Computer Security and Privacy where Human Factors meet Engineering

Franziska Roesner

Department of Computer Science & Engineering
University of Washington

franzi@cs.washington.edu

ABSTRACT

As our world becomes more computerized and interconnected, computer security and privacy will continue to increase in importance. In this paper, I overview several computer security and privacy challenges faced by end users of the technologies we build, and how we can design and build technologies that better match user expectations with respect to security and privacy. I close with open challenges in computer security and privacy where engineering meets people.

INTRODUCTION

Over the past several decades, new technologies have brought benefits to almost all aspects of our lives, transforming how we work, how we communicate, how we interact with each other, and how we live our lives. Unfortunately, however, new technologies also bring new and serious security and privacy risks. For example, smartphone malware is on the rise, often tricking unsuspecting users by appearing as compromised versions of familiar, legitimate applications (Ballano 2011); by recent reports, over 350,000 variants of Android malware have been identified (Sophos 2014). These malicious applications incur indirect and direct costs by stealing financial and other information or by secretly sending costly premium SMS messages. On the web, privacy is also a growing concern as advertisers and others secretly track user browsing behaviors, resulting in ongoing efforts at "Do Not Track" technology and legislation (Do-Not-Track Online Act of 2013). Such concerns cast a shadow on modern and emerging computing platforms that otherwise provide great benefits.

Addressing these and other computer security and privacy risks will continue to increase in importance as our world becomes more computerized and interconnected. To that end, it is vital that we approach the engineering design process with a "security mindset" – attempting to

anticipate and mitigate the unexpected and potentially dangerous ways our technologies might be (mis)used. Computer security and privacy research aims to systematize this process and tackle these challenges. Its efforts can be characterized along several (often overlapping) axes: (1) theoretical cryptography, (2) analyzing or attempting to attack deployed technologies, (3) measuring deployed technical ecosystems (e.g., the web), (4) studying human factors, and (5) designing and building new technologies. Some of these axes themselves represent entire academic subdisciplines – e.g., cryptography or usable security – but all work together and inform each other to improve the security and privacy properties of existing and emerging technologies. This paper focuses specifically on computer security and privacy at the intersection of human factors and the engineering of new computer systems.

Indeed, many computer security and privacy challenges arise when end users' expectations don't match the actual security and privacy properties of the technologies they use – for example, when installed applications secretly send premium SMS messages or leak a user's location to advertisers, or when invisible trackers observe a user's behavior on the web. There are two general approaches we can take to try to mitigate these discrepancies. On the one hand, we can try to help users change their mental models about the technologies they use to be more accurate, e.g., to help users think twice before installing suspicious-looking applications, by educating them about the risks and/or by carefully design the user interfaces of app stores. Recent work by Bravo-Lillo et al. (2013) on designing security-decision UIs to make them harder for users to ignore is a nice example of this approach. On the other hand, however, we can try to *(re)design technologies themselves* so that they better match the security and privacy properties that users intuitively expect – in other words, "maintaining agreement between a system's security state and the user's mental model" (Yee 2004). Though both approaches are valuable and complementary, this paper explores the second approach: designing security and privacy properties in computer systems in a way that does not simply take human factors into account but that actively removes from users the burden of explicitly

managing their security and/or privacy at all. To illustrate the power of this approach, I will first overview a case study in user-driven access control and then highlight other recent examples.

CASE STUDY: USER-DRIVEN ACCESS CONTROL

As an example, consider smartphones (such as iOS, Android, or Windows Phone), other modern operating systems (such as recent versions of Windows and Mac OS X), and browsers. These platforms allow users to install arbitrary applications, and they limit the capabilities of those applications in an attempt to protect users from potentially malicious or buggy applications. Thus, by default, applications cannot access sensitive resources or devices like the file system, the microphone, the camera, or GPS. However, in order to carry out their intended functionality, applications do need access to these resources. Thus, an open question in modern computing platforms in recent years has been: how should untrusted applications be granted permissions to access sensitive resources? Today's platforms typically handle this by explicitly asking the user to make that decision. For example, iOS prompts users to ask whether an application may access a sensitive feature such as location, and Android[1] asks users to agree to an install-time manifest of permissions requested by an application (Figure 1).

Unfortunately, these permission-granting approaches place too much burden on users. Install-time manifests are often ignored or not understood by users (Felt, Ha, et al. 2012), and permission prompts are disruptive to the user's experience, teaching users to ignore and click through them (Motiee et al. 2010). These issues lead users to accidentally grant applications too many permissions, thereby failing to protect users from applications that use these permissions in malicious or questionable ways (e.g., secretly sending SMS messages or leaking location information). Indeed, in addition to outright malware (Ballano 2011), studies have shown that

[1] As of Android M, Android will use runtime prompts similar to iOS instead of its traditional install-time manifest.

even legitimate smartphone applications commonly leak or misuse private user data, such as by sending it to advertisers (e.g., Enck et al. 2010).

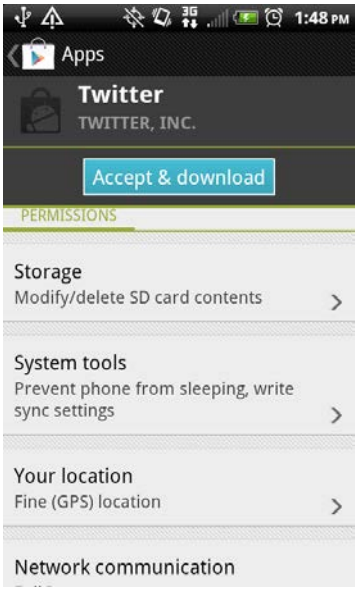

Figure 1: *Existing permission-granting mechanisms that require the user to make explicit decisions: runtime prompts (as in iOS, above) and install-time manifests (as in Android, right).*

One approach to reducing the scope of an application's illegitimate permissions is to attempt to better communicate application risks to users (e.g., Kelley et al. 2013) or to redesign permission prompts (e.g., Bravo-Lillo et al. 2013). However, we found in a user survey (Roesner, Kohno, Moshchuk, et al. 2012) that people have existing expectations about how applications use permissions – many people believe, for example, that an application cannot (or at least will not) access a sensitive resource like the camera unless it is related to the user's activities within the application. In reality, however, after being granted the permission to access the camera (or another sensitive resource) once, Android and iOS applications can continue to access the camera in the background without the user's knowledge.

This finding speaks for an alternate approach: modifying the system to better match user expectations about permission granting. To that end, we developed *user-driven access control* (Roesner, Kohno, Moshchuk, et al. 2012) as a new model for granting permissions in modern operating systems. Rather than asking the user to make explicit permission decisions, user-

driven access control grants permissions automatically based on existing user actions within applications. The underlying insight is that users *already implicitly indicate the intent to grant a permission though the way that they naturally interact with an application.* For example, a user who clicks on the "video call" button in a video chat application implicitly indicates the intent to allow the application to access the camera and microphone until the call is terminated. If the operating system could interpret the user's permission-granting intent based on these actions, it would not need to additionally prompt the user to make an explicit decision about permissions, and it could limit the application's access to a time intended by the user. This is challenging, however, because the operating system cannot by default interpret the user's action within the custom user interfaces of arbitrary applications.

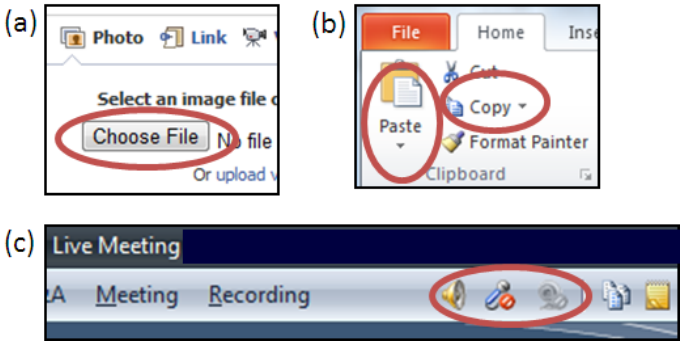

Figure 2: *Access control gadgets (ACGs) are special system-controlled user interface elements that allow the operating system to capture a user's implicit intent to grant a permission (e.g., to allow an application to access the camera).*

To allow the operating system to interpret such a permission-granting intent, we thus developed access control gadgets (ACGs) – special, system-controlled user interface elements that grant permissions to the embedding application. For example, in the video chat application, the application's "video call" button is replaced by a system-controlled ACG. Figure 2 shows additional examples of permission-related user interface elements that can be easily replaced by ACGs to enable user-driven access control. Though the general principle of user-driven access control has been introduced before (discussed below), ACGs make it practical and

generalizable to multiple sensitive resources and permissions, including location, the clipboard, files, the camera, the microphone, etc.

User-driven access control is powerful because it improves a user's security and privacy experience by changing the system to better match their expectations, rather than the other way around. That is, the user's experience interacting with his or her applications is unchanged, but the underlying permissions granted to applications match his or her expectations.

OTHER EXAMPLES

User-driven access control follows philosophically from Yee (2004) and a number of other earlier works. For example, CapDesk (Miller 2006) and Polaris (Stiegler et al. 2006) were experimental desktop computer systems that applied a similar approach to file system access – both give applications minimal privileges, but allow users to grant applications permission to access individual files via a "powerbox" user interface (essentially a secure file picking dialog). In a similar vein, Shirley and Evans (2008) proposed a system (prototyped for file resources) that attempts to infer a user's access control intent from the history of user behavior; BLADE (Lu et al. 2010) attempts to infer the authenticity of browser-based file downloads using similar techniques. Related ideas have recently appeared in mainstream commercial systems, including Mac OS X (Apple 2011) and Windows 8 (Microsoft n.d.), whose file picking designs also share the underlying user-driven access control philosophy. (Note that automatically managing permissions based on a user's interactions with applications may not always be the most appropriate solution – Felt, Egelman, et al. (2012) recommended combining a user-driven access control approach for some permissions (e.g., file access) with other approaches (e.g., prompts or post-facto auditing) that work more naturally for other permissions or contexts.)

Stepping back, the approach of designing systems to better and more automatically meet users' security and privacy expectations is much more general than the challenges surrounding application permissions discussed so far. For instance, consider the challenge of

securing communications between two parties, where available tools like PGP have long faced usability challenges (Whitten and Tygar 1999). Example efforts in this space that remove the burden from users while providing stronger security and privacy properties include Vanish (Geambasu et al. 2009), which supports messages that automatically “disappear” after some time; ShadowCrypt (He et al. 2014), which replaces existing user input elements on websites with ones that transparently encrypt and later decrypt user input; and Gyrus (Jang et al. 2014), which ensures that only content a user has intended to enter into a user input element is what is actually sent over the network ("what you see is what you send"). Commercially, communication platforms like email and chat are increasingly moving towards providing transparent end-to-end encryption (e.g., Somogyi 2014), though more work remains to be done. For example, the security of journalists’ communications with sensitive sources has come under question in recent years and requires a technical effort that provides low-friction security and privacy properties to these communications (McGregor et al. 2015).

Another security challenge faced by many systems is that of user authentication, which is typically handled with passwords or similar approaches, all of which have known usability and/or security issues (Bonneau et al. 2012). One approach to securing a user’s accounts that is seeing commercial uptake is two-factor authentication, in which a user must provide a second factor in addition to a password (e.g., a code provided by an app on the user’s phone). Two-factor authentication provides improved security but decreased usability; efforts like PhoneAuth (Czeskis et al. 2012) aim to balance these factors by using the user’s phone as a second factor only opportunistically when it happens to be available.

As a final example, another important way in which user expectations about security and privacy don’t match the reality of today’s systems is with respect to privacy on the web. As people browse the web, their behaviors are invisibly tracked by third-party advertisers, website analytics engines, and social media sites. In our prior work (Roesner, Kohno, and Wetherall 2012) we discovered that social media trackers, such as Facebook’s “Like” or Twitter’s “tweet”

button, represent a significant fraction of trackers included on popular websites. To mitigate the associated privacy concerns, we applied a user-driven access control design philosophy to develop ShareMeNot, a new defense for social media trackers that allows tracking *only when* the user clicks the associated social media button. ShareMeNot's techniques were integrated into the Electronic Frontier Foundation's Privacy Badger tool, which automatically detects and selectively blocks these and other types of trackers without requiring explicit user input.

Finally, of course, there are additional examples not covered here in which systems are intentionally designed to better and more automatically match users' security and privacy expectations – and this approach is often well complemented by work that attempts to better communicate with or educate users, e.g., by changing the designs of existing user interfaces.

CHALLENGES FOR THE FUTURE

As new technologies become widely adopted, they not only improve and transform our lives but also bring with them new and serious security and privacy concerns. Balancing the desired functionality provided by our technologies with security, privacy, and usability will remain challenging. This paper has provided examples across several contexts where we can come closer to achieving this balance by removing the burden from the user and designing computer systems that better and more automatically meet people's expectations about security and privacy. However, more work remains to be done in all of these and other domains, particularly in areas of emerging technologies. For example, emerging augmented reality technologies like Google Glass and Microsoft HoloLens will raise new security and privacy challenges, as applications and sensors become more ubiquitous and integrated into people's lives. By understanding and anticipating these challenges early enough, and with the right insights and design philosophies – such as by designing systems that better and more automatically meet people's expectations – we can improve the security, privacy, and usability of emerging technologies before they become widespread.

REFERENCES

Apple. (Nov. 2011). App sandbox and the Mac app store. https://developer.apple.com/videos/wwdc/2011/.

Ballano, M. (Feb. 2011). Android Threats Getting Steamy. Symantec Official Blog. http://www.symantec.com/connect/blogs/android-threats-getting-steamy.

Bonneau, J., Herley, C., van Oorschot, P. C., & Stajano, F. (2012). The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *Proceedings of the IEEE Symposium on Security and Privacy.*

Bravo-Lillo, C., Cranor, L. F., Downs, J., Komanduri, S., Reeder, R. W., Schechter, S., & Sleeper, M. (2013). Your Attention Please: Designing security-decision UIs to make genuine risks harder to ignore. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS).*

Czeskis, A., Dietz, M., Kohno, T., Wallach, D., & Balfanz, D. (2012). Strengthening User Authentication through Opportunistic Cryptographic Identity Assertions. In *Proceedings of the 19th ACM Conference on Computer and Communications Security.*

Do-Not-Track Online Act of 2013, S.418, 113th Congress. (2013). Retrieved from https://www.congress.gov/bill/113th-congress/senate-bill/418.

Enck, W., Gilbert, P., Chun, B., Cox, L. P., Jung, J., McDaniel, P., & Sheth, A. N. (2010). TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *Proceedings of the USENIX Conference on Operating System Design and Implementation.*

Electronic Frontier Foundation. Privacy Badger. https://www.eff.org/privacybadger.

Felt, A. P., Egelman, S., Finifter, M., Akhawe, D., & Wagner, D. (2012). How to Ask For Permission. In *Proceedings of the Workshop on Hot Topics in Security (HotSec).*

Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., & Wagner, D. (2012) Android Permissions: User Attention, Comprehension, and Behavior. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS).*

Geambasu, R., Kohno, T., Levy, A., & Levy, H. M. (2009). Vanish: Increasing Data Privacy with Self-Destructing Data. In *Proceedings of the 18th USENIX Security Symposium*.

He, W., Akhawe, D., Jain, S., Shi, E., & Song, D. (2014). ShadowCrypt: Encrypted Web Applications for Everyone. In *Proceedings of the ACM Conference on Computer and Communications Security.*

Jang, Y., Chung, S. P., Payne, B. D., Lee, W. (2014). Gyrus: A Framework for User-Intent Monitoring of Text-Based Networked Applications. In *Proceedings of the Network and Distributed System Security Symposium (NDSS).*

Kelley, P. G., Cranor, L. F., & and Sadeh, N. (2013). Privacy as part of the app decision-making process. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI).*

Lu, L., Yesneswaran, V., Porras, P., & Lee, W. (2010). BLADE: An Attack-Agnostic Approach for Preventing Drive-By Malware Infections. In *Proceedings of the ACM Conference on Computer and Communications Security.*

McGregor, S. E., Charters, P., Holliday, T., & Roesner, F. (2015). Investigating the Computer Security Practices and Needs of Journalists. In *Proceedings of the 24th USENIX Security Symposium.*

Microsoft. (N.d.). Accessing files with file pickers. http://msdn.microsoft.com/en-us/library/windows/apps/hh465174.aspx.

Miller, M. S. (2006). Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control. PhD thesis, Johns Hopkins University, Baltimore, MD, USA.

Motiee, S., Hawkey, K., & Beznosov, K. (2010). Do Windows Users Follow the Principle of Least Privilege?: Investigating User Account Control Practices. In *Proceedings of the Symposium on Usable Privacy & Security (SOUPS).*

Roesner, F., Kohno, T., Moshchuk, A., Parno, B., Wang, H. J., & Cowan, C. (2012). User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems. In *Proceedings of the IEEE Symposium on Security and Privacy.*

Roesner, F., Kohno, T., & Wetherall, D. (2012). Detecting and Defending Against Third-Party Tracking on the Web. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI).*

Shirley, J., & Evan, D. (2008). The User is Not the Enemy: Fighting Malware by Tracking User Intentions. In *New Security Paradigms Workshop.*

Somogyi, S. (Dec. 2014). An Update to End-To-End. Google Online Security Blog. http://googleonlinesecurity.blogspot.com/2014/12/an-update-to-end-to-end.html.

Sophos. (Nov. 2014). Our top 10 predictions for security threats in 2015 and beyond. http://www.sophos.com/en- us/threat-center/security-threat-report.aspx.

Stiegler, M., Karp, A. H., Yee, K.-P., Close, T., & Miller, M. S. (Sept. 2006). Polaris: Virus-Safe Computing for Windows XP. *Communications of the ACM* 49, 83–88.

Unger, N., Dechand, S., Bonneau, J., Fahl, S., Perl, H., Goldberg, I., & Smith, M. (2015). SoK: Security Messaging. In *Proceedings of the IEEE Symposium on Security and Privacy.*

Whitten, A. & Tygar, J. D. (1999). Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *Proceedings of the USENIX Security Symposium.*

Yee, K.-P. Aligning Security and Usability. (Sept. 2004). *IEEE Security and Privacy* 2(5), 48–55.