An Approach to Usable Security Based on Event Monitoring and Visualization

Paul Dourish and David Redmiles
Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
USA
{jpd,redmiles}@ics.uci.edu

ABSTRACT

The thorny problem of usability has been recognized in the security community for many years, but has, so far, eluded systematic solution. We characterize the problem as a gap between theoretical and effective levels of security, and consider the characteristics of the problem. The approach we are taking focuses on visibility – how can we make relevant features of the security context apparent to users, in order to allow them to make informed decisions about their actions and the potential implications of those actions?

Keywords: Usability, mental models, visualization, event monitoring.

1. INTRODUCTION

Networked computer systems are increasingly the site of people's work and activity. So, for example, millions of ordinary citizens conduct commercial transactions over the Internet, or manage their finances and pay their bills online; companies increasingly use the Internet to connect different offices, or form virtual teams to tackle mission-critical problems through entirely "virtual" interaction; and we see increasing discussion of online voting systems for political elections.

However, these new opportunities have costs associated with them. Commercial, political and financial transactions involve disclosing sensitive information. The media regularly carry stories about hackers breaking into commercial servers, credit card fraud and identity theft. Many people are nervous about committing personal information to electronic information infrastructures. Even though modern PCs are fast enough to offer strong cryptographic guarantees and high levels of security, these concerns remain.

Participation in activities such as electronic commerce requires that people be able to trust the infrastructures that will deliver these services to them. This is not quite the same as saying that we need more secure infrastructures. We believe that it is important to separate *theoretical* security (the level of secure communication and computation that is technically feasible) from *effective* security (the level of security that can practically be achieved in everyday settings). Levels of effective security are almost always lower than those of theoretical security. There are many reasons for this disparity, including poor implementations of key security algorithms (Kelsey et al., 1998), insecure programming techniques (Wagner et al., 2000; Shankar et al., 2001), insecure protocol design (Kemmerer et al., 1994; Schneier and Mudge, 1998), and inadequate operating systems support (Ames et al., 1983; Bernaschi et al., 2000). One important cause of the disparity, though, is the extent to which users can comprehend and make effective use of security mechanisms.

New Security Paradigms Workshop '02, September 23-26, 2002, Virginia Beach, Virginia.
Copyright 2002 ACM ISBN 1-58113-598-X/02/0009 ...$5.00

2. CURRENT PROBLEMS

Although much effort has been devoted to the development of algorithms and technologies for secure communication and computation, much less time has been spent investigating how to integrate these techniques into real-world use in meaningful ways. In fact, it often seems that security and usability trade off against each other – the complexity and overhead of traditional security mechanisms are barriers to their effective deployment. A few simple examples from our own experience will illustrate:

- A research group designing a system for mobile code needed a security solution. A highly qualified academic security expert designed and implemented an elegant scheme based on SPKI/SDSI in which the system servers would determine transaction rights based on cryptographically secure certificates exchanged over an SSL RPC infrastructure. However, in actual use, this resulted in a performance reduction of 5-10X. As a result, in day-to-day use, everyone simply turned it off, rendering the system less secure than it had been in the first place.
- A research laboratory used S/Key one-time pads to allow terminal access through a firewall host. Researchers would periodically use private passwords and local client programs to generate themselves new one-time password pads. However, the system was soon discontinued when it became clear that people could not tell whether their connections were secure enough to make it safe to generate the new pads.
- Norton's Anti-Virus software offers an option to check incoming email for viruses before you download it to your computer. The actual mechanism for doing this is not directly disclosed. When this option is turned on, the user's login and password are sent to a Norton server, which downloads the user's email and reads it, checking

for viruses, before sending it on to the user. Inserting Norton's own servers as an intermediary makes great technical sense, allowing Norton to respond rapidly to new virus attacks. However, users are typically shocked to learn that their password and their email are being shared with Norton; it damages their trust in the system and in the software.

These brief examples are intended merely to be suggestive of the range of difficulties that people encounter putting security technology into practice, but they also express a number of common themes. Three are particularly important here.

First, security in practice is *not an all-or-nothing matter.* Rather, in practice, the question is not "is this system secure?" but instead, "is it secure *enough* for my current tasks?" This is a quite different question. It suggests a continual tuning of the degree of security required, and a process of matching security to task. Too much security can be as much of a problem as too little. Systems that inflexibly offer absolute security are likely as useless as those that offer none (but generally more difficult to configure and use).

Second, *visibility of mechanism* plays a critical role. If the key problem is determining whether the current configuration of systems and services is secure enough for the task at hand, then it is critically important that security features and potential threats be visible so that this determination can be made. Hidden features of infrastructure, including mechanisms designed for secure computation, are inherently unavailable for this sort of examination.

Third, security in end-user applications is *an end-to-end phenomenon*, even though it arises out of the interactions between many different components (Saltzer et al., 1981; Blumenthal and Clark, 2001). Effective security potentially depends upon each application or infrastructure component involved, as well as on the relationships between those components. Although the end-to-end element is a known issue in traditional security circles, it is particularly problematic when we consider visibility and usability as central issues for security infrastructures. When we talk of "distributed applications" or "networked applications", we mean to include not simply the application, but the entire "slice" through the infrastructure needed for that application to work – client, server, network services, protocol implementations, etc.

Our hypothesis is that a technical infrastructure which makes visible the configuration, activity, and implications of available security mechanisms will enable end users to make informed choices about their behavior; and that these informed choices, in turn, will yield more effective, more secure system use. To test this hypothesis, we are developing a "trustable" infrastructure that makes information and security policy and configuration available to end users in ways that are visible, usable, and integrated with their normal activities.

3. PREVIOUS APPROACHES

It is broadly recognized that one of the major challenges to the effective deployment of information security systems is getting people to use them correctly. Psychological acceptability is one of the design principles that Saltzer and Schroeder (1975) identify. Even beyond the domain of electronic information systems, there are many examples of the fact that overly complex security systems actually reduce effective security. For example, Kahn (1967), cited by Anderson (1993), suggests that Russian military disasters of the Second World War were partly due to the fact that Russian soldiers abandoned the official army cipher systems because they were too hard to use, and instead reverted to simpler systems that proved easier to crack. Scheiner's (2000:373) sums up the situation: "Security measures that aren't understood by and agreed to by everyone don't work."

However, despite this broad understanding of the significant relationship between security and usability, little work has been carried out in this area to date. We discuss some exceptions here.

3.1 Usability of Security Software and Mechanisms

In a series of studies, researchers at University College, London have explored some of the interactions between usability and security (Adams, Sasse and Lunt, 1997; Adams and Sasse, 1999). They focused on user-visible elements of security systems, such as passwords. Although many information systems professionals regard users as being uninterested in the security of their systems (and, indeed, likely to circumvent it by choosing poor passwords, etc), Adams and Sasse's investigations demonstrate that users are certainly motivated to support the security of the system, but often unable to determine the security implications of their actions. The specific problems that they identify with passwords have also led to interesting design alternatives (Brostoff and Sasse, 2000; Dhamija and Perrig, 2000).

In some cases, the complexity of making security work is as much a matter of interface design as anything else. Whitten and Tygar (1999) present a usability analysis of PGP 5.0, demonstrating the difficulties that users have in completing experimental tasks (in their user study, only 3 out of 12 test subjects successfully completed a standard set of tasks using PGP to encrypt and decrypt email.) 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." Like us, 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. While this approach is clearly helpful, it is limited to a particular area of system security, and lacks the real-time feedback.

3.2 Control Over Security

One area at the intersection of usability and security that has received some attention is the role of access control in interactive and collaborative systems. For example, Dewan and Shen (Shen and Dewan, 1992; Dewan and Shen, 1998) have explored the use of access control and meta-access control models as a basis for describing and controlling degrees of information access and management in collaborative systems. This is not simply a technical matter, since the structure and behavior of these "internal" components can have a significant effect on the forms of interactivity and collaboration they can support (Greenberg and Marwood, 1994).

Many collaborative systems involve privacy issues and need to provide users with control over the disclosure of information. This has spurred a number of researchers to explore the development of privacy control systems that are tailored to the needs of end users. For instance, Dourish (1993) describes the relationship between three different security mechanisms for similar multimedia communication systems, each of which reflects assumptions and requirements of the different organizations in which they were developed. Bellotti and Sellen (1993) draw on experiences with multimedia and ubiquitous computing environments to identify the source of a number of potential privacy and security problems. Their primary concepts – disembodiment and dissociation – are both visibility problems, related to the disconnection between actors and actions that renders either actors invisible at the site of action, or actions invisible to the actor.

Based on their investigations of privacy problems in online transactions, Ackerman and colleagues propose the idea of *privacy critics*, semi-autonomous agents that monitor online action and can inform users about potential privacy threats and available countermeasures (Ackerman et al., 1999; Ackerman and Cranor, 1999). Again, this mechanism turns on the ability to render invisible threats visible.

One important related topic is 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 characaterize degrees and qualities of security.

3.3 Visualizing Networked Systems

Although there has been a certain amount of research investigating ways of visualizing distributed systems structure, behavior and performance, most of this work has been aimed at system managers and operators. Systems such as Pulsar (Finkel, 1997) or Planet MBone (Munzer et al., 1996) are designed to convey information to highly technical audiences. One exception is in the System Health project (Dourish et al., 2000), which monitored the activity of complex distributed systems in order to convey some understanding of the state of the system to end-users whose work might be affected by outages, slowdowns, and other mysterious "internal" events. However, this work was directed towards fairly general characterizations of systems, rather than focusing on an issue like security. In a more focused area, we anticipate being able to apply heuristics, which can inform a more specialized interpretation of events.

4. THE VISIBLE SECURITY APPROACH

Our Visible Security approach involves bringing together a number of elements, including visualization technology, system- and interface-monitoring components, and security-specific heuristic evaluation components, to provide users with a coherent, real-time picture of the state of the system and their applications with respect to security needs. Figure 1 illustrates a layered architecture describing our approach.

The architecture is separated into four levels. At the top level are security gauges – visualization widgets that present dynamic visual representations of system state and activity. The next layer is in two parts: *security monitoring* and *event monitoring*. Together, they support the specification and monitoring of security and privacy conditions. The component for event monitoring is based on an existing system for testing and tracking user interface and system activity. The security monitoring component embodies a set of security heuristics allowing it to translate these basic event notifications into security-relevant interpretations. Like intrusion detection systems (e.g. Denning, 1987; Lunt and Jagannathan, 1988; Smaha, 1988), our monitoring agents detect relevant conditions to be communicated to users; but unlike intrusion detection systems, they integrate information across multiple machines, incorporate application knowledge, and assess the general degree of protection rather than looking for specific problematic conditions.

The question arises, what events can be monitored? The next layer of Figure 1 incorporates event notification servers as serving as a source of events for the event monitoring layer. The event monitoring layer can be adapted to various notification servers. Finally, in the last layer, we show a set of information sources. These include applications in use by the end users and they may also include agents monitoring facets of the end users' work environments that are not directly observable by monitoring events from a single application.

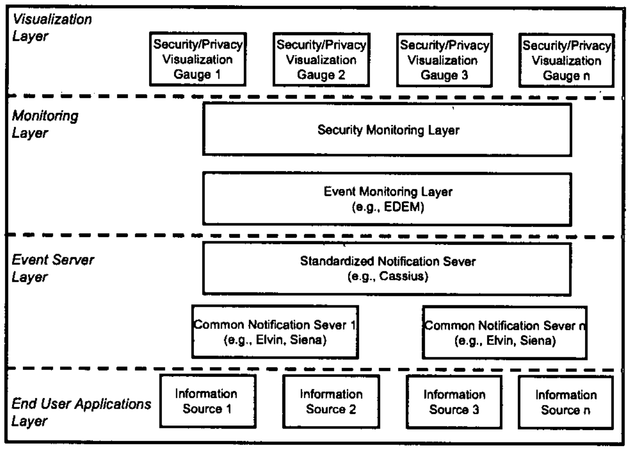
Visualization Layer
Security/Privacy Visualization Gauge 1
Security/Privacy Visualization Gauge 2
Security/Privacy Visualization Gauge 3
Security/Privacy Visualization Gauge n
Monitoring Layer
Security Monitoring Layer
Event Monitoring Layer (e.g., EDEM)
Event Server Layer
Standardized Notification Sever (e.g., Cassius)
Common Notification Sever 1 (e.g., Elvin, Siena)
Common Notification Sever n (e.g., Elvin, Siena)
End User Applications Layer
Information Source 1
Information Source 2
Information Source 3
Information Source n

Figure 1: Architecture for the Visible Security Approach

So, our architecture is designed to gather, integrate, and interpret information about security, which is distributed across a large number of systems and components; and then, to present this information as a set of real-time visual displays. We now turn to a more detailed account of the research problems to be solved.

4.1 Levels of Expression

One of the primary challenges in designing visual accounts of system security is to achieve an appropriate level of expression or description. Clearly, the visual presentations provided must be expressive enough to be useful in making security-relevant decisions. However, at the same time, it is not our goal to provide people with large amounts of information, nor do we intend to require all users to understand all the relevant technical characteristics of security in their system.

A helpful analogy can be found in driving a car. Very few competent drivers could provide a full and scientifically accurate description of the operation of the internal combustion engine (especially modern computer-controlled engines, which would require a computational as well as a mechanical account.) However, they are nonetheless able, in the course of driving, to make use of information such as the sound of the engine, the stiffness of the steering and the feel of the clutch as they drive. Coupling these sounds to actions does not require a full technical model, but relies on informal understandings, practice, and experience. Similarly, our goal is to provide people with information – visual depictions of the system's action – which they can incorporate into their assessments through practice and experience, but which do not rely on complete technical descriptions of the system's operation.

An important element of our strategy, then, is not to attempt to represent the users' intent, nor their interpretation of current threats. While user modeling approaches of this sort have achieved some degree of success in online applications such as web site personalization, we feel that the domain of user actions in networked systems is insufficiently constrained to apply this approach. Instead, our approach is to have the system present information that it can validly "talk about" – its own internal structure and action. The question to be addressed, then, is in what terms this account should be constructed.

4.2 Mental Models of Security

One sensible place to begin, at least for calibration, is with end users own accounts of system action and system state. In cognitive science, "mental models" refers to the conceptual understandings that users hold of the domains in which they operate. Actions are planned and interpreted with respect to these models. These same models can be a valuable source for design in this effort. Some existing work has begun to look into operative models of system action. For instance, Sasse and colleagues (Rimmer et al., 1999; Weirich and Sasse, 2001) present some features of the mental models supporting end users' descriptions of the behavior of networked software systems. Maglio and Matlock (1999) draw on contemporary explorations of metaphor and cognition (Lakoff, 1992) to uncover the metaphorical structure of reasoning and talking about networked system use.

Our interest in models is two-fold. First, they help articulate a level of description that makes sense, since it is close to the level in which system action is currently understood. In this, we want to identify a meaningful level of abstraction, rather than the specific details of people's understandings. Second, these identify areas where the visualizations can be used correctively. Just as in other areas, such as the oft-studied domain of "naïve physics," mental models can often be wrong (diSessa, 1983.) One area where we hope that our visualizations can help is not simply in helping people understand what their system is doing, but also in developing more accurate intuitions about the system's behavior.

Building on the existing studies of these models, we are currently investigating end users' mental models of security in information systems. Interestingly, our early results show the importance of domains beyond the technical in understanding security; accounts of the behavior of software systems and the models of security and threat assessment depend on social and organizational factors as much as technical ones. Any adequate system response must similarly integrate these elements. This investigation is ongoing.

5. CURRENT STATE

Parts of the architecture represented in Figure 1 have been instantiated and others remain open for investigation. The monitoring of information sources, distribution of interesting events via event notification servers, the detection of event patterns, and some rudimentary visualization have all been achieved in a research scenario (de Souza et al., 2002). In this previous work, components of a large distributed avionics software system were instrumented with *probes*. The probes reported events to an event notification server, CASSIUS (Kantor and Redmiles, 2001). Various single-purpose visualizations of the events were displayed in small windows and were referred to as *gauges*. In this work, gauges provided information primarily about the load of events on a system. However, other gauges provided alarms to warn of adverse sequences of events. This latter kind of gauge is one that should directly support the detection of security or privacy breaches.

The probe and gauge approach was expanded from the research event notification server, CASSIUS, to work with two more widely available event notification servers: Elvin (Fitzpatrick et al., 1999) and Siena (Carzaniga et al., 2001). This expansion was critical for at least two reasons. First, it was necessary to verify the generality of the approach. Many kinds of event notification servers exist in the world and most have their merits with respect to a particular environment. It was important to demonstrate that the probe and gauge approach could function more robustly. Second, it is becoming evident that a diverse distributed software workspace will have applications linked to specific servers already. If the Visible Security architecture outlined above is to work in the real world, it will have to be built upon an architecture of diverse, communicating servers. Thus, Figure 1 shows at least three typical servers that might be involved in an implementation of an adaptation of the probe and gauge approach for making security and privacy events more visible.

The detection of specific events plays an important role in the scenarios that the Visible Security approach should apply to. There have been a large number of languages proposed for monitoring and detecting specific event sequences. From our previous work, we intend to adapt an approach called EDEM for Expectation-Driven Event Monitoring (Hilbert and Redmiles, 1998; Hilbert and Redmiles 2001). Expectations play a key role in monitoring. Although it is useful to provide general visualizations of events for end users' awareness, these can become ignored out of habit. It is critical to be able to specify alarm conditions that perform more intrusive cues to direct end users and others to problem situations. While other researchers have focused on the theoretical expressiveness of an event language (e.g. Luckham (1998) and Cohen and colleagues (1997)), we acknowledge the need for a balance between expressiveness and pragmatism. The specification of events needs to useful but it must also be usable by end users. In EDEM, leaving out more esoteric features made it possible to have a much simpler specification of events of interest. Although EDEM provides for creating a hierarchy of event specifying different levels of abstraction, only the monitoring of software program events has been specified. In the Visible

Security approach, heuristics encoding typical security or privacy violations still need to be specified. Another area to be explored further is that EDEM allowed probes to be specified in a declarative fashion independent of an applications' implementation. The range of applications that can be automatically monitored versus those that require manual instrumentation with probes needs to be explored further.

Continuous monitoring of the state of security and privacy was emphasized earlier. In complex distributed software environments that today's end users work with, many kinds of events and information affect them. There are events about others work, such as the checking in and out of shared documents, progress toward the completion of a process, and, as noted here, events violating expectations about security and privacy. In previous work, knowing information that affected one's work was termed *awareness* (Dourish and Bellotti, 1992). The gauges approach is very compatible with this concept of awareness (Kantor and Redmiles, 2001). One of our long-term goals is to provide an environment of ubiquitous awareness, integrating security and privacy issues with other kinds of "awareness" information.

In a related investigation, we have also begun to explore the notion of dynamically-coupled visualizations of system state (Dourish and Byttner, 2002). Our initial work has concentrated on the use of dynamic visualizations of Java programs as a means to help novice programmers understand the dynamic structure of the programs they develop. As a proof-of-concept, though, this demonstrates the value of dynamic visualization of system behavior, as well as providing a testbed for applying the same techniques to different domains and different sets of users. For example, this work is currently being extended to visualization of network and file activity for end users of Java programs, in support of the research outlined here.

6. CONCLUSION

Computer and communication security has been an important research topic for decades. However, the pressing concern at the moment is not simply with advancing the state of the art in theoretical security, but with being able to incorporate powerful security technology into the kinds of networked computational environments that more and more people rely on every day. We see the problem of creating a trustable infrastructure – one that end users can see is visibly trustworthy – as a major problem for both the security and the HCI research communities.

We have described an approach that we are currently developing. This work builds on earlier investigations, by ourselves and others, into mental models of network application behavior, dynamically-coupled visualizations of system state, and event monitoring and distribution infrastructures. Our early experiences with these, and with investigations in progress into end users' mental models of system security, suggest that that this approach can provide an effective mechanism to address effective, rather than theoretical, security.

The novel paradigm we have been exploring is to conceptualize security as a practical problem that end users encounter and routinely solve in the course of their daily activity. This turns our attention from security-as-it-can be to security-as-it-is; and in making that shift, we also change the problems to be addressed. We see the fundamental problems of security as those surrounding the usability not of special-purpose-security software, but of networked applicatios and systems more generally. By addressing these problems as they occur in the real world, our research aims to advance the achievable levels of security in actual settings of use. We look forward to presenting our future results.

7. REFERENCES

[1] Ackerman, M. and Cranor, L. 1999. Privacy Critics: UI Components to Safeguard Users' Privacy. *Adjunct Proceedings of CHI'99* (Short Papers), 258-259.

[2] Ackerman, M., Cranor, L., and Reagle, J. 1999. Privacy in E-Commerce: Examining User Scenarios and Privacy Preferences. *ACM Conf. on Electronic Commerce*, 1-8. ACM.

[3] Adams, A. and Sasse, M.A. 1999. Users Are Not The Enemy: Why users compromise security mechanisms and how to take remedial measures. *Comm. ACM*, 42(12), 40-46.

[4] Adams, A., Sasse, M.A., and Lunt, P. 1997. Making Passwords Secure and Usable. In Thimbleby, H. O'Connaill, B., and Thomas, P. (eds), *People and Computers XII: Proceedings of HCI'97*, 1-19. Springer.

[5] Ames, S., Gasser, M., and Schell, R. 1983. Security Kernel Design and Implementation: An Introduction. *IEEE Computer*, 16, 7, 14-22.

[6] Anderson, R. 1993. Why Cryptosystems Fail. *Proc. ACM Conf. Computer and Communication Security CCS'93*, 215-227. ACM.

[7] Bellotti, V. and Sellen, A. 1993. Design for Privacy in Ubiquitous Computing Environments. *Proc. European Conf. Computer-Supported Cooperative Work ECSCW'93*, 77-92. Kluwer.

[8] Bernaschi, M., Gabrielli, E., and Mancini, L. 2000. Operating System Enhancements to Prevent the Misuse of System Calls. *Proc. ACM Conf. Computer and Communication Security*, 174-183. New York: ACM.

[9] Blumenthal, M. and Clark, D. 2001. Rethinking the Design of the Internet: the end-to-end arguments vs. the brave new world. *ACM Trans. Internet Technology*, 1(1), 70-109.

[10]Brostoff, S. and Sasse, M.A. 2000. Are Passfaces more usable than passwords? A field trial investigation. In S. McDonald, Y. Waern & G. Cockton (Eds.): *People and Computers XIV - Usability or Else! Proceedings of HCI 2000*, 405-424. Springer.

[11]Carzaniga, A., Rosenblum, D., and Wolf, A. 2001. Design and Evaluation of a Wide-Area Notification Service. *ACM Trans. Computer Systems*, 19(3), 332-383.

[12]Cohen, D., Feather, M., Narayanaswamy, K., Fickas, S. 1997. Automatic monitoring of software requirements. *Proceedings of the 1997 International Conference on Software Engineering, ICSE 97* (Boston, MA), 602-603.

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

[14]Dewan, P. and Shen, H. 1998. Flexible Meta Access-Control for Collaborative Applications Primitives for Building Flexibile Groupware Systems. *Proceedings of ACM Conference on Computer-Supported Cooperative Work CSCW'98*, 247-256. ACM.

[15]Dhamija, R. and Perrig, A. 2000. Deja Vu: A User Study. Using Images for Authentication. In *Proceedings of the 9th USENIX Security Symposium*, Denver, Colorado.

[16]Dourish, P. 1993. Culture and Control in a Media Space. *Proc. European Conf. Computer-Supported Cooperative Work ECSCW'93*, 125-137. Kluwer.

[17]Dourish, P. and Bellotti, V. 1992. Awareness and Coordination in Shared Workspaces. *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW'92*, 107-114. New York: ACM.

[18]Dourish, P., Swinehart, D., and Theimer, M. 2000. The Doctor is In: Helping End-Users Understand the Health of Distributed Systems. *Proc. 11th IEEE/IFIP Workshop on Distributed Systems Operation and Management DSOM 2000.* IEEE.

[19] Dourish, P. and Byttner, J. 2002. A Visual Virtual Machine for Java Programs: Exploration and Early Experiences. *Proc. ICDMS Workshop on Visual Computing* (San Francisco, CA.)

[20]Finkel, R. 1997. Pulsar: An Extensible Tool for Monitoring Large UNIX Sites. *Software Practice and Experience*, 27(10). 1163–1176.

[21] Fitzpatrick, G., T. Mansfield, et al. 1999. Augmenting the workaday world with Elvin, *Proceedings of 6th European Conference on Computer Supported Cooperative Work ECSCW'99*, 431-450. Kluwer.

[22] Greenberg, S and Marwood, D. 1994. Real-Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface. *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW'94*, 207-218. ACM.

[23]Henning, R. 2000. Security Service Level Agreements: Quantifiable Security for the Enterprise? *Proc. New Security Paradigm Workshop* (Ontario, Canada), 54-60. ACM.

[24] Hilbert, D. and Redmiles, D. 1998. An Approach to Large-Scale Collection of Application Usage Data Over the Internet, *Proceedings of the Twentieth International Conference on Software Engineering (ICSE '98)*, Kyoto, Japan), IEEE Computer Society Press, 136-145.

[25] Hilbert, D. and Redmiles, D. 2001. Large-Scale Collection of Usage Data to Inform Design, *Eighth IFIP TC 13 Conference on Human-Computer Interaction INTERACT 2001* (Tokyo, Japan), 569-576.

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

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

[28] Kahn, D. 1967. The Codebreakers. Macmillan.

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

[30]Kelsey, J., Schneier, B., Wagner, D., and Hall, C. 1998. Cryptanalytic Attacks on Pseudorandom Number Generators. *Proc. Intl. Workshop on Fast Software Encryption*, 168-188. Springer-Verlag.

[31]Kemmerer, R., Meadows, C., and Millen, J. 1994. Three Systems for Cryptographic Protocol Analysis. *Journal of Cryptology*, 7(2), 79-130.

[32]Lakoff, G. 1992. The Contemporary Theory of Metaphor. In Ortony (ed), *Metaphor and Thought (2nd Edition).* Cambridge University Press.

[33]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.

[34]Luckham, D. 1998. Rapide: a language and toolset for causal event modeling of distributed system architectures. *Proc. Second International Conference Proceedings Worldwide Computing and Its Applications - WWCA'98* (Tsukuba, Japan), 88-96.

[35]Maglio, P. and Matlock, T. 1999. The Conceptual Structure of Information Space. In Mundo, Benyon, and Hook (eds), *Social Nagivation of Information Space*, 155-173. Springer.

[36] Munzer, T., Hoffman, E., Claffy, K., and Fenner, B. 1996. Visualizing the Global Topology of the MBone. *Proc. of the Symposium on Information Visualization* (San Francisco, CA). New York: IEEE.

[37] Rimmer, J., Wakeman, I., Sheeran, L., and Sasse, M.A. 1999. Examining Users' Repertaoir of Internet Applications. In Sasse and Johnson (eds), *Human-Computer Interaction: Proceedings of Interact'99.*

[38]Saltzer, J. and Schroeder, M. 1975. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9), 1278-1308.

[39]Saltzer, J., Reed, D., and Clark, D. 1981. End-to-End Arguments in System Design. *ACM Trans. Computer Systems*, 2(4), 277-288.

[40] Schneier, B. 2000. *Secrets and Lies: Digital Security in a Networked World.* Wiley.

[41]Schneier, B. and Mudge. 1998. Cryptanalysis of Microsoft's Point-to-Point Tunnelling Protocol (PPTP). *Proc. ACM Conf. Computer and Communication Security*, 132-141. New York: ACM.

[42]di Sessa, A. 1983. Phenomenology and the Evolution of Intuition. In Gentner and Stevens (eds), Mental Models. Hillsdale, NJ: Laurence Erlbaum.

[43]Shen, H. and Dewan, P. 1992. Access Control for Collaborative Environments. *Proc. ACM Conf. Computer-Supported Cooperative Work CSCW'92*, 51-58. ACM.

[44] Smaha, S. 1988. Haystack: An Intrusion Detection System. *Proc. Aerospace Computer Security Applications Conference*, 37-44.

[45] de Souza, C., Basaveswara, S., Redmiles, D. 2002. Lessons Learned Using with Notification Servers to Support Application Awareness, Department of Information and Computer Science, University of California, Irvine, Technical Report #02-11.

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

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

[48] Wagner, D., Foster, J., Brewer, E., and Aiken, A. 2000. A First Step Towards Automated Detection of Buffer Overrun Vulnerabilities. *Proc. Networked and Distributed Systems Security Symposium*. Internet Society.

[49] Weirich, D. and Sasse, M.A. 2001. Pretty Good Persuasion: A first step towards effective password security for the Real World. *Proceedings of the New Security Paradigms Workshop 2001* (Sept. 10-13, Cloudcroft, NM), 137-143. ACM Press.

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

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