In Proceedings of the 2003 Workshop on Human-Computer Interaction and Security Systems, April 2003

Safe Staging for Computer Security

Alma Whitten
University of California at Berkeley
School of Information Management & Systems
102 South Hall #4600
Berkeley, CA 94720-4600
alma@sims.berkeley.edu

J.D. Tygar
University of California at Berkeley
School of Information Management & Systems
102 South Hall #4600
Berkeley, CA 94720-4600
tygar@sims.berkeley.edu

ABSTRACT
In this paper, we introduce the technique of safe staging for computer security, which is adapted from conventional user interface staging to meet the specific needs of computer security in consumer software. Safe staging can reduce the initial complexity of security concepts for novice users while providing continuous protection against dangerous security errors and supporting exploratory learning.

Keywords
Security, usability, learnability, staging, training wheels.

INTRODUCTION

This paper presents three things:

1. A user interface design technique called safe staging which is highly appropriate for creating learnable security software.
2. A high-level discussion of how safe staging might be applied to applet security.
3. A brief overview of a detailed experiment in which safe staging was applied to public key certification in a secure electronic mail application.

What is staging, and how can it be done safely? Staged user interfaces shepherd the user through a sequence of stages of system use to increase user understanding and protect against errors. In computer security, especially in software that is intended for general consumer use, learning how to make effective use of the security mechanisms often presents an insurmountable threshold to novices [6]. If that threshold can be safely broken down into a series of smaller steps, through user interface staging, then the usability of the security, and the security of the user, will be greatly improved.

Existing research on varieties of user interface staging, such as [2] and [4], relies on explicitly exercised control over the user's progress, and enforced restrictions on user actions in the earlier stages. Such coercive control can be appropriate for training in a business software environment, and is expected in educational software that follows a tutorial model, but is unlikely to be appreciated by general consumers, who want to accomplish their primary tasks and may already be disposed to view security as an annoying interference. To develop a technique for staging security that does not rely on coercive control, we adapt the model provided by existing standards for consumer warning labels to create a template for designing levels of security use that create a natural sequence, support conscious exploration, and protect the user at all times.

For some security concepts, the significant obstacle to user understanding is the distance between the human concern underlying the security goal and its abstract representation by the security mechanism. Public key certification is an example of such a concept. We show how our staging technique can be usefully adapted to guide the user through successive levels of abstraction, and present experimental results that strongly support the effectiveness of the staging technique.

USEFUL TERMS

User interface staging might be designed in a wide variety of ways. In order to more clearly discuss which of those ways are useful and appropriate for computer security, we define the following terms:

hard staging: explicitly enforces requirements for when the user may progress to the next stage, and restricts user actions in the earlier stages.

soft staging: gives the user freedom to decide when to progress to the next stage, and encourages progression by presenting it as the conceptual path of least resistance.

function restricted staging: restricts or otherwise avoids the use of certain functions until the user is competent to manage the security necessary for protection.

data restricted staging: avoids exposure of private data or other valuable resources to the software until the user is competent to manage the security necessary for protection.

We have already mentioned the reasons why hard staging is unlikely to be appropriate in software intended for general consumer use. Function restricted staging faces a similar obstacle in that, as security designers, we are often attempting to support the adoption of security use by people who are already accustomed to using those functions without security. We cannot realistically expect users to accept security software that asks them to suspend or even delay their access to instant messaging, downloadable executables, or electronic mail, yet those are some of the applications for which we would most like to design truly usable security. For those reasons, our focus here will be on techniques for soft, data restricted staging.

MAKING SOFT STAGING SAFE

We assert that the use of a security mechanism can be safely postponed by a stage if the user interface can be designed to meet the following requirements as soon as the stage begins:

1. The user must know which of the available actions are potentially dangerous.
2. The user must know what bad consequences may result from the danger.
3. The user must know how to use a temporary strategy to avoid the danger.
4. The user must know how to begin learning about the postponed security mechanism once the limitations imposed by the temporary strategy become unacceptable.

These requirements correspond closely to the ANSI standard for consumer product warning labels [1], so it is reasonable to expect that this amount of information can be successfully conveyed in a brief and eyecatching manner. Furthermore, presentation of this amount of information is unlikely to significantly delay or annoy a user who already understands the security mechanism.

The temporary strategy for avoiding the danger may be based on either data restriction or function restriction. The success of the security user interface as a whole also requires that the security mechanism, once the user decides to investigate it, be presented in a usable and learnable manner, but that is true regardless of whether or not staging is used.

EXAMPLE: STAGING APPLET SECURITY

When Java applets were first introduced, it was with the expectation that they would be used in a wide variety of ways, including, at one extreme, as subscription-based replacements for application software such as word processors, desktop publishers, and checkbook managers. This prospect required web browsers to incorporate applet security mechanisms that were flexible enough to allow known, authorized applets to access system resources on the user's machine, such as files, directories, and printers, while blocking randomly encountered applets from doing anything dangerous or perhaps even from executing at all.

The Java applet security model [3] presents a challenging usability problem, since it includes the full complexity of digital signature based trust management and access control. Furthermore, because of the design goal of encouraging the adoption of applet technology, a restrictive default configuration would be problematic, and would in any case most likely be quickly circumvented by users. The security user interface design in the HotJava web browser [5] attempted to solve this dilemma by setting a default configuration that restricted applets to running in a "sandbox" without access to system files or peripherals, but also allowed any digitally signed applet to request additional access permissions directly from the user. This design delayed the presentation of trust management and access control issues to the user until the first time a signed applet made a request for additional permissions, but it did not do so safely, since users were thereby set up to have to make a sophisticated trust decision on the spot, without any advance opportunity to develop the necessary expertise. It is a truism in computer security that users in such a situation will nearly always press the "Okay" button and hope for the best.

A design approach that uses soft staging with data restriction, by contrast, leads us to begin with a default state that allows users to safely postpone understanding of both digital signature based trust management and of access control policies, while still permitting applets to store and access local files within safe limitations. In the default state, or initial stage, applets would be permitted access only to a single, dedicated directory, created for that use alone and containing no pre-existing files. The user would need to be briefed with the following information, in accordance with the safety requirements:

1. That permitting an applet access to their data is dangerous.
2. That an applet might steal, damage, or otherwise misuse their data.
3. That the danger can be avoided by withholding all valuable information from the dedicated applet directory and from direct interaction with applets.
4. That when they wish to consider whether an applet may be trusted with valuable information, they can investigate the use of digital signatures for establishing trust.

Once the user decided to proceed to the next stage, they would have the option of granting a digitally signed applet its own dedicated directory, accessible only by applets signed with the same private key. They would need to be briefed with the following updated information to maintain compliance with the safety requirements:

1. That permitting an applet access to data that it did not itself store is dangerous.
2. That an applet might steal, damage, or otherwise misuse data it does not own.

3. That they can avoid the danger by continuing to withhold any data that they do not want a particular applet to have.
4. That when they wish to allow an applet limited, specific access to data outside its own directory, they can investigate the access control policy mechanism.

In the final stage, the user would be able to define fine grained access control policies for applets signed with particular keys, just as they would in the original HotJava design. In this design, however, they would have followed a natural, well-supported learning path to arrive at that stage, and would at all times have had enough information to proceed and explore safely.

SOFT STAGING FOR THE ABSTRACTION PROBLEM

Computer security is often confusing to novices not only because of the number of new mechanisms to be understood, but also because many security mechanisms are abstract, pared down formal representations of messy, human social concerns such as trust, identification, and permission. One way to help users become comfortable with those abstractions might be to design an initial stage that combines data restriction with an intuitive, social temporary strategy for protecting against the danger. This would allow users to practice thinking about the process for which the security mechanism is an abstraction, preparing them to recognize the mechanism itself as sensible and useful once they decide to investigate it. Such a temporary strategy will probably provide weak protection at best, but that is not a safety issue as long as it is combined with data restriction and accurate presentation of the remaining danger, in accordance with the safe staging requirements.

EXPERIMENTAL RESULTS

We designed, implemented, and performed usability testing on a secure electronic mail application that included soft, data restricted staging to help users understand the highly abstract mechanism of public key certification. A detailed discussion of our design and of our testing methodology is beyond the scope of this extended abstract, but we briefly cover the main points and summarize the results here. A full description of these experiments will appear in a forthcoming publication.

Staging design for public key certification

We staged public key certification using our method of soft staging for the abstraction problem. Key certification is a particularly difficult concept to convey to users, because it takes the already complicated social process of having a known and trusted individual vouch for another individual's identity, and represents it as an abstract mechanism. To stage it, we attempted to first accustom users to thinking about the problem of verifying identity, before encouraging them to progress to a second stage in which the use of trusted certifiers was available as a solution to that problem. This required a temporary strategy for verifying identity that might provide only weak verification, but would be easy for users to understand. The obvious candidate for such a strategy was the use of writing style and references to shared personal data as material for verifying identity when trading public keys by electronic mail. The resulting initial stage was thus designed to meet the following safety requirements:

1. The user must know that accepting a public key without verifying identity is dangerous.
2. The user must know that they could be spied on or tricked into accepting forged messages if they accept the wrong public key.
3. The user must know that they can get some protection by using the weak verification method described above, but that it will not protect them against a skilled, targeted attack.
4. The user must know that they can investigate key certification when they decide that they want stronger security.

Comparison testing

We constructed a short paper presentation that explained the basics of how to use a public key based electronic mail security system such as PGP. To create a variant corresponding to our staged design, we inserted an additional two paragraphs of text, describing the use of our temporary strategy for weak verification, before the explanation of key certification. We also constructed a third, unstaged variant, similar to the presentation of SSL in most web browsers, which downplayed the role of secret keys. Test participants were randomly assigned one of the three variant presentations, followed by a series of written questions about how the security system would be used in a variety of scenarios. In answering those questions, 45% of the participants who received the staged variant were able to correctly describe the use of key certification, versus 10% of those who received the basic variant, and none of those who received the third variant. Since the presentation of key certification itself was identical across all three variants, this strongly suggests that the staging was responsible for the greatly increased success rate.

Proof of concept testing

We created a fully functional software simulation of our staged user interface design and used it to perform formal scenario based user testing, similar to that described in [6]. Two of our twelve test participants experienced difficulties that prevented them from trading public keys successfully and thus did not progress to the portion of the test involving key certification. Of the remainder, nine out of ten were able to successfully and appropriately get their own public keys certified, and six out of ten appropriately rejected an imposter's proffered public key for lack of certification. Success on the latter point was also strongly correlated with having engaged with the staging by making use of the weak verification strategy in an earlier scenario, as was success at answering true/false questions about key certification in the post-test debriefing questionnaire. These are excellent results for what is probably the most difficult usability challenge in public key cryptography, and appear to accord with the expected usefulness of the staging in increasing user understanding.

CONCLUSIONS

The technique of user interface staging, appropriately adapted to the particular needs of computer security in consumer software, can serve as a powerful method for safely reducing the immediate complexity of security concepts for novice users.

ACKNOWLEDGEMENTS

We gratefully acknowledge funding support for this research. This research was sponsored in part by the Defense Advanced Research Projects Agency under DARPA contract N6601-99-28913 (under supervision of the Space and Naval Warfare Systems Center San Diego), by the National Science foundation under grant FD99-79852, and by the United States Postal Service under grant USPS 102592-01-Z-0236. Views and conclusions contained in this document are those of the authors and do not necessarily represent the official opinion or policies, either expressed or implied of the US government or any of its agencies, DARPA, NSF, USPS.

We also thank Ka-Ping Yee for serving as a second coder for our user testing results, and James Landay and James Lin for aid with user testing laboratory space and equipment.

REFERENCES

1. American National Standards Institute. *ANSI Z535.4 Product Safety Signs and Labels*, 1998.
2. J.M. Carroll and C. Carrithers. *Training Wheels in a User Interface.* Communications of the ACM, Vol. 27(8):pages 800—806, August 1984.
3. J. Steven Fritzinger and Marianne Mueller. *Java Security.* Sun Microsystems White Paper, 1998.
4. Guzdial, M. Software-realized Scaffolding to Facilitate Programming for Science Learning. Interactive Learning Environments, Vol 4. No. 1, 1995, 1-44
5. Sun Microsystems. *HotJava Browser: A White Paper.* Sun Microsystems White Paper, 1995.
6. Alma Whitten and J.D. Tygar, *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0.* Proceedings of the 9th USENIX Security Symposium, August 1999.