Two Experiences Designing for Effective Security

Rogério de Paula, Xianghua Ding, Paul Dourish, Kari Nies, Ben Pillet,
David Redmiles, Jie Ren, Jennifer Rode and Roberto Silva Filho

Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3425

{depaula, dingx, jpd, kari, bpillet, redmiles, jie, jen, rsilvafi}@ics.uci.edu

ABSTRACT

In our research, we have been concerned with the question of how to make relevant features of security situations visible to users in order to allow them to make informed decisions regarding potential privacy and security problems, as well as regarding potential implications of their actions. To this end, we have designed technical infrastructures that make visible the configurations, activities, and implications of available security mechanisms. This thus allows users to make informed choices and take coordinated and appropriate actions when necessary. This work differs from the more traditional security usability work in that our focus is not only on the usability of security mechanism (e.g., the ease-of-use of an access control interface), but how security can manifest itself as part of people's interactions with and through information systems (i.e., how people experience and interpret privacy and security situations, and are enabled or constrained by existing technological mechanisms to act appropriately). In this paper, we report our experiences designing, developing, and testing two technical infrastructures for supporting this approach for usable security.

Keywords

Privacy practices, effective security, theoretical security, usable security, visualization, event-based architecture, peer-to-peer file-sharing application, YANCEES, Vavoom, Impromptu.

1. INTRODUCTION

Networked computer systems have become critical not only for conventional commercial and financial transactions, but for ad-hoc informal, social interactions. These two uses of network infrastructures represent the two ends of a range of possible ways in which these infrastructures are being appropriated and utilized as sites of organizational practices as well as people's work and everyday accomplishments. By the same token, they demonstrate the range of challenges in designing and implementing usable private and secure systems. On the one hand, institutions are concerned with the protection of their sensitive information, and consequently are shaping the creation of new technologies, and policies (internal and otherwise) as well, to guarantee high levels of secure transactions. On the other hand, individuals, often workers of these institutions, are primarily concerned with getting the work done, which often involves collaborating with other individuals, disclosing information, and exchanging documents in effectively safe manners. This poses a tension in the implementation of usable, secure systems – the need for systems that are more secure and those that are more useful and trustable.

One major challenge in designing private and secure system rests on the dualism between protection and disclosure (Palen and Dourish, 2003). Sensitive information is regarded as assets that should be protected in order to ensure institutions' business advantages, but at the same time commercial and financial transactions involve collaboration and consequently the disclosure of this information, which might unintentionally expose valuable assets to other parties. For the most part, this problem of making private information secure has been tackled through the development of "stronger" and more reliable encryption, access control, and intrusion detection mechanisms – the user interface thus becomes the way through which users configure these mechanisms.

Dourish et al. (2004) conducted an empirical investigation of people's attitudes and practices around security and observed that they display considerable concerns about security as well. These concerns manifested in various ways – concern over disclosure of sensitive information, concern about viruses, concerns about hackers, marketers, and other threats – but at the same time they demonstrated frustration at the difficult of managing information and the ways that clumsy security interfaces prioritize computer and information protection at the cost of interfering with their work.

We have argued that the critical problem of usable security is not simply about designing more secure systems and infrastructures, but designing useful and trustworthy system that are *effectively secure*, rather than *theoretically secure* (Dourish and Redmiles, 2002). The levels of effective security are almost always lower than those of theoretical security because effective security represents the levels that can practically be achieved in everyday settings, rather than those that are technologically feasible. This distinction is the cornerstone to a clearer understanding of the theoretical and design approaches introduced and discussed in this paper – in particular our argument that usable security is not only about the “usability” of security mechanisms, but a broader concern with the ways in which users experience privacy and security (of and through a technology) in everyday life (Dourish and Anderson, 2005).

Hence, we are concerned with the question of how to make relevant features of a security situation visible to users so as to allow them to make informed decisions about potential privacy and security problems, and about their actions and potential implications of these actions. That is, we are interested in how to improve privacy and security by creating conditions for users to recognize and understand privacy and security situations, make informed decisions, and act accordingly. This has led to the design of technical infrastructures that make visible the configurations, activities, and implications of available security mechanisms to allow users to make informed choices and take coordinated and appropriate actions when necessary. The main contribution of this paper is thus to report on our experience in putting these theoretical concerns into practice.

Toward this goal, we focused on two design explorations that combine three major design principles – visualization mechanisms, multi-source event-monitoring architecture, and the integration of action and configuration. The first is an event monitoring application, based on visualization and event-based architectures. It uses a fairly simple network monitor to allow users see network activities when they connect to a web site. The second is a graphical interface for ad-hoc face-to-face collaborative activities. It is a peer-to-peer file-sharing application based on the principles of visualization and integration of action and configuration. It allows users to directly visualize everyone connected to the local network (the same subnet), the files being shared and their degree of sharing, and the operations being performed on these files, as well as to easily control the degree of sharing of their files. These are the first steps toward better understanding privacy and security as practical concerns, and the implications to the design of usable security.

This paper thus offers a detailed description of those two software applications designed to address the major goal of improving privacy and security by making security and privacy features “apparent” rather than “transparent.” The overall experience has helped us attain a deeper understanding of the challenges in designing usable private and secure infrastructures as well as the challenges of conducting privacy and security studies – both will be further discussed in the paper.

2. THEORETICAL APPROACH

Our theoretical approach draws empirical investigation into everyday security practices (Dourish et al., 2004), which looked at how people manage security as a practical, day-to-day concern, and exploring the context in which security decisions are made. It looked at a wide range of computer users, with different needs and working in different settings. A number of common issues arose in our interviews – a broad summary would be that “security,” both as a need and a practice, extends beyond the domain of the computer system itself. This empirical work thus provided a foundation for our reconsideration of the problems of security to a large degree as an interactional and situated problem – privacy and security concerns are created and interpreted, for example, by the interplay among one’s sharing needs and goals, the technology at hand (and one’s understanding of it), and the physical and social settings.

2.1 Security in the Wild

In everyday settings, we can assess and control security more easily and more flexibly. Participation in everyday social life inevitably involves disclosing information. When we walk down the street, we disclose our presence to others; when we talk in public, we run the risk that others will overhear us. However, we can choose which path to follow, and modulate our speech so that we are not overheard. We have ways to understand how our information is being disclosed, and to whom, and ways of managing or controlling that disclosure. In particular, security management in the everyday world displays the following properties:

- Everyday access management is *continuous*. That is, rather than being restricted to discrete choices for information management, in the everyday environment people can manage their degree of information disclosure or withdrawal along a continuum.
- Everyday access management is *continual*. That is, rather than being restricted to particular moments in time, it is being constantly monitored, controlled, and adjusted to the circumstances in which we find ourselves. The content of a conversation with a colleague may change our ideas of what is appropriate to disclose.
- Everyday access management is *co-extensive*. That is, the mechanisms by which information is shared are the same as those by which information is withheld, and the process of controlling disclosure is unobtrusively part of the work that people are already engaged in. There is no separation between providing information to others and describing information to be kept private.
- Everyday access management is *coherent*. Although we may be carrying out many tasks, and available through many channels (visual, auditory, etc), we can achieve a coherent effect. In contrast, even though our electronic identity is frequently distributed across many devices (e.g. laptop, handheld, cellphone), technological solutions treat each device or network component individually.
- Everyday access management is *contingent*. The actual requirements on disclosing and withholding information depend on the circumstances in which people find themselves, the settings in which they are working, and even the context of the conversation that takes place amongst them. While we can specify some of this in advance, the actual details are negotiated in the moment.
- Everyday access management is *comprehensible*. The structure of the world and our place within it is available at-a-glance.

Technological solutions in general lack these properties. The difficult of balancing sharing with privacy has two consequences to the design of collaborative systems. First, it limits participation – Sheehan (2002) documents non-participation in electronic interaction because of privacy concerns. Second, it may undermine participation because of inadvertent information disclosure – Good and Krekelberg (2003) detail the high degree of accidental information sharing in peer-to-peer file-sharing applications.

This discrepancy between the everyday face-to-face interactions and those mediated by technology can be attributed to two features of conventional design approaches – the temporal, spatial, and functional separations between security configuration and information sharing interfaces, and the separation between underlying architecture and the interface (the hiding approach for dealing with complex interactions and systems).

2.2 Usability, Security, and Something Else

Interface design is as critical as anything else in making the complexity of security work. Whitten and Tygar (1999) present a usability analysis of PGP 5.0, demonstrating the difficulties that users have in completing experimental tasks. The problems that they uncovered were largely problems of interface design, and in particular the poor matching between user needs and the structure of the encryption technology provided to meet these needs. Zurko and Simon (1996) explore similar concerns in their focus on "user-centered security". They are concerned that the inscrutability of conventional security mechanisms makes it less likely that users will employ them effectively. The approach they outline focuses on graphical interfaces and query mechanisms to MAP, an authorization engine.

Our concern is not, however, simply with the usability of security mechanisms, but more broadly on how security can manifest itself as part of people's interactions with and through information systems. Usability researchers have long argued that "usability" cannot be an afterthought in information system design. Security researchers have made the same argument about the design of secure systems; insecure systems cannot be turned into secure ones merely by the addition of a layer of encryption. Both of these argue, then, that security and usability need to be understood as a holistic design problem.

A holist design approach includes considerations of technical nature as well as those of social, institutional, and political ones. For example, one important related topic has been control over the degree of security available. One of our criticisms of traditional security systems has been their "all or nothing" approach. However, there has been some work that attempts to characterize degrees of security provision, as embodied by the idea of "quality of security service" (Irvine and Levin, 2001; Spyropoulou et al., 2000). This builds on earlier work establishing a taxonomy of security service levels (Irvine and Levin, 1999). The fundamental insight is that organizations and applications need to trade-off different factors against each other, including security of various forms and degrees, in order to make effective use of available resources (Thomsen and Denz, 1997; Henning, 1999). While this work is directed towards resource management rather than user control, it begins to unpack the "security" black box and characterize degrees and qualities of security. In a work on privacy and security issues in a highlight institutionalized institution, de Paula (de Paula, 2004) showed that privacy and security concerns may arise from users' perception of risks in using a technology that was shaped by existing tensions between different social groups, organizational norms, and privacy policies.

In all these respects, we have postulated that studies on usable security should not be limited to a focus on improving the usability of security mechanism, but should also explore new design approaches in which security and privacy is understood and performed by users when carrying out their everyday practices. In this respect, issues of privacy and security are contingent not only on security mechanisms embedded on a technology, but also on users' needs and goals, and the sociotechnical context in which their activities take place. In addition, we focus on privacy and security issues as experienced by users in everyday, ad-hoc face-to-face interactions with and through technology, rather than as experienced and defined by network administrators (which in itself is an important and complex issue to be further studied).

2.3 Usable Security Design and a Motivating Problem

Addressing privacy and security as a holist design encouraged us to think more broadly about existing problems users face nowadays as new networked, mobile systems start to permeate the various aspects of our everyday life. For example, we have been interested in the problem of sharing documents in collaborative, often ad-hoc practices. We see it not only relevant to distributed collaborative practices but also to everyday ad-hoc face-to-face encounters. For example, with the widespread deployment of wireless networking infrastructures, and the ubiquitous use of mobile and handheld devices, people increasingly exchange documents, multimedia files, and personal information through peer-to-peer (P2P) file-sharing applications. Studies show major problems in these systems – users find them difficult to understand, configure, and use, which often leads to the high degree of accidental document sharing and inadvertent disclosure of private information (Good and Krekelberg 2003; Sheehan 2002).

Traditionally, the goal of interface design and usability practices for security mechanism has been to make the architectural complexity invisible and control and configuration interfaces easier to use. These practices play critical roles in isolating the system complexity from users, and creating interfaces for them to effectively and efficiently interact the system, which for the most part attempts to take all necessary actions itself to protect those users. Such approaches that attempt to make the provision of system security "automatic" or "transparent" essentially remove security from the domain of the end-user. However, in situations where only the end user can determine the appropriate use of information or the necessary levels of security, then this explicit disempowerment becomes problematic.

We believe that these approaches are insufficient or inadequate to deal with privacy and security problems. The more the architecture disappears into the background, the harder it is for users to understand the implications of their actions and the extent to which these actions may cause privacy and security problems. As a consequence, when unexpected events occur (e.g., an unexpected user joins the network and accesses shared private information), users are often unable to make prompt and informed decisions and take appropriate actions. In addition, these approaches create unnecessary and ineffective disconnections (temporal, spatial, and functional) between security configuration (e.g., setting file permissions) and sharing interfaces, which induce users to make premature, uninformed decisions, or to inadvertently disclose private information in a sharing situation. For example, users are often required to define critical privacy and security parameters of a P2P application, for example, such as sharing directories and control access, at the installation of the application; or they are required to shift to a configuration/preference dialog-box to specify certain privacy or security policies, when they are not required to turn to the operating system for doing so.

Our approach is concerned with the extent to which networking systems, such as P2P file-sharing applications, can more effectively support users' privacy and security by exposing certain features of the architecture to allow users to observe and understand certain events on the network, take appropriate actions, and better perceive the implications of their actions.

We believe that the extent to which people understand the privacy and security issues and the implications of their decisions and actions depends on the situation at hand as well as their technological frames (Orlikowski, 1994) – privacy and security concerns, from our perspective, is not a manner of actual levels of technological protection, but a judgment of risks and payoffs based on people's needs and goals, and their understanding of the available sociotechnical infrastructure available to support them. In this respect, we have reconsidered the problem of security and privacy in large as an interactional and situated problem (de Paula et al., 2005).

3. DESIGN EXPLORATIONS

To further understand the implications of our theoretical approach, we have explored three design principles: *visualization mechanisms*, *integration of configuration and action*, and *event-based architecture*. The goal of these initial design explorations was to help us further understand how to support informed decision-making regarding privacy and security – i.e., helping users better understand the consequence of their actions by making visible different aspects of network operations, such as exchanging file or simply viewing a web site.

The underlying goal of using visualization mechanisms is to allow users to see and assess the outcomes of their actions. By providing dynamic feedback on relevant but hidden aspects of system activities, such as network traffic and configurations, people are more likely to understand the relationship between their actions and the technology configuration through which they are performed. It is noteworthy that this visualization does not take the form of network monitoring that might be employed by system administrators or network managers. Clearly, end users neither understand nor care to understand the details of network operation, and so we cannot assume this level of technical expertise. Nonetheless, we find that people can understand and appreciate the temporal and structural correlations between their activities and the system's behavior. The major challenge we face is to achieve the appropriate level of expression and description. These initial test-beds have helped start explore some of the trade-offs and challenges in designing visualization mechanisms for usable security that offer enough information for users' everyday decision-makings without overwhelming them with unnecessary technical details. The goal is then not to represent users' intent, or depict particular interpretation of privacy or security events or concerns, but to account for particular privacy and security events and help users construct valid interpretations that reflect their current privacy and security needs (i.e., the effective security).

As has been pointed out, we are primarily concerned with the ways in which people express and demonstrate security needs through everyday actions (Dourish and Anderson, 2005). The integration of configuration and action principle reflects this goal by unifying onto a single UI two interrelated activities, namely the act of sharing and controlling. Conventional interfaces separate configuration and action in both space and time, although in everyday practice they are one and the same activity. This separation manifests itself, in current operating system designs, for example, in a separation between a control panel where preferences are set, and some separate window or windows within which the activity of the system is performed. This separation is doubly problematic. Not only does it separate two coextensive forms of activity (the act of "sharing" being distributed across the preference window and the system window), but it also separates the expression of preferences from the occasion or situation in which those preferences are to be invoked.

These principles have thus informed the design of the two applications that we will present next. Our initial experiment, largely as a proof of concept, was based on two technical platforms – Vavoom and YANCEES. The goal was to understand the extent to which the use of visualization, event-based architecture, and consequently the exposing of the underlying network events and infrastructures. The second experiment focused on the design and implementation of a prototype to explore the concept of integrating action and configuration in face-to-face file-sharing activities as a way of helping users better understand system configurations and their consequence to user actions.

3.1 Events and Visualization

The first approach we took for helping users assess the situation at hand, and consequently make informed decisions rests on the visualization of events on the network that are often hidden from the end-users. As opposed to creating "agents" or "critiques" that monitor the network, attempt to detect "abnormal" events, and inform users, we decided to expose underlying activities so that users can better perceive the underlying mechanisms and activities on the network and consequently better understand the implications of their actions. To this end, we implemented a visualization engine, Vavoom, which was coupled with an event-based architecture, YANCEES, which provides a high-level notification channel. That created a test-bed for demonstrating our assumption concerning the use of visualization as usable security.

3.1.1 Vavoom

Vavoom is a visualization engine for the Java virtual machine (Dourish and Byttner, 2002). The first version of Vavoom was an extension of an open source Java virtual machine implementation, augmented to report various statistics and events about the execution of Java class files to a separate process, responsible for demultiplexing the events and feeding them to a range of components capable of visualizing aspects of the run-time behavior of the class files (e.g. their memory usage patterns, inter-class call patterns, method invocation, instance allocation, etc.) The key feature of this system was that it could visualize the behavior of arbitrary unmodified class files, including Java system components for which source was not available. Vavoom was originally developed as a pedagogical tool, but its ability to produce dynamic, real-time visualizations of Java software operation clearly made it appropriate to this application.

However, the initial version of Vavoom had a number of problems, primarily performance and compatability. These problems both stemmed from a single design problem, which was that Vavoom was tied to a particular JVM implementation. Relying on the instruction stream, it was based on the interpreted version of this JVM, which was no longer being actively maintained, as well as having obvious performance issues itself. For this project, we adopted a different implementation strategy

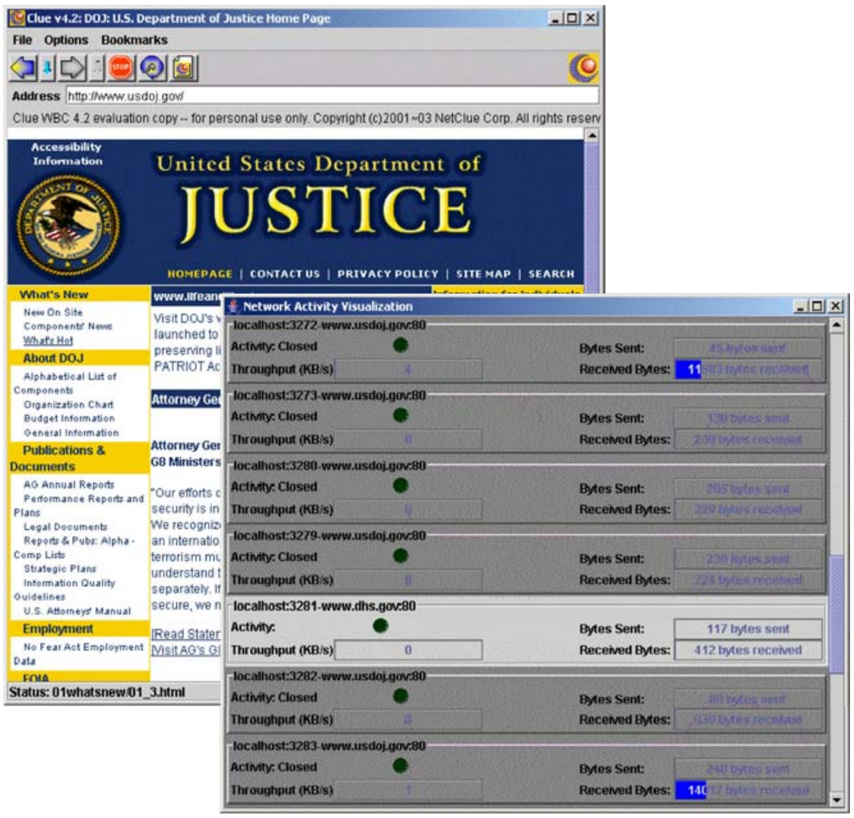
Clue v4.2: DOJ: U.S. Department of Justice Home Page
File Options Bookmarks
Address http://www.usdoj.gov/
Clue WBC 4.2 evaluation copy -- for personal use only. Copyright (c)2001~03 NetClue Corp. All rights reserv
United States Department of JUSTICE
HOMEPAGE | CONTACT US | PRIVACY POLICY | SITE MAP | SEARCH
Status: 01whatsnew/01_3.html
Network Activity Visualization
localhost:3272-www.usdoj.gov:80
localhost:3273-www.usdoj.gov:80
localhost:3280-www.usdoj.gov:80
localhost:3279-www.usdoj.gov:80
localhost:3281-www.dhs.gov:80
Activity:
Throughput (KB/s) 0
Bytes Sent: 117 bytes sent
Received Bytes: 412 bytes received
localhost:3282-www.usdoj.gov:80
localhost:3283-www.usdoj.gov:80
Activity: Closed
Throughput (KB/s)
Bytes Sent:
Received Bytes:

Figure 1: Vavoom and YANCEES - Event visualization of Department of Justice website

for Vavoom, creating a new custom classloader, which would dynamically rewrite Java byteloaders at load-time, instrumenting the class files with calls to our visualization engine. This largely preserved the features of the original Vavoom system, including the ability to work with arbitrary Java class files, even in the absence of source. More to the point, though, it made the system independent of JVM implementation, and now able to work with high-performance JVMs such as HotSpot. One problem that this new approach introduced was that the visualizer would operate only over application class files, not on the system's internal classes, which had been loaded before our classloader could be installed. This is relevant to our security concerns, but was a relatively minor consideration in the initial implementation.

3.1.2 YANCEES

The second component of our system was YANCEES. YANCEES (Silva Filho et al., 2003) is an open infrastructure for event-based publish/subscribe distributed architectures. Event-based infrastructures provide an effective mechanism for flexible, loosely-coupled distributed systems integration. Note that by event-based systems here, we refer to a particular style of distributed software architecture in which "events" are data structures that flow through a collective software bus to coordinate the activity of multiple components; this is not the event detection associated with, for instance, intrusion detection systems (Denning, 1987; Lunt and Jagannathan, 1988). YANCEES is the latest in a line of event-based architectures developed by our research group; it is a versatile infrastructure designed for flexibility in extensibility and configurability of its functionality. It also interoperates with a range of other event-based infrastructures such as CASSIUS (Kantor and Redmiles, 2001), Siena (Carzaniga et al., 2001) and Elvin (Segall and Arnold, 1997), and provides a pluggable architecture which can support additional services such as event persistence, event sequence detection, and other features that may be needed by different applications. Event architectures are particularly appropriate for our approach to usable security (Dourish and Redmiles, 2002). They provide an integration platform for sharing and visualizing "end-to-end" security-related information. Moreover, through event correlation and analysis, they can be used to detect high-level patterns arising out of sequences of low-level events.

For system developers, the YANCEES connector provides a high-level event notification channel. This connector delivers relevant events to interested subscribers. The subscription model defines a subscription language and its commands – event sequence detection and content-based filtering. The notification model expresses how these events are delivered to the subscribers, for example, using push or pull notification of events. The resource model defines where the event filtering is performed, whether in the client-side or in the server side. The event model expresses how events are represented, for example, as plain text, as objects, as attribute/value pairs or data structures; and the protocol model defines different interaction mechanisms with the server, for example, roaming protocols, federation of servers and others.

YANCEES allows programmers to implement their own subscription languages and event representations using the extensibility provided by XML (Extensible Markup Language). These languages are implemented by plug-ins installed in the publish/subscribe service. Once installed, the plug-ins are automatically loaded by YANCEES in order to service the queries posted by the clients, based on the subscriptions they post to the service. YANCEES also provides access to the main components of a publish/subscribe system (the routing engine, the publication and subscription stubs), allowing the modification and replacement of those strategic points. In other words, it provides an open implementation that allows developers to define their own extensions to the publish/subscribe model.

3.1.3 Vavoom and YANCEES Test-bed

YANCEES was used to handle all communication between system components, including between the Vavoom JVM itself and the visualization displays. In addition to the visualization displays designed as part of the initial Vavoom implementation, we created specialized displays customized to security needs, particularly focused on web browsing as our initial scenario. The focus of the proof-of-concept was the question: can we visualize network activity as part of Web browsing, so that users could become aware of the ways in which aspects of their activity might be tracked while visiting web sites?

By tracking the bytecode patterns, this prototype monitored network activity, maintaining a view of active connections and indicating when they were read or written, opened or closed.

Figure 1 shows the visualizations of various connections that were established when members of our group connected to the Department of Justice (DoJ) website. When the users visited the website, they expected that the target site would be the only site to which they would be connecting. However, as shown in the highlighted bar, the DoJ site also established a connection to the site of Department of Homeland Security. This connection is not evident to the users during a normal visit to the targeted site, but our visualization showed such "hidden" connections to the users. This visualization would help them to further assess risks and security associated with her browsing behavior.

This particular prototype was used only for demonstrations and internal activities; its function was not to be the basis of user trials, but rather to demonstrate the fundamental principle, and provide a test-bed for experimenting with implementation ideas. Although this was a very preliminary demonstration, the application was able to show its potential to uncover aspects of network activity otherwise hidden, such as the use of off-site images and “web bugs” to maintain records of web site visitor activity. By making visible the pattern of network activity that leads to a particular page rendering, this system could begin to help people understand the consequences of their actions. More than the specific application or the particular design of the visual tools, this was the initial goal.

Although this application implemented a very elementary awareness mechanism, it served the purpose of helping us “visualize” aspects as well as patterns of the network activities that are usually hidden and unexplored. It has then guided us our future development effort toward designing more elaborate and effective visualization mechanisms. For example, we are currently working on a series of visualization mechanisms for representing patterns of network activities around the concept of “normality.” By normality, we mean that we are exploring activity patterns and representations that help users build over time a “sense” of normal network activities. For example, we intend to study: what is a “usual” pattern of network connections? What is a “normal” number of connections? Or what is a “normal” connection to a specific host when accessing a particular web site, as the DoJ example showed.

3.2 Integration of Action and Configuration

Impromptu was the first prototype designed and implemented to help us evaluate the concept of integrating action and configuration within the same GUI. It is an ad-hoc file sharing application. Each Impromptu user can share files and decide how the shared files can be accessed by other users. A file can be “see-only”, which means other users will only know its existence but cannot access its content. A file can be “read-only”, where other users will be able to read its content but not modify it. A file can also be “read-write”, allowing other uses to read and modify its content. Finally, a file can be “persistent”, which means that it will still exist for read/write access even after the original owner has left the ad-hoc sharing group.

3.2.1 Interface Design

Impromptu’s interface is based on the principles of visualization and direct manipulation to give users a clear representation of “who is around,” what files are shared and in what degree, and what actions by other users are being taken at a given moment. In addition, the interface allows users to easily configure the sharing levels of their files by directly moving them in and out from the Pie GUI. In so doing, the interface integrates visualization of events (e.g., new user joining the sharing workspace, files being accessed, and files being make more or less available) and configuration (e.g., user moving files to a persistent repository before leaving the sharing workspace) –clearly implementing the concept of the integration of action and configuration. Figure 2 depicts what a user will see when Impromptu launches. The “pie” designates the entire ad-hoc file-sharing group in which each slice corresponds to a single participant’s shared area of the workspace. A participant’s own slice is represented by the darker shaded slice. The organization and orientation of this circular region are consistent for all users, so that informal references (e.g. to “left”, “right,” or “top corner”) can be oriented towards by all (Tatar et al., 1991). Files, represented by labeled dots, are placed in and around the circular region. Each area is tagged with a unique color for each user; this color is also associated with that user’s files, and with indicators of that user’s activity.

The interface is separated into multiple concentric regions; the metaphor corresponds to the idea that the closer the files are to the center, the “more shared” they are. Various degrees of sharing might be implemented. The particular mappings we have been using are that files outside the circle are not shared at all, but available to the local user; files in the outer region are visible but not readable or writable to others; files in the next region are readable but not writable; in the next, readable and writable; and in the center, readable, writable, and available persistently.

Persistent access means that, even when someone leaves the session, his or her files remain accessible to others in the group; by default, files are non-persistent, meaning that when the user leaves the session, their files will disappear from others’ interfaces. The provision of persistence serves two functions here, one pragmatic and one research-oriented. The pragmatic motivation is that persistence is a necessary feature of many of our usage scenarios (e.g. information sharing in group meetings); the research motivation is that we wanted to be sure that our different “sharing degrees” did not simply correspond to conventional file access rights. File access is managed by moving

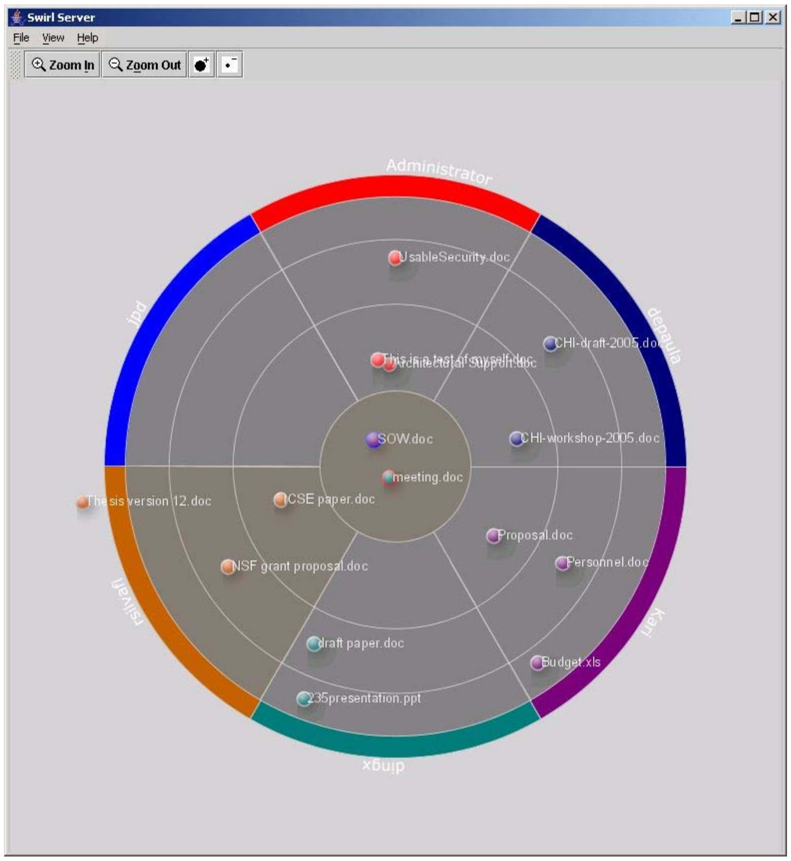

Figure 2: Impromptu interface - Six person file-sharing collaboration

the files between the levels. People can only control the accessibility of their own files.

The dots that represent files do more than simply convey the position of an object in the Pie; they also represent activities over those files. Remote file accesses to local files cause a ring around the icons for the files to blink in colors that indicate the identity of the user accessing them. This dynamic visual display draws attention to current activity and allows for a quick overview of access patterns. In so doing, it implements the concept of integration of action and configuration and of dynamic visualization of activity.

3.2.2 System Implementation

Internally, the Impromptu application consists of the following components: the graphical user interface, the Jetty web server, the Impromptu WebDAV proxy, and the Slide WebDAV repository. The secure WebDAV connector and the YANCEES event notification connector connect these components together. The architecture is depicted in Figure 3. Jetty and Slide are external open source software components. The user interface component, the proxy component, the secure WebDAV connector, and the YANCEES connector (see above) are developed by us.

Jetty serves as a dynamic application server that allows an add-on component to decide what a response will be when Jetty receives a request. Slide is such an add-on component that provides WebDAV repository support. WebDAV (Goland et al., 1999) is an HTTP extension that provides Internet-scale resource storage, retrieval, and modification capability. It is an open standard, easily available in different platforms, and is thus chosen as the foundation storage for the ad-hoc file sharing application.

Each participant stores his/her files in his/her own Slide server. However, this local storage is not directly seen by the participant. A participant only interacts with the Impromptu proxy server, using the Pie GUI depicted in Figure 2. The proxy provides an illusion of a unified, shared file storage workspace. When an Impromptu proxy receives a file operation request, it determines whether the request is directed at a local file or a remote file belonging to another participant. In the former case, it retrieves the file from the local Slide server, using a standard WebDAV request. In the latter case, it performs the operation against the remote Impromptu proxy, which will accomplish the operation using its own local Slide server.

The implementation of the GUI is based on SVG (Scalable Vector Graphics), a W3C recommendation, which defines an XML grammar for rich 2D graphics including features such as transparency, arbitrary geometry, filter effects (shadows, lighting effects, etc.), script and etc. Since the graphics are vector-based, they will not lose any quality if they are zoomed or resized, which is desirable for Impromptu to run on different devices with a variety of screen sizes. The Batik toolkit we used in our implementation enables us to generate, parse, view or convert SVG contents using Java technology. Impromptu uses then YANCEES, configured for a peer-to-peer setting, to maintain the client Pie views in sync by informing each client of events taking place on the others.

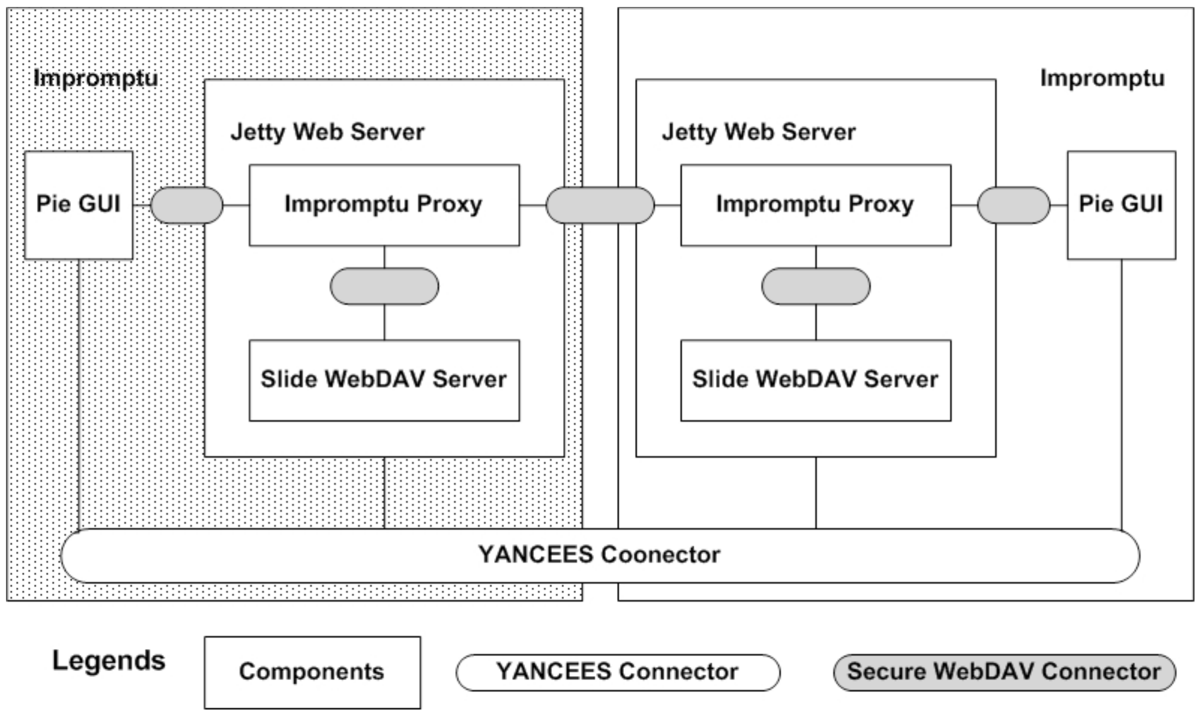

Figure 3: Impromptu Architecture

We designed this application for a relatively-security-friendly, ad-hoc file-sharing environment. The participants are not malicious, and the major risk in such an environment is unintentional disclosure of information. In traditional file sharing applications, when a user operates on files, it is not always clear to the user what files are shared, how they might be accessed and changed, and who is currently reading and changing files. However, we want to avoid requiring users to use a rather complex configuration operation to express such intentions. Such complexity might be overwhelming to the user, and thus affect usability. In summary, the security goals for the Impromptu file sharing application are 1) make security visible; 2) ease security configuration.

3.2.3 Impromptu Test-bed

We conducted a series of cognitive walkthrough activities within our research group in order to flag some initial interface problems. CW offered important information concerned the "usability" of Impromptu, but provided limited insights with respect to our primary concern with how users' privacy and security could be demonstrated through and supported by the application. We then conducted a series of informal pilot studies where we observed pairs of individuals (drawn from our department) collaborating. Each pair was working on a self-selected task. Thus, they were highly motivated to complete tasks given their realism, and tasks varied in structure and type of collaboration required. Sessions lasted one hour each. Our observations underline the necessity of integrating action and configuration, as well as the usefulness of providing a real time visualization of activity.

Four particular issues stand out from these initial usage experiences.

First, we noted that each group that tried Impromptu used it in a different way. Some groups adopted highly integrated working style, while others used Impromptu more as a means to coordinate separate activities. Some shared information to the highest degree, while others used the sharing levels more selectively.

Second, the integration of action and configuration creates a strong sense of embodiment and sharing. People respond to the shared space of the Impromptu interface as a shared and active space, and the objects within it as truly shared and seamlessly available

Third, we were pleased to see that the interface provides people with a strong sense of the presence of others. During some of our trials, we unexpectedly joined a session in progress, so that suddenly a new user would appear in the interface. This arrival was clearly visible in the interface, and was apparent to people glancing at the Impromptu window. Further, people's responses indicated that they were, first, clearly aware of the consequences for their own activities, and, second, able to take action in response by, for example, taking shared files and moving them to a read-only space to prevent the new arrival from gaining full access to the content of their work.

Fourth, we can see that it is important to understand aspects of the context in which the system may be put to use. There are two relevant contexts – a physical context and a working context. The physical context of use is face to face collaboration; Impromptu was not designed to support distance or distributed collaboration (although it does not preclude it), but rather as an adjunct to face to face work, permitting people to share information more easily than they might do using other physically co-present mechanisms (e.g. flash drives.) People talked to each other a great deal while using Impromptu, commenting on their actions, describing their plans, and of course talking about the work that they were doing. The use of Impromptu as a support, rather than a replacement, for face-to-face interaction is clearly important in the design. The working context is slightly more problematic. File sharing is rarely an end in itself; it is a means to support other working activities. Impromptu, then, is expected to be used alongside other applications. In our early trials, we noted that these other applications would sometimes obscure the Impromptu interface, making it harder to notice changes and updates. We are looking, therefore, at a range of ways of conveying information about shared activities to people, not only through a dedicated interface but also through ancillary displays that can augment other interfaces.

4. DISCUSSION AND FUTURE DIRECTIONS

Although this is an ongoing research, it is still useful to reflect on some of the experiences we gained designing, developing, and having others as well as ourselves using the applications. We have obtained promising results from these experiences that point to future design and development directions and activities. We were also able to observe some challenges in the performance of evaluative research in the areas of privacy and security in the context of everyday uses of technologies.

We have so far developed and explored the two approaches for usable security separately – visualization of network events and the integration of action and configuration – but our goal is to integrate them. For example, our informal study with Impromptu has shown that because of the seamlessness of the interface, users got drawn into their activities and were unable to perceive activities on the network, such as new users joining the workspace, when the application was hidden behind another application. So, we are now exploring new forms of awareness visualization that will allow users to perceive activities on the network. A well-discussed problem with awareness mechanisms is the tension between notification of important events and disruption. We do not intend to create yet another notification system, but to take advantage of the convergence of new wireless hand-held devices. For example, we are exploring the use of a PDA or a cell phone running Impromptu in support of activities taking place on one's notebook – while users carry out their collaborative tasks on their computer, they will be able to change sharing levels of their files through the PDA as well as monitor who else might be "around." By the same token, we are exploring the use of the environment as a situated and continuous awareness mechanism. The goal is to create mechanisms that increasingly match our everyday physical interactions with privacy and security (as described in Section 2).

The integration of the two applications will require extending and improving existing technological infrastructures. YANCEES is currently underutilized by Impromptu. The use of YANCEES as a configurable and extensible service allows the infrastructure to be customized to different requirements and permits the incorporation of future extensions that is necessary for our approach for usable security as well as the integration of various sources of awareness mechanisms. For example, YANCEES can be configured with fewer resources in order to fit in a Palm-size computer. It can also incorporate new commands in the subscription language, such as sequence detection, abstraction and other event processing features, allowing a better manipulation of events, which may be necessary for more sophisticated clients.

Impromptu at the moment implements a single security model – the visualization of user presence on the interface and the ability to directly control the sharing levels of files on the workspace. There may be a need in the future for implementing other models of privacy and security that will allow us to study other forms of privacy and security practices. We plan to implement different security policies and privacy models by extending YANCEES security models. This will allow us not only to further study the issues of designing effective private applications but also to evaluate the underlying technical infrastructure developed in this research project.

The ability of implementing and exposing different privacy and security configurations and policies will also allow us to further understand the implications of exposing different aspects of the underlying technical infrastructure. We intend then to further explore the ways in which users construct trust with each other as well as with the system itself.

The design and implementation of the Impromptu evaluation has also offered relevant insights about the limitations of existing standard usability evaluation methodologies and challenges in conducting studies on privacy and security practices that are mediated by networked applications. In particular, one major challenge that became clearly evident was the well-known tension between studying users in naturalistic settings and more controlled settings. This tension nevertheless assumes a more critical dimension in settings of privacy and security as we have conceptualized in this paper – an interactional and situated problem. For example, we were concerned with the extent that users were able to attune to the fact that outsiders inadvertently joined the workspace. This use of deception is more applicable in more controlled situations, such as in lab-experiments, rather than ad-hoc spontaneous interactions. Our challenge nevertheless lies in our understanding that privacy and security problems only reveal themselves in the situation in which they unfold.

5. CONCLUSION

Although technical infrastructures, such as wireless ad-hoc networks over 802.11b or Bluetooth, are becoming stable, common component of everyday interactions, their current implementations and more important the integration of these various components are still awkward, hard to use, and difficult to understand. This has serious implication to the privacy and security of these systems. Usability research and practices play a critical role in addressing this problem, as well as new ways of thinking about the problem itself and new approaches for the design of these systems.

In this paper, we offered an alternative theoretical and design approach for usable security. We started with a question: to what extent will making relevant features of security situations apparent to users allow them to make more informed decisions about potential privacy and security problems, and about their actions and potential implications of these actions?

To address that, we designed and implemented two applications that make visible the configurations, activities, and implications of available security mechanisms. The goal was to allow users to make informed choices and take coordinated and appropriate actions when necessary. This work differs from the more traditional security usability work in that our focus is not simply on the usability of security mechanism, but how security can manifest itself as part of people's interactions with and through information systems.

Our experiences designing, developing, and testing these two technical infrastructures have offered promising directions for future design, implementations, and research on usable security. In so doing, we hope to improve privacy and security, not necessarily by hiding complexity but by creating conditions for users act appropriately.

6. ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under awards 0133749, 0205724 and 0326105, and by a grant from Intel Corporation.

7. REFERENCES

Carzaniga, A., Rosenblum, D. S., and Wolf, A. L. 2001. Design and Evaluation of a Wide-Area Event Notification Service. ACM Transactions on Computer Systems, August, Vol. 19 Issue 3, pp. 332-383.

Denning, D. 1987. An Intrusion-Detection Model. IEEE Trans. Software Engineering, 13(2), 222-232.

de Paula, R. (2004). The construction of usefulness: How users and context create meaning with a social networking system. Unpublished Unpublished Ph.D. Dissertation, University of Colorado at Boulder, Boulder, CO.

de Paula, R., Ding, X., Dourish, P., Nies, K., Pillet, B., Redmiles, D., et al. (2005). In the eye of the beholder: A visualization-base approach to information system security. International Journal of Human-Computer Studies (to appear).

Dourish, P., & Anderson, K. (2005). Privacy, security… and risk and danger and secrecy and trust and identity and morality and power: Understanding collective information practices: Irvine, CA: Institute for Software Research. Technical Report UCI-ISR-05-1.

Dourish, P. and Byttner, J. (2002). A Visual Virtual Machine for Java Programs: Exploration and Early Experiences. Proceedings of the ICDMS Workshop on Visual Computing (Redwood City, CA).

Dourish, P., Grinter, R., Delgado de la Flor, J., and Joseph, M. (2004). Security in the Wild: User Strategies for Managing Security as an Everyday, Practical Problem. Personal and Ubiquitous Computing, 8(6), 391-401.

Dourish, P. and Redmiles, D. (2002). An Approach to Usable Security Based on Event Monitoring and Visualization. Proceedings of the New Security Paradigms Workshop 2002 (Virginia Beach, VA). New York: ACM.

Goland, Y., Whitehead, E., Faizi, A., Carter, S. and Jensen, D., 1999. HTTP Extensions for Distributed Authoring - WEBDAV. Internet Engineering Task Force, Internet Proposed Standard Request for Comments 2518, February.

Good, N., and Krekelberg, A. 2003. Usability and Privacy: A study of Kazaa P2P file-sharing. Proc. ACM Conf. Human Factors in Computing Systems CHI 2003 (Ft Lauderdale, FL). New York: ACM.

Henning, R. 1999. Security Service Level Agreements: Quantifiable Security for the Enterprise? New Security Paradigm Workshop (Ontario, Canada), 54-60. IEEE.

Irvine, C. and Levin, T. 1999. Towards a Taxonomy and Costing Method for Security Services. Proc. 15th Annual Computer Security Applications Conference. IEEE.

Irvine, C. and Levin, T. 2001. Quality of Security Service. Proc. ACM New Security Paradigms Workshop, 91-99.

Kantor, M., Redmiles, D. 2001. Creating an Infrastructure for Ubiquitous Awareness, Eight IFIP TC 13 Conference on Human-Computer Interaction (INTERACT 2001—Tokyo, Japan), 431-438.

Lunt, T. and Jagannathan. 1988. A Prototype Real-Time Intrusion-Detection Export System. Proc. IEEE Symposium on Security and Privacy, 59-66. New York: IEEE.

Orlikowski, W. J., & Gash, D. C. (1994). Technological frames: Making sense of information technology in organizations. *ACM Transactions on Information Systems (TOIS), 12*(2), 174-207.

Palen, L. and P. Dourish (2003). Unpacking "privacy" for a networked world. Proceedings of the SIGCHI conference on Human factors in computing systems, Ft. Lauderdale, Florida, USA, ACM Press.

Segall, B. and Arnold, D. (1997). Elvin has left the building: A publish/subscribe notification service with quenching Proceedings AUUG97 (Brisbane, Australia).

Sheehan, K. 2002. Towards a Typology of Internet Users and Online Privacy Concerns. The Information Society, 18, 21-23.

Silva Filho R. S., De Souza C. R. B., and Redmiles D. F.(2003). The Design of a Configurable, Extensible and Dynamic Notification Service. Proc. Second International Workshop on Distributed Event-Based Systems (DEBS'03),

Spyropoulou, E., Levin, T., and Irvine, C. 2000. Calculating Costs for Quality of Security Service. Proc. 16th Computer Security Applications Conference. IEEE.

Tatar, D., Foster, G., and Bobrow, D. (1991). Designing for Conversation: Lessons from Cognoter. International Journal of Man-Machine Studies, 34, 185-209.

Thomsen, D. and Denz, M. 1997. Incremental Assurance for Multilevel Applications. Proc. 13th Annual Computer Security Applications Conference. IEEE.

Whitten, A. and Tygar, J.D. 1999. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. Proc. Ninth USENIX Security Symposium.

Zurko, M.E. and Simon, R. 1996. User-Centered Security. Proc. New Security Paradigms Workshop. ACM.