Editor: Cecilia Mascolo • *cecilia.mascolo@cl.cam.ac.uk*

Usefulness Is Not Trustworthiness

Landon P. Cox • *Duke University*

Mobile phones have placed communication, sensing, and computation at the center of nearly all human activity. A great deal of the software written for this new platform is extraordinarily fun and useful. However, the question of how much of it we can trust remains unanswered.

My undergraduate operating systems class spends a lot of time thinking about computer systems that don't work properly. When discussing ways to ensure that a system remains predictable in the face of bugs, attacks, and failures, the students inevitably reach a point where they have to trust that some thing or set of things behaves as advertised. If a system is well-designed, then this set of trusted components will be small, and interactions between trusted and untrusted components will be constrained by a set of narrow, well-understood interfaces. Of course, even when a designer carefully abides by these principles, everything can and will go wrong if his or her trust is misplaced.

Unexpected Behavior

One system that I discuss in my class to underscore the role that trust plays in system design is the Secure Sockets Layer (SSL), which provides the basis for secure communication on the Web. SSL's roots of trust are the certificate authorities (CAs) that vouch for the authenticity of a website's public key. When I cover SSL, my students and I work through questions such as how a Web browser knows if a CA has vouched for another website's key, and why we should believe what a CA says in the first place.

This past fall, with the SSL lecture coming up, I went online to research CAs' trustworthiness so that I could update my slides. Using my office PC, I searched for the name of an overseas CA who had been involved in several high-profile data-confidentiality incidents. Through my searches, I read several reports about the CA from both well-known news organizations and technology-policy blogs. I updated my slides with this new information, and my SSL lecture was well received.

The next day, something unexpected happened: I received an unsolicited text message about job opportunities at the company I had been researching on my desktop the day before. I don't consider myself a paranoid person, but this was too strange and specific to be a mere coincidence. What made this unwanted message even stranger was that it was delivered to my private mobile-phone number. I hide this number behind a Google Voice number that I share with others. Only a handful of family members know my private number, and, as far as I know, no public record links it to me. Nonetheless, someone or something was able to link my private mobile-phone number to my Web searches from the day before. How could this have happened? Who would have had access to both my private number and my Web searches?

Google certainly knows both because I route phone calls to my mobile phone through Google Voice, and Google obviously knows everything I search for. However, it seems highly unlikely that Google was responsible given that, as far as I know, it doesn't sell SMS-based ads based on recent Web searches. My best guess is that my surprise message was a result of analytics performed on the websites I visited and data collected by apps installed on my phone. A Web analytics company would know what search

 1089-7801/11/$26.00 © 2011 IEEE Published by the IEEE Computer Society

term led me to a particular website via the "referer[sic]" field in an HTTP request. And, as my collaborators at Penn State, Intel, and Duke documented in our work on TaintDroid,[1] many mobile apps are linked with third-party libraries that forward information about a user to remote advertising and analytics servers. Any of the apps I use on my device could easily have collected my mobile-phone number.

Searching for Bases of Trust

Mobile developers might feel pressure to collect and sell their users' personal information for several reasons. First, mobile apps are much more similar to Web applications than desktop applications, in that they represent a form of inexpensive, light entertainment for many people. This puts pressure on developers to price their apps low enough to remain an "impulse buy" for users browsing their device's app store. The standard 30/70 split imposed on developers by app store maintainers undercuts sales revenue even more. Given these market realities, no one should be surprised that developers have turned to advertising and harvesting users' information to earn a living. Yet our apps still seem to do things we don't expect.

Google instructs users to "only install applications you trust," but it's unclear on what basis a user can decide whether an app is trustworthy. If an app is attractive and useful, and requires access to a device's location to provide its core functionality (for example, searching for nearby restaurants), then, for most users, it makes perfect sense to give the app access to a device's location. Unfortunately, an app's attractiveness and utility says very little about its trustworthiness.

Security researchers have described "Trojan horse" attacks on computer systems since at least 1972, when the term was used in an appendix of James Anderson's report for the US Air Force entitled *Computer Security Technology Planning Study*.[2] In his report, Anderson describes a Trojan horse as a piece of software that is "so useful" that a system's operator will use it even if he or she isn't completely sure how the software was developed or what it's doing. Anderson insightfully observed that Trojan horse attacks exploit vulnerabilities in how software is admitted into a system, rather than vulnerabilities in the mechanisms used to control the software's actions once it starts to execute. A Trojan horse attack relies on us to misplace our trust in something just because it's useful or attractive.

Are mobile apps Trojan horses? Not all apps misuse their users' data, and even for the apps that share data in unexpected ways, calling them Trojan horses seems harsh. We think of a Trojan horse as something that's truly malicious and actively tries to damage someone or something. The vast majority of mobile apps aren't interested in causing damage. Yet when an app isn't clear about how it will use the personal data to which it has access, then its behavior will closely match the textbook definition of a Trojan horse.

Short of denying ourselves all the ways that apps make our mobile devices useful and fun, how can we ensure that our data is handled properly? Three methods come to mind: economic, legal, or technical frameworks. Given that the economics of mobile app stores has created strong incentives for developers to harvest information instead of charging consumers more directly for the cost of creating and maintaining services, an economic framework capable of securing users' data is unlikely to emerge. Rumblings have arisen that the government might try to enact laws to ensure that personal information is handled properly, but I'm not a legal expert, and the legislative process is hard to predict.

However, we can easily imagine a technical approach to ensuring that users have a sound basis for trusting an app. Simply giving users more information about what an app will do when they decide to install it and providing good tools for monitoring what it does after installation would improve the current situation a great deal. Quite a bit of early, good work is occurring on both approaches,[3] and several opportunities exist for start-ups and researchers in human–computer interaction, operating systems, and networking to help create the right framework for ensuring that sensitive data is handled properly.

References

1. W. Enck et al., "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," *Proc. 9th Usenix Conf. Operating Systems Design and Implementation* (OSDI 10), Usenix Assoc., 2010, pp. 1–6.
2. J.P. Anderson, *Computer Security Technology Planning Study Volume II*, tech. report ESD-TR-73-51, vol. 2, Electronic Systems Division, Air Force Systems Command, Oct. 1972.
3. M. Egele et al., "PiOS: Detecting Privacy Leaks in iOS Applications," *Proc. 18th Ann. Network and Distributed System Security Symp.* (NDSS 11), 2011; www.mylookout.com.

Landon P. Cox is an assistant professor at Duke University. His research interests include security and privacy in operating systems and mobile computing. Cox has a PhD in computer science and engineering from the University of Michigan, Ann Arbor. He's a member of Usenix and the ACM. Contact him at lpcox@cs.duke.edu.