	**Mean**	**Median**	**Standard deviation**	**Wilcoxon-Mann-Whitney test**
With code snippets	97.7	12.0	163.9	$W = 319.5$, $p = 0.002$,* $\alpha = 0.025$
Without code snippets	3.9	2.5	4.4	
With secure code snippets	204.3	145	209.3	$W = 73$, $p = 0.19$
With insecure code snippets	70.2	14.0	122.4	
With security implications	135.2	16.0	207.0	$W = 239.5$, $p = 0.03$,* $\alpha = 0.05$
Without security implications	17.4	3.0	37.0	

*Statistically significant at $p < 0.05$.

examples that developers can directly copy and paste. But simply adding examples isn't enough—the genius of Q&A sites is that, rather than trying to predict problems, the examples explicitly address problems that at least one real developer has encountered. Software engineering researchers have mined Stack Overflow posts to identify trouble spots in APIs, including cryptography APIs.[9–11] We advocate for writers of official documentation to adopt this approach, especially for security-sensitive topics. We propose an ongoing cycle of documentation and API improvement, with feedback from Q&A sites as one key step to identify areas needing further attention. Results from our free-choice participants suggest that they often visited the official documents before resorting to Stack Overflow for working examples; we hope that bringing more hot-topic examples into the official documents can ultimately reduce the naive copying-and-pasting of insecure code from other resources. ■

References

1. E. Chin et al., "Analyzing Inter-application Communication in Android," *Proc. 9th Int'l Conf. Mobile Systems, Applications, and Services* (MobiSys 11), 2011, pp. 239–252.
2. M. Egele et al., "An Empirical Study of Cryptographic Misuse in Adroid Applications," *Proc. ACM SIGSAC Conf. Computer and Communications Security* (CCS 13), 2013, pp. 73–84.
3. S. Fahl et al., "Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security," *Proc. ACM Conf. Computer and Communications Security* (CCS 12), 2012, pp. 50–61.
4. A.P. Felt et al., "Android Permissions Demystified," *Proc. 18th ACM Conf. Computer and Communications Security* (CCS 11), 2011, pp. 627–638.
5. M. Georgiev et al., "The Most Dangerous Code in the World: Validating SSL Certificates in Non-browser Software," *Proc. ACM Conf. Computer and Communications Security* (CCS 12), 2012, pp. 38–49.
6. B. Reaves et al., "Mo(bile) Money, Mo(bile) Problems: Analysis of Branchless Banking Applications in the Developing World," *Proc. 24th USENIX Conf. Security Symp.* (USENIX Sec 15), 2015, pp. 17–32.
7. S. Komatineni and D. MacLean, *Pro Android 4*, Apress, 2012.
8. N. Elenkov, *Android Security Internals*, No Starch Press, 2015.
9. S. Nadi et al., "Jumping through Hoops: Why Do Java Developers Struggle with Cryptography APIs?," *Proc. 38th Int'l Conf. Software Eng.* (ICSE 16), 2016, pp. 935–946.
10. W. Wang and M.W. Godfrey, "Detecting API Usage Obstacles: A Study of iOS and Android Developer Questions," *Proc. 10th Working Conf. Mining Software Repositories* (MSR 13), 2013, pp. 61–64.
11. W. Wang, H. Malik, and M.W. Godfrey, "Recommending Posts Concerning API Issues in Developer Q&A Sites," *Proc. 12th Working Conf. Mining Software Repositories* (MSR 15), 2015, pp. 224–234.

Yasemin Acar is a PhD candidate in computer science at the Center for IT-Security, Privacy and Accountability (CISPA), Saarland University. Her research interests include establishing practices for ecologically valid usable security and privacy research and supporting developers in making secure programming choices. Contact her at acar@cs.uni-saarland.de.

Michael Backes is a full professor of computer science, director of CISPA, and head of the Information Security and Cryptography group at Saarland University.

He's also a Max Planck Fellow at the Max Planck Institute for Software Systems. Backes' research interests include the design, analysis, and verification of protocols and systems; mechanisms for protecting end-user privacy; new attack vectors; and universal solutions in software and network security. Contact him at backes@mpi-sws.org.

Sascha Fahl is head of the Usable Security and Privacy Research group at CISPA, Saarland University. His research focuses on the intersection of computer security and privacy mechanisms with human factors, involving large-scale analyses of the Internet and software repositories and user studies with end-user and information security workers. Fahl received a PhD in computer science from Leibniz University Hannover. Contact him at fahl@cs.uni-saarland.de.

Doowon Kim is a PhD candidate in computer science at the University of Maryland. His research interests include computer security and privacy, and usable security. Kim received an MS in computer science from the University of Utah. Contact him at doowon@cs.umd.edu.

Michelle L. Mazurek is an assistant professor in the Department of Computer Science and the Institute for Advanced Computer Studies at the University of Maryland. Her research interests include understanding and designing tools and techniques to improve security- and privacy-relevant decision-making. Mazurek received a PhD in electrical and computer engineering from Carnegie Mellon University. Contact her at mmazurek@umd.edu.

Christian Stransky is a PhD candidate at CISPA, Saarland University. His research interests include usable security and privacy with a focus on developers. Stransky received an MSc in computer science from Leibniz University Hannover. Contact him at stransky@cs.uni-saarland.de.