

Barriers to Usable Security? Three Organizational Case Studies

Deanna D. Caputo | MITRE

Shari Lawrence Pfleeger | Pfleeger Consulting Group

M. Angela Sasse | University College London

Paul Ammann, Jeff Offutt, and Lin Deng | George Mason University

Usable security assumes that when security functions are more usable, people are more likely to use them, leading to an improvement in overall security. Existing software design and engineering processes provide little guidance for leveraging this in the development of applications. Three case studies explore organizational attempts to provide usable security products.

Security products are designed to keep individual and corporate users' systems safe from threats, but such products are effective only if employees and customers are able and willing to use them—and use them properly. When asked about involving usability experts in their work, software project managers often explained why they didn't find such experts useful. One manager said, "I would rather hire a Wiccan, because at least they have a spell book." Security developers are finally realizing through experience and data that many users ignore or circumvent unusable security products. Prompted by lost sales, lost time, and a profusion of misuse errors, developers want to build usable security into their projects. The July 2009 National Academy of Sciences Workshop identified several challenges to advancing research in usability, security, and privacy, including inconsistent terminology and definitions; limited access to data; scarcity of expertise; unfamiliarity with work at the intersection of usability, security, and privacy; and difficulty of moving security usability research results into practice.¹

To date, many developers claim to have thoughtfully integrated user-centered design into their processes to reach a usable security goal. However, usable security involves much more than interfaces. It addresses how people think about and use their computer systems, particularly in the context of how they do their jobs. Good usable security accounts for differences in user experience, needs, skills, and attitudes as well as changing tasks and business needs. In addition, successful developers will need to build usable security into processes spanning the system's life cycle, from conception through design, implementation, and evolution, rather than addressing only a single user at one point in the software development process.

A rich body of behavioral science results can be mined to improve the likelihood of successful usable security. Two of the authors of this article, Shari Lawrence Pfleeger and Deanna D. Caputo, have mapped out several promising areas where behavioral science results could be applied to significantly improve usable security, including understanding heuristics, biases, framing, and cognitive load.²

Other researchers point to studies of human–computer interaction that can form a basis for improving usability.³

This article describes the findings of three case studies, each of which explores a unique organization’s attempts to improve the usability of its security products.

Usable Security Needs a Different Approach

In August 2011, the Institute for Information Infrastructure Protection (I3P) convened a National Institute of Standards and Technology (NIST)-sponsored workshop to encourage practitioners to take the next step in advancing research in usable security: integrating actions promoting usable security in all stages of the software development life cycle.⁴ Participants in the Software and Usable Security Aligned for Good Engineering (SAUSAGE) Workshop observed that the intersection of usability and security is problematic because making software robustly secure often entails controlling or limiting user actions in ways that reduce perceived usability (and perhaps utility and adoption of the software). This has led to an unproven yet persistent belief on the part of some developers that there is a tradeoff between usability and security, which makes usable security unachievable if true. Therefore, one of the workshop’s recommendations was to conduct a series of case studies to investigate how organizations deliver usable security and how they evaluate the effects of changes they implement. As a consequence, NIST and the US Department of Homeland Security funded a multidisciplinary team of researchers to develop a uniform case study methodology and apply it to three different organizations. This article describes the results of this two-year project to understand what leads to or hinders the development of usable security.

Although some developers know—from experience or published research—that most users ignore or circumvent unusable security products, others still erroneously think there’s a tradeoff between usability and security and that users should make extra efforts to be secure. For example, Adam Beaument and colleagues found that users will ignore or circumvent security that takes too much time and effort and therefore undermines productivity.⁵ Unfortunately, existing software design and engineering processes provide little guidance for leveraging usability and security in the development of next-generation end-user applications—neither do they suggest how to make existing security products and services more usable in cost-effective ways. All software engineering processes should address human decision making and behavior, but system developers and designers are rarely taught how to improve system characteristics while maintaining decision-making effectiveness. To address these gaps, this research focused on

developing case studies of usable security design and implementation that could be used both in understanding the problems and in informing organizations about solutions. In particular, these case studies investigated how to incorporate usability and security in software engineering processes.

The case studies were based on a series of interviews with key developers, designers, and managers. The interview questions drew on principles of *activity theory*,^{6,7} which support identification of the organizational drivers and constraints that shape team behavior and influence design and development processes. Activity theory, coupled with questions about human–computer interaction, ensures that broader contextual drivers are accounted for in both the design and evaluation of technological products. It’s valuable in directing researcher attention beyond individual cognitive tasks to identify and assess how contextual factors, such as organizational structure, the build environment, project funding, and reward systems, influence team and individual approaches to task accomplishment.

Consequently, the research team looked at organizational factors in software development and deployment that could affect activities intended to integrate usable security into software development. For example, how a software development team perceives or relates to end users is likely to influence how it balances and implements usability and security in its software designs. In many large organizations, software teams work at a significant distance from the real-world end users of their products. Other times, software functions are developed in response to development requests from customers who aren’t end users and who might be poorly informed about what end users actually do, resulting in software that fails to meet the usability needs of real end users.

For this study, usable security is motivated by the assumption that when security functions are more usable, users will be more likely to use them, leading to an improvement in overall security. In this work, the research team adopted the International Organization for Standardization (ISO) definition of *usability*: “The extent to which a product can be used by specified users to achieve goals with effectiveness, efficiency, and satisfaction in a specific context of use.”⁸ Based on this definition, the team defined *usable security* as “delivering the required levels of security and also user effectiveness, efficiency, and satisfaction.” Recognizing the importance of performing rigorous case studies and drawing on solid behavioral science, this multidisciplinary research team brought together computer, social, and behavioral scientists from three different institutions with expertise in software engineering, including software security, testing, process, design, and usability, as well as experience in applying social and behavioral science methods to security.

These three case studies are intended to form the basis of a growing corpus of studies that will help the security and usability communities understand how to make security products more usable. Indeed, it is hoped that eventually the lack of usability will no longer be an excuse for users to circumvent security software, systems, or procedures.

Methodology

Any credible investigation into what makes security more usable must necessarily involve an understanding of the users, their motivations, and their tasks. As David Alan Grier noted, “We can’t understand our current state of affairs without knowing the way we thought and acted in the past any more than we can fully understand a neighborhood without remembering the forces that shaped it.”⁹ To explore the environment in which usable security is embedded, we employed behavioral science techniques that explore context and motivation. In particular, the intent was to document the organizational efforts and expertise involved in developing a security product that has been constructed or enhanced to make it more usable. Because the goal was to identify factors that lead to success, the case study methodology involved three things: stating tentative hypotheses, evaluating the degree of control over variables, and taking steps to ensure the meaningfulness of the investigation.

Michael Agar noted, “In its classic form, a hypothesis is a statement of the co-variation between two variables. ... On the other hand, hypothesis has a broader sense as ‘an idea to check out.’ ... Something learned in a conversation becomes a hypothesis to check in further conversations or observations.”¹⁰ Unlike experiments, in which sampling and control over all variables drive the research design, it’s not appropriate in this type of exploratory research to expect every variable to have precise theoretical and operational definitions. An initial definition may be the starting point for discussion of a concept, but researchers might find that some people use the same language with very different meanings. Thus, this usable security case study methodology blends several approaches drawn from psychological and anthropological case study techniques so that systematization and study design, combined with careful elicitation of data from appropriate sources, can ensure that the evidence supporting the hypotheses is credible. In this way, as the corpus of usable security case studies expands, findings from individual cases can be explored and tested in other environments, leading to a broad understanding of how, when, and under which conditions to provide usable security.

The three organizations studied were selected to help explain three things: why each organization added usability and security elements to its software

development process, how and where they added them, and how the organization determined that the resulting software was usable and secure. For each organization, we sought findings that could be supported by arguments that support or refute hypothesized theories. Initially, the research team hypothesized three possible explanations of why changes in the software development process might lead to more usable security:

- *The “key individual” theory:* Improved outcomes resulted not from the process changes but instead from the efforts of a single individual who cares about usable security.
- *The “experienced team” theory:* Improved outcomes resulted not from the process changes but instead from the team’s prior experience in building usable security.
- *The “incentives” theory:* Improved outcomes resulted not from the process changes but from incentives placed on team performance with respect to usable security.

To ensure the meaningfulness of the investigation at multiple levels, we developed a set of interview questions for each of three units of analysis levels: developer, product manager, and senior manager. The final interview protocol can be found in the “Interview Questions for Case Studies” sidebar. To verify that the investigation was meaningful and confirm anticipated control over variables before carrying out the reported case studies, the team first performed a pilot study of the full protocol. This test of the questions, ordering, and interactions not only helped us understand and practice the steps but also demonstrated that the original sets of questions were too long and repetitive. The questions in the sidebar represent the revised questions, updated after the pilot test.

Once the case study methodology and interview questions were piloted and well documented, the team obtained approval of the study design from each research organization’s Human Subjects Review Committee to ensure that sensitive data was adequately protected. The research team took many measures to protect the volunteering organizations’ intellectual property and reputations. These measures included having both subjects and research team members sign nondisclosure agreements, anonymizing the organization names, encrypting all documents, avoiding cloud services where control over documents could be lost, and limiting or shredding all paper documents, such as notes or printouts. A more complete discussion of the usable security case study methodology developed during this research program can be found elsewhere.¹¹

Interview Questions for Case Studies

These questions were asked of participants at three levels of inquiry that correspond to the three levels of analysis: software development, product management, and higher management. Within each level, we asked questions about security, usability, and software development.

Section 1: Development Level

Please give us an overview of the product that we are discussing today. Is this a new or legacy product? If possible, as part of your overview, can we see a demo?

Security

1. Did the specifications you were given cover security aspects? For instance, was a risk/threat analysis provided? Were security mechanisms specified? What were the security challenges in the project?
2. Were you given any specific security criteria to meet? (When would the software be “secure enough”?)
3. Who were security tasks assigned to? Did a team member have the role of “security champion”? If decisions were made, can you walk us through the process?
4. Did you use any security design methods and tools? (Example: misuse case analysis.)
5. Were any security assessments carried out during the project? (Examples: code walkthrough, pen testing.) If so, what were the results? What changes did you make in response?
6. Did you consult other team members, or other security resources in the company, at any stage? If so, what was your question, and what insights/help did you obtain?
7. Did you consult any external security resources—security guidelines, CERT pages, threat analyses provided by consultants—at any point? If so, did the resources help you to answer the question?

Usability

1. Did the specification you were given cover usability aspects? For instance, was the UI design provided? What were the usability challenges in the project?

2. Were you given any specific usability criteria to meet? (When would the software be “usable enough”?)
3. Who were usability tasks assigned to? Did a team member have the role of “usability champion”? If decisions were made, can you walk us through the process?
4. Did you use any usability methods and tools? (Examples: task analysis, prototyping.)
5. Were any usability assessments carried out during the project? (Examples: heuristic evaluation, user testing.) If so, what were the results? Did you make any changes in response to the findings?
6. Did you consult other team members, or other usability resources in the company, at any stage? If so, what was your question, and what insights/help did you obtain?
7. Did you consult any external usability resources—literature, design guidelines—at any point? If so, what was your question? Did the resources help you to answer the question?

Software Development

1. Did you follow a particular software development process on this project? If so, how did it diverge from the official process?
2. Who produced the usability and security requirements for the project?
3. Did you encounter conflicting usability and security requirements in this project? If so, what were the choices? Can you walk us through your decision process?
4. Did the software development process support the decision? If yes, how? If not, why not?
5. Did you consult any internal or external resources—colleagues, developer forum—to help you solve the problem? If so, what information was helpful to you?
6. How did you conduct the testing during the software development process? What were the results? Did you test your security mechanisms for usability?
7. Did you perform an evaluation of usability and/or security of

Continued on page XX

Participating Organizations

We identified three providers of cybersecurity systems willing to participate in our case studies. Each organization agreed to provide access to documentation of its perceived need for usable security, the steps taken to build usable security into its development process, and data useful in evaluating the effects of using the enhanced development process. Before the interviews, the research team made sure that the organizations understood the steps of the research process. In

particular, we asked about how problems and opportunities were recognized and addressed, and how the corrections or enhancements were evaluated. In return for their time and sharing, the organizations received a confidential, informal, outside, usable security evaluation by the senior research team of computer scientists, psychologists, and an anthropologist. This evaluation included a description of skills, processes used, gaps in team composition (such as usability expertise), and lessons learned during the development cycle that

Continued from page XX

your project? If so, how did you conduct the evaluation? Did you contact actual users?

8. In the software development process, when was usable security considered and implemented? (Examples: at the same time as other regular features, or after everything is done.)
9. How were usability and security handled during maintenance activities?
10. What percentage of the total project effort was spent on usable security?

Section 2: Product Level

Please give us an overview of the product that we are discussing today. If possible, as part of your overview, can we see a demo?

Security

1. What were the specific security goals/requirements for this project?
2. Why were they important, and for whom? (Example: reputation of customer organization.)
3. Who identified the security goals, and how?
4. Were specific criteria for meeting security goals identified (when the software would be “secure enough”)?

Usability

1. What were the specific usability goals/requirements for this project?
2. Why were they important, and for whom? (Example: productivity of employees in customer organization.)
3. Who identified these usability goals, and how?
4. Were specific criteria for meeting usability goals identified (when the software would be “usable enough”)?

Software Development

1. Were there specific goals for usability of the security mechanisms in the software?
2. Why were they important, and for whom? (Example: so customer organization would retain its own customers.)

3. Did you contact actual users?
4. Who identified the specific goals, and how?
5. What percentage of the total project effort was spent on usable security?
6. Did you test your security mechanisms for usability?

Level 3: Higher Management Level

Please give us an overview of the product that we are discussing today. If possible, as part of your overview, can we see a demo?

Security

1. What were the specific security goals/requirements for this project?
2. Why were they important, and for whom? (Example: reputation of customer organization.)
3. Who identified the security goals, and how?
4. Were specific criteria for meeting security goals identified (when the software would be “secure enough”)?

Usability

1. What were the specific usability goals/requirements for this project?
2. Why were they important, and for whom? (Example: productivity of employees in customer organization.)
3. Who identified these usability goals, and how?
4. Were specific criteria for meeting usability goals identified (when the software would be “usable enough”)?

Software Development

1. Were there specific goals for usability of the security mechanisms in the software?
2. Why were they important, and for whom? (Example: so customer organization would retain its own customers.)
3. Did you contact future users?
4. Who identified the specific goals, and how?
5. What percentage of the total project effort was spent on usable security?
6. Did you test your security mechanisms for usability?

produced the more usable software. These documents could be used by the organizations to raise awareness internally about good practices, teach other software developers how to adopt them, and identify areas of skill or process that are ripe for further improvement. The interviews and discussions provided both managers and practitioners with an opportunity to reflect on and document their activities and outcomes—actions often not taken when staff members are busy.

The three case study organizations had several commonalities. All three are large companies (between

14,000 and 300,000 employees), and all use a large number of applications and products (some home-grown, others purchased). All three companies have more than one business location.

The organizations also had significant differences. The three companies have very different customers, including federal and private. More important, they prioritize security and usability very differently. The organizations ranged from a “security first” corporate culture with a low tolerance for deliberate security violations to one in which security isn’t typically the initial focus of

each business unit; in the latter, a drop in product sales drove a focus on usability.

Data Collection and Analysis

The research team made a two-day visit to each organization's site to learn both why and how the security was designed to be usable and how the product's security and usability were evaluated. We questioned representative software developers, product managers, and senior managers using interviews, so that no staff member's work was interrupted for more than an hour or two. Some staff members were asked follow-up questions by email or telephone. An important part of the case study methodology is the fact that we intentionally didn't define usability or security for the interviewees so that we could get a sense of how they perceived those characteristics. Instead, we asked them to provide criteria for successful security and usability. In addition, their responses to interview questions revealed their working definitions and assumptions, especially concerning usability.

At least three team members took detailed notes at each interview. Each team member read every set of notes, highlighted the important statements, identified key actors and processes, and resolved differences among datasets for each interviewee. Then, the reconciled notes were qualitatively coded independently by three team members and analyzed, using Atlas.ti software, to assist in identifying relationships across concepts and interviews. *Qualitative coding* is a methodology in the behavioral sciences that refers to the process of reading text and then assigning labels to words and phrases to classify concepts and assign meaning. Coding had at least two steps: assigning initial labels to text, and then reviewing the codes to focus the descriptors and extract meaning. For example, the second, more focused step might eliminate some codes, combine others into categories, and identify links or themes that connect some codes to others. The goal is to discover relationships that wouldn't likely be found simply by reading the text.

Then, one organization at a time, we used the coding results for each organization to evaluate key factors contributing to some aspect of usable security. We sent findings to each organization separately, so that its representatives could correct any errors in understanding, transcription, attribution, or fact. Because of the sensitivity of these individual organizational findings, we don't discuss them individually in detail. Rather, analysis focuses on the data across organizations to identify patterns that improve usable security or inhibit its success.

Cross-Case Findings

Although each case study is designed to be pursued on

its own, the research team created a consistent methodology to enable multiple-case study analysis. A multiple-case comparison not only strengthens results when commonalities are identified but can also suggest new hypotheses based on variation in context and results. Across cases, the focus was on why and how case results differed or were the same.

The three organizations agreed to participate in the case studies because, at some point in their business processes, each not only recognized the importance of usable security but also made changes to try to improve its security products' usability. The analysis revealed additional important commonalities:

- each had small development teams, even though all three were large companies;
- each focused mainly on making the remote access process or the access review and revocation process more usable;
- each had an agile-inspired, informal development process, in which developers followed the spirit though not necessarily the letter of a particular agile process;
- the staff in each organization had different ideas about what usability is; and
- the staff in each organization had different ideas about the relationship between usability and security.

None of the organizations had defined criteria or measurements for usability or security. They didn't perform formal usability testing, so the teams couldn't assess if usability really improved for the users they were specifically targeting. Similarly, there was little or no formal security evaluation, so the teams also couldn't assess if better usability improved security. Most interviewees stated that they intended to conduct security evaluations (and some actually thought they had) but were unable to articulate any such process when asked. In addition, no organization used business modeling to determine if investment in usable security would pay off, and in what ways. But even without this key information, several important conclusions can be drawn.

Pleasure versus Pain

Each organization responded more to pain than to pleasure; that is, in each organization, usability was mostly a "grudge sale," in which managers responded more to financial pressures—reduced sales, increased competition, or escalating cost of use—than to reasoned arguments about the need for usability. The pain was usually evidenced by user complaints (internal or external) from large numbers of customers or from influential individuals, such as corporate executive officers who couldn't use an application. Most often, usable security was seen as an optional cost, not as an opportunity or

a competitive advantage. As one interviewee summarized, “Usability groups are available to anyone who has money to support them. We usually pass on the costs to our customers. We let the customers decide whether usability is important enough to pay for.”

Economics and Incentives

None of the case study organizations were motivated simply by providing usable security as a dimension of quality and as the right thing to do in building a system. Instead, based on the research team’s observations, economics and incentives were the key factors that initiated a push for usable security. Because those who deliver secure applications with poor usability generally don’t bear the resulting cost, complaints about unusable security are relayed to developers and then often ignored. Moreover, additional budget isn’t allocated to development for usability unless it affects the organization in a big way. Thus, addressing, measuring, and tracking these costs is necessary but very difficult. There’s little reward for usable security, and there are few consequences for bad usability or failed attempts at usable security. In other words, developers simply don’t feel the pain inflicted by bad usability.

Differing Definitions

When asked what *usability* really means, respondents offered radically different definitions. Some equated usability with *productivity*: fewer keystrokes leading to lower cost, or novel features leading to business improvements. Some defined usability in terms of *access* for new users, devices, or platforms. Only a few individuals thought about usability as a way of increasing security: fewer user errors or increased willingness to comply with security rules. One interviewee described a seamless user experience in which security doesn’t get in the way: “Users need to get to the resources they need, and do what they want and when without thinking.” These differences in understanding mean that consistent measurement of improvements in usability can be daunting, and there’s no one-size-fits-all metric to apply.

This diverse view of what usable security means was exacerbated by the dearth of developers who realize that delivering usable security goes deeper than the interface. Developers interviewed rarely had any understanding of

- general user characteristics, such as the capabilities and limitations of human perception and cognition;
- their target users’ primary tasks and the associated performance constraints; or
- the context of use (physical, social, and temporal) of security applications or processes.

Consequently, they had no idea how a particular security solution might affect an individual or an organization.

Achieved Usability

Another finding is that usable security wasn’t always achieved. Without clear evaluation criteria or anything measurable, “usable security” was in the eye of each individual beholder. Even within the same organization, different participants had different ideas of what this meant.

Delivery

Even when they understood what usability might look like, developers didn’t always see a need to deliver it. In particular, developers didn’t understand the impact of lack of usability on individual performance and well-being, organizational productivity, or the effectiveness of security. For example, one interviewee said, “This technology was built to address some of these difficulties to solve existing problems without negatively changing the user experience. Hence, input from the actual users didn’t seem relevant.” The developers confused user knowledge with their own in-depth knowledge of the product, falsely assuming that user needs were the same as “how we [the developers] think it should be used.” Many also viewed usability knowledge and methods as “common sense” (which they naturally felt they possessed in abundance), not as a specialist discipline with relevant knowledge and methods.

“Developer Knows Best”

Underlying the view of usability as common sense was the belief that “developers know best.” They considered themselves as users of the product in question, and thus saw little or no need to engage with target users. The following quotes capture this perfectly: “I can see a problem before the users see it; I can go in and troubleshoot. I use it myself,” and “A lot of our background comes from running around helping users out. That’s given us a good perspective of what users need and what their complaints will be. We kind of have a sense based on our experience.” Yet, all developers admitted that they were not taught usability in their computer science or security programs and didn’t have any on-the-job or work-sponsored training, either. Despite not knowing how a product might be used “in the wild,” developers believe they know a product well and know how to improve it. This gap in training and awareness leads the developers to focus on optimizing features they understand while ignoring potentially important improvements that could be guided by user feedback. This bias toward “developer knows best,” in addition to the belief that there is a tradeoff between usability and security, also means that many developers accept guidance only

from other developers. In particular, these developers don't value recommendations from usability specialists, who are viewed as people who "cannot code," and thus have no understanding of what would be required to deliver the designs they recommend. We found that these cultural differences were very real and common and created barriers to communication and delivery of usable security.

Discussion

The cross-case findings are fascinating in many ways, including the degree to which they reveal cultural biases hinted at in the original National Academies study. Differences in jargon and concept were only a small part of the cultural divide; differences in training and thinking led to dismissal of important user feedback.

Revisiting the three theories hypothesized in the "Methodology" section of this article, we found no support for theory 1 (the key individual)

or theory 2 (the experienced team). However, the case studies did find some support for the incentives theory: the companies were motivated to improve the usability of their security products only when it was clear that such improvement would decrease the negative consequences of the usability problems. To be specific, in one case the goal was to decrease calls to the help desk, which was staffed by the security team. In another case, the goal was to reduce the loss in market share that was due to a public negative report on the product's usability. In the third, the goal was to reduce the institutional cost of thousands of employees wasting time with unusable systems.

In other words, if usable security was important to the organization in terms of increased compliance with security policies or increased sales, then a way was found to make security more usable. Indeed, in one case, when the motivation was withdrawn, the organization reverted to less usable security. Please keep in mind that only three case studies have been completed to date, so significantly more data is necessary before these theories can be appropriately evaluated.

Increasing Usable Security

Based on the findings, at least two options appear worth exploring to potentially increase usable security.

Assign responsibility for usable security to those who can deliver it. Currently, in most organizations, the IT help desk is the first place a user goes when experiencing system problems. Thus, the help desk is taxed with the costs

of poor usable security. The problems are measured in user complaints, and the help desk's effectiveness is measured by how quickly it can remedy concerns, even when it can't fix the problem. What if the number of user complaints was a metric affecting the performance reviews of the software's designers or developers? Or, what if usable security was defined to include not only features but also lack of failures? Developers might then take increased ownership of failures in usable security and eventually take steps to design usable security into products during early development stages.

Provide training on usability and usable security—in school and on the job. Developers need a basic understanding of the complexity of human capabilities

and limitations, as well as human activity and productivity, to appreciate the complementary expertise offered by a usability expert. One developer familiar with usability said, "Focus

What if the number of user complaints was a metric affecting the performance reviews of the software's designers or developers?

groups are important. But I want to engage people and test it out because there will be ways in which they use it that are never thought of. ...The moral of the story is that I've got experience and it doesn't account for all the ways in which a user is going to use something that we haven't anticipated." Mutual respect between developers and usability experts might encourage the developers to see through different eyes and observe that unusable security isn't secure because users will find workarounds to get their primary tasks done that will reduce intended security.

Open Questions

In addition, our cross-case findings suggest several important open questions that need answers if usable security is to be respected and accepted by developers.

Is motivation more important than process? We began this study expecting to find places in the software development process where usability knowledge and processes could be best inserted. But what if a solution is not only about process, or even mainly about process? The role of process is to make sure people do what their organization wants, regardless of personal motivation. What we saw is that textbook or mandated software development processes weren't followed routinely or even valued; their adoption occurred only when developers valued them and were personally motivated to use them. The three case study organizations were motivated by time or cost pressures, which caused them

to avoid usability processes. However, poor usability meant reduced access and thus reduced productivity for users. Poor usability led to inefficiencies, and one aspect of usability is efficient use of user time, especially when the task is secondary. A senior development team member recognized that, consequently, poor usability meant sales were dropping. So, what is the role of motivation in encouraging consideration of usable security during system design, development, and sustainment and how do we determine what those motivations are?

“Today’s organizations try to improve usability by framing security choices in terms of risk rather than compliance.”

How do we overcome cultural barriers? The case studies identified and described barriers among development, security, and usability experts, but there are also barriers among different parts of a company that have different perceptions of the value of usable security. For example, developers might not value usable security, but sales and marketing staff could understand its value in distinguishing the product or service in the marketplace. Perhaps a shared lexicon about usability, productivity, security, and related concepts would reduce translation errors in communication among these experts.

How do we encourage developers to value usability when they don’t bear the cost of not addressing usable security? Although organizations must clearly specify the usable security requirements, that isn’t enough. To developers, usability often is considered only after functionality and security. What if the value of usable security were monetized so that the cost of putting usability analysis into the process is weighed against the expense of the help desk support needed when products aren’t usable? In so doing, creating a business case for usable security might help. An organization can motivate developers by providing incentives or disincentives, or it can implement a process that ensures that usability is considered, regardless of whether people are motivated. If developers knew there was a formal usability evaluation, and products not meeting the usability threshold wouldn’t get released, they would very likely pay a lot more attention to the test criteria.

Does risk-based security make security more usable than compliance-based security? When reviewing the appropriateness of assigned system accesses, some managers focus their review only on those systems or data exhibiting the most operational, reputational, or financial risk. This attempt at efficiency replaces early, more straightforward approaches in which each manager is

required to do a complete review of everyone’s access to everything. In other words, today’s organizations try to improve usability by framing security choices in terms of risk rather than compliance. The advantages go beyond reducing the size of the set needing scrutiny. Some organizational managers argued that risk-based security is more flexible and can more quickly handle changes

to software development requirements. It’s not clear whether this risk-based approach actually improves security. What’s clear is that compliance increases: managers are most likely

to complete the review if they have a smaller set of accesses to evaluate. What are some necessary and sufficient actions that will increase security in a risk-based system? That is, how small can the size of the evaluated accesses be so that it encourages compliance, and how small is so small that it sacrifices security?

Lessons Learned and a Call for More Case Studies

At the beginning of the project, we documented several hypotheses about usable security to visit after analysis and see what had been learned. Based on the data findings, three of these hypotheses are false:

- Usability is common sense; no experts needed.
- If we make a product harder to use, then it’s more secure.
- There’s always a tradeoff between usability and security.

However, the data collected leaves uncertainty about the truth or falsity of one hypothesis, which therefore needs further investigation: usable security is expensive. Measures of impact are needed to decide if this hypothesis is true or false.

Some of these hypotheses might be true in some situations and false in others. Indeed, each of the three motivators for usable security posited at the beginning of our studies (key individual, experienced team, and incentives) could well be important to some extent in every organization. Although our three cases are too few for generalization, a larger corpus of case studies based on the same methodology is likely to suggest whether, when, and how these and other characteristics contribute to usable security.

Replicability

To perform the case studies, we purposely defined a methodology for three types of interviews that other

organizations can use at least to evaluate themselves, if not to enable other researchers to evaluate other organizations and compare the results with other studies in the corpus. Some clear lessons learned can be drawn from the creation, application, and evaluation of these three case studies that will be useful as other researchers apply the methodology to other organizations.

Legal Concerns

In finding willing organizations to study, legal concerns kept several organizations away. We approached a handful of organizations that had clearly been successful in making their security applications more usable. At the development and product levels, staff members were eager to share their stories. But the organization's legal staff was concerned that important corporate intellectual property or secrets might be revealed, so permission to proceed wasn't granted. The lesson? Make sure to talk to someone senior about the benefits of the research so he or she directs the lawyers to make it happen and manage the associated risks.

Costs

In finding willing organizations to study, time and cost were major concerns. The organizations donated significant amounts of time from expensive individuals, and in some cases, real money for travel. It's imperative to make a strong case for the benefits to the organizations of the study and to provide them with valuable feedback.

Diversity

In building a case study team, having diverse backgrounds of team members is a strength. Our research team included several types of behavioral scientists, software engineering experts, and usability experts. Their different perspectives enriched discussions and strengthened the methodological approach.

Interviews

In the organizations being studied, having a single point of contact is helpful in setting up interviews, finding the right people and projects, and arranging logistics. Many interviews must be done remotely, so reliable technology is essential to enable both the researchers and the interviewees to concentrate on the interview content. Be aware, though, that project demands often trump research demands, so to keep interviews short, repeat visits or follow-up questions might be needed. Finally, different cultures are interacting, so it's important to obtain interviewees' background information (education, training, and experience), which can help to explain what's observed.

In applying the methodology, get background information after the interviews, rather than before. Interviewees usually have limited time to spend in an interview, so the background information can be gathered through paper and electronic means after the fact. In addition, interviewees were more interested in sharing their background after they met with the research team and had a positive interaction that built trust.

Piloting

As we noted earlier, piloting is extremely important, even if only to use the methodology to learn how it works, what works, and what doesn't work well. The pilot study in this project led to numerous significant improvements in the interview questions.

Focus

It's important to be clear from the start that the study relates to only one security product; gathering data on more than one product can introduce too many variables.

Other Considerations

A review of the findings also suggests that other variables, such as team size and organizational culture, should be identified and used to scope out where additional case studies should be done. Barbara Kitchenham and her colleagues point out that case studies sample from the variables (rather than over the variables, as experiments do), so additional cases are more useful when they increase the representativeness of the corpus of organizations studied.¹² These additional case studies should use the same methodology, enabling more effective cross-case analyses. The goal is to increase knowledge and understanding as this corpus grows. Based on the findings from these three case studies, future case studies that could add tremendous value to a corpus would include smaller companies, larger development teams, government or nonprofit organizations, and nonaccess control products.

One valuable addition to the executed design would be formal testing to determine whether the resulting products improved in usability and security. We requested the products for such testing, but these organizations were concerned about their intellectual property and weren't willing to hand over the products for independent testing. Instead, we had to rely on demonstrations and specifically ask "how do you know" questions when the organizations claimed improved usability. This is a good example of how a theoretical study design must be modified to reflect organizational constraints. Future case studies should continue to request formal testing. ■

Acknowledgments

This work was prepared under cooperative agreement 70NANB11H169 from the National Institute of Standards and Technology (NIST) and the US Department of Commerce, and under award 2006-CS-001-000001 from the US Department of Homeland Security. The statements, findings, conclusions, and recommendations are those of the authors and do not necessarily reflect the views of NIST, the US Department of Commerce, or the US Department of Homeland Security. In addition to the authors, researchers on this project include Richard Pietravalle (MITRE) and Laura McNamara (Sandia National Laboratory).

References

1. National Research Council, *Toward Better Usability, Security, and Privacy of Information Technology: Report of a Workshop*, Nat'l Academies Press, 2010.
2. S.L. Pfleeger and D.D. Caputo, "Leveraging Behavioral Science to Mitigate Cyber Security Risk," *Computers & Security*, vol. 31, no. 4, 2012, pp. 597–611.
3. M.A. Sasse, S. Brostoff, and D. Weirich, "Transforming the 'Weakest Link': A Human/Computer Interaction Approach to Usable and Effective Security," *BT Technology J.*, vol. 19, no. 3, 2001, pp. 122–131.
4. S.L. Pfleeger, Draft report on NIST Workshop, 2011; upon request of program manager mary.theofanos@nist.gov.
5. A. Beautement, M.A. Sasse, and M. Wonham, "The Compliance Budget: Managing Security Behaviour in Organisations," *Proc. 2008 Workshop New Security Paradigms (NSPW08)*, 2008, pp. 47–58.
6. B. Nardi, ed., *Context and Consciousness: Activity Theory and Human-Computer Interaction*, MIT Press, 1996.
7. Y. Engeström, "Innovative Learning in Work Teams: Analysing Cycles of Knowledge Creation in Practice," *Perspectives on Activity Theory*, Cambridge Univ. Press, 1999, pp. 377–406.
8. ISO 9241-210: *Ergonomics of Human-System Interaction—Part 210: Human-Centered Design for Interactive Systems*, Int'l Organization for Standardization, 2010; www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=52075.
9. D.A. Grier, "The Tenor of Our Times," *Computer*, vol. 46, no. 8, 2013, p. 128.
10. M.H. Agar, *The Professional Stranger: An Informal Introduction to Ethnography*, 2nd ed., Academic Press, 1996.
11. S.L. Pfleeger et al., "Studying Usable Security: How to Design and Conduct Case Studies," submitted for publication to *Computers & Security*.
12. B. Kitchenham, L. Pickard, and S.L. Pfleeger, "Case Studies for Method and Tool Evaluation," *IEEE Software*, vol. 12, no. 4, 1995, pp. 52–62.

Deanna D. Caputo is a principal social psychologist at the MITRE Corporation. Her research interests include

the intersection of behavioral science and computer science, such as insider threats, cybersecurity, and effective ways to change behavior. Caputo has a PhD in social and personality psychology from Cornell University. Contact her at dcaputo@mitre.org.


Shari Lawrence Pfleeger was the project manager for this usable security project. Pfleeger has a PhD in information technology from George Mason University. She is now retired. Contact her at shari@pfleeger.com.

M. Angela Sasse is a professor at University College London. Sasse has a PhD in computer science from the University of Birmingham. Contact her at a.sasse@cs.ucl.ac.uk.

Paul Ammann is an associate professor in the Department of Computer Science at George Mason University. His research interests include software testing, usability, and security. Amman has a PhD in computer science from University of Virginia. Contact him at pammann@gmu.edu.

Jeff Offutt is a full professor of software engineering at George Mason University. His research interests include test automation, Web application engineering, mutation testing, and critical software. He is editor in chief of Wiley's Software Testing, Verification, and Reliability and coauthor of Introduction to Software Testing. Offutt has a PhD in information and computer science from the Georgia Institute of Technology. Contact him at offutt@gmu.edu.

Lin Deng is a PhD candidate in information technology at George Mason University. His research interests include testing mobile applications. Contact him at ldeng2@gmu.edu.

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.