Privacy Requirements in an Age of Increased Sharing

Travis Breaux

IN 2006, a close industry colleague of mine speculated that “privacy is dead” amidst the difficulty he encountered when convincing management about the importance of privacy in software. What my colleague and I could not predict was the coming transformations due to the confluence of scalable cloud computing, faster wireless networks, smaller mobile platforms, new app markets, and big data. Today, it only takes a handful of developers and six weeks to turn a barebones idea into a deployed application with tens of thousands of users and minimal startup costs.

Third-party services help lower the barrier to entry by providing “free” or low-cost access to software project hosting, user authentication, data storage and analytics, and even advertising revenue. Applications are increasingly social, in that they allow family, friends, and complete strangers to organize themselves around mutually shared interests. Some applications further address ephemeral requirements that arise in the moment of an individual’s needs: finding a nearby restaurant, hiring a rideshare or taxi, or meeting friends after work. More sensitive activities, such as communicating with a doctor about a health issue, photographing a bank check for remote deposit, or monitoring a home security system, are more easily integrated into lightweight applications built on open frameworks and platforms. Across these applications, however, the boundary between what constitutes sensitive and non-sensitive personal information is blurring and becoming redefined by software.

Today, privacy is thriving. Many companies have chief privacy officers, and several large companies that primarily develop software, including Google, Microsoft and Facebook, are recruiting to fill positions for “privacy engineers,” people skilled in both the principles of privacy and the technical know-how needed to construct reliable software. Privacy engineers understand a range of privacy requirements and can discuss design trade-offs between privacy and business goals. Furthermore, they can conduct routine development activities, including programming and code inspections to verify that software conforms to privacy requirements. To help train these engineers, universities are developing software engineering courses and professional degrees in privacy. The impetus for this shift in software development culture might have been initially driven by compliance with privacy law, but privacy is increasingly becoming a value-add in emerging business models.
Privacy Is Not Only about Compliance

The US, EU, and several other countries have enacted privacy legislation that drives the need to link privacy requirements to software. While the EU has an omnibus privacy law that covers personal information in general, the US has a menagerie of laws that separately regulate financial and health information, information about children under 13 years of age, unsolicited email, and law enforcement access to personal information, among many others. For certain software projects, privacy is thus driven by compliance with laws. In this approach, best practice entails a requirements “smoke test” or questionnaire that asks developers to internally report the kinds of information that the system will encounter. These checklists can trigger meetings with in-house legal counsel aimed at studying the risk of noncompliance, and the result of such meetings could be new requirements aimed at reducing legal risk. While this form of regulation is necessary to keep business practices in line with societal norms, it doesn’t generally empower developers to innovate with respect to privacy.

Alternatives to legal compliance include compliance with voluntary standards. The “OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data,” which is an international standard, and the Fair Information Practice Principles (FIPPs), which are better known in the US, provide high-level principles that developers can use to guide their design decisions. For example, the OECD guidelines recommend limiting the use and sharing of personal information to those purposes for which it was originally collected, called the use limitation principle. This principle aims to prevent repurposing, which is a privacy threat, because individuals who are willing to share information for one purpose (such as communicating with a doctor) might be less willing to share the same information for another purpose (advertising). Several of these principles have also been aligned with the ISO 29100:2011 standard. The US National Institute of Standards and Technology (NIST) recently published Special Publication 800-53, Appendix J, which describes several critical privacy controls that software developers can align with their system requirements using traditional requirements trace matrices.

While compliance-based approaches provide software developers an important and necessary starting place, leaders in privacy are taking a more design-driven approach, wherein compliance is a consequence of design and not the driving factor. This includes techniques to express designer intent and trace privacy requirements across multitier applications to identify potentially conflicting requirements.

WHAT IS PRIVACY?

While some people might feel privacy is difficult to define, privacy experts generally refer to a handful of prominent definitions established in law scholarship. Samuel Warren and Louis Brandeis, in their seminal 1890 law review article entitled “The Right to Privacy,” define privacy to mean “the right to be let alone.” In computer science, privacy often equates to secrecy or confidentiality (that is, a subset of security), which is perhaps too conveniently addressed by access control and encryption technology. Related to secrecy is anonymity, or the ability of an individual to act without the corresponding ability to attribute his or her actions to their identity; this includes sharing de-identified information about individuals using techniques to anonymize sensitive datasets, such as k-anonymity, differential privacy, and so on.

Whereas privacy as secrecy and anonymity shift the responsibility to software to deliver these system qualities, a third definition concerns privacy as control, in which users are given notice and choice to participate in how their personal information is used. This third definition includes techniques for opting into or out of data collection, use and transfer, or nudges that encourage users to think about the impact of their decisions to share information at the time of those decisions.

Finally, other scholars have defined privacy as personal autonomy: the freedom to discover one’s identity and preferences without persecution or unwanted interference. In this respect, privacy concerns how companies and government agencies use datasets to assess creditworthiness, restrict air travel, or filter Internet search results based on a user’s behavioral profile.

Reference

Identifying the Operating Principle

In the book entitled *What Engineers Know and How They Know It*, Walter Vincenti highlights the importance of design selection in satisfying desirable operating principles or system qualities. These qualities are sometimes called *soft goals* in requirements engineering and are prefaced by directional verbs, for example, minimize response time or maximize throughput. A significant challenge to adapting Vincenti’s approach to designing privacy-preserving software is first recognizing that privacy is frequently secondary to the primary operating principles for most software-based systems. In other words, most software is first designed to address business goals or stakeholder needs, such as finding a job, purchasing a product, and so on.

In some cases, these goals can introduce privacy risks, because the means to satisfying those goals must leverage personal information in a way that exposes individuals to unwanted disclosures or loss of control. Consider an online retail recommender system that aims to maximize sales by recommending related products while minimizing returns due to customer dissatisfaction. To achieve these two business goals, the recommender service has access to customer reviews concerning product satisfaction, and it uses various weighting strategies that group shoppers by interests or life stages, such as a love of books or new mothers, or by preferences, such as the desire for lowest cost versus enhanced product features. To calculate these weights, the recommender system must collect and retain sensitive information about a person’s shopping habits and potentially how the person him- or herself would rate products. Privacy challenges arise if the recommendation interferes with the customer’s shopping experience by suggesting products based on inaccurate information, such as misunderstanding onetime “gift” purchases for a friend or correlating purchases across shoppers that aren’t representative of the user, such as linking interests in self-defense with hunting wild game. To minimize this kind of misrepresentation, for example, Amazon recommendations are paired with links that users can click to edit their browsing history and delete items used to make recommendations; these links instantiate a privacy requirement to transfer control to the user to reduce inaccuracies in recommendations.

Balancing Information Utility and Privacy Risk

Individuals experience privacy differently; some individuals are gregarious and outgoing, whereas others are more reserved; some individuals are part of the mainstream, while others are part of a minority. Designers need not assume that privacy is a zero-sum game. Instead, it can be layered to allow users to engage with systems at different intervals and extents that they deem appropriate. In other words, software designers can let users personalize their level of privacy to meet their own perceptions of privacy risk.

Privacy need not be a secondary principle. To improve privacy, designers can choose to maximize intimacy, which lets users have candid interactions with their most intimate family members and friends, or to maximize anonymity, which lets users explore new ideas and spaces without fear of misrepresentation or unwanted attention. Furthermore, developers can apply several functional requirements to enhance privacy and balance utility and risk. To name a few, developers can aim for the following:

- **Minimize data.** Only disclose the minimum set of information needed to complete a transaction. This includes using suppression, which means removing sensitive data elements, and generalization, which means substituting data elements with more abstract alternatives (such as changing US ZIP code to a city or state). Service-level agreements can be used to prevent re-identification by third parties.

- **Increase awareness.** Legal privacy notices are often long and difficult to read for most users. Provide short notices at the time of collection and nudge first-time users to understand the benefits and risks of their sharing decisions. Help users visualize what sharing means before they share...
by presenting visual prototypes and narrative use cases.

- **Maximize user control.** In some industries, one-click opt-out is the death knell of information sharing. This form of control is often too coarse, and users might never see an opportunity to opt-in later. In addition to opt-out, provide users with controls to shape information sharing in ways that let them mitigate their own perceived risks based on context without abasing permanently. This includes allowing users to limit sharing based on certain times, topics, or recipients as they deem appropriate.

- **Reduce inaccuracies.** Especially when data is used to make important decisions, implement consistency checks that serve to surface anomalies or potential inaccuracies. Errors in data can arise due to data entry error, merging datasets, and restoring from backups where errors can persist long after correction. When appropriate, let users participate in the filtering and correction of their own data.

As a matter of best practice, companies can develop a set of privacy design guidelines similar to those described above that serve to standardize privacy practices across projects.

Although some might argue that privacy is about compliance, business needs often dictate how value is derived from personal information, and that’s a potential source of privacy risk. Engineers need not succumb to an “all or nothing” approach to privacy; instead, they can balance privacy risk with their software design’s primary operating principles. This includes linking operating principles to privacy risks and finding ways to increase user participation in reducing exposure to risks. In the end, users benefit from personalized software and developers benefit from a more diverse, trusting, and resilient user base over time.

References


TRAVIS Breaux is an assistant professor of computer science at Carnegie Mellon University. His research interests include privacy, requirements engineering, and legal compliance. Breaux received a PhD in computer science from North Carolina State University. Contact him at breaux@cs.cmu.edu.