In Search of Usable Security: Five Lessons from the Field

A new system reduces the time to enroll in a secure wireless network by two orders of magnitude, and it also gets high marks for usability and user satisfaction. This real-world example reveals five general lessons for usable, secure system design.

Dirk Balfanz, Glenn Durfee, and D.K. Smetters
Palo Alto Research Center

Rebecca E. Grinter
Georgia Institute of Technology

As early as 1975, Jerome Saltzer and Michael Schroeder argued that usability was an essential component of secure systems.[1] In their seminal paper, "The Protection of Information in Computer Systems," they developed eight basic principles of information protection, the last of which called for "psychological acceptability" of information protection systems. Curiously, many of today's security systems seem to ignore this idea.

To be fair, human-computer interaction (HCI) research in 1975 was in its infancy, so few people knew how to design and evaluate computer systems for usability. Almost 30 years later, though, HCI design and evaluation techniques have left the research lab: corporations regularly use such techniques to assess their products' usability. In spite of this advance, researchers and practitioners have only recently begun applying HCI usability evaluation techniques to security systems. Their findings show that end users struggle to comprehend the security decisions with which they are presented, so they're more likely to misconfigure—and thus jeopardize—their security.[2] Moreover, users often deliberately disable or ignore security to get their work done—for example, Anne Adams and Angela Sasse found that people often deliberately disclose or share passwords to ease system access.[3] A lack of usability causes them, unwittingly or not, to turn secure systems into highly insecure ones.

Luckily, professionally managed infrastructures such as corporate firewalls protect most of us while at work. However, as users take a wider variety of mobile devices into their homes or out on the road, they increasingly encounter security decisions that only they can make, not their systems administrator back at the office.[4] We must develop technologies that put security decisions into users' hands. To address the challenge of building such technologies, we've spent the last several years working as a team combined of both security and HCI researchers. In this article, we'll present some lessons we've learned while building usable, secure systems; we illustrate these lessons by contrasting two different versions of a public key infrastructure (PKI)-based secure wireless network.

Usable PKI: Making the impossible easy

When we first set up a wireless network at the Palo Alto Research Center (PARC), we opted for a PKI-based solution because it promised better security, the flexibility of public-key cryptography, and better ease of use than password-based systems. The idea was to give 200 users an X.509 certificate[5] and use the 802.1x Transport Level Security authentication method of the Extensible Authentication Protocol (EAP-TLS)[6,7] to authenticate them. Once we set up such a network, we can get a high degree of network security without requiring continued user (or administrator) intervention.

We knew that setting up a PKI might be tricky. Studies show that many people find PKI deployment incomprehensible, complex, and unusable.[8] Consequently, most organizations currently don't use PKI technology, despite the expectation that PKIs would be in widespread use by now. Because we're security researchers with extensive experience with PKI technology, we offered to help the administrative staff roll out the PKI. We thought we could avoid the pitfalls of PKI deployment. We were wrong.

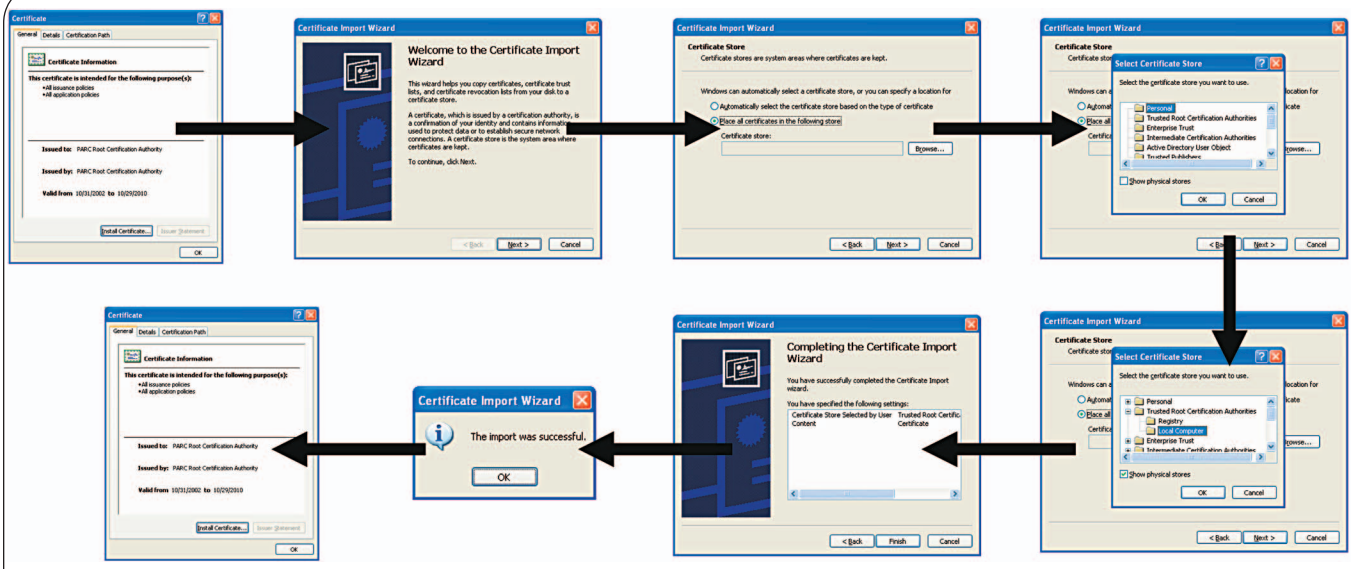

Figure 1. Manual setup. These dialog boxes show eight of the 38 steps required when manually configuring the secure client setup for a wireless network.

Experiences with traditional PKI deployment

In the first version of our wireless LAN, users had to request and install an X.509 certificate from an internal certification authority and then configure the 802.1x client software provided by their operating system to use the EAP-TLS authentication protocol. In order to submit a certificate request, users had to determine and provide their wireless card's media access control (MAC) address and install the internal certificate authority's (CA's) root certificate.

A team of system administrators and security researchers set up the wireless network and the PKI. The security researchers spent considerable time configuring and managing the CA software and keys (establishing the corporation's "root" for the PKI), providing training for the systems administrators, and then supporting the transition of responsibility to the central computer support group.

Our next step was to conduct a study to determine what usability problems the users encountered. Once the wireless network and the PKI were in place, our HCI researcher studied eight subjects' enrollment experiences. All the subjects had advanced degrees, typically PhDs in computer science and related disciplines, but the average time it took for them to request and retrieve their certificates and then configure their systems was 140 minutes.[9] More significantly, despite using a fairly automated Web-based enrollment system (similar to those used by commercial certificate vendors such as Verisign) and the GUI-based 802.1x wireless configuration software provided by Microsoft Windows XP, the process involved a total of 38 steps to complete enrollment. Each step forced the user to make a decision or take an action (see Figure 1).

To help the subjects enroll, our administrators produced an elaborate set of instructions that detailed each step. Almost all the subjects printed out the instructions, but even those who started the process determined to understand what they were doing soon abandoned that goal and instead followed the instructions mechanically. In the end, many test subjects described the enrollment process as the most difficult computer task that PARC had ever asked them to do. They had little, if any, idea of precisely what they had done to their own computers. Several commented that if something were to go wrong, they couldn't perform even basic troubleshooting, which for some subjects marked the first time that they had ever experienced the inability to self-administer their own machines—not a positive experience. Although PKI technology secured the subjects' machines for wireless use, it simultaneously reduced their ability to configure and maintain their own machines.

This study's results surprised us: we thought the experience would be relatively simple, but the study confirmed the conventional wisdom that deploying a PKI is difficult. Even the commercial software vendors supplying parts of the system expressed surprise that we would take on the complexity of a PKI-based deployment to gain added security, rather than opting for a simpler and less secure password-based system.

The results also highlighted the hidden costs of making PKI work. To compensate for its poor user interface, we used a combination of documentation and training to

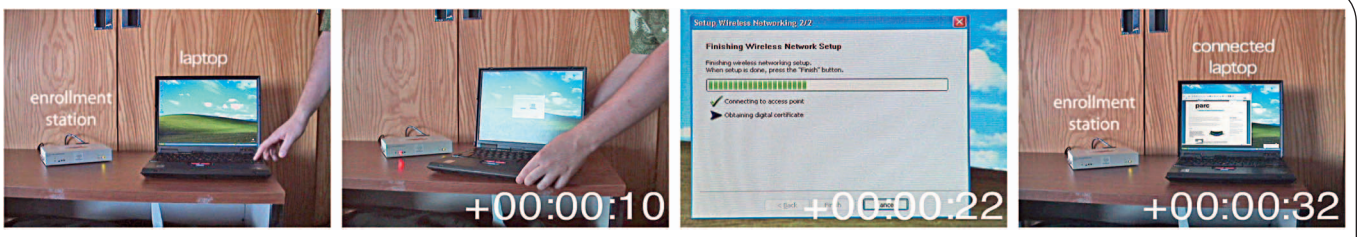

Figure 2. Gesture-directed automatic configuration. The user briefly aligns the infrared ports of a laptop and an enrollment station to start a fully automated setup.

help users enroll in the wireless network. Such a solution is painful and hardly practical even in a well-supported enterprise setting, making it completely infeasible for smaller home or office networks.

User-friendly PKI deployment

Based on our previous work in designing user-friendly security technology,[10,11] we felt we could do much better. We posed a difficult challenge: is it possible to build a system that lets an average end user join a device to a wireless network using the strongest, PKI-based security standards available, simply, easily, and intuitively?

The system we built is described in detail elsewhere,[9] but we'll summarize it here briefly. The setup process is largely automated—a small wireless setup application takes care of tasks such as requesting and installing certificates or configuring the 802.1x client. The setup application also collects additional information (user names, MAC addresses, WLAN service set identifiers [SSIDs], and so forth).

We introduce new devices to the network using *location-limited channels*.[10] The user takes a laptop to an *enrollment station* and initiates a brief infrared exchange between the former and the latter (see Figure 2). During this exchange, enough public information is transmitted to facilitate a secure wireless connection between the laptop and the station. (Sending only public information over infrared makes it difficult to attack a location-limited channel.) Over the resulting secure wireless connection, the laptop requests and later receives its client certificate (the enrollment station forwards certificate requests to the same CA we mentioned earlier). Once the laptop receives the certificate, the setup application automatically installs it and configures the 802.1x client software. The (built-in) 802.1x client software then takes over and authenticates the laptop to the wireless network, using the newly installed certificate.

Two important aspects of this design stand out: first, we use a gestural user interface to solve the key distribution, or trust-bootstrapping, problem. By simply pointing out the laptop to the enrollment station, the two devices exchange trust information—the gesture captures the user's intention to enroll *this* particular laptop into *that* particular enrollment station's network. We call this idea *gesture-directed automatic configuration*. Second, we give the user an intuitive trust model: our enrollment station is locked in a room, so only someone who can get into that room and walk up to the enrollment station can get onto the wireless network. Such a model ties digital security to physical security. At PARC, for example, users must present their badge to a system administrator before that administrator will unlock the enrollment room.

Usability studies demonstrate that this approach is much simpler and more intuitive to users than setting up security via traditional methods. It took users an average of 1 minute and 39 seconds—and a total of four steps—to add a new device to our secure corporate wireless network using gesture-directed automatic configuration. We also got much higher marks in user satisfaction and confidence. Even our system administrators (who by this time had become quite familiar with the manual system) prefer the new system: they now use it exclusively to enroll other users' computers.

Five lessons

After three years of designing, implementing, and deploying secure systems that put usability first, we learned a number of lessons that apply across many efforts.

You can't retrofit usable security

The security community has long argued that security must be designed into systems from the ground up; it can't be "bolted on" to an existing system at the last minute. The same is true for usability—usability of security in particular. Adding explanatory dialog boxes to a confusing system is not the solution, and forcing a better GUI onto a fundamentally unusable design is like applying a Band-Aid to a broken leg. Developers must think about usability, security, and their interplay during the very first stages of system design. The decision not to bother users with passwords, for example, could affect applications' protocols or data-protection mechanisms.

A gestural user interface lets users intuitively express their application needs while simultaneously supporting

strong security. This intuitive interface stands in strong contrast to our original, "traditional" interface, which focused on using friendly GUI components to walk the user through the process of manually configuring certificates and wireless network settings. Although we put a great deal of effort into making that GUI-based approach easy to use, our effort failed completely. Only by starting from completely different interaction principles did we manage to build a usable and secure system. Such fundamental design decisions must be made at the very beginning of the development process.

Tools aren't solutions

Tools such as SSL or IPSec are great resources in the hands of developers because they mean that we can rely on proven protocols and implementations to give our applications certain security properties. What such tools can't provide, though, is the solution to a user's problem: they're like Lego bricks, whereas the application that solves the problem is like a whole building built from Lego. Using the same Lego brick set can help us either build a functional, sturdy home or a useless, brittle shack.

Recognizing that available technologies such as SSL or security APIs are nothing more than tools is only part of this lesson; the other part is appreciating that our current portfolio of available tools is rather incomplete. We're missing building blocks for the higher layers of application design. Earlier, we saw a glimpse of what those higher-level building blocks might be (location-limited channels such as infrared provide a user-friendly way to bootstrap trust for collocated devices). Additionally, small application-specific PKIs can give us all the benefits of public-key cryptography without the drawback of a global PKI's nonexistence. These tools proved highly successful in the context of the wireless security application discussed in this article, but like other good tools, they're also applicable to a wide variety of other applications.[11] We must find more high-level building blocks that application designers can employ to create user-oriented solutions.

Mind the upper layers

Security is not something to handle only in the lower layers of the networking stack or in the depths of the operating system. If we try to solve the security problem purely in those lower layers, users inevitably have to deal with those layers when something goes wrong (such as when they need to understand a security-related event or want to change their security settings). If we design security into all of an application's layers (in particular, its upper layers), it becomes *implicit* and hence much more user-friendly. For example, in a recent study,[11] users created shared virtual spaces for collaborating with each other. They invited each other to join a space and let members add objects to the space (such as files, cameras, or speakers). This metaphor provides a natural basis for security: only users who are members of a space can access objects in that space. By making sharing a top-level primitive of the application, users can understand what they are sharing and with whom, and application developers can understand the security requirements implied by this intuitive model.

In general, the security mechanisms an application implements must be compatible with what the user needs to accomplish. A user who finds that one of her software's security features prevents her from actually getting her job done is much more likely to turn that security feature off.

Users typically think about security in terms of their application goals—"I don't want anyone but the recipient to read this email," or "I don't want my credit-card number stolen"—not in terms of keys, certificates, or access control lists. Where possible, applications should be designed to make security implicit, to take advantage of actions a user must take anyway to discover what security operations must take place.[4] In the example presented earlier, users wanting to join a particular wireless network did so by pointing out the enrollment station serving that network via infrared, as if they were using a remote control. Software could take advantage of that action to exchange authentication information and securely join the network. The user, however, could focus only on his high-level goal: getting a particular device onto a particular wireless network.

Taking advantage of users' intentions requires understanding the modes of expression a typical user might employ to indicate his or her intentions and then piggybacking security actions on those modes, rather than expecting the user to learn new behavior. However, this also requires understanding the mental models users employ to reason about their security environment.

This implies that security now becomes the application developer's responsibility, which runs counter to the traditional approach of abstracting away security in the lower layers of the system. However, it doesn't mean that application developers must now create their own encryption algorithms. They have a rich portfolio of tools at their disposal, and it's their job to apply those tools in a way that solves a particular problem for the user.

Keep your customers satisfied

To put your users' needs first seems like an obvious lesson, well understood in other fields of system design. The security community, however, often believes that security is more important than users' other needs, even when such dogged devotion results in a system that doesn't let users accomplish the tasks for which that system was designed.

Expertise can blind even those most sensitive to user concerns. As security researchers, we believed that we had designed an easy-to-enroll-in traditional PKI deployment. After all, the design we picked was similar to that chosen by others in the field (such as Verisign,

www.verisign.com), and when we used the system ourselves to obtain certificates and configure machines for the secure wireless network, it took us only a couple of minutes. Imagine our surprise when usability studies revealed it took users on average two hours to join the secure wireless network. Other security researchers agreed with our expectations, to the point of suggesting that our empirical results[9] must be incorrect—it couldn't possibly take users that long to enroll.

When designing a system, developers must keep in mind that they aren't average users, so after they finish the system, their target audience should test it. Such studies can often be performed relatively simply—a small population of subjects will provide much of the information necessary to evaluate a system[12] and can provide the basis for effective iteration cycles of design, implementation, and evaluation. Even simply discussing their experiences with the designated user population gives security and systems designers the opportunity to see through the eyes of their end users.

To gain maximal user feedback and discover lingering usability (and security) flaws, system designers and implementers should follow the system or application through usability testing and deployment. Support questions provide another window into what users find difficult or unintuitive, and all of this information can feed back into redesign iterations and implementation and interface refinement. This data provides a valuable basis for the design of future systems, applications, and primitives.

Another somewhat surprising aspect of this lesson is that usability failures sometimes hide behind apparent success stories. After all, the test subjects at PARC used our manual enrollment scheme, and we successfully managed to deploy a PKI-based wireless security solution. Yet, our usability study revealed many problems behind that success. Outright failures might present tempting problems for the HCI community and security researchers interested in making secure and usable systems, yet torturous successes such as the one described earlier also have much to offer this same group. Usability studies can focus on the work all participants performed in designing, deploying, and using the system; it can also help interested parties find obvious places to reduce the effort required.

Think locally, act locally

Application security often seems to require generic, universal solutions to problems—solutions that don't exist in practice. Many academic papers on novel cryptographic protocols, for example, assume that all players in the protocol reliably know each other's public key. Secure email, to this day, relies on a global public-key infrastructure. (To be fair, Pretty Good Privacy [PGP] does not. However, the trust model underlying it is even more convoluted than that of a traditional X.509-based PKI, which is one reason why it's even more marginalized.)

The lack of a global PKI renders all applications requiring one immediately unusable, but even if we all participated in a global PKI, this PKI—necessarily being generic in nature—would be part of the plumbing, the underlying security system that application security is built atop. Unfortunately, users would be exposed to details of this plumbing—to send an encrypted email to a friend, you would first need to get a certificate, which is not something the average user will comprehend.

We found that by thinking locally, we could avoid much of the exposure to global plumbing. In our current example, we rolled our own mini-PKI for a wireless network; in another study,[13] every Web site acts as its own CA. As a result, in both cases, we could completely hide from the user the fact that certificates were being requested, installed, and used.

On the contrary, if a certificate is a generic tool that can be used independently of applications, it also becomes an entity that users must deal with directly, independent of the application's context. We (and others[14]) learned that this profoundly confuses users. Systems that follow the "think locally" principle are also often easier to deploy, because they don't require administrators to coordinate with some larger infrastructure or organization. As a result, they can offer greater opportunities for automatic configuration—for example, we built a standalone version of our secure wireless enrollment system that incorporates gesture-based enrollment, a CA, and a wireless access point in one box.[9] Such a device can automatically configure itself, creating a highly secure wireless network and a small PKI that can be used transparently, even in a home setting.

Interest in the usability of information security has finally picked up in the research community. Information security often fails because of the lack of usability: users either misunderstand the security implications of their actions, or they consciously turn off security features to "fix" usability problems.

To rectify this situation, we must design systems that are simultaneously usable and secure. Not only will this better protect users, it will actually enable them to accomplish tasks they couldn't accomplish before due to the lack of a trustworthy infrastructure.

The lessons we've described here are applicable to usable secure system design in general, and we encourage the community to draw from and refine them. □

References

1. J.H. Saltzer and M.D. Schroeder, "The Protection of Information in Computer Systems," *Proc. IEEE*, vol. 63, no. 9, 1975, pp. 1278–1308.
2. A. Whitten and J.D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," *Proc. 8th Usenix Security Symp.*, Usenix Assoc., 1999, pp. 169–184.

3. A. Adams and M. Angela Sasse, "Users Are not the Enemy: Why Users Compromise Computer Security Mechanisms and How to Take Remedial Measures," *Comm. ACM*, vol. 42, Dec. 1999, pp. 40–46.
4. D.K. Smetters and R.E. Grinter, "Moving from the Design of Usable Security Technologies to the Design of Useful Secure Applications," *Proc. New Security Paradigms Workshop '02*, ACM Press, 2002, pp. 82–89.
5. *Recommendation X.509: The Directory/Authentication Framework*, Consultative Committee on Int'l Telegraphy and Telephony, 1988; www.itu.int/publications/index.html.
6. B. Aboba and D. Simon, "PPP EAP TLS Authentication Protocol (EAP-TLS)," IETF RFC 2716, Oct. 1999; www.ietf.org/rfc/rfc2716.txt.
7. *ANSI/IEEE Std. 802.1x, Port-Based Network Access Control*, IEEE, 2001.
8. P. Doyle and S. Hanna, "Analysis of June 2003 Survey on Obstacles to PKI Deployment and Usage," 2003; www.oasis-open.org/committees/pki/pkiobstaclesjune2003surveyreport.pdf.
9. D. Balfanz et al., "Network-in-a-Box: How to Set up a Secure Wireless Network in under a Minute," *Proc. 13th Usenix Security Symp.*, Usenix Assoc., 2004, pp. 207–221.
10. D. Balfanz et al., "Talking to Strangers: Authentication in ad hoc Wireless Networks," *Proc. 2002 Network and Distributed Systems Security Symp.* (NDSS'02), Internet Soc., 2002, pp. 23–35.
11. W.K. Edwards et al., "Using Speakeasy for ad hoc Peer-to-Peer Collaboration," *Proc. ACM 2002 Conf. Computer Supported Cooperative Work* (CSCW 2002), ACM Press, 2002, pp. 256–265.
12. J. Nielsen and T.K. Landauer, "A Mathematical Model of the Finding of Usability Problems," *ACM Conf. Human Factors in Computing Systems* (INTERCHI '93), ACM Press, 1993, pp. 206–213.
13. D. Balfanz, "Usable Access Control for the World Wide Web," *Proc. Ann. Computer Security Applications Conf.*, IEEE CS Press, 2003, pp. 406–415.
14. P. Gutmann, "Plug-and-Play PKI: A PKI Your Mother Can Use," *Proc. 12th Usenix Security Symp.*, Usenix Assoc., 2003, pp. 45–58.

Dirk Balfanz *is a security researcher at the Palo Alto Research Center. His technical interests include end-to-end security and ubiquitous computing. He received a PhD in computer science from Princeton University. Contact him at balfanz@parc.com.*

Glenn Durfee *is a security researcher at the Palo Alto Research Center. His technical interests include applied cryptography, usable security, and security for mobile and wireless devices. He received a PhD in computer science from Stanford University. Contact him at gdurfee@parc.com.*

Rebecca E. Grinter *is currently an associate professor at the College of Computing, Georgia Institute of Technology, but she worked at the Palo Alto Research Center at the time of this study. Her technical interests include computer-supported cooperative work, human-computer interaction, security, and ubiquitous computing. She received a PhD in information and computer science from the University of California, Irvine. She is a member of the ACM. Contact her at beki@cc.gatech.edu.*

D.K. Smetters *is a senior member of the research staff at the Palo Alto Research Center. Her technical interests include security for mobile and wireless devices, usable security, and applied cryptography. She received a PhD from the Massachusetts Institute of Technology. She is a member of IACR, Usenix, and the ACM. Contact her at smetters@parc.com.*