Security Automation Considered Harmful?

W. Keith Edwards Erika Shehan Poole Jennifer Stoll

School of Interactive Computing and GVU Center
Georgia Institute of Technology
85 Fifth Street NW, Atlanta, GA 30332-0280

{keith, erika}@cc.gatech.edu, jstoll@gatech.edu

ABSTRACT
End-users are often perceived as the weakest link in information security. Because of this perception, a growing body of research and commercial activity is focused on automated approaches to security. With these approaches, security decisions are removed from the hands of the users, and are placed instead in systems themselves, or in remote services or organizations that establish policies that are automatically enforced. We contend that although security automation is potentially beneficial in theory, in practice it is not a panacea for end-user information security. A number of technical and social factors mitigate against the acceptance and efficacy of automated end-user security solutions in many cases. In this paper, we present a discussion of the inherent limitations of automating security for end-users. We then discuss a set of design guidelines for choosing whether to automate end-user security systems. We conclude with a set of research directions focused on increasing the acceptance and efficacy of security solutions for end-users.

Categories and Subject Descriptors
D.4.6 [**Operating Systems**]: Security and Protection—*access controls, information flow controls*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces— *interaction styles, user-centered design*.

General Terms
Design, Security, Human Factors.

Keywords
Usable security, security policy, automation, design guidelines.

1. INTRODUCTION
In 2001 and 2004, Hannu Kari of the Helsinki University of Technology provocatively predicted that the Internet would collapse by 2006 due to being completely overrun by malicious users [19]. In other words, users' dissatisfaction with the state of security on the Internet would be so high such that they would abandon it [6]. Though Kari's prediction proved incorrect, it rings true in that many users have indeed been unhappy with their online experience. The Pew Foundation reports that 77% of email users labeled the act of being online as "unpleasant and annoying" [37].

Paradoxically, despite such intense user dissatisfaction with a range of security-related issues in the online experience, users themselves are often perceived as being the "weak link" in the information security chain, as many neglect to adopt or correctly use even the most basic security measures [41]. A growing number of researchers (see, for example, [16, 38, 39, 44, 47, 52]) have noted that this apparent neglect is a natural outcome of the orientation users have toward security *vis-à-vis* the other tasks that they must accomplish. For most users, security is not a task in itself. That is, managing their security is not a goal for users, and is often only incidental to their task at hand; consequently, their motivation to actively manage their security may be low at best.

From this perspective, the absence of users' motivation to take charge of their own online security would seem to suggest that the logical course of action is to completely *remove* such security decisions from the hands of users—in other words, to *automate* security for them by moving policy decisions to systems or expert administrators. Such automation promises not only to protect individuals' security, but also to serve the greater good by more easily preventing threats that can lead to botnet infections or distributed denial-of-service attacks.

The appeal of this thinking can be seen across both the commercial and research landscapes. For example, according to a guide for IT executives produced by NetworkWorld, "Security automation has clearly become an IT mandate, with anti-virus protection, intrusion detection and prevention, and patch management as the primary concerns" [30]. In fact some go so far as to recommend automation to the extent that all end-user devices be made non-programmable [6]. Trends such as the move toward automated identity management [23] and service-oriented security architectures (in which a paid network service provider is responsible for the security of end-user nodes) also highlight the appeal of automation. Similar support for security automation is seen in the human-computer interaction (HCI) community. Jakob Nielsen, a well known figure in the HCI community, has stated that attacks "cannot be thwarted by placing the burden on users to defend themselves at all times. Beleaguered users need protection, and the technology must change to provide this" [32].

These trends illustrate the perceived promise of automating security for end-users. In this view, security experts and systems designers have already made the security choices for end-users. Thus, by making end-user security invisible, users can pursue their primary tasks without the annoying disruption of dealing with purportedly "secondary" security tasks.

We agree that relieving end-users of the burden of security management through automation may be desirable. However, we also believe the automation approach is predicated on a set of (sometimes incorrect) assumptions about technical, social, and environmental contexts in which security decision making takes place. We therefore contend that there are inherent limitations to how well automation can succeed in practice *even if the technology behind it is faultless*. These limitations can lead to "failures" of automation that are not only technical (i.e., when the automation system simply stops working, for example), but are also failures of meeting the actual needs of the users. We further contend that such failures to meet users' needs are not simply annoying nuisances, but rather are at the heart of many of the problems described under the rubric of "usable security" [53]. We therefore ask, "To what extent is it appropriate to automate security for end-users? And in what cases does automation actually create more problems than it promises to solve?"

Our intent is not to cast stones at current or past research on automated approaches to system security, and acknowledge that automation does have a role to play. Rather, we seek to illuminate the design space of security solutions by exploring when automation may (or may not) be appropriate for end-users. While there is undoubtedly a role for automation in ensuring that security is both more usable *and* more effective for end users, we believe that the appropriate place of automation may necessarily be more limited than is commonly anticipated.

In this paper, we explore what researchers can do to shift the balance either toward or away from automation in order to yield systems that successfully negotiate the tensions between usability for end-users and security. Our approach in exploring the limitations of security automation is grounded in perspectives from the social sciences, human-computer interaction, and economic, historical and policy traditions. Automation has been studied extensively in domains other than information security, and we believe that this knowledge has applications to the design of security systems.

In the following sections, we define more precisely what we mean by "security automation," explore the different forms that automation may take, and identify innate limitations to automated security approaches for end-users. We then explore the situations for which security automation *may* be a workable approach as well as situations for which it seems inappropriate. The factors for determining appropriateness are based on: 1) contextual and social dependencies of use, 2) the interaction costs of automation and its failures, 3) end-users' ability to understand the underlying security systems infrastructure, and 4) the inappropriateness of rigid policy definitions for inter-personal communication and sharing. These factors are based on arguments derived from the literature of human-computer interaction, sociology, usable security and the history of science and technology. We then use these arguments to explore guidelines for determining how security decision making and policy setting can be automated appropriately for end-users. We posit that specific design choices regarding the appropriate automation of security technology can lead to greater acceptance by users, and suggest several corresponding research directions related to improving end-user security.

2. DEFINING SECURITY AUTOMATION

We define "security automation" as any system or technology that effectively removes the security decision process from the user. In this paper, we are agnostic to the specific domain of security in which automation may be implemented. We include in our discussion a number of systems, including end-user firewalls, spyware detection and removal tools, intrusion prevention systems, spam filters, and access control mechanisms.

Across these domains, there are a range of strategies for how security automation for the end-user can be implemented. Figure 1 illustrates these along a spectrum of rigidity, with more rigid automation strategies on the left and the more flexible strategies on the right.

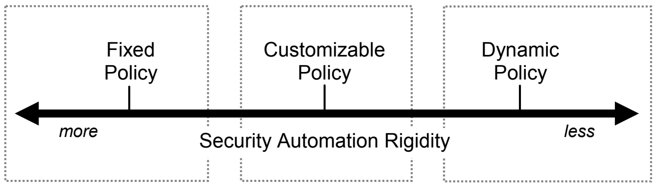

Figure 1. The spectrum of automation approaches

2.1 Fixed policy

At the far left is the case where security decision policies are explicitly embedded directly in the code of a particular tool or application. Examples of such systems include security kernel implementations, Kerberos servers, and virtual private networks (which, for example, allow or deny connections based on a key verification). Often, these "decisions" are so straightforward that they may be rarely though of as policies at all: because they are expressed in the code of the system, there is little ability or need to change them. The rationale for this approach is that such systems embed mechanisms to protect against known malicious behavior or to enable known benign system behavior, in cases where the decision to be made is relatively unambiguous.

2.2 Customizable policy

Next are systems that allow customization of security policies, many of which rely on system-wide defaults configured by a system administrator. Examples include network firewalls, intrusion detection and prevention systems, and patch management systems. In some systems these policies can be dictated enterprise-wide by a centralized corporate network and system management organization; individual hosts then automatically enforce these policies. Examples of such systems include Consul's Insight Security Manager [11], BMC's Marimba [29] and IBM Proventia Management SiteProtector [26]. One distinguishing characteristic of this class of systems is that some entity other than the end-user—a more expert user, an administrator with responsibility for defining and enforcing policy, or even a paid "outsourced" security service such as VeriSign—has responsibility over the end-user's security policy. Once defined, these policies are enforced automatically and are generally not customized (or even seen) by end-users.

Much research has centered on increasing the power of customizable policy systems. Examples include "policy language" systems that express security policies in machine-parseable and enforceable forms (e.g. SPARCLE [9]). Other approaches leverage community knowledge rather than relying on a particular individual or administrative organization (e.g. the Acumen system for cookie management [25]).

Yet another class of systems in this category is that in which the user explicitly *sets* (and often forgets) the policy. This is perhaps the most common form of security automation for end-users, including standard system file permissions (users set permissions on files and folders, which are then enforced by the system until the user changes them) and most consumer firewalls (the end-user chooses to allow access to a given host by a given program, and the firewalls rules are configured to allow this in the future). Although these tools allow end-user configuration, after this configuration step they essentially become fully automated. Thus, we consider them to be in the "customizable automation" class of systems, albeit with end-user control over the customization. The defining property of all of these systems is that there is some degree of adaptability compared to the hard-coded, "fixed policy" approach, independent of whether the policies are set by the users themselves or by another entity.

2.3 Dynamic policy

Finally, the most flexible form of security automation includes systems that are designed to provide dynamic policy adaptation. In contrast to the center of the spectrum, in which adaptation generally occurs infrequently, these systems are designed to be more-or-less continually adaptive. Moreover, unlike systems in which a security service or a systems administration organization sets policies for entire groups of users, these systems often allow personalized tailoring of the security environment to individual users. Some examples include Bayesian spam filters [43] and a range of dynamic access control systems that attempt to adapt to the context of the given situation (including [8, 17, 28, 49, 50]).

Defining security automation for end-users along this spectrum of policy rigidity highlights several issues. First, it shows the extent to which end-users are (or are not) involved in their security, that is, the extent to which so many security decisions have been made for them. Note that this spectrum encompasses automation at all levels of the stack—from low-level hardware to end-user applications. Second, this spectrum begins to highlight some of the tensions of security automation, especially around how well either rigid or adaptable policies accommodate the needs of their users. In the next section, we explore elements of the automation approach, and the contexts in which it is used, to uncover limits to the power of automating security.

3. LIMITS OF AUTOMATION

In this section, we discuss why security automation may be neither feasible nor desirable from an end-user perspective, even in situations in which it is technically possible to automate. Note that factors mitigating against automation do not necessarily *prevent* designers from pursuing an automation approach, but may make it more difficult for them to create an *effective* automated system for end-users, or may make it more problematic for such a system to find acceptance and use among a particular user community.

3.1 Social and environmental contexts of security

Along the spectrum of security automation, the more rigid the security policy is, the more it assumes a "one-size-fits-all" strategy. The fixed policy approach assumes that the security decisions embedded in the design are a fit for the widest range of users; in contrast, the dynamic policy approach assumes a fit for individual users. In other words, the extent to which a security policy is predefined depends on where it lies on the spectrum of policy rigidity.

Predefining security policy or decisions can have significant limitations. In the extreme case, such predefined security policy can become a "one-size-fits-none" strategy to end-user security, because many security decisions are contextually dependent [39]. That is, they are difficult to describe ahead of time, for all cases, and all possible situations for end-users. Although some security decisions almost never change (e.g. blocking the execution of known malicious code), there are a number of cases where security decision-making is not as clear cut. For example, in online transactions, making decisions about which vendors to trust for certain actions cannot be predetermined in many cases [36]. Common firewalls also illuminate problems with flexibility—while firewall configuration policies can perhaps be more easily set for applications such as email clients, similar rules do not easily apply to user browsing activities, in which the system may not be able to make *a priori* determination of a user's intent or level of trust in a site.

Automation failures resulting from overly rigid, predefined security policies can become especially apparent in social (person-to-person) interactions. This is because many security decisions are necessarily socially (not just contextually) dependent, and social activity is fluid and nuanced. This mismatch between the fluidity of social activity and the rigidity and formalization of technical systems has been referred to as the "social-technical gap" by Ackerman, and arises in a range of contexts in which technical systems mediate human communication [1]. For instance, Gaw *et al*. note that choosing whether or not to encrypt an email message may have nothing to do with the technical capabilities or usability of encryption functions in email software. Rather, the people in the organization they studied at times refrained from using email encryption due to perceptions of a message's unimportance or a fear of being perceived by co-workers as being "paranoid" [24].

One can argue that this case illustrates not only the importance of social factors in users' decisions with regard to security, but also serves as an example of how automation "done right" may be effective here. In the email system used in Gaw's study, both the act of encryption and its effects were visible to users; if these were hidden, it is arguable whether these social factors would have come into play. Of course, for many security situations it may be impossible to automate to the point that the user is *completely* removed, which may be necessary in order to eliminate such effects. This is a point we return to in the section "Effects of Automation on User Experience."

In some cases, such as access control for human principals, security decisions are inherently social. Palen and Dourish [35], point to problems driven by social requirements, particularly relating to the need for selective variation of policies over time, and the negotiated, progressively revealed nature of social disclosure.

All of these social effects make security automation policies hard to define *a priori*. Ackerman, for example, points to the example of the Platform for Privacy Preferences Project (P3P), which allows users to specify preferences about disclosure of personal information while browsing the web: users formulate an *a priori* policy description which is then applied automatically while they surf. As Ackerman points out, this is essentially an intractable problem from a human perspective: the P3P model does not support basic principles of progressive social disclosure, presents an essentially infinite decision space to users, and fails to handle the inevitable exceptions to the pre-formulated policy [1].

Some systems have attempted to better support interpersonal activities by grouping users into a set of pre-defined roles that dictate their access rights and abilities, in effect, attempting to mirror social relationships in the access control system. Some of these provide formalized role-based access control (RBAC) mechanisms (such as [18]), which generalize security decisions by grouping principals into a (possibly extensible) set of roles with certain defined privileges. However, studies of these systems in practice seem to indicate that even they do not provide the necessary flexibility for day-to-day social practice. For example, in a study of a networked video conferencing and awareness system, Dourish and Bellotti [14] note that the predefined policies embedded in roles tend to be relatively static and are not a workable solution for many real-world uses. Neuwirth, *et al.*, [31] describing a role-based collaborative writing tool, note how the cost of setting up and maintaining the constantly shifting role relationships quickly outweighs any potential benefit to be had from the access control system. Smetters and Grinter [44] and others have noted similarly problematic experiences of assigning and using roles for access control mechanisms that govern the interaction of human principals.

These examples demonstrate that the more rigid a set of predefined security policy is, the less it is able to accommodate contextual and social dependencies: when the technology is not able to adapt to users' needs at the moment, it is easy to envision situations in which users will work around or actively subvert the technology. So how can the security community cope with the problems of social and contextual dependence? Researchers, designers, and developers have attempted to overcome these problems by pushing automation closer to the dynamic end of the spectrum as shown in Figure 1, making them more adaptable to the situation at hand. These automation systems are necessarily more complex than the static solutions: they attempt to provide more descriptive power, through more fine-grained rules, and through inference mechanisms to adapt to situational demands (see, for example, a number of systems that provide more dynamic approaches to policy setting, including [8, 17, 28, 49, 50]).

However, these highly dynamic systems can themselves be problematic, sometimes producing policy settings that are a poor fit for the social or environmental situation at hand. Systems that use extensive inferencing have been heavily studied in the domain of context-aware computing (see, for example, [4, 27]); these systems can cause usability issues when the inferencing mechanisms fail, leaving users with little understanding of the inner workings of the system, and possibly exposed to risks due to incorrect threat assessment (an issue we shall return to in the section "Effects of Automation on User Experience"). For the most part, security systems in this class remain untested from a user perspective. We do not know whether such fine-grained policy or role definitions will "solve" the limitations caused by overly rigid automation, or whether they simply produce more boundary conditions, as seems to be the case in other domains.

3.2 Whose values are valued?

When security decisions are automated, the values behind these decisions are those of "empowered" sources rather than the users who will themselves be affected by, and must ultimately live with, these policies. By *values* we mean assumptions about intended or proper use that have been embedded into the security automation system—for example, whether the ability to be anonymous is acceptable, or whether individuals in an online system must be accountable and identifiable; whether digital rights management systems are an acceptable form of control, or whether such control is best left to the policy, economic, or legal spheres; and the degree to which governments (or other bodies) can and should be allowed to observe communications. Problems arising from embedded values can range from subtle distinctions around perceptions of proper use, up to flagrant abuse of power.

A key property of security automation systems is that an innate power differential exists between the person who sets or defines the security automation and the end-user who has to live with this decision; problems may arise when the values of these two stakeholder parties do not align. Recent examples from the service-provider domain dramatically illustrate such mismatches of stakeholder values, and how easily such mismatches can be abused. As an example, an action by Canada's second largest telecommunications provider prevented its subscribers and downstream ISPs from accessing the website of the Telecommunications Workers Union [5]. In this case, as well as others such as the so-called "Great Firewall of China," the policies of the network were used to further and support the values of the providers rather than those of the end-users. These examples demonstrate how easily the goals and intentions of end-users can be trampled by automation shaped by the policy of another entity; again, the loss of trust prompted by such a values mismatch may cause users to work around the security system in order to accomplish their goals.

These embedded values can reach into the deepest levels of security technologies. For example, cryptographic protocols designed to allow key recovery implicitly make statements about the legitimacy of outside parties to be able to read encrypted communications (modulo possible legal considerations) [12]. Other technologies—such as digital rights management systems, identity management systems, and others—may similarly embed certain values that may be at odds with those who must use them.

These effects are not unique to security automation; the embedding of values into other technologies has been extensively studied in other domains (see [22, 48] for example), Once embedded, the values represented by these technologies sometimes become "invisible" in the sense that they may appear to be a natural consequence of other, more "value-neutral" design choices.

3.3 Effects of automation on user experience

There are a range of subtle problems arising from automation systems that are only "mostly accurate." Here, by accuracy we mean the degree to which a security automation system makes the "correct" choice (in determining that a potential threat is, indeed, malicious and should be prevented, or in assessing a user's intent in wishing to provide access to some resource). Problems arising from lack of accuracy—the costs of having to deal with automation when it "fails"—exist all along the rigidity spectrum of automation.

Some of these problems are due simply to the poor accuracy of some security tools. For example, work by Axelsson has pointed to the difficulty of creating effective intrusion detection systems because of theoretical limits on these systems' abilities to suppress false alarms [2]; a human generally must attend to these alarms to ascertain whether the threat is valid, of course. In other cases poor accuracy may arise because of the difficulty in keeping security technologies current with the threats that may arise. For example, new "stealth" techniques by malware authors create difficulties for generic tools such as stateless, signature-based spyware detection and removal systems [20, 51]. Note that we are not suggesting here that anti-malware tools not be used; rather, that accuracy problems with such tools impact users' experience of them, and may lead to circumvention on the part of users.

However, merely increasing accuracy is not a complete solution to the problem, since even automation systems that only fail *rarely* must still contend with what to do when one of those rare failures *does* occur. Generally speaking, there are two options available to system designers when automation fails. The first option is to ask the user to disambiguate the decision that must be made; the second is to simply continue as normal. Both options have user experience implications, as we describe here.

The first option of deferring to the user to resolve an ambiguous decision that the system cannot make for itself has obvious user experience costs. The classic examples of this sort of approach are current firewalls, which ask users to indicate whether or not access should be granted to or from a particular remote host. Such notifications are intrusive and disruptive; further, if they are not directly tied to an action the user is doing (trying to play a game, connect to a website, or send an email), these notifications seemingly pop up at random, with no connection to the users' task at hand. This is one of the classic usable security problems, which leads to users ignoring, misusing, or even subverting the security warnings.

However, there are other, and more subtle, problems that result from automation failures in which users must be responsible for deciding the correct course of action. By shielding users from having to make most of the underlying security choices, these systems do not provide good models for correct behavior when automation fails. The sudden "foregrounding" of the underlying system and security infrastructure becomes problematic when users are suddenly asked to cope with and understand this substrate [33, 34]. Sociologist Susan Leigh Star has noted similar issues in her work on the interaction implications of infrastructure [46]. When security automation fails, these decisions will necessarily fall into the hands of users. Hence the system *must* provide mechanisms for dealing with the exception cases where automation makes incorrect security decisions for end-users [33, 34]. In the spyware case, for example, end-users are not provided with mechanisms that could help them make informed decisions when confronted by a choice from their spyware detection system: what happens if I allow this program to continue? Is this program actually spyware? Paradoxically, the very *success* of automation systems in shielding users from having to make security decisions means that they are *even more* ill-equipped to understand and cope with these decisions in the rare times they are faced with them.

The second strategy available to security automation designers is to simply continue as if the automation system had not failed. Indeed, in many cases, this may be the *only* option, as a failure of automation may not be detectable by the automation system itself. Some systems, such as spam classifiers, for example, may simply set a cut-off threshold at a certain confidence value; without user feedback, these systems may have no way to tell that the automation failed.

These systems raise huge concerns from a user experience perspective. For example, if the action the system takes is irreversible, users may come to mistrust (or even fear) the automation system, even though it may fail in only a rare percentage of cases. A spam filtering system that irreplaceably deletes messages will be turned off by many users the first time an important message is lost. Further, if users cannot easily perceive the actions of the system, they may lose trust in it. Obviously, there is a trade-off here: if we require users to monitor the system, we lose many of the benefits of automation. But we do need to ensure that the activities of the system are available and intelligible to users, so that they can determine if it is indeed performing correctly, and possibly to allow them to recover from automation failures. In the next section, we explore some of these user experience tradeoffs resulting from accuracy in more detail, and describe strategies for negotiating these tensions.

4. GUIDELINES FOR AUTOMATING APPROPRIATELY

The limits of automation discussed in the previous section and the understanding that not all end-user security can or should be automated underscore a caveat from Bainbridge: "…the designer who tries to eliminate the [user] still leaves the [user] to do the tasks which the designer cannot think how to automate…[this] means that the [user] can be left with an arbitrary collection of tasks, and little thought may have been given to providing support for them" [3].

To avoid burdening users with such unsupported, arbitrary tasks that may result from improper automation, we present here a set of guidelines that researchers and systems developers can use to decide whether or not to automate aspects of end-user security decision making and policy setting.

(1) How accurate is the system?

If, for any given situation, an automation technology can be made 100% accurate, and if the system is sufficiently flexible to cope with the nuances of any contextual and social dependencies that may be present, then automation may be appropriate. Additionally, a system may also be a candidate for automation if its accuracy is less than perfect, but the benefit of the automation exceeds the risk of failure. For example, in some contexts, spam filters that occasionally improperly classify which messages are spam have benefits that exceed the risks of misclassification (assuming, of course, that the user is equipped to be able to identify and correct any mistakes the automated system makes; this places user interface requirements on the system, that its actions must be intelligible to users, and it must provide the proper affordances for users to be able to tell what it is doing, and correct bad behaviors as necessary).

(2) How are stakeholder values embodied in the system? What roles do social and environmental contexts have in this particular application?

There may be tension between the values of the various stakeholders of a system that automates end-user security decision making. For example, a systems administrator who has to "clean up the mess" after a security break-in may prefer more rigid security controls, whereas an end user may find the same controls chafing. In order to understand the values of stakeholders and subsequently design appropriate end-user interfaces, system designers should perform a stakeholder analysis to assess values, and analyze how these values are or may be embedded into the system in question. In addition to understanding the concerns of system stakeholders, system designers should also examine the potential social and environmental contexts of system use. For all of these questions, methodologies from human-computer interaction such as value sensitive design [21] or participatory design [42] may be particularly helpful, as may design approaches such as AEGIS [40], which incorporate end-user values and intentions into the system's design from its inception.

(3) Does automation reduce end-user information overload or otherwise simplify the task of security decision making?

Automation may be useful when it would be virtually impossible for the user to do the work. For example, intrusion prevention systems inspect packets at a rate which system administrators would no be able to do. The inspection is necessary but impossible to perform without the help of automation. Additionally, automation may be appropriate when an end user is facing a known malicious entity. For example, the use of blacklists and signatures can enable a system to automatically block certain types of attacks an exploits in intrusion prevention system. Note, however, that there should be workarounds for edge cases; that is, automated systems for protecting end-user security should also be flexible enough to allow them to override the automation if necessary or desired [45].

(4) Are there alternatives to automation that are at least as appropriate for end-users?

Instead of automating a security task, system designers should ask, "Is it possible to change the nature of the task so as to avoid this problem all together?" One way of changing the nature of the task is to make security "implicit," as described by Smetters and Grinter [44]. With implicit security, user security decisions are not automated *per se*, yet not left for users to deal with firsthand. Rather, security decisions made by the system are inferred *directly* from actions end users take. If it is not possible to redesign the task, there are still other alternatives to automation. The nature of the task could remain the same, but instead the system could provide end-users with relevant contextual information about the task at hand and its security implications. The benefit of this approach is that the user is kept more aware of the system's state [33, 34].

(5) If automating, are there mechanisms to "keep the human in the loop"?

Norman [34] contends that one of the major problems of automated systems is that improper feedback from the system to the end-user may lead to errors and difficulties when the automation is not performing properly. He further contends that appropriate design assumes that errors will occur, and provides necessary feedback under this assumption. Keeping the human in the loop is an essential element of effective control, especially when the task being automated needs "assistance" from the end-user, as is the case with many security decisions; the challenge is doing this without increasing the complexity of the interface to the point that users turn it off or ignore it [7]. Hence, system designers should ask the following questions when designing systems that automate security decision making for end users: Do end-users understand what the automation is doing? Are there system state indicators? Are these indicators understandable to end users?

(6) If the automation mechanisms fail, are there user interfaces for gracefully dealing with these situations?

The user interaction *costs of automation failures* must be considered in the overall analysis of the worth of a system, not simply the benefits to be accrued when it is working perfectly. This is because when security automation fails, these decisions will necessarily fall into the hands of users, and hence the system *must* provide mechanisms for dealing with the exception cases where automation makes incorrect security decisions. However, user interfaces for recovering from automation failures (which will appear rarely if the automation system is reasonably accurate) may be disruptive or intrusive since they are likely to be unexpected; such interfaces may encourage users to simply turn off the automation system to prevent further disruption. This behavior is commonly seen in end-user firewall systems, for example, where many users ignore, or worse, disable these systems to prevent distracting (and often incomprehensible) messages from appearing at inappropriate times [39].

5. SUGGESTED RESEARCH DIRECTIONS

The above guidelines are far from complete. Our purpose in listing them is primarily to point towards a need for further research. We believe it is at the intersection of security, human-computer interaction, the social sciences, and public policy where the understanding for automating security appropriately for end-users can be more fully developed.

Our perspective on the social and human limits to security automation suggests a number of research directions for the security community. Research in these areas may improve the end user experience of information security by encouraging the use of automation that better accommodates the contextual and social constraints of end-users and their values.

5.1 Accounting for system behavior

As noted in our discussion of the implications of automation accuracy and failure—guidelines (1) and (6) – systems often do not impart strong models to the user that can be applied when automation fails. If the underlying security mechanisms and infrastructure are hidden away most of the time, then users are ill-equipped to deal effectively with automation exceptions. These situations when automation cannot determine the correct course of action are, of course, likely to be the very situations that are rare or highly complex, and thus require strong knowledge on the part of users to choose the correct course of action.

We propose that a better design strategy is to not hide away the security infrastructure in all but the exception cases, but rather to selectively and artfully expose it to users. Instead of having as our goal a completely "transparent" or "seamless" experience, in which the underlying security machinery is *always* hidden (which, as we have argued, is unlikely to be achievable in all circumstances), we propose that exposing these "seams" to the user can provide better user understanding of the (necessary) underlying systems concepts that they will need in order to make better informed decisions.

These design principles have been examined by a number of researchers in the HCI community. For example, Dourish's notion of "system accounts" provides a means by which systems can "account" for their behavior—in other words, provide a rationale for their actions and internal state that is intelligible to users [15]. Such accounts may be especially powerful in situations where policy setting is highly dynamic; for example, when machine learning approaches are used, and the system has a complex internal state that governs its selection of actions. Chalmers *et al.* have explored a number of ways that underlying infrastructure "seams" can be exploited in applications that have a user-visible component. Their approach to "seamful design" shows that, when properly aligned and appropriated by the interface, such infrastructure and systems concepts can be exposed as features in the end-user interface, rather than as "rough edges" to be worked around or hidden [10].

We are not arguing that users must understand the minute details of, for example, public key cryptography. Rather, we argue that in many cases the underlying security infrastructure may suggest certain metaphors, visible at the user level, which may be appropriated by users to enhance and further their abilities to work with the system, even in "exception" cases. In situations where users must be closely involved in the decision process—in other words, in situations where the "exception" cases are somewhat less exceptional—this ability to understand the inner workings of the system to some degree is even more essential. In such cases, it may be valuable to try to ensure that the underling systems abstractions and the user-visible metaphors are as closely aligned as possible, to minimize the discrepancy between "the interface" and "the system."

5.2 End-to-end causality and contextualization

Another research direction (related to guidelines (3), (4) and (5)) can be found in one of the tensions in usable security research: the fact that security interfaces are often seen as intrusive and disruptive [7, 52]. In order to "do security" users must take some action that is orthogonal to the task at hand and use a set of tools that are unrelated to those that they need in order to get their work done.

We propose that a solution to this tension is to provide mechanisms for what we term "end-to-end causality," and use these mechanisms to contextualize security decisions within the framing of user's applications. To illustrate what we mean by end-to-end causality and contextualization, we use the case of a common firewall. When an outbound connection is attempted, most firewalls—separate tools that are only used to manage security policy, not to "get work done"—will display a message requesting that the user decide whether to allow or deny access. This is an example of exception handling, in that the user must be involved in telling the system what the correct behavior is, and which the system will possibly then internalize as a set of persistent rules for future outbound connections. This interaction happens outside of the context of the user's application, forcing them to stop what they are doing, deal with the security decision, and then return to their work. This disruption in workflow and distraction of attention are problematic from a user experience perspective. Perhaps worse, the messages provided by these tools are often incomprehensible to users ("winamp.exe is attempt to contact 128.54.13.122 on port 1222; allow or deny?") because neither the benefits nor the potential risks are framed in terms of the applications that they use day-to-day.

End-to-end causality means tracing operations that require security decision making back to their root causes in users' applications and actions. In those cases in which a given operation is associated with *an action taken by the user,* we have the opportunity to embed that decision into the context of the application that caused the decision event. Since identifying actions caused by user agency is not enough, we must contextualize the decision points raised by these actions *within the applications that caused them*. In other words, we must frame the decision that users face in terms of the concepts embodied in that application. This means that the risks and the benefits of any given decision must be explained in application-centric terms, rather than semantically-neutral terms.

Building on our firewall example, this means that an outbound connection attempt would not cause a separate dialog to appear from the firewall itself, but rather would cause a decision interface to be created within the application that shows, where possible, the root cause of the outbound connection. This interface would not just be contained within the application, but would ideally express the decision in terms of the application itself (perhaps: "You're connecting to a music store I haven't seen before. Since I don't know this store, please be sure that it's legitimate, and does not contain pirated music. Are you sure you want to continue?")

Our concept of causality and contextualization builds on the work of Smetters and Grinter, presented at NSPW 2002 [44] on "implicit security." Implicit security approaches leverage end-user actions to create security policy. This notion likewise depends on causality (being able to correctly identify that a user has taken an action) and is inherently contextualized in that application. While this tracing certainly cannot be done in all cases (underlying systems processes, for example, may connect to the network periodically without any user action being the root cause), we believe that it can provide a powerful approach to addressing the tension between usability and security.

Our framing of causality and contextualization is different from classic systems and network security. In these classic approaches, a trusted infrastructure agent is often charged with protecting the user's system from intrusion and compromise. We propose that more of the decision process—even for decisions that have classically been systems-security level decisions—must move into the application itself, where system designers can take advantage of application semantics, and allow their users to make more informed decisions about what the correct or incorrect choice of action may be, and what the risks and benefits of that action may be.

5.3 Toward socially relevant forms of security

A final research direction is suggested by the issues of policy flexibility in reflecting end-user values as discussed in guideline (2). As we have argued, many automation approaches suffer from a "one-size-fits-all" strategy, in which a generalized policy is called upon in a huge range of specific cases for which it may be ill-suited.

Many of these problems, we believe, arise purely out of the need to craft *a priori* policies that can be easily created (since they are often generated by humans using, for example, policy specification languages) and can be reused (security policies across a corporation or other organization, for instance). In such situations, the fluidity of human work can collide with the comparatively brittle and fixed security policies.

One solution may be to allow highly dynamic and personalized policies for automation; this is, essentially, the space on the far right of Figure 1. While this approach holds promise to overcome the "one-size-fits-all" nature of many automation approaches, it carries with it its own innate drawbacks. For example, such dynamic policies may be generated using statistical machine learning or other approaches that are highly complex, and often unpredictable from the user's perspective. These systems, in trying to anticipate users' needs, are also likely to have more exception conditions than other automation approaches, and so still have issues with accuracy as described earlier.

Another approach may be to incorporate notions of identity of human principals into security infrastructure. While identity management systems have long promised the ability to flexibly identify resources (including people) in online systems, the reality is still short of that promise. As an example, in many online systems today, we authenticate the identity of a *device* rather than a *user*.

Systems such as Bluetooth pairing, for example, rely heavily on unique device IDs (such as MAC addresses, or on-device certificates); they do not directly identify the human user of that device, even though these devices are often treated as proxies for their users. We propose that, with the increase in human-to-human communication and applications on the network, we need to conceive of security—and identity management—more in terms of its person-centricity rather than simply device-centricity. Although this communication will, of course, always be mediated by machines (and is thus always, at some level, machine-to-machine communication), we believe that device identification should play a secondary role to human identity management. One goal of this approach should be to minimize exception handling (which users will likely have to cope with) by framing policy decisions in terms of the people with whom a user interacts, rather than simply the devices those people might be using.

Framing security decisions in social terms allows social navigation and social networking structures to be applied to the challenge of usable security, an avenue that is already being explored by a number of researchers [13, 25]. Of course, any system that uses social identity as a key concept must be flexible in how it defines principals, allowing users to take on a range of identities to suit various social demands (as suggested by [16]).

While such an approach is not a panacea, and will certainly not deal with all of the problems of inflexible automation systems, it does move closer to a world in which the socially-dependent aspects of security can be more readily expressed—by bringing social actors into the security infrastructure as first-class abstractions, rather than simply the devices that they may use.

6. CONCLUSIONS

In this paper, we have argued that although automation is often touted as a means to achieving better security by taking the user out of the security decision process, there are inherent limits to automation, based on human and social factors. Many of these factors are *independent* of the specific security technology being used. The significance of this is that more or better automation will not necessarily "fix" the problems, because of inherent limitations that arise from issues of flexibility and accuracy, as well as with fundamental mismatches to the social contexts in which automation may be deployed.

While automation may be useful in some contexts, an understanding of the *non-technical* constraints around its use is essential for effectively applying it. Understanding the inherent limits to automation suggests research directions that can improve the efficacy of automation when used appropriately.

Towards this end, we suggest a definition of security automation for end-users along a spectrum of rigidity. This spectrum provides a basis for discussing three sources of limits in automation: situational and social dependencies, accommodation of end-user values, and user interface costs deriving from automation failures. We then use these limitations as a basis for discussing six guidelines to help ensure the appropriate use of security automation for end-users. These six guidelines serve to highlight three broad areas for further research: providing accountability to end-users for system behavior, further incorporation of the implicit security approach by providing end-to-end causality and contextualization, and a more socially relevant form of security for end-users.

7. ACKNOWLEDGEMENTS

We thank our reviewers, as well as our colleagues at Georgia Tech including Jonathon Giffin, for their insightful feedback, which has helped us to strengthen this paper. This work was supported by the National Science Foundation (CNS #062681), Symantec Corporation, and the US Department of Homeland Security.

8. REFERENCES

[1] Ackerman, M. 2000. The Intellectual Challenge of CSCW: The Gap Between Social Requirements and Technical Feasibility. *Human-Computer Interaction Journal*, vol. 15, pp. 179-203.

[2] Axelsson, S. 1999. The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection. In *Proceedings of the 6th ACM Conference on Computer and Communications Security,* November 1-4, 1999. Singapore, pp. 1-7.

[3] Bainbridge, L. 1987 Ironies of automation. In *New Technology and Human Error* (ed. J. Rasmussen, K. Duncan, & J. Leplat). New York: Wiley.

[4] Bellotti, V., and Edwards, W.K., "Intelligibility and Accountability: Human Considerations in Context Aware Systems," *Journal of Human-Computer Interaction*, 16:2-4, 2001. Thomas Moran, ed., Lawrence Erlbaum Associates, NJ. pp. 193-212.

[5] Benkler, Y. Wealth of Nations. Yale University Press, 2006, pp. 133-175.

[6] Berinato, S. How to Save the Internet, March 15, 2005, *CIO Magazine.*

[7] Bishop, M., "Psychological Acceptability Revisited". In L. Cranor & S. Garfinkel (Eds.), *Security and Usability: Designing Secure Systems That People Can Use*, O'Reilly & Associates, 2005.

[8] Blaze, M., Feigenbaum, J., and Lacy, J. "Decentralized Trust Management." In *Proceedings of the IEEE Symposium on Security and Privacy*. Oakland, CA. May, 1996. pp. 164-173.

[9] Brodie, C. A., Karat, C., and Karat, J. An empirical study of natural language parsing of privacy policy rules using the SPARCLE policy workbench. In *Proceedings of the Second Symposium on Usable Privacy and Security*, PA, 2006.

[10] Chalmers, M. Dieberger, A., Höök, K., Rudström, A. "Social Navigation and Seamful Design." *Cognitive Studies*: *Bulletin of the Japanese Cognitive Science Society* 11(3), pp 171-181, 2004.

[11] Consul: http://www.consul.com/Content.asp?id=58

[12] Denning, D.E., and Branstad, D.K. "A Taxonomy for Key Escrow Encryption Systems," *Communications of the ACM*, 39:3, March 1996.

[13] DiGioia, P., Dourish *P.* Social Navigation as a Model for Usable Security. *Proceedings of the Symposium On Usable Privacy and Security (SOUPS)*, Pittsburgh, PA, 2005.

[14] Dourish, P. and Bellotti, V. 1992. Awareness and Coordination in Shared Workspaces. *Proceedings of the Conference on*

Computer-Supported Cooperative Work (CSCW'92). Toronto, Ontario. pp.107-114. New York: ACM.

[15] Dourish, P. and Button, G. On "Technomethodology": Foundational Relationships between Ethno-methodology and System Design. *Human-Computer Interaction*, 13(4), 1998, pp. 395-432

[16] Dourish, P. Grinter, R.E., Delgado de la Flor, J., and Joseph, M. "Security in the Wild: User Strategies for Managing Security as an Everyday, Practical Problem." *Personal and Ubiquitous Computing* 8, pp. 391-401. Springer Verlag, 2004.

[17] Edwards, W. K. "Policies and Roles in Collaborative Applications." *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW)*, Boston, MA, 1996.

[18] Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R. and Chandramouli, R. "Proposed NIST Standard for Role-Based Access Control," *ACM Transactions on Information and System Security*, vol. 4, no. 3, pp. 222–274, August 2001.

[19] Fisher, K. "Internet about to collapse says Finnish scientist". *ars technica*, Oct. 18, 2004. Also:www.ars technica.com/news.ars/post/20041018-4318.html

[20] Ford, R. and Allen, W. H. 2007. How Not to Be Seen. *IEEE Security and Privacy* 5, 1 (Jan. 2007), 67-69.

[21] Friedman, B., Kahn, P. H., Jr., & Borning, A. (2006). Value Sensitive Design and information systems. In P. Zhang & D. Galletta (eds.), *Human-computer interaction in management information systems: Foundations* (pp. 348-372). Armonk, New York; London, England: M.E. Sharpe.

[22] Friedman, B., and Kahn, P.H. Human Values, Ethics, and Design. *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications.* Lawrence Erlbaum Associates, Mahwah, NJ. pp. 1177-1201.

[23] Garigue, R. Between Chaos and Order: Managing the Risk in Organizations, BMO Financial Group, https://www.expotiperformance.com/Sites/expoIT/Multimedias/garigue.pdf

[24] Gaw, S., Felten, E. W., Fernandez-Kelly, P.. Secrecy, Flagging, and Paranoia: Adoption Criteria in Encrypted E-Mail. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, 2006.

[25] Goecks, J., Mynatt, B. Social Approaches to End-User Privacy Management. In L. Cranor & S. Garfinkel (Eds.), *Security and Usability: Designing Secure Systems That People Can Use*, O'Reilly & Assoc 2005.

[26] IBM: http://www.iss.net/products/Proventia_Management_SiteProtector/product_main_page.html

[27] Isbell, C., Pierce, J. Using an IP Continuum for Adaptive Interface Design. *Proceedings of the 11th International Conference on Human-Computer Interaction*, 2005.

[28] Kanawati, R., and Riveill, M. "Access Control Model for Groupware Applications." In *Proceedings of Human-Computer Interaction*. Huddersfield University, UK. August, 1995. pp. 66-71.

[29] Marimba: www.bmc.com/products/attachments/MarimbaClientManagementSecurity.pdf

[30] NetworkWorld. Executive Guide: Security Evolves. (2004). Also at:www.enterasys.com/corporate/pr/2006 /060123-networkworld.pdf

[31] Neuwirth, C., Kaufer, D.S., Chandhok, R., and Morris, J. 1990. "Issues in the desigin of computer support for co-authoring and commenting," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*. CA, pp. 183-195

[32] Nielsen, Jakob. *User Education is not the Answer to Security Problems*. http://www.useit.com/alertbox/ 20041025.html (2004).

[33] Norman, D. A. The Design of Everyday Things. Basic Books, NY, 2002.

[34] Norman, D. A. (1990). The "problem" of automation: Inappropriate feedback and interaction, not "over-automation". In D. E. Broadbent, A. Baddeley & J. T. Reason (Eds.), *Human factors in hazardous situations* (pp. 585-593). Oxford: Oxford University Press.

[35] Palen and Dourish "Unpacking Privacy for a Networked World", *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, Ft. Lauderdale, FL, 2003.

[36] Patrick, A.S., Briggs, P. & Marsh, S. Designing systems that people will trust. In L. Cranor & S. Garfinkel (Eds.), *Security and Usability: Designing Secure Systems That People Can Use*, O'Reilly & Associates, 2005.

[37] Rainie, Lee. The CAN-SPAM Act Has Not Helped Most Users So Far. *Pew Internet Life Foundation Report,* 2004.

[38] Sasse, M. A., Has Johnny Learnt to Encrypt by Now? Examining the Troubled Relationship Between a Security Solution and Its Users. *5th Annual PKI R&D Workshop (2006)*. middleware.Internet2.edu/pki06/ proceedings/sassejohnny_usability.ppt

[39] Sasse, M. A., Brostoff, S., Weirich, D. Transforming the 'weakest link' – a human/computer interaction approach to usable and effective security. *BT Technol J* Vol 19 No 3, July 2001.

[40] Sasse, M.A., Flechais, I., Usable Security: Why do we need it? How do we get it? In L. Cranor & S. Garfinkel (Eds.), *Security and Usability: Designing Secure Systems That People Can Use*, O'Reilly & Assoc 2005.

[41] Schneier B: *Secrets and Lies*, John Wiley and Sons, 2000.

[42] Schuler, D., Namioka, A., *Participatory Design: Principles and Practices*, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 1993

[43] Sehami, M., Dumais, S., Heckerman, D., Horvitz, E. "A Bayesian Approach to Filtering Junk Email." *AAAI Workshop on Learning for Text Categorization*, 1998.

[44] Smetters, D., Grinter, R. Moving from the Design of Usable Security Technologies to the Design of Useful Secure

Applications. *Proceedings of theNew Security Paradigms Workshop (NSPW)*, 2002.

[45] Spiekermann, S., Pallas, F., "Technology Paternalism -Wider Implications of Ubiquitous Computing." *Poiesis & Praxis: International Journal of Ethics of Science and Technology Assessment*, Vol. 4, 2005

[46] Star, S.L., The Ethnography of Infrastructure. *American Behavioral Scientist* 43:3, pp. 377-391. 1999.

[47] Whitten, A., Tygar, J., Why Johnny Can't Encrypt. *Proceedings of the 8th USENIX Security Symposium* (1999).

[48] Winner, L. "Do Artifacts Have Politics?" *The Whale and the Reactor: A Search for Limits in an Age of High Technology*. Chicago: The University of Chicago Press, 1986. pp. 19-39.

[49] Wong, R.K., Chau, H.L., and Lochovsky, F.H. "A Data Model and Semantics of Objects With Dynamic Roles." *13th International Conference on Data Engineering (ICDE)*. Birmingham, UK, April 1997. IEEE Computer Society, pp. 402-411.

[50] Woo, T., and Lam, S. "Designing a Distributed Authorization Service." In *IEEE Computer and Communications Societies (INFOCOM)*, CA. 1998.

[51] Wu, M., Huang, Y., Wang, Y., and Kuo, S. 2006. A Stateful Approach to Spyware Detection and Removal. In *Proceedings of the 12th Pacific Rim international Symposium on Dependable Computing* (December 18 - 20, 2006). PRDC. IEEE Computer Society, Washington, DC, 173-182.

[52] Yee, K. Guidelines and Strategies for Secure Interaction Design. In L. Cranor & S. Garfinkel (Eds.), *Security and Usability: Designing Secure Systems That People Can Use*, O'Reilly & Associates, 2005.

[53] Zurko, M. User-Centered Security: Stepping Up to the Grand Challenge. *21st Annual Computer Society Security Applications Conference*, 2005.