A Prototype User Interface for Coarse-Grained Desktop Access Control

A. Chris Long, Courtney Moskowitz, and Greg Ganger

November 13, 2003
CMU-CS-03-200 3

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This work was supported by grants from the Center for Computer and Communications Security (C3S) at Carnegie Mellon University and from the Critical Infrastructure Protection Information Assurance Fellowship at the Department of Defense. We thank the members and companies of the PDL Consortium (including EMC, Hewlett-Packard, Hitachi, IBM, Intel, Microsoft, Network Appliance, Oracle, Panasas, Seagate, Sun, and Veritas) for their interest, insights, feedback, and support. We also thank the participants in the studies.

Keywords: User interface, security, desktop, low-fidelity prototyping, role-based access control, sandboxing, compartmented mode workstation

Abstract

Viruses, trojan horses, and other malware are a growing problem for computer users, but current tools and research do not adequately aid users in fighting these threats. One approach to increasing security is to partition all applications and data based on general task types, or "roles," such as "Personal," "Work," and "Communications." This can limit the effects of malware to a single role rather than allowing it to affect the entire computer. We are developing a prototype to investigate the usability of this security model. Our initial investigation uses cognitive walkthrough and think-aloud user studies of paper prototypes to look at this model in the context of realistic tasks, and to compare different user interface mechanisms for managing data and applications in a role-based system. For most participants, our interface was simple to understand and use. In addition to a security model that is intrinsically useful, we believe development of this system will inform issues in the design and implementation of usable security interfaces, such as refinement of design guidelines.

1 Introduction

Many security mechanisms, methods, systems, etc., have been developed to provide protection against a variety of threats. Unfortunately, much of this work is compromised in practice because it ignores a crucial link in the security chain: the user [14]. The main goal of most users is not security, but, for example, to process invoices, send email, read `cnn.com`, or view the dancing bears animation sent by a friend. There is increasing evidence that security that does not account for users' desires, needs, and abilities will fail[1] [1, 8, 14, 19].

In the physical world, we have reasonable security in spite of a lack of fine-grained security mechanisms. For example, we routinely allow only partly-trusted people into homes, such as friends for socializing and repairmen to fix our utilities. We often only monitor them loosely, since that is all that is required to insure that the plumber is not looking through our CDs instead of fixing the pipes. This level of supervision is easy because CDs and pipes are typically not near one another. In general, the varied locations for different types of objects and activities in the home provide a convenient, albeit coarse, security mechanism.

In information space, no analogous mechanism exists. On typical home computers, all programs run by the user have access to all data, and in some circumstances set security policies they do not intend [8]. Users do not necessarily store files for different purposes in separate places. Even when they do, current security mechanisms do not automatically take advantage of this, and it is too much to expect users to set good permissions on every file or directory (e.g., using permission bits under Unix™[9] or access control lists under modern versions of Microsoft Windows™[15] and other operating systems). However, just as people coarsely sort their possessions in the physical world into separate rooms, they may be willing to coarsely sort their files and applications. This sorting may enable the system to protect data from malware more effectively than it does at present. For example, on a typical home computer today, malicious code (e.g., obtained as an email attachment, downloaded from the web) has freedom to do anything to any part of the computer. However, if the user always reads mail or browses the web in one role, only data in that role, rather than the whole computer, will be vulnerable to email- or web-transmitted malware. Another potential benefit of coarse-grained file classification is that we may find it easier to do and more accurate than fine-grained classification.

We have begun developing a system designed to give the user increased security by providing easy-to-understand and -use coarse-grained access control. In our security model, all files and applications are partitioned into a handful of categories, such as "Personal," "Work," and "Communications." Applications running in one role will not have access to files belonging to a different role. We have built a paper prototype and investigated its usability with think-aloud user studies, a cognitive walkthrough, and a keystroke-level-model. Preliminary results show that this system is usable and desirable.

We believe development of this system will not only directly yield better protection from malware, but also indirectly improve security by informing HCISEC (HCI + security) guidelines (e.g., [21]), design (e.g., the ideal level of user control and awareness of security), and implementation (e.g., whether security should be at the application, toolkit, or operating system level).

[1] In most situations, that is, such as home and office. In some scenarios, such as the military, users can be forced to use security even when it is cumbersome, time-consuming, and hard to use.

Figure 1: The Security Manager

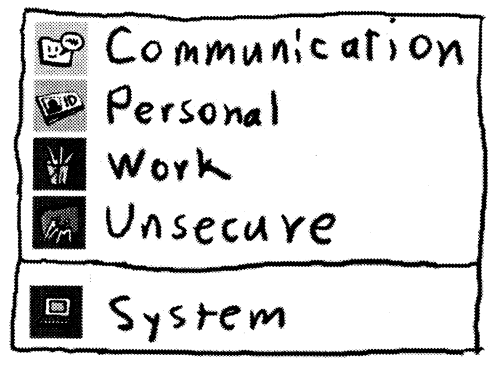

Figure 2: Role menu

The next section describes how roles work in our system. Next, we describe the user studies, followed by non-experimental analyses of the system. We then discuss the studies, our system, and larger issues of HCISEC. Finally, we conclude and summarize future work.

2 Working with Roles

The system protects data by partitioning the computer into a handful of roles. In the filesystem, this means that there is a set of files shared read-only by all applications (e.g., system-wide shared libraries) and a set of files for each role that can only be accessed by that role (except the system role, which is analogous to an administrator account and can access all roles). At runtime, applications are prevented by the operating system (OS) from interacting with applications running in other roles. Each role has its own clipboard so that copying or cutting to the clipboard does not automatically make the data available to all roles.

We have designed interface mechanisms for performing common desktop operations in the role-based context, such as copying the clipboard from one role to another and opening a file from a different role.

Part of our user interface is a new top-level window, the *Security Manager* (see Figure 1), which runs in the System role. The windowing system is trusted and ensures that I/O goes only to the Security Manager or the appropriate role as indicated by the user.

Files can be moved across roles in several ways: by dragging to a folder window belonging to another role or to a role icon in the Security Manager; with a context menu item "Copy to" (which brings up a sub-menu of the roles, shown in Figure 2); or by copying the clipboard from one role to another (using menus on the Security Manager, shown in Figures 4 and 5).

For directly accessing files across roles from within an application, we modified the typical file dialog box (see Figure 3).

There are many slight variations in how many of the above techniques work; these were investigated in the user studies (described below).

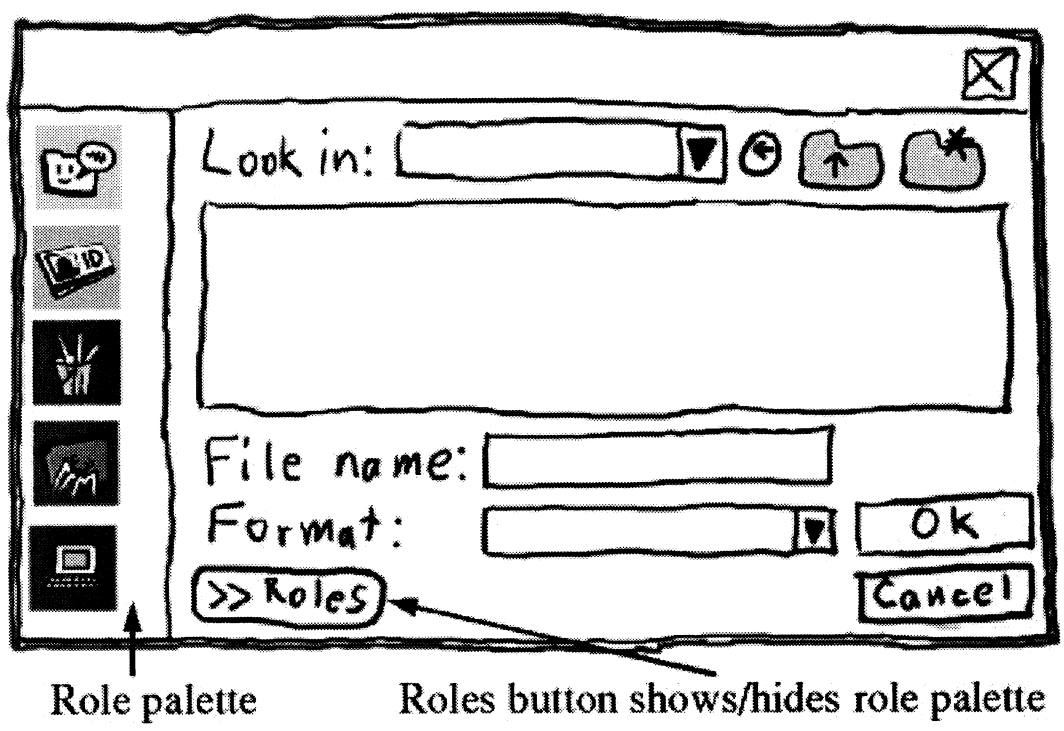

Figure 3: Role-aware file dialog box. The role palette on the left side chooses the role to browse. The "Roles" button hides and shows the role palette.

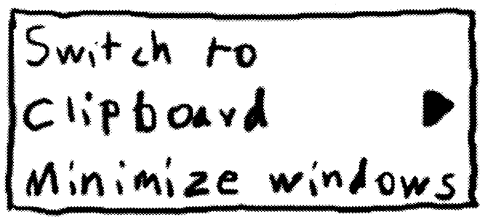

Figure 4: Security Manager context menu

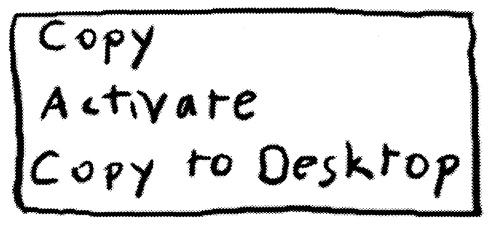

Figure 5: Security Manager menu for clipboard option (see Figure 4)

3 User Studies

This section describes two think-aloud user studies of our interface, both conducted using a paper prototype.

3.1 First User Study: Security in Context

The goal of this study was to investigate how well people could use the interface in the context of activities typical of home computer use. We also wanted to test whether manual or automatic (implicit) role-switching would be preferred.

3.1.1 Participants

We recruited participants from the general student body at Carnegie Mellon University. Five men and five women, ranging in age from 19 to 26, participated in this study. Each participant was paid for their time.

3.1.2 Equipment

The interface was a paper prototype. The questionnaire at the end of the study was filled out on a desktop computer using a web browser.

3.1.3 Procedure

Participants performed the experiment one at a time. We described the general concept of a paper prototype and how to interact with it. Next, the participant read an overview of the interface, including a description of the security model and each role. Participants were encouraged to think aloud.

Participants were given a tutorial consisting of three tasks: copying a URL from one role to another using the clipboard; filing an email attachment into a different role from the one the email client is in; and transferring a file from one role to another using the clipboard and then attaching it to an email message. The tutorial gave explicit, step-by-step instructions on how to perform each task, such as "Right-click the Unsafe button in the Security Manager."

In the tutorial and the experimental task, for half of the participants, clicking on an object of an inactive role automatically selected the object and made its role active; for the other half, only objects belonging to the current role could be accessed so the role had to be switched manually. After the tutorial, participants were given an opportunity to ask questions.

Next, participants did the experimental tasks. Unlike the tutorial, participants were given only a general description of the task, and it was up to them to decide how to accomplish it. We took away the interface overview and the tutorial at the beginning of the experimental task so we could more easily monitor how well they remembered how to use the interface. The tasks were:

Downloading a game Given a URL in an email message, open the link and download the game from that site.

Play an animation A friend sends you a file using instant messaging. Play the file, but first download a player for it (you are prompted for where to go to get the player).

Company document You've received a work-related email to file away.

Bill payment You've received an email about a bill with a link to your bank's website. Pay the bill.

Personal notes Write some personal notes and save them to a file in the Personal role. Attach the file to an email message in the Communications role.

After the experimental tasks, participants filled out a web-based questionnaire about their experiences using the interface and general demographics. Many of the questions asked the participant to rate the interface, tutorial, etc., on a scale of 1–9 for a pair of words. For example, one question asked for overall reaction to the system as

difficult 1 2 3 4 5 6 7 8 9 easy

The questionnaire also included requests for free-form comments. Participants took 45–60 minutes to complete the study.

3.1.4 Results

We obtained two types of results from the study: numerical scores from the questionnaire and verbal or written comments. Numerical ratings of the security features were positive overall (see Table 1).

We found that participants in the automatic role-switching condition verbally expressed approval more frequently than participants in the manual role-switching condition. Requiring the user to switch roles manually before accessing a file or application caused some users to become confused when their action failed. Although the numerical survey data showed no significant difference between approval ratings for the two groups, we felt that the qualitative evidence was sufficient to recommend automatic role-changing for future testing.

Seven participants made positive comments (e.g., "I definitely would like it," "Easy to operate") and two expressed interest in finding out when the software was ready. However, four made negative ones (e.g., "annoying for the lazy user who doesn't want to deal with the 'roles' each time they use the computer"). The main issues were convenience of transferring files across roles and of switching roles. Participants varied in their opinion of the likelihood

Word pair		Median rating	
1	9	Study 1	Study 2
intrusive	helpful	6.0	8.0
annoying	pleasant	5.5	6.0
hostile	friendly	7.5	7.5
cumbersome	convenient	6.0	6.0
confusing	clear	6.5	7.0
difficult	easy	7.5	7.0

Table 1: Reactions to security features in the user study on a scale of 1–9 (1 being the most negative, 9 being the most positive).

of others using it. Some participants thought it was very similar to current interfaces and would be easily learned, while others thought it would not be easy for a computer novice or that only people who had been affected by malware would want to use it.

Several participants commented that they would be very motivated to use our system because they had been affected by viruses or were concerned about being affected.

There was disagreement among participants about how desktop file icons should be displayed. Some preferred that only icons for the active role be displayed, whereas others wanted all icons to be displayed at all times. One participant said she did not store any files on the desktop, so it did not matter.

Similarly, some participants thought that dividing files among the roles would be very helpful in organizing files and finding them later, but others did not.

Most participants did not think the roles we used in the study were ideal. Of those who specified why, all said that they wanted fewer roles. One only wanted two roles, one for "files I absolutely knew were safe" and the other for "files that were suspect." A real implementation should allow users to add and remove roles.

Our observations of participants using the system and their comments about it confirmed our belief that a role-based desktop was workable. However, it also highlighted the importance of making the low-level UI mechanisms convenient. To determine the best mechanisms to use for role-related operations, we conducted a second user study.

3.2 Second User Study: UI Security Mechanisms

The goal of this study was to compare different methods of performing key role-related operations, such as copying files from one role to another. Knowing how users perceive different methods will help us design our interface to be learnable, efficient, and convenient.

3.2.1 Participants

We recruited participants from the general student body at Carnegie Mellon University. Four men and two women, ranging in age from 19 to 25, participated in the study. Each participant was paid for their time.

3.2.2 Equipment

The interface was again a paper prototype. The questionnaire at the end of the study was filled out on a desktop computer using a web browser.

3.2.3 Procedure

We described the general concept of a paper prototype and how to interact with it. Next, the participant read the same interface overview used in the first study. Participants were encouraged to think aloud. Unlike the first study, a tutorial was unnecessary since the experimental tasks were given as step-by-step instructions.

Each experimental task consisted of a high-level goal and 2–4 methods for reaching the goal. Each participant performed all methods for each task. An abridged description of the tasks and methods follows[2]:

1. Open a URL in an email message (which is in the Communications role) in the Unsafe role.
   A. Copy to clipboard with context menu and use activation option in Security Manager menu (see Figures 4 and 5).
   B. Use context menu in application and select "Activate in/Unsafe".
2. Move the "Test.doc" file from the Communications to the Personal role.
   A. Drag and drop to the Security Manager and use a directory dialog box (see Figure 3).
   B. Copy to clipboard, transfer clipboard to new role with the Security Manager (see Figures 4 and 5), and paste.
   C. Use context menu in application, select "Copy to/ Personal", and use a directory dialog box.
   D. Change to new role, open "My Documents", and drag and drop file from old role to new role's "My Documents".
3. Save an open text file that's being edited into the Communications role.
   A. Use menu item "File/Save to/Communications" and use Save As dialog.
   B. Use menu item "File/Save as" to save the file, then move it to the Communications role.
   C. Use menu item "File/Save as" and open Roles to save direct to Communications role (see Figure 3).
4. You are in the Communications role. You want to open the file "to do" that is in the Personal role.
   A. Files for all roles are shown, with role-appropriate colored borders. Double-click the "to do" file.
   B. Files for all roles are shown, with role-appropriate colored borders, but icons for non-active roles are dimmed. Double-click the "to do" file.
   C. Only files for active role are shown. Change to Personal role, then Double-click the "to do" file.
5. Some applications are installed in more than one role. You are in the Communications role and want to open the web browser in the Personal role.
   A. There is one web browser icon on-screen, with a border in the active role color. Change to the Personal role; the web browser icon border changes to match. Double-click to start the browser.
   B. Several icons for the web browser are visible, each with a border of a different role's color. Double-click the Personal one to open it.
6. You want to send a file to a friend. The file is in the Personal role, but the email you want to reply to is in the Communications role.

[2]Participants were given more detailed instructions.

PID	Task	1		2				3			4			5		6		
	Rank	1	2	1	2	3	4	1	2	3	1	2	3	1	2	1	2	3
1	Method	B	A	C	D	B	A	C	A	B	A	C	B	A	B	C	A	B
2		B	A	A	D	C	B	A	C	B	A	B	C	A	B	B	C	A
3		B	A	A	B	C	D	AB	AB	C	C	B	A	A	B	C	B	A
4		B	A	D	A	C	B	C	A	B	C	B	A	B	A	B	C	A
5		B	A	CD	CD	A	B	B	C	A	C	B	A	A	B	C	A	B
6		B	A	A	D	C	B	A	C	B	AB	AB	C	A	B	A	C	B
	Winner	B		A				A			C			A		C		

Table 2: Method rankings by participants in second study. Two methods together (e.g., "AB") indicate that the participant ranked them equally.

A. Move the file to the Communications role. Use the "Attach" button in the email client and the Open dialog box that appears.

B. Click the "Attach" button. Open Roles to load directly from the Personal role.

C. Move the file to the Communications role. Drag and drop the file into the email message.

For tasks with two or three methods, we balanced the method ordering across participants. This was not possible for task 4 because it has four methods, so we chose unique orders for the participants to be as different as possible.

Upon completion of each task, the participant was asked to rank the methods by preference. As in the first study, participants filled out a web-based questionnaire at the end. Participants took 45–70 minutes to complete the experiment.

3.2.4 Results

The ratings of the security features in this study were comparable to those from the first study (see Table 1). We also received comments from participants verbally and on the questionnaire, although fewer—perhaps because the tasks were simpler and step-by-step. The biggest concern participants expressed was that the operations have few steps. The last participant was not worried about getting a virus and thought the roles were extra, unnecessary work. However, the other five were positive about the interface and said they would use it. One said, "I wish some of [your] designs... would be common practice amongst big leading software companies."

Unlike the first study, participants ranked the interface methods for performing each task. The rankings and the winning method for each task are given in Table 2. To determine the winners, we looked at plurality vote and a point-based system (1st = 7 points, 2nd = 5, 3rd = 3, and 4th = 1). Interestingly, both ways yielded the same winner for all tasks, although, as the table shows, the margin of victory varied greatly across tasks. Specifically, the winners for tasks 1 and 5 were clear and for 6 fairly clear, but for 2–4 the results were close. Especially interesting votes were those cast for task 4, in which methods A and C are mostly first and last choices and B is mostly the middle choice.

Some methods shared common interface mechanisms of clipboard use, context menus, drag and drop, and role-aware file dialog boxes. Overall, the strongest preference was against using the clipboard. The most common comment

about specific mechanisms was that they liked drag and drop, although in the rankings it is similar to other popular methods.

4 Non-Experimental Analyses

In addition to measuring user response to the tasks and methods in the second study, we also analyzed them using cognitive walkthrough and keystroke-level modeling (KLM).

4.1 Cognitive Walkthrough

In this portion of the study, we obtained user preference data for several ways of doing the same task that are explicitly compared side by side. However, there are other facets to be considered. Let us assume for the moment that one of the methods we provide is the only correct pathway to the goal. Does the pathway provide enough context and feedback for a novice user who is learning by exploration to figure it out? In order to gain a deeper understanding of this issue, we applied the Cognitive Walkthrough usability method [12]. This analysis does not require participants to test the interface. Instead, the analysts answer a series of four questions about each step of the method:

1. Will the user try to achieve the right effect?
2. Will the user notice that the correct action is available?
3. Will the user associate the correct action with the effect?
4. If the correct action is performed, will the user see that progress is made?

We posit relevant assumptions about pre-existing user knowledge and use these to decide whether each step meets the criteria. If it does not meet the requirements on any of the four questions, the step is classified as a failure. Knowledge that these failures exist can lead to fixes that we may not have thought of otherwise.

Two of our tasks use a method that involves copying something to the clipboard and interacting with the Security Manager to transfer the clipboard contents to another role (1A, 2B). This method fails on question 1 because the next goal to accomplish after copying to the clipboard is unclear. A user learning through exploration will have a tough time if he does not know what effect he is trying to achieve. Similarly, any pathway that requires the user to complete two separate tasks will be confusing. This informs us that a more unified pathway should be used.

A drag-and-drop method appears in three places in our study (2A, 2D, 6C). Interestingly, the success of this action varies depending on the context. We can assume that users are familiar with the Windows metaphor of dragging a file onto an icon. When our interface requires this type of interaction, the step succeeds. We have seen that users frequently prefer dragging and dropping to other kinds of interaction. However, if we attempt to use drag-and-drop in combination with our role system (such as allowing a user to change a file's role by dragging it into a folder belonging to another role), the method will fail because the correct action is not clearly linked to the effect (question 3). Dragging may be the correct action, but there is nothing in the interface to help the user associate that action with the desired effect. It creates an inconsistency with the rest of the interface, where objects in one role cannot interact directly with

objects in another role. Although this was an attempt to add convenience for the user, the result warns against creating an unclear situation. Therefore, drag and drop should not be the only method for transferring files across roles.

Two tasks use a role-aware dialog box for locating a directory or file. Although in both cases the cognitive walkthrough analysis indicated success on all the steps, this method was not very popular among users, because they felt that it involved too many steps. If a method is cognitively correct but still disliked, how can we fix it? One possibility is to reduce the number of steps involved, assuming that this does not result in a new failure. A second possibility is to include multiple correct pathways, each stemming from a valid initial goal on the part of the user.

We looked at two tasks that focused on the way files and applications are displayed rather than the sequence of actions (4, 5). Here, the method preferred by users in both tasks contained a failure. The options that succeed in this analysis take up more desktop space, which several participants disliked. It may be the case that none of the options we provided are adequate. More work is needed for the file display options.

4.2 Keystroke-Level Model

We were also interested in finding out how quickly an experienced user could complete the tasks with each of the methods. A Keystroke-Level Model, which is often applied when the experimenter is concerned with maximizing efficiency, is appropriate for this purpose [3]. We applied the KLM to tasks 1–3 and 6 (i.e., excluding tasks 4 and 5, which focused on the display). To use KLM, we started by creating a fine-grained list of all the necessary operations for each individual method. We assigned each operation a time value (in seconds) based on the rules and constants given in Card, Moran, and Newell. For the button or key-press constant we used the value for an average skilled typist. Because of the low-fidelity nature of our prototype, we chose to use the average value for pointing to a target on the screen rather than measuring each distance and calculating the value using Fitt's Law. In cases where the participants are instructed to use "any method they remember" to accomplish a sub-goal of a larger task, we assumed the method with the shortest value. Using the total time values, we can compare the efficiency and speed across the possible pathways for each task.

The KLM showed that in each case, the method with the fastest time for experienced users was the one selected by our participants most often. In some ways our participants were like experienced users, because they were not required to explore the interface as a novice would. Many users verbally stated that they preferred the quickest method. These data suggest that their perceptions were accurate even though the time differences involved are relatively small (several seconds at most). Also, it shows that this perception was directly related to their preferences.

Using drag-and-drop as part of a method leads to a faster total time. The "Save To" or "Copy To" mechanism, which simply adds one extra step to the standard Windows behavior, has a smaller but similar effect.

The KLM presented evidence that using the role-aware dialog box for saving and opening files may be less efficient than we originally supposed. When we designed this mechanism, we hoped it would save time by allowing the user to access files in other roles with just one button press, while keeping the current role as the default, all within the same window. However, it appears that the extra step does increase the time significantly enough to impact the user. The task takes 2.6 seconds longer with the extra role-aware capability than it would take to use the same dialog

box to save a file in the current role.

One last aspect that may be useful to examine is the proportion of total time which comes from direct operations (e.g., moving the mouse across the screen, clicking) vs. the time taken by mental preparation (a constant which is added to certain types of operations according to the KLM rules). For example, the task where users are asked to transfer a file to a different role (task 2) contains four possible methods. For three of these methods, the total time varies but the time contributed by mental preparation remains the same (2.7 seconds each). The method that uses drag and drop from one folder to another (D) has a larger mental preparation time (4.05 seconds), though the total time is almost exactly the same as that of the method that uses the "Copy to" menu (C). User data shows that method D was listed as one of the top two choices twice as often as method C. The KLM data suggests that forcing the user to move the mouse around the screen and click many times is the more detrimental factor.

5 Discussion

Nearly all participants believed that the approach of using roles on the desktop was a good one, and three-fourths said they would use such an interface. Based on our studies, we have identified the following factors as most important for the acceptance of the role-based desktop:

- Concern about malware
- Ease-of-use of role interface
- Personal organization style

The first two factors are common to many interfaces, especially security interfaces. People who are concerned about malware perceive greater benefit from the interface. The easier the interface is to use, the lower its cost. With further development, we believe we can increase the convenience and therefore acceptability of the system even further.

The last factor derives from the fact that to benefit from the role-based desktop, the user must organize applications and data by role. Some participants in our studies said they already organize their computers in a similar way, and like the idea of improving security by doing something they (almost) already do. However, a few participants seemed resistant to the idea that they would have to organize their files according to role. One participant asked about storing a file in multiple roles simultaneously. Thus far, requiring a file to belong to exactly one role does not seem to be a big limitation, although we will keep this issue in mind in the future.

We found in the second study that it may not be possible to match an interface to everyone's personal organizational style simultaneously. Specifically, in the task about whether to display file icons for all roles simultaneously or to display only those for the active role, participants were clearly divided. In cases such as this, the interface should allow the user to change the view to match their preference. Nearly all participants rated method B (dimming inactive role icons) as their second choice, so it may be a reasonable default since it is most acceptable to the widest audience.

User customization may also be desirable for controlling what happens when files are dragged and dropped on a role icon in the security manager to move them across roles. Currently, the interface brings up a dialog box that allows the user to choose a target folder for the files. Most participants did not comment on this behavior, but a few found

it cumbersome and requested that the files automatically be sent to some predefined location, such as My Documents or the desktop. A simple customization mechanism in this case would be to show a folder dialog box, but provide an option to make the chosen directory the default.

An issue we plan to investigate in the future is how flexible our security model needs to be. We think most users will likely only need the simple, strictly partitioned model we have outlined here, but some may desire more flexibility. Possible extensions include: having files belong to more than one role, which was requested by one participant; mixing files of different roles in the same folder; sharing roles across users; and role-based restriction of non-file actions, such as network access. However, the additional complexity this flexibility may add to the interface could make it too hard to use.

One type of flexibility that was not in the paper prototype but we have planned for software versions is the ability to add, delete, and rename roles. The value of this was confirmed by participants in both studies. This feature will allow users to tune the system somewhat to give the tradeoff they want between usability and security. That is, users who are very concerned about security could have many roles and those who are not so concerned could have few.

A more complex extension to the system would be to automatically create temporary roles for certain actions, such as running just-downloaded executables or opening email attachments. This feature may require application-level knowledge and participation, as well as support from the operating system for an application to create a role and control it.

Although in this paper we have emphasized the usability of our system, we have considered its security as well. Our system ultimately puts trust in the user, since in general the system is not intelligent or knowledgeable enough to make autonomous security decisions. However, this makes it vulnerable to poor user decisions, such as from trickery by malicious software. For example, a malicious program could draw graphics in its window, making it appear to belong to a different role [20]. Although the correct active role would be shown in the Security Manager and the spoofed window would not have privileges of the other role, the user could be tricked into entering sensitive information. A possible solution for our system would be to allow windows from only one role on-screen at a time. However, participants in our study were strongly opposed to that, so we are seeking alternatives. Others have addressed this issue and proposed solutions[20, 21], which may be useful in our system as well.

This project touches on broad issues in HCISEC. One issue is what level of control over security by users is ideal. Users need to be able to control the security of their system. However, we believe one reason current security mechanisms fail is that they require the user to make too many decisions. This issue is difficult because often security is a trade-off that only the user can make. We believe having security based on a small number of roles will significantly reduce the burden of decisions, both by making them less numerous and by making each decision easier. Our experience with the paper prototype confirms this belief.

A similar tension lies in providing the appropriate level of awareness of security to the user. If indicators of role membership, for example, are too visually stimulating, they will be annoying. On the other hand, if they are too subtle, users will have insufficient awareness of security status and possibly perform an operation in a role different from the one they intend to use. In this area, the paper prototype did not provide much guidance because awareness will be

influenced by the entire visual experience, which the prototype only roughly approximates. We anticipate investigating the awareness issue further in future high-fidelity prototypes.

An implementation issue we plan to investigate in this project is where security software should reside. To broadly categorize, three choices are the operating system, the application, and a toolkit or library. On today's common platforms, most security infrastructure is in the operating system. The advantage is that there is a single implementation, but it is harder to make usable because there is no contextual information available (e.g., about the user's high-level goal or task). At the other end of the spectrum is application-level security (e.g., encryption in email clients). Security here can, at least in principle, be much more usable since the application knows much more than the OS about what the user is trying to accomplish. However, it requires each application writer to implement their own security, which increases the opportunity for inconsistency in the interface and bugs in implementation. We believe that more work needs to be done in the middle ground, on toolkits and libraries for security. At this level, the software has some knowledge of the user context, while allowing for a small number of implementations. The current project will allow us to explore what infrastructure application-level software needs to easily implement easy-to-use security.

6 Related Work

This work is related to previous work in three areas: HCI and security, file organization, and security technologies.

6.1 HCI and Security

Our prototype attempts to make security more accessible by incorporating good security practices into the general file organization system. Our motivation stems from the increasing realization that although the end user represents a vulnerable spot in the security chain, blaming the user for security breaches misses the point [13]. The user should not have to face a conflict between the goals of work tasks and the goals of security. As Yee puts it, "a more secure system is more controllable, more reliable, and hence more usable; a more usable system reduces confusion and is thus more likely to be secure [21]." He proposes a set of design principles intended to help designers resolve user interaction dilemmas, with special emphasis on the principle of least authority, which states that it is better for an application to begin with no capabilities and receive only those capabilities it needs to carry out its function. This is not necessarily the only way to design a more secure system [1], but it presents an interesting basis for thought. We concur with Whitten that the design of security interfaces presents challenges different from typical UI design [19]; our system provides another setting in which to explore principles such as Yee's.

CapDesk is an example of an independently developed security system that fits in with Yee's proposed principles [4]. It is implemented with the E programming language, which naturally lends itself to confining the capabilities of software applications. With CapDesk, each application is granted the authority it needs to carry out a task by the user's clicks, which cannot be forged by a malicious program. CapDesk uses dialog boxes to convey authority in a similar way to our system. A security analysis of one aspect of this work (the web browser) found a number of security flaws but concluded that the overall architecture is sound [17].

WindowBox also attempts to combine security with usability [1]. The idea is that while the user may not understand complex security settings, he or she will be able to understand the idea of "absolute physical separation" between multiple desktops. Multiple desktops are the functional equivalent of allowing several users to log in at the same time and easily switch back and forth between their desktop views. Our work elaborates on this idea by emphasizing user testing and analysis. Also, our system departs from the idea of absolute physical separation by allowing multiple roles simultaneously on the desktop. Another important feature is the ability to create a trusted path for data [20]. To make this possible, we attempt to create a strong visual relationship between an unspoofable portion of the screen (the Security Manager) and the data and applications themselves.

6.2 File Organization

Our work draws on existing research into strategies for file organization. While there is plenty of information to be found regarding methods of organizing both electronic and paper files [2, 10, 11, 18], there is much less information available about the way people partition files according to their content, which would have been useful for choosing default roles. The existing work yields insight into the functions of finding and reminding [2], the nature of archives and the relationship between filing and piling [11, 18], and the possibilities of automatic document clustering [10]. Some locations are important because they remind the user to take an action, while others are important because of they represent a grouping. All of these bring home the point that a file's location is quite significant.

6.3 Security Technologies

Our approach is a combination of sandboxing (e.g., [6]) and Role-Based Access Control (RBAC) [5]. The goal of sandboxing is to run untrustworthy programs in a separate "sandbox" environment where they cannot interact with or affect the main system. Many different sandboxing schemes have been proposed; popular sandbox technologies include virtual machines (e.g., [16]) and Java [7]. Our system can be viewed and could be implemented as a collection of sandboxes, one for each role.

RBAC is primarily used for controlling access to a multi-user system where each person has their own role(s) and is allowed to wield only the privileges assigned to their role(s). Our concept of RBAC differs slightly. We are dividing the desktop file system into several different roles which can all be accessed by the same user without further authentication, but the roles cannot be accessed by each other without an explicit action on the part of the user.

7 Conclusions

We have presented a new system for mitigating the damage malware can cause for average users, by partitioning the desktop computer system into roles. Our preliminary user studies and non-experimental analyses indicate that this approach is simple enough to be comprehensible and acceptably easy to use. Most participants (12/16) liked the concept and would be interested in using such a system.

For the next step, we plan to implement a software prototype of the interface. We will use it to continue exploring the best mechanisms for role-management and to evaluate how to best provide role-awareness. After that, we plan to prototype a system with working security and study it in actual use. As it matures, it will provide an opportunity to explore HCISEC guidelines (e.g., [19, 21]).

We believe the development of this system will lead not only to improved desktop security, but will also illuminate larger issues in the design and implementation of usable security interfaces, such as finding appropriate levels of control and awareness of security and easing the implementation of usable security for application writers.

References

[1] Dirk Balfanz and Daniel R. Simon. WindowBox: a simple security model for the connected desktop. In *Proceedings of the 4th USENIX Windows Systems Symposium*, pages 37–48. USENIX, USENIX, August 2000.

[2] Deborah Barreau and Bonnie A. Nardi. Finding and reminding: file organization from the desktop. *ACM SIGCHI Bulletin*, 27(3):39–43, 1995.

[3] Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*, pages 264–265. L. Erlbaum Associates, 1983.

[4] Combex, Inc. Combex, E and CapDesk: POLA for the distributed desktop. `http://www.combex.com/tech/edesk.html`.

[5] David Ferraiolo and Richad Kuhn. Role-based access controls. In *15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[6] Ian Goldberg, David Wagner, Randi Thomas, and Eric A. Brewer. A secure environment for untrusted helper applications. In *USENIX Security Symposium*, San Jose, CA, July 1996. USENIX.

[7] Li Gong. Java 2 platform security architecture. `http://java.sun.com/j2se/1.4.2/docs/guide/security/spec/security-spec.doc.html`, 2002. Sun Microsystems.

[8] Nathaniel S. Good and Aaron Krekelberg. Usability and privacy: a study of Kazaa P2P file-sharing. *Proc. CHI 2002, CHI Letters*, 5(1):137–144, December 2003.

[9] Fred T. Grampp and Robert H. Morris. UNIX operating system security. *AT&T Bell Laboratories Technical Journal*, 63(8):1649–1672, October 1984.

[10] Han-Joon Kim and Sang-Goo Lee. A semi-supervised document clustering technique for information organization. In *Proc. 9th Int'l Conference on Information and Knowledge Management*, pages 30–37. ACM Press, 2000.

[11] Thomas W. Malone. How do people organize their desks?: Implications for the design of office information systems. *ACM Transactions on Information Systems (TOIS)*, 1(1):99–112, 1983.

[12] Peter G. Polson, Clayton Lewis, John Rieman, and Cathleen Wharton. Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *International Journal of Man-Machine Studies*, 36(5):741–773, 1992.

[13] M. A. Sasse, S. Brostoff, and D. Weirich. Tranforming the 'weakest link'—a human/computer interaction approach to usable and effective security. *BT Technical Journal*, 19(3):122–131, 2001.

[14] Bruce Schneier. *Secrets & Lies: Digital Security in a Networked World*. John Wiley & Sons, New York, NY, 2000.

[15] Karanjit S. Siyan. *Windows NT Server 4 : Professional Reference*. New Riders Publishing, Indianapolis, IN, 2nd edition, 1997.

[16] VMware, Inc. Vmware. `http://www.vmware.com/`, 2003.

[17] David Wagner and Dean Tribble. A security analysis of the Combex DarpaBrowser architecture. DARPA-sponsored security review. Available at `http://www.combex.com/papers/darpa-review/security-review.html`.

[18] Steve Whittaker and Julia Hirschberg. The character, value, and management of personal paper archives. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 8(2):150–170, 2001.

[19] Alma Whitten and J. D. Tygar. Usability of security: A case study. Technical Report CMU-CS-98-155, Carnegie Mellon University, Pittsburgh, PA 15213, December 1998.

[20] Zishuang (Eileen) Ye and Sean Smith. Trusted paths for browsers. In *Proceedings of the 11th USENIX Security Symposium*, pages 263–279. USENIX Association, 2002.

[21] Ka-Ping Yee. User interaction design for secure systems. Technical Report CSD-02-1184, U.C. Berkeley, Berkeley, CA, May 2002.