Reaching the Masses: A New Subdiscipline of App Programmer Education

Charles Weir
Security Lancaster
Lancaster University, UK
+44-7876-027350
c.weir1@lancaster.ac.uk

Awais Rashid
Security Lancaster
Lancaster University, UK
+44-1524-510316
a.rashid@lancaster.ac.uk

James Noble
Victoria University
Wellington, NZ
+64-4-4635233
kjx@ecs.vuw.ac.nz

ABSTRACT

Programmers' lack of knowledge and interest in secure development threatens everyone who uses mobile apps. The rise of apps has engaged millions of independent app developers, who rarely encounter any but low level security techniques. But what if software security were presented as a game, or a story, or a discussion? What if learning app security techniques could be fun as well as empowering? Only by introducing the powerful motivating techniques developed for other disciplines can we hope to upskill independent app developers, and achieve the security that we'll need in 2025 to safeguard our identities and our data.

CCS Concepts

• **Social and professional topics~Software engineering education** • *Security and privacy~Software security engineering* • *Security and privacy~Human and societal aspects of security and privacy*

Keywords

app developer, app development, app programmer, app security, application security, continued learning, mobile app, programmer education, secure app, secure app development, security technique, software development, software security, whole system security

1. INTRODUCTION

Mobile apps are increasingly becoming the lynch pins of our lives. We use apps to communicate, apps to plan, apps to manage our finances, apps to do our shopping, and apps to remember all our security information.

Creating our apps are more than 2.9 million app developers, of whom only some 25% are professionals developing apps for companies [12]. In those apps, cloud-based connectivity and social networking functionality are making trust and security issues fundamentally important. So security expertise – and hence effective security practices – in those developing such apps is vital.

Yet there is considerable evidence that such expertise is lacking. Analysis of the top five payment apps by Bluebox, a security solution provider, found significant security failures in each [3]; analysis of a range of Android apps by Enck et al found privacy problems in most of them [6]. Both analyses highlighted that it was the choices that the app programmers had made that were causing the problems; given the same environment and cloud services they could have chosen problem-free implementations.

And indeed a recent IBM-driven survey of opinions about app security in American companies [9] revealed that more than 70% percent believed that the developer inexperience was a major threat to their business.

2. EXPLORING THE PROBLEM

To address this problem, the authors instigated open-ended interviews with a dozen experts in app security. Table 1 lists the participants along with the organization each worked with most; it shows an indication of the organization size and a subjective estimate of the organization's position in on a 'secure software capability maturity model'. Throughout this paper we've quoted from interviewees, giving their identifiers.

Table 1: Interviewees and their organizations

Identifier	Organisation type	Org. size	Est. CMM
P1, P12	Bespoke app developer	Solo	Low
P2, P7	Mobile phone manufacturer	Med.	High
P3, P11	Operating system supplier	Large	High
P4	Smart card specialists	Small	Med.
P5	Security-related software-as-service supplier	Med.	High
P6	Promoting industry	Gov't	Low
P8	Telecoms service provider	Large	Med.
P9	Bank	Large	Med.
P10	Secure app technology provider	Small	Med.

The conclusions were daunting. Most of the interviewees, naturally, were working in areas where there was considerable effort put into security. But they saw little interest or activity related to app security in other companies and areas.

"Very very few developers are actually interested in security… You can see that from the Apps World [exhibition] where there's no mention of security at all." (P1)

And where app programmers are learning about security, the experts saw them as typically concentrating on low-level, checklist based, approaches to getting app code secure, and ignoring the wider picture. Our interviewees were clear that in order to get

FSE'16, November 13–18, 2016, Seattle, WA, USA
ACM. 978-1-4503-4218-6/16/11...$15.00
http://dx.doi.org/10.1145/2950290.2983981

effective app privacy and security, programmers would need to have some awareness of the wider issues of security.

“Businesses need to take a realistic approach – it's a business decision on their part – 'I understand what my assets are and I understand what things I need to protect and I understand how much I am willing to pay for that – and understand how much risk I am going to take.' None of these things do you need to know what a buffer overflow is for!

But nevertheless that is what lots of people, including PCI as it happens, seem to think computer security is about – and it really shouldn't be.” (P6)

2.1 Difficulty of Learning Security

Developers in the large, security-aware, companies are already well catered for with on-the-job training in app security.

“The internal training, and tools and technologies for [software privacy] are good. Mostly through internal training, I guess, nowadays.” (P3)

Our experts indicated, however, that app developers in other contexts are not generally learning what they need to know. They have no colleagues to learn from; training is expensive and not felt necessary.

“I’m probably not unusual in terms of software people in that you don’t really take courses”. (P12).

Most security writing has a further problem, highlighted by Conradi and Dyba [4]: programmers resist learning from the output of process improvers, and particularly from formal written routines. Yet much of the existing security literature is of this kind.

Furthermore many undergraduate university courses do not cover app security well, so even those trained there are not being well catered for.

“So for the majority of people who are currently going through various computer science degrees, security doesn't really come into it at all, in any real context”. (P10)

2.2 Relationship to Other Learning

Let’s consider the typical approaches that app programmers do have for learning. As Enes [7] found, most professional learning is on-the-job. Our interviewees confirmed this.

“We use external consultancies but we haven't really done any formalized training” (P2)

So the main sources of information are those normally available to app programmers. These are:

- Web searches, typically leading to developer sites such as Stack Overflow.
- Popular guides such as the O’Reilly series
- Occasional shows and industry events
- Blogs on app development
- Operating system websites

Guide books for app programmers do exist, such as *Application Security for the Android Platform* [10] or *Learning iOS Security* [1]; however since few app programmers are interested in security, they’re not well motivated to buy them – and both books are restricted to exploring the security features of their respective platforms. Similarly operating system manufacturers’ websites and blogs on app development aren’t helpful unless programmers actively seek them out.

Unfortunately, as a learning resource, Stack Overflow and similar bulletin boards have a significant flaw: they are poor for gaining an overview to a topic, and actively discourage questions that don’t have focused answers. A detailed analysis of the topics on the Stack Overflow site [2] found little in the way of overview discussions.

Thus sites like Stack Overflow are valuable in helping programmers sort out problems they know they have, but don’t help programmers with problems they don’t know they may have; most security problems are likely to be of this second type.

3. WHAT DO PROGRAMMERS NEED TO LEARN?

From the interviews we have identified a set of key security techniques that app developers need to know. These fall naturally under five categories: analysis, communication, dialectic, feedback and upgrading. The following sections explore each in turn.

3.1 Analysis

Security analysis is thinking outside the simple scope of programming. In particular it is the need to make security-aware choices of programming environment, tools and components. It is the need to review the system from the point of view of an attacker, and to identify likely exploits. It is understanding and thinking through the motivations of possible attackers, in order to understand how best and most cheaply to deter them.

“I think the things that are the most challenging around security really are trying to understand the threat landscape and trying to understand how threats are realized.” (P2)

3.2 Communication

The primary communication the experts identified is the discussion with project stakeholders about security. Our experts were clear that there is no such thing as perfect security, and that the trade-offs between the various costs of security (such as development time, usability issues, tool costs and expertise costs) require discussion with stakeholders. There are particular skills in representing security decisions in terms that non-programmers will understand.

“It goes from there: how secure do you want it to be. You have to show that there’s a problem first I think, that’s how it’s phrased”. (P1)

A second place where communication is important in the context of security is the discussion with other teams about responsibilities and the impact of different exploits.

“[In a project with successful security] everybody who was close to the project, lived through the project life cycle to delivery, was very comfortable picking the phone up to anybody else and discussing any aspect, and everyone reported back quite openly what they were seeing, and when we came together”. (P8)

3.3 Dialectic

Dialectic is a word from Greek that means learning by interrogating. Many of the best and most often mentioned security techniques are dialectic and involve critiquing what the programmer has produced: penetration testing, code reviews, code analysis tools, and even automated testing tools. Obviously the choice of approach depends on the programmer’s situation; penetration testing can be expensive. But free code analysis tools ensure that every app programmer has access to at least one of these.

“We do code reviews as much as possible”. (P7).

"[Security] tends to get handed off, in most companies I've worked with, to a white-hat hacking team." (P8).

"And so [when] the tools do the code inspection review for you, for free, constantly, all the time, so you can't skip it, then yes, that's a huge win".(P3)

3.4 Feedback

Many interviewees stressed the importance of gathering information from the deployed apps to detect and evaluate security issues. This is much more difficult with mobile apps than servers, since apps don't have continuous connections and are on devices under the control of other people. But there are a range of tools available, and so programmers need to be aware of the possible need for feedback mechanisms.

"I think one of the problems with remote devices is that these devices are intended to be robust against all attackers if you lose your device. So the builds that we produce – we're trying to look at them and see what is happening on them, [but they are built] by design such that [you] can't get raw access to those devices" (P11)

3.5 Upgrading

Many apps are released with a "fire and forget" mentality. This does not work well with security issues, where the security landscape is changing continuously and the nature of threats changes too. So app developers need to consider mechanisms to ensure upgrading. There are practical issues: though the 'App Stores' support upgrading, this feature is often not actioned by users. And there are commercial issues: once a project has ended who will implement upgrades? So app developers need to consider ways to support and enforce upgrading.

"And the patches and updates are basically what modern security is about – mistakes will be made and when the mistakes are found – how do you get the updates out?" (P3)

4. REACHING THE MASSES

Thus there is a vast population of isolated developers who are not being reached by the existing resources and information about app security: both developers on their own, and those developing apps within organizations that don't currently see security as an issue.

We've identified five key points we need to teach to these isolated app developers: analysis, communication, dialectic, feedback and upgrading – each in the context of app development security.

And we've uncovered three important roadblocks hindering the learning of app security: first, programmers don't learn well from process-improvement styles of literature; second, the main programmer sources of knowledge ('Googling Stack Exchange') won't give programmers the information they most need to know; and finally, most app developers don't appreciate that they may need to learn about the subject.

To improve the situation we need to reach out to a group of individuals, without having direct access to them. It's a different problem from teaching other aspects of software development, such as internationalization or DevOps, since app developers won't necessarily access good resources or training even if these are available. We need a new paradigm; we need a new way to reach these people.

4.1 A Different Approach

Different programmers learn in different ways and are interested in different things, so we believe a single form of intervention, however effective, is unlikely to reach all of our target audience. Also, since we are in effect teaching new attitudes, few of the traditional mechanisms such as books are likely to work.

We propose instead 'engaging' interventions likely to appeal to programmers for their own sake. We anticipate that these will be publicized via the web: expert blogs, and security OS websites.

The following sections explore some possibilities for these interventions.

4.2 Games that Teach

One popular approach is games. A great deal of work has been done on gamification, with books such as Kapp's [8] explaining the techniques involved. Tillman et al's game Code Hunt [11] teaches vast numbers of programmers through an online game. Code Hunt's approach is to provide a unit test that the programmer's code must pass; this certainly demonstrates the dialectic aspect, but will not be very good for teaching the other security techniques. Other researchers, including the authors, have had success with group games to teach aspects of software security, such as Denning et al's Control-Alt-Hack game [5]. These work very well in a classroom or conference context, but do not naturally extend to reach to an online audience.

Picture Angry Birds meeting Stack Overflow! Create an online game where players implement security aspects to defend against attacks? Perhaps crowd source both attacks and defenses, where each player gets to both take the role of attacker on other players' code, and defender on their own? It's an enchanting possibility; even if it risks taking too much time to engage the typical target solo programmer.

4.3 Story Telling

A different approach is story-telling. The British radio soap opera, The Archers, has been running for 65 years, and has over 5 million regular listeners; its main purpose, at which it is highly successful, is to teach farming knowledge to a community that is unreachable by any other form of education. Taking a similar approach here would suggest a podcast (and blog) narrating a plot that would cover and teach each of these aspects.

More ambitious would be a storyline in an appropriate existing series ('Mr. Robot', and the UK's 'IT Crowd' come to mind), to be created if the opportunity arose.

4.4 Adapting Business as Usual Approaches

More conventional is to tailor direct teaching and group learning approaches to the distributed nature of the target audience. This suggests implementing a massively open online course (MOOC) on app security using audio, written text, and video along with interactive discussion groups. Organizations such as edX and Futurelearn provide frameworks to make this straightforward.

Another possibility is a short video along the lines of – or indeed actually – a TED talk by a suitable expert;

Both possibilities leverage the 'professional skills gaining' motivation present in programmers, which suggests promoting them via professional organizations too.

5. RESEARCH AGENDA

Whilst each of these interventions has promise, we don't know which are likely to be effective, nor which techniques and variants of each will have the most impact. This leads us to a set of research questions, as follows.

RQ1 How best to design and implement the interventions to convey the security techniques discussed in Section 3? This is a complex problem, involving amongst other aspects elements of design, gamification, and measurement of impact.

RQ2 Which interventions – and dissemination techniques – are most effective at conveying each technique to the largest population of programmers? Implementing all the interventions at scale will be costly; we'll need to evaluate which ones offer the most value.

RQ3 Which interventions provoke a wider interest in the programmers reached? To achieve a lasting effect we do not just need to engage programmers initially, but need also to encourage further interest in the subject so they learn also from existing sources of security education.

This approach is very different from others in the field of programmer education, making this an entirely new subdiscipline. The research will require a multi-disciplinary team, with varied skills including at least the following:

Programming: To implement code based interventions such as games.

Psychology: To achieve the 'attractiveness' of the content; to structure measurement of the results; to use psychological techniques to 'nudge' programmers towards more effective security practices.

Creative writing: For the storyline.

Narration: For an engaging verbal version of the storyline.

Marketing: To establish and develop the channels to bring the content to the target audience.

6. EVALUATING TECHNIQUES

The research will require objective measurement. In particular we can identify four aspects to measure:

Success using the interventions with a sample group of students or similar, and evaluating their learning based on the intervention (RQ1).

Reach the number of downloads, accesses, or to the resource (RQ2)

Engagement the number of accesses of later parts of the resource. (RQ2)

Coverage attending exhibitions such as Apps World frequented by the target solo programmer audience and asking via a simple questionnaire of delegates which if any of the interventions they have encountered and their impact (RQ2 RQ3).

Ideally we'll want to extend our research to measure outcomes as well as these outputs. Whilst we can argue that the combination of 'success' with 'engagement' and 'coverage' implies a positive impact, better still would be evidence of an improvement in the code produced by programmers in the target group.

To achieve that, we might collect app identifiers, where possible, from participants for a 'before and after' anonymous evaluation of their released apps' security along the lines of that by Enck et al. [6]. Other possibilities would include extending the questionnaires in the 'coverage' evaluation to estimate interviewees' awareness of app security or access to other learning resources, and correlating that with exposure to the interventions.

7. CONCLUSION

In this paper we explored the need to improve the security skills of isolated app developers. We explored the resources currently available and concluded that they were insufficient for the job of both learning and motivation.

To address these issues we propose a research agenda for a new subdiscipline: research into ways to motivate and teach app security for the isolated developer. We propose a set of research questions, a multi-disciplinary team, several ways to reach the developers in question, three approaches to teaching, and four aspects we can measure to evaluate the success of each approach.

It's important that our future app security is not left to chance; we believe this new subdiscipline will make a substantial contribution.

8. REFERENCES

[1] Banks, A. and Edge, C.S. *Learning iOS Security*. Packt Publishing, Birmingham, UK, 2015.

[2] Barua, A., Thomas, S.W., and Hassan, A.E. *What are developers talking about? An analysis of topics and trends in Stack Overflow*. 2012.

[3] Bluebox Security. *'Tis the Season to Risk Mobile App Payments - An Evaluation of Top Payment Apps*. 2015.

[4] Conradi, R. and Dybå, T. An empirical study on the utility of formal routines to transfer knowledge and experience. *ACM SIGSOFT Software Engineering Notes 26*, 5 (2001), 268–276.

[5] Denning, T., Lerner, A., Shostack, A., and Kohno, T. Control-Alt-Hack: the design and evaluation of a card game for computer security awareness and education. *CCS '13: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, (2013), 915–928.

[6] Enck, W., Octeau, D., McDaniel, P., and Chaudhuri, S. A Study of Android Application Security. *Proceedings of the 20th USENIX conference on Security*, (2011).

[7] Enes, P. and Conradi, R. Acquiring and Sharing Expert Knowledge. 2005. http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2005/aanes-fordyp05.pdf.

[8] Kapp, K.M. *The gamification of learning and instruction: game-based methods and strategies for training and education*. John Wiley & Sons, San Francisco, 2012.

[9] Ponemon Institute. *The State of Mobile Application Insecurity*. 2015.

[10] Six, J. *Application Security for the Android Platform*. O'Reilly, Sebastapol, CA, 2011.

[11] Tillmann, N., de Halleux, J., Xie, T., and Bishop, J. Code Hunt: Gamifying teaching and learning of computer science at scale. *Proceedings of the first ACM conference on Learning@ scale conference*, ACM (2014), 221–222.

[12] Vision Mobile. *Developer Economics Q3 2014: State of the Developer Nation*. London, 2014.