Engineering long-lived systems is hard, and adding privacy considerations to such systems makes the work harder.

Who may look at private data that I put online? Certainly I may look at it, plus any person I explicitly authorize. When may the online system’s operators look at it? Certainly when customer service representatives are assisting me in resolving a problem, they might look at the data, though I would expect them to get my permission before doing so. I would also expect my permission to extend only for the duration of the support transaction and to cover just enough data elements to allow the problem’s analysis and resolution.

When may developers responsible for the software’s evolution and maintenance look at my data? Well, pretty much never. The exception is when they’re called in during escalation of a customer service transaction. Yes, that’s right: developers may not, in general, look at private data contained in the systems that they have written and continue to support. In practice, it’s probably infeasible to make developer access impossible, but we should make it highly visible.

Doesn’t the code have a role in this? Of course it does, but the code isn’t generally created by the consumer and isn’t private. Insofar as consumers create code—and they do when they write macros, filters, and configurations for the system—it’s part of this analysis. The system life cycle and privacy implications of user-created code are beyond the current state of the art and merit significant attention in their own right.

So what happens when an online system is forced to migrate data from one version of the software to another version? This happens periodically in the evolution of most long-lived systems, and it often involves a change to the underlying data model. How do software engineers ensure that the migration is executed correctly? They may not spot-check the data, of course, because it’s private. Instead, they build test datasets and run them through the migration system and carefully check the results. But experienced software engineers know very well that test datasets are generally way too clean and don’t exercise the worst of the system. Remember, no system can ever be foolproof because fools are way too clever. So we must develop tests that let us verify that data migration has been executed properly without being able to examine the result and spot-check it by eye. Ouch.

What’s the state of the art with respect to this topic? Our community has produced several documents that represent a start for dealing with private data in computer systems. By and large, these documents focus on foundational issues such as what is and isn’t private data, how to notify consumers that private data will be gathered and held, requirements of laws and regulations governing private data, and protecting private data from unauthorized agents and uses.

Rules and regulations concerning privacy fall along a spectrum. At one end are regulations that attempt to specify behavior to a high level of detail. These rules are well intended, but it’s sometimes unclear to engineers whether compliance is actually possible. At the other end are rules such as HIPAA (Health Insurance Portability and Accountability Act) that simply draw a bright line around a community of data users that comprise doctors, pharmacies, labs, insurers, and their agents and forbid any data flow across that line. HIPAA provides few restrictions on the handling or use of this data within that line. Of course, one irony with HIPAA is that the consumer is outside the line.

Given the current state of engineering systems for online privacy, regulations like HIPAA are probably better than heavy-handed attempts to rush solutions faster than the engineering community can figure out feasibility limits. This is an important area of work, and some promising research is emerging, such as Craig Gentry’s recent PhD thesis on homomorphic encryption (http://crypto.stanford.edu/craig/craig-thesis.pdf), but full rescue looks to be years off. We welcome reports from practitioners and researchers on approaches to the problem of maintaining data that may not be examined.

Marc Donner