Sesame: Informing User Security Decisions with System Visualization

Jennifer Stoll, Craig S Tashman, W. Keith Edwards, Kyle Spafford
Georgia Institute of Technology, School of Interactive Computing
Atlanta, Georgia 30332
jstoll@gatech.edu, {craig, keith, kyle}@cc.gatech.edu

ABSTRACT
Non-expert users face a dilemma when making security decisions. Their security often cannot be fully automated for them, yet they generally lack both the motivation and technical knowledge to make informed security decisions on their own. To help users with this dilemma, we present a novel security user interface called Sesame. Sesame uses a concrete, spatial extension of the desktop metaphor to provide users with the security-related, visualized system-level information they need to make more informed decisions. It also provides users with actionable controls to affect a system's security state. Sesame graphically facilitates users' comprehension in making these decisions, and in doing so helps to lower the bar for motivating them to participate in the security of their system. In a controlled study, users with Sesame were found to make fewer errors than a control group which suggests that our novel security interface is a viable alternative approach to helping users with their dilemma.

Author Keywords: Security usability, security interface design, system visualization

ACM Classification: H.5.2 User Interfaces, User-centered design; K.6.5 Management of Computing and Information Systems: Security and Protection

INTRODUCTION
"AVG Update downloader is trying to access the Internet"

"The firewall has blocked Internet access to your computer [FTP] from 192.168.0.105 [TCP Port 57796, Flags: S]"

"[Your] AntiSpyware has detected that the Windows NetBIOS Messenger Service is currently running. (This service should not be confused with the peer-to-peer Windows Messenger service, or MSN Messenger service which are used for Internet Chat). Beginning with Windows XP Service Pack 2, the Windows NetBIOS Messenger service… …What would you like to do?"

The above are examples of actual alerts [15] that users are given from their anti-virus, anti-spyware and firewall tools. While some alerts are purely informational, most require users to make a decision. The choices they face are often "Always", "This one time" or "Never," posing the quandary of whether to suffer through even more messages or perform an action that may be irreversible. Further, the information given to help users make these decisions is often highly technical or vague (e.g., "*Destination IP: 192.168.0.1: DNS*" or "*This program has changed since the last time it ran!*"). Even when tools have a "More Info" button to provide access to more detailed information, that information is often confusing as well.

How do users cope with such security decisions? Some turn to online research in an attempt to comprehend the alerts. This strategy is evident in the multitude of online forums where users ask questions, sharing their collective wisdom about such decisions (e.g., antionline.com or forumz.tomshardware.com). In contrast, some cope by simply ignoring pop-ups or warnings from their security tools [20]. In fact some security books even advise users to turn off the annoying alerts; for example, one self-help security book quips, "*the [stop alerts] button should say Shut Up, You are Driving Me Crazy*" [15].

Simply put, users are asked to make decisions about things they do not understand, based on information that is difficult to comprehend. The poor decision making that (expectedly) is an outcome of this can result in dire consequences [18], including phishing attacks, bot infestations, and various forms of malware

End-user security decisions present a troubling dilemma. On the one hand, because users must be involved in deciding how to balance security risks against the work they want to accomplish, many of these decisions are impossible to effectively automate [2, 3, 4, 5, 6] (e.g., as in the case of personal firewalls). On the other hand, the users who must make these decisions are generally uninterested in security as an end in itself [18] and, as noted, often have little useful information to help them make good decisions [4, 16]. Further, most of these decisions require a level of technical knowledge not possessed by most end-users. The key question this paper then explores is: since users *must* make security decisions (in particular, ones requiring system-level

CHI 2008, April 5–10, 2008, Florence, Italy.

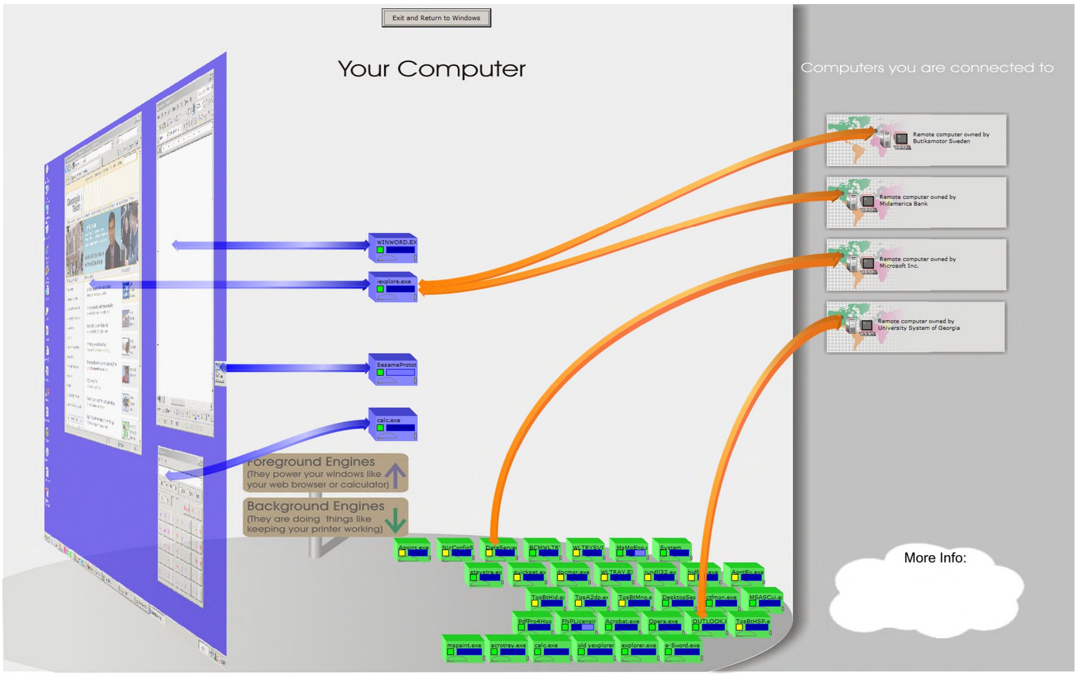

Figure 1: The Sesame 'Behind-the-Scenes' View running on a live system

knowledge), how can we help them understand their system well enough to make better-informed security choices?

We explore this issue of informed security decision making through *Sesame*, an interactive, visual, firewall-like tool designed to assist non-experts in making better informed security decisions. Sesame addresses threats similar to those of consumer firewalls, but uses a visual, direct manipulation interface that exposes system-level information in a meaningful, comprehensible way. To this end, Sesame provides users a 'behind-the-scenes' view of their computer, integrating existing elements of the desktop UI (such as windows) with previously hidden systems-level components (such as the processes that own those windows, the network connections those processes are making, and so forth). Essentially, the desktop metaphor is extended to convey system-level concepts in terms of their relationship to familiar desktop-level abstractions. We conjecture that this view can help to inform user security decisions and, by lowering barriers to this information, may help to motivate them as well [18].

Our initial study results suggest that this behind-the-scenes view of their system's underlying architecture seems to help users make better security decisions involving system-level knowledge. In the sections that follow, we briefly survey current research and commercial security tools. We then describe the design of the Sesame UI and present the results of our user study. We conclude with a discussion of our study results and implications for future work.

RELATED WORK

While there are many security tools available, few are designed specifically to support end-users in their security decision making process. However, of the tools available for security decision making, many can be grouped into two categories: those for expert users and those for non-experts.

Experts

Visualization tools: The majority of prior research focuses on visualization-based approaches intended for the expert user. While inappropriate for non-experts, they do illustrate types of information that experts find useful in order to detect security problems and make informed decisions with regard to security. For example, many of these tools support monitoring of network connections; these include Rumint, IDS Rainstorm, VisAlert [7], and others that provide experts with a variety of useful network data representations [1]. However, these tools are highly technical and complex (presenting data at the level of individual packets), and do not integrate with, or build on, existing metaphors of the desktop GUI.

Text-based tools: Another set of very common tools provides extensive system behavior and status information in a text-based approach. Although these systems often ship on consumer computing platforms, they are generally intended for knowledgeable users or even system- or network-administrators. These include tools like ProcessExplorer, tcpview, and Windows Task Manager [22]. Again, these tools present system-level information useful in making informed security decisions. However, they convey information in a piecemeal fashion (different tools for different information), leaving users to assimilate and make sense of it; and the textual presentation neither supports visual metaphors nor integrates with the existing and well-known features of the desktop UI.

Non-Experts

In contrast to the wealth of tools available for network and security professionals, research in the area of supporting security decision making for end users remains nascent [4, 18]. Discussed below are two classes of such tools.

Tools for specific activities: Some tools for non-experts provide support only for security decisions that relate to a specific user activity, e.g., browsing, searching, making online transactions, sharing files, and so forth [2, 3, 21]. For example, Web Wallet [19] can be used to ensure that a user's information will be sent to the desired site rather than to a spoof. While these tools help support security decisions in the context of a specific activity, they do not address the range of 'system level' security situations that are independent of what the user is doing. Consequently, security decisions which do not map to a specific activity are largely overlooked by these tools. For example, a spyware or bot infection can severely endanger a user's privacy and security regardless of what actions the user is performing [13,9]. Further, since different tools are employed for different activities, the level of security, as well as the user interface, can easily be inconsistent. This is not to say that security tools are only effective if they support *all* security decisions *across* all activities, but rather that activity-specific tools are not a complete answer to end-user security.

Tools for specific threats: Other tools provide support for user decisions based on specific types of threats such as anti-phishing toolbars, anti-virus, -adware, and –spyware systems. These tools tend to be based on heuristics or black lists and will only provide protection for against attacks matching their heuristics or that are on their lists. Users must constantly update these tools in order for them to be effective; and the way in which they alert users to problems is also problematic, as evidence shows that users often find the information in these alerts to be difficult to understand and so ignore them [20, 4].

Consumer firewalls are perhaps the most common types of security software that involves explicit user decision-making. These systems protect against a range of network-based attacks, such as worms that exploit software vulnerabilities. These systems are not task-specific, but present system-level information in a way that is often undecipherable to users (e.g., describing connection attempts in terms of process names, IP addresses, and port numbers). Users therefore have little actionable information on which to base security decisions when faced with firewall popups.

Thus, there are few systems that attempt to provide users with a framework for making security decisions that are not tied directly to a specific activity, but instead help with overall system security. Our goal with Sesame is to fill this void; to provide general, firewall-like security (not, for example, detecting of malware at the point of installation), while helping users to make informed decisions. We focused specifically on handling common classes of attacks where user action is necessary to determine the correct course of action. These include situations where there is no *universally* correct action, but rather a tradeoff between security and convenience that the user must consider, or situations where the correct action is dependent on context outside of the system. Thus, unlike current firewalls—which convey systems-related information textually, using low-level technical details—Sesame provides a holistic graphical representation of security-related system information in a way designed to be understandable by end-users. Sesame also provides a means for exploration of the underlying system, allowing it be used for a greater range of purposes than traditional consumer firewalls.

DESIGN PROCESS

In this section we discuss our iterative design process and choice of the representational paradigm for Sesame.

Representational Paradigm

Our overarching goal with Sesame is to convey a visual model that allows security-related system information to be meaningfully interpreted by the user. To do so, we chose a direct manipulation, model-world paradigm, as this is known to have significant benefits for learnability [8]. Within this interface we sought to 1) show how important but unfamiliar abstractions (e.g., processes and networks) relate to abstractions that are familiar and meaningful to users (e.g., windows and real-world places); and 2) provide users with actionable controls that enable them to affect their security state using the information given.

We believed a spatial, direct-manipulation interface would yield a number of important benefits, the first of which is to *leverage existing knowledge*. Our target, non-expert user already typically interacts with the computer through the spatial metaphor of the desktop GUI, which allows us to leverage that experience to convey other, more complex relationships via a similar spatial GUI. More subtly though, desktop objects are familiar and meaningful to users; by employing an interface that is metaphorically compatible with the desktop, it should be easier to represent relationships between esoteric security abstractions and the familiar windows and things found on the desktop. The second reason to use a visual representation is *speed*. The human visual system has an enormous facility for the rapid assessment of visual scenes [14]. By presenting data in this way, as opposed to a more verbose text-based approach, we hope to minimize the time users need to assess their security situation in any particular instance—thereby reducing one of the known barriers to user motivation for understanding security [18].

Information Content

Portraying *all* low-level information about the state of the user's system would not only potentially be overwhelming, but also likely unnecessary for protection against the classes of threats we are targeting. Thus, we do not attempt to depict all possible system information, but focus instead on a smaller subset selected through an analysis of whether the information is both practically accessible and relevant in addressing the common types of security threats we are targeting (spyware, phishing, and bot infections).

Thus, the information we depict includes:

- Process characteristics, including depictions of which windows are associated with a given process, average and current CPU usage, putative vendor, whether that vendor can be confirmed, and process installation date;

- Network characteristics, including incoming and outgoing connections;
- Remote systems, including domain, putative owning organization, and putative geographical location.

Although far from comprehensive, the above data set provides enough information to identify and potentially mitigate many variants of the three types of attacks on which we focus. Spyware, for example, could be identified by routine attempts at establishing connections from suspicious processes targeted at suspicious remote servers. A bot infection could be detected similarly, except in some cases the remote systems would be initiating the connections. In this sense, Sesame goes further than current firewalls, enabling users to proactively explore security risks. For example, some phishing scams could be identified by observing one's web browser connecting to servers that appear unaffiliated with the nominal proprietor of the website. We used situations like these to evaluate Sesame as described below in our User Study section.

Formative Study

To get feedback on potential approaches for visual presentation and terminology, we conducted a small formative study early in our design process. This formative study was intended to provide feedback on (1) user preferences with respect to concrete versus abstract representations of system information, (2) ability of users to decipher relationships among system information, and (3) preferred terminology for technical concepts. We showed three users paper prototypes of two designs, one using a concrete, spatial visualization similar to that of figure 1, and another using more a more abstract presentation. Participants overwhelmingly preferred the more concrete representation, and understood many of the relationships being conveyed (such as relationships between processes and windows) without explanation by the experimenters.

This brief, early study was also helpful in revealing what participants did *not* understand. For example, participants had difficulty understanding that a part of the visualization represented physically remote computers on the Internet. Our discussions with participants also informed a number of elements of our final design, such as the use of geographic maps to suggest remoteness. Other feedback led to significant changes in the arrangement of processes used in the final version of Sesame. Finally, an additional discovery we made during the formative study was that the term 'process' caused significant confusion for our non-technical users. Based on participant feedback, we chose the term 'engine' as one that made more sense to them. (We acknowledge, however, that using such non-standard terms may confuse the more 'technical' users.)

SESAME: EXTENDING THE DESKTOP

In this section we describe how the system level information is presented in Sesame, and the controls given to users for interacting with the tool.

Invocation - Getting 'behind-the-scenes'

Among the first design choices we faced was deciding where to place Sesame in relation to the desktop. Our requirement to connect existing elements (such as windows) to Sesame-provided elements (such as processes) meant that the tool somehow had to be integrated into the desktop. However it could not appear to be 'just another application' running *on* the desktop, because we needed to convey that Sesame is a level of abstraction 'below' the conventional GUI.

To resolve this challenge, users can access Sesame either by invoking the always-present "Open Sesame" button on the desktop, or through a dialog box that appears when Sesame needs to interact with the user—for example, when a process requests a connection (the user may ignore the dialog box if they do not wish to respond). When Sesame is invoked, the user's desktop GUI rotates about the vertical axis to reveal the processes, networks, and other system-level elements behind it (Figure 1). To exit from the view, users click the exit button at the top of the screen. A benefit of this smooth rotation effect is that it provides a continuous transition from the desktop interface to Sesame to keep users from becoming disoriented.

Rotating the entire desktop to display the Sesame visualization raises several problems. The first is that a full-screen visualization is heavyweight, and may dissuade users from invoking it when they do not have to. It further means that the user cannot have the Sesame visualization visible while performing other tasks. These issues of motivation and efficiency may be important to explore in the future, but at this stage we are principally concerned with effective information representation. A full screen visualization helps us with this by offering conceptual advantages in showing the desktop/system-level separation, as well as the practical advantages of providing abundant space for the system-level components we wish to show.

Division of Space

Once behind the desktop, we faced another design challenge. The effect of rotating the desktop is intended to suggest that the revealed objects are part of the user's PC. Yet to convey abstractions such as remote computers, we also needed to represent areas that lie *outside* the user's PC. Thus, Sesame provides a clean division where one side of the visualization (furthest from the rotated desktop) contains all external elements such as remote computers, and the other side (containing the rotated desktop) represents the internal elements of the user's PC; both regions are labeled and are colored differently to emphasize the distinction.

Visual Elements

In addition to the rotated desktop, the Sesame visualization contains several distinct types of visual elements: processes, remote computers and connection requests. As shown in Figure 1, blue cubes (representing window-owning processes) are connected via arrows to each of the windows on the rotated desktop. Directly beneath them are the nonwindow-owning processes, referred to as ‘background’ processes. On the right, in the ‘external’ region, are representations of remote computers that are connected via arrows to the process with which they are communicating. However, before the remote computers can connect to the user’s system, a connection request is given to the user with options to ‘Allow’ or ‘Forbid’ the connection. Also, in the lower right corner is a ‘More Info’ space providing brief descriptions when the user hovers over the various visual elements.

Processes

Processes are arranged either floating parallel to the rotated desktop or lying on the ground, depending on whether they own windows (‘foreground’ processes) or not (‘background’). To highlight the difference between them, the foreground processes are larger and in blue; the background processes are smaller and in green. This difference in representation was the result of our earlier formative user studies: we found that users preferred significant visual differences between the process types.

To avoid deluging users with processes, only certain ones are shown: 1) we show all processes that own windows as we want to suggest to users that all of their interaction with the computer is mediated by processes; 2) we show all processes that have ever connected to the network; and 3) we cull any process that is a known-safe component of the Windows OS, unless it is being controlled by another, untrusted process.

Much of the security-related, system information provided is actually conveyed with the process cube. On its face is the name of the executable from which it was started. There is also a small square, a rectangular bar graph, and a gauge (Figure 2a). The colored square indicates whether the vendor of the executable could be verified (green for yes, yellow for no); the bar graph indicates how long the executable has been installed on the user’s system; and the gauge indicates both current and average CPU usage.

These three indicators provide users with a fast way of judging if a process is behaving abnormally. Part of Sesame’s design intent is to help users identify abnormal behavior by developing a sense of what normal behavior looks like via the three indicators. To augment their ability to have a sense of ‘normal’ and in keeping with the safe-staging principle [17], we also use in-place semantic zooming, allowing users to learn more information when they want. When users hover over a process cube with the mouse, it expands slightly to show small explanatory text labels next to each of the glyphs on the cube’s face (Figure 2b). To obtain an even more complete explanation, users can click the cube, expanding it to a full-sized representation, which also contains an editable list of security policies applied to the process (Figure 2c). Consequently, users can visually determine whether a given process’ characteristics are unusual, then zoom in to learn what they actually mean.

Coming out from the edges of processes are wide arrows joining them to the windows they own and the remote computers with which they are communicating (Figures 3a, 3b). The intention here is to connect—both conceptually and diagrammatically—concepts in the Sesame interface with familiar related concepts on the user’s desktop and in the real world. The arrows are rendered in bright colors with an extruded, gradated appearance to draw attention and make them feel concrete. We chose double-headed arrows to imply that information flows both ways.

Figure 2a: unexpanded process

Figure 2b: view of process when hovered over

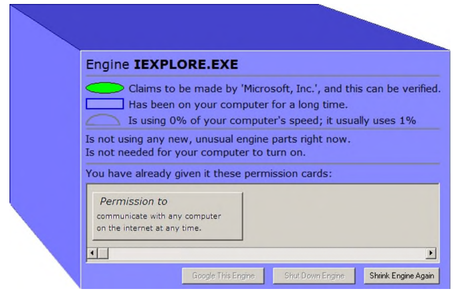

Figure 2c: process when clicked upon

Figure 3a: Process-to-window arrow

Figure 3b: Process-to-remote system arrow

Remote Computers

The remote computers that act as either servers or, less likely, clients to the user’s computer are represented as stacked rectangular tiles on the right of the screen. In earlier versions of Sesame, we tried to depict the remote computers as actual concrete renderings of computers; but as it was not clear to users that these were separate from their PCs. Users tended to feel that a better way to suggest distantness would be to use maps, which prompted us to render remote computers as abstract map images overlaid with pointers to the putative location of the remote system. The arrows leading from the processes to the remote system tiles terminate over the geography of the remote system, and the panels include a textual description (e.g., “Remote computer owned by XYZ Corp”).

As with processes, remote system tiles also allow semantic zooming. When the mouse hovers over a tile, it expands to reveal the domain, owner, and more precise geographic location associated with the remote system; the information is obtained via a reverse DNS lookup and query of a

Figure 4: expanded remote system tile

WHOIS database of domain registrations (Figure 4).

Connection Requests

When an attempt is made to establish a new incoming or outgoing connection, the firewall component of the Sesame backend intercepts the request and Sesame asks the user for permission. Sesame represents the connection attempt by showing a double headed arrow between the local process and remote system tile, but the arrow is solid only on the side initiating the request, with the other end of the arrow dotted (Figure 5). There is a gap between the two ends of the arrow to suggest that they could be, but are not yet, connected. To make the user's options more clear, the buttons where the user can accept or forbid the request appear at this solid/dotted junction. In order to draw the user's attention to the choice, the red border around the buttons flashes for several seconds after Sesame is invoked. To ensure that users see the process associated with the connection request, a border flashes around it as well.

Like current firewalls, Sesame allows the user to set policies for allowing or denying connections; but unlike most firewalls, we try to make changes affecting future policy obvious by representing policies concretely, and visually associating them with the objects to which they apply. When the user selects the 'accept' or 'forbid' buttons, a policy dialog box is shown animating out of the relevant process cube and expanding to full size on screen. The user can select from four policy choices, allowing or denying connections on a one time basis or indefinitely, and with respect to any remote computer or just those associated with a particular domain. Once the user confirms their policy choice, the selected policy (represented as a card) is animated to disappear directly into the associated process, conveying that the policy is 'inside' that cube and is later accessible from it.

Usage Scenarios

To demonstrate how Sesame might be used, we include two usage scenarios in which we show similar situations, one where the appropriate action is to allow a connection so that the user can continue with her task; and one where the appropriate action is to deny it in order to fend off an attack. In the first, the user is asked to answer an explicit question by Sesame about whether to allow access to a remote system. In the second, the user invokes Sesame herself to try to decide whether a website is fraudulent.

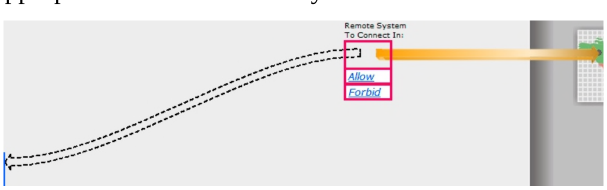

Figure 5: Remote computer requesting connection to local process.

Scenario 1: Should I allow this connection?

In the first scenario, the user is browsing a web page containing an embedded video. Upon selecting the 'Play' button in the embedded video player, Sesame brings up a dialog box indicating a remote computer is trying to initiate communication with a local process, and the user is directed to click a button in the dialog box to open Sesame. When clicked, the user's screen rotates to reveal that the process, to which the remote system is trying to connect, is the same process that is running her web browser. Noting the temporal coincidence between selecting the 'Play' button and the appearance of the dialog box, she suspects that the new connection is needed to allow the video to play. Unsure if watching the video is worth the risk of letting a remote system connect to her computer, she hovers over the remote system and observes that it is owned by the same company associated with the website—a company she is familiar with and feels reasonably comfortable trusting. She clicks the 'Allow' button to permit the connection to proceed. However, as she does not feel comfortable giving the company free reign to connect to her computer, she assigns a permission card allowing the connection just once. The yellow connection arrow now becomes completely solid and the user clicks the 'Exit and Return to Windows' button which rotates her desktop back to its normal position and the video plays as expected.

Scenario 2: Is this a phishing site?

In the second scenario, we begin with a user who is directed to a website claiming to be affiliated with their bank. When the site asks for personal information, the user becomes suspicious and invokes Sesame. Sesame shows the rotated desktop allowing the user to see which process is connected to their web browser window. They notice that the same process is connected to one remote computer. The small map shows that the remote computer is located in an unfamiliar geography and the name of the owner of the remote computer seems unrelated to the user's bank. Still uncertain, the user moves their mouse to the remote computer to find out more information. When the mouse hovers over the remote system, it zooms to show it is associated with an unknown domain, and located at an address in a distant country. The user becomes concerned that they may not really have visited their bank's website, and therefore closes their browser and calls the bank.

IMPLEMENTATION

Sesame is implemented in C++ and runs on unmodified Microsoft Windows XP. All screenshots in this paper were taken from live execution of the system. The front-end interface of the system uses the standard windows GDI to render most graphics, with GDI+ used for more complex effects, such as the gradient fills on the extruded arrows. The front-end also includes a series of hooks allowing it to be driven from synthetic data sets, as this capability was necessary for our user study. In order to modify the datasets in real time, Sesame registers several global hotkeys allowing us to use a second keyboard to control the data Sesame shows to study participants. The Sesame front-end gathers process list data; the back-end is used to retrieve network status. It also includes a simple open-source firewall that is used to intercept attempts at initiating network connections, and performs reverse-DNS and WHOIS lookups to obtain information about remote systems. As we did not need complete functionality for our evaluations, certain backend features related to firewall operations, process culling, CPU usage, and vender verification were not fully implemented, but were mocked up in the front end.

USER STUDY

We conducted a study to evaluate how well users could judge potential security threats using the visualization provided by Sesame. Our primary interest was in how well users could cumulatively leverage the concepts presented in Sesame to make security-related decisions. We chose to evaluate these concepts together rather than piecemeal, as the experience of our formative study suggested that the different concepts underlying Sesame are heavily interdependent. To perform this cumulative evaluation, we asked users to make security decisions in an environment with Sesame, as well as a more conventional environment using the ZoneAlarm firewall instead. To assess users' understanding of the different ideas in Sesame, we asked various questions about different parts of the visualization and about why users made particular choices. Note that our focus in this study was to understand how the quality of decision-making was affected by our visual interface, not on evaluating Sesame's efficiency as a UI (whether it allows faster decision-making), or its impact on user motivation (whether it incents people to be more active in security management); we leave these as future work.

Participants

Our study included a total of 20 participants recruited from a university campus: 45% female (9 subjects) and 55% male (11 subjects). All were undergraduate students; none were computer science or computer engineering majors, and none considered themselves to be experts in computer operation. While those pursuing undergraduate degrees may not ideally represent typical users, we found that our participants were indeed unfamiliar with basic concepts of computer operation such as processes, and security threats such as phishing. We therefore felt they were adequately representative. We used between-subjects testing, so participants were divided evenly between control and experimental groups.

Security Tasks

To determine how well participants could identify security threats, we gave them several tasks. For each task, they effectively had to judge whether a given situation posed a security threat. The tasks were based on the types of decisions users must make in real world use; the first four were common personal firewall configuration decisions, the latter two required judging the authenticity of websites. The tasks included: T1) allow or forbid an incoming connection from Microsoft.com after clicking on a video player link; T2) allow or forbid an outgoing connection from a spyware process named loadsys.exe; T3) allow or forbid an incoming connection from a bot server requesting connection with a bot process named intmonp.exe; T4) allow or forbid an outgoing connection from the process named outlook.exe; T5) determine if the website claiming to be Mid America Bank is a phishing site; T6) determine if the website claiming to be CitiBank is a phishing site. T5 and T6 are intentionally similar but whereas T5 represents a threat, T6 does not—this allows us to test for false positives as well as false negatives. We balanced the tasks between situations that are triggered by user activity (T1, T5, and T6) and those with no direct relationship to the user's actions (T2, T3, and T4).

Participants performed the security tasks in either the control or the experimental environment. The control environment was made to reflect the most common configuration used by non-experts [15], a typical personal computer with the ZoneAlarm firewall. In addition to its ubiquity, we used ZoneAlarm in the control because, like Sesame, it provides firewall functions and therefore offers many of the same actionable controls, and asks many of the same questions, as Sesame. It also embodies the textual, indirect interaction paradigm employed in most end-user security tools. Further, the firewall tasks (T1-T4) were derived from actual prompts ZoneAlarm presents to users. Also, rather than use a live installation of ZoneAlarm, we made an interactive mockup using actual screenshots from ZoneAlarm alerts—this was to ensure that we could control the timing and exact content of the dialog boxes. We integrated interactive widgets into our mockup as well so that it would behave, as well as appear, virtually identical to the actual tool.

Participants in the experimental group used the same system as those in the control, except that Sesame was provided in place of ZoneAlarm. Because Sesame offers an unusual user interface, we gave participants about 90 seconds to explore it before beginning the tasks. We provided no explanation about the meaning of Sesame's behind-the-desktop visual elements; we only demonstrated how one could hover over and click on some objects to semantically zoom in on them.

Similarly to ZoneAlarm and other firewalls, for tasks T1-T4, Sesame alerted the user to the security situation by bringing up a dialog box. From there the user would click a button to enter the Sesame UI. For T5-T6, the experimenter explained the situation and asked the user to click the always-present "Open Sesame" button to enter the Sesame interface.

Study Details

As each participant began the study, they were asked a series of background questions to assess their familiarity with computers and security tools. Before any tasks were presented to them, participants were instructed to think aloud about what information they would be using in making their decisions. All participants used the same Dell laptop with a 15” monitor running Windows XP Home edition. An external keyboard was used by the experimenters to bring up the alerts for each security task. The participants were asked to make security decisions as if the laptop being used was their own.

As discussed above, for the first security task participants were asked to download a video from the Microsoft website which caused an alert dialog box from the firewall to pop-up. After completing the first task, participants were instructed to browse to any website of their choosing. They were then presented with the next three tasks while they browsed online. Upon the completion of each task, the participant continued browsing until the next one was presented to them. We were careful during these tasks not to cause a popup to appear immediately after the user took an action (e.g., clicking a link) so that they would not erroneously think they caused the popup in those cases where they did not. For the final two tasks, none of the participants were familiar with the term ‘phishing site’ so a brief explanation was given.

Since we were assessing the intelligibility of the Sesame visualization, we declined to answer participant questions about the meanings of Sesame’s visual elements. We asked them to infer as best they could with the information given.

Participants were asked to think aloud during their decision making process for the six tasks, in order to aid us in understanding their reasoning processes. Upon completion of all tasks participants were asked follow-up questions in a semi-structured interview regarding the choices they made. We asked about the clarity of the decisions they had to make, the choices presented to them and the information provided by the security tool they used (i.e., Sesame or ZoneAlarm). We also asked for suggestions for improving the security tools. Each subject participated in the study individually and was voice recorded to capture responses to the interview questions and the think aloud.

RESULTS

For each participant, we gathered the following data: 1) background information, 2) the participant’s ‘miss-rate’ for the six tasks, and 3) responses to follow-up questions regarding comprehension of the system information provided. We define miss-rate to be the number of potential threats the participant evaluated erroneously (i.e., judging a threatening situation to be safe or vice versa).

Participant security background: Based on the background information gathered, both the control and experimental groups were similar in terms of experience with computers and security tools. Participants in the control group had an average of 10.4 years of experience using the computer and only 2 out of 10 participants did not use any security tool (e.g. Norton Anti-virus, firewalls or anti-spyware). Similarly, participants in the experimental group had an average of 9.2 years of experience using the computer and 2 out of 10 did not use any security tools. Subjects in both groups were asked to briefly describe their security practices besides using security software.

Based on the responses in both groups, the majority of the participants did not have or could not recall additional security practices they performed besides making occasional updates to their security software when alerted to do so by their tools. Further, it seems that the amount of time spent using a computer does not necessarily translate into familiarity with the system knowledge needed to make security decisions. Although participants’ computer usage averaged 9-10 years, all were unfamiliar with basic system-level concepts, e.g., processes, network connections.

Security task miss-rate: Since our data were non-parametric, discrete, and did not appear to fit a known distribution, we used a two-tailed Mann-Whitney test to determine if the difference in the miss-rate between the ZoneAlarm (control) and Sesame (experimental) groups was statistically significant. In the table below, we summarize the miss rate for both groups of participants:

	ZoneAlarm group miss-rate (6 males, 4 females)	***Sesame group miss-rate (5 males, 5 females)***
Females	45.8%	25% (with outlier* 30%)
Males	36%	20%
Miss-rate	40%	22.2% (with outlier* 26.6%)

**Outlier miss-rate: 66.7%*

Table 1. Participant miss-rate for security tasks

Of the 20 participants, we identified one outlier whose miss-rate was 66.7%. During the study, this participant expressed concern about completing the tasks because s/he was very unfamiliar with the Windows environment given that the participant’s primary computer was a Macintosh.

Cumulatively over all of the tasks, the experimental group performed significantly better than the control group. Including the outlier, the miss-rate for the group using Sesame is 26.6% while the miss-rate is 40% for the control group (Z=1.97, P=0.05). (When the outlier is excluded, the Sesame group miss-rate is reduced further to 22.2%, Z=2.53, P=0.05). Over just the firewall-like tasks (T1-T4), the experimental group using Sesame performed 41% better than the control group; but if the outlier is included, the experimental group performs just 20% better. However, when tasks were considered individually or in smaller sets, we did not have statistically significant differences between groups due to our relatively small sample size. Figure 6 shows the success rates by task, including the outlier.

Information Comprehension: After the completion of the six security tasks, we interviewed participants to understand their reactions to the tasks they performed and, for the experimental group, to assess how much of Sesame they understood. Most of the participants in the control group stated that they were uncertain how to use the information provided by ZoneAlarm in making their security decisions. Also, many of the participants (seven out of ten) said they

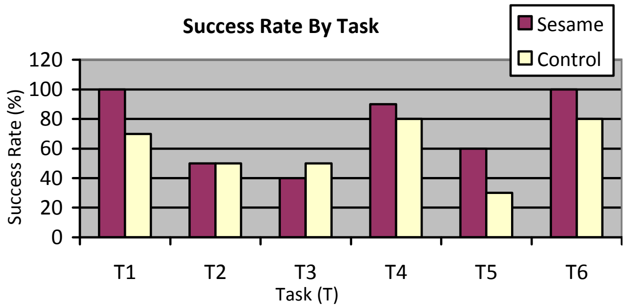

Figure 6: Participant hit-rates for security tasks

were following a specific strategy in making allow/deny decisions. Five of the seven were either allowing or denying every connection request. For example, one participant said *"I don't want to read all this stuff so I'm just going to deny it. If I allow it, I may have to do something else."* The remaining two of the seven allowed or denied based on whether they could recognize the process name. Even though all the participants had access to a lot of information provided by Zone Alarm, only five participants in total looked at the process name given to help with their decision. The *more info* button was consistently ignored.

Cumulatively, as discussed above, the experimental group performed significantly better than the control group. One contributing factor for this may have been that none of the participants reported using a predetermined strategy in making decisions. But perhaps more importantly, Sesame users appeared to employ more of the information provided than the control group in making their allow/deny decisions. Sesame users reported using information about the processes themselves as well as the remote systems to which they were connecting. In contrast, control users tended to rely on process names only. We suspect that this may be because Sesame provided a more accessible explanation of the general significance and specific facts about remote systems. For example, five participants found the geography information provided on remote systems to be helpful. Participants also reported that the type of language used in Sesame was helpful, as well as the permission cards in the allow/deny dialog box—though it is not clear that the latter specifically aided them in performing the tasks.

To assess the extent to which participants understood the conceptual model Sesame provides, we asked them to describe the visualization. We specifically inquired about the nature of the visual elements such as processes, and about the distinction between areas representing things within the computer versus outside of the computer. We found that 8 of the 10 users understood the basic significance of the cubes representing foreground processes. Background processes posed a challenge, with only 2 users understanding their purpose. We were surprised by this finding, as we explicitly included a textual description of the distinction between the two types of processes within the UI. The representations of the remote computers were understood well, with 8 participants recognizing their meaning. Eight participants also understood the basic purpose of the arrows connecting processes to the remote computers. Finally, 8 of the 10 participants were also able to identify the areas of the Sesame UI that represented things considered to be within the computer, versus the areas representing things that were 'outside' the computer.

DISCUSSION AND FUTURE WORK

The results of our study are encouraging, suggesting that on the whole, Sesame's novel UI helps users make better security decisions than with typical security environments with a traditional firewall. We were especially pleased that our representation could be reasonably well interpreted without explanation. E.g., nearly all users understood the division between the internal and external regions of the design—this was a particular challenge we faced in earlier prototypes. The results suggest our basic representational approach to be a viable alternative to conventional textual approaches; and that novice users seem to rapidly learn a system-level structure when it is framed visually and in terms of more familiar concepts as was done in Sesame. While there is evidence that Sesame's visual presentation is more effective than traditional firewalls, additional, larger studies are needed to confirm this.

Besides the seemingly successful aspects of Sesame, the shortcomings of the UI were also informative. A common difficulty for participants was inferring causal relationships. Many participants felt that actions they took caused Sesame to bring up dialog boxes, even in cases where the Sesame visualization itself gave indications to the contrary. Additionally, participant comments suggested they often had difficulty grasping the idea of their computer's software environment as a collection of quasi-independent causal agents, instead inferring strong relationships among the different processes. These difficulties suggest that future versions of Sesame might use metaphors that better suggest the agent-causal nature of processes—such as depicting them with animated, anthropomorphic figures.

While we evaluated Sesame as a holistic combination of its features, we do believe that there are generalizable design principles from our experiences that could be applied to other systems.

First, given our desire to provide a direct manipulation interface, swiveling the desktop to show the underlying system seemed to be an accessible way to provide context to help users understand otherwise hidden features. That there are other possible approaches is certain; however, we believe that the generalizable principle here is that contextualization of new information with familiar, known concepts is a key for non-expert use. Our swivel metaphor is one (but not the only) way one might accomplish this.

Second, Sesame's policy cards provide a persistent visual indication of policy settings, allowing them to be easily seen and accessed by users in the future. The utility of persistent visual indicators—to support awareness of system state, to serve as an affordance for reversibility, and so forth—seems particularly important for conveying security state, where re-accessibility and intelligibility are especially difficult. We believe this to be a severe failing of current firewalls, in which settings—once configured—are often difficult to re-access.

Third, we believe that metaphorical 3D models may aid in helping non-experts understand system information where necessary; particularly when it involves concepts underlying the desktop metaphor. Such modeling need not be limited to a firewall UI. Like file management models (e.g. two file folders with pages transferring from one to the other to show the status of copying), other system models can be embedded in applications, perhaps with individual components rotating aside to reveal relevant underlying system information. Our study suggests that security information embedded in such a model can be effectively leveraged by end-users.

Finally, Sesame shows the viability of direct manipulation (DM) in low level, security and configuration interfaces, whereas DM is traditionally only used for windowing and within applications. We believe that this area is ripe for the use of other specific design techniques from the HCI community, such as the use of 2.5D UI's to extend the desktop metaphor. Our work suggests that users can understand relationships between different levels of abstraction through such well-proved techniques, even when dealing with the complex information necessary for security and system decision-making.

CONCLUSION

Sesame brings a direct manipulation graphical interface to end-user security to help non-experts make better informed security choices. Most prior work in end-user security tended to be highly task or threat-specific; or largely text-based and/or designed for experts. There are few tools if any which focus explicitly on helping non-expert users to better understand the technical, system-level concepts needed to make security decisions. With Sesame, we investigate an approach to providing non-experts with a general, firewall-like tool that addresses a wider range of threats independent of specific tasks or applications.

Further, we move toward making system-level concepts accessible to non-experts by representing them as concrete objects and relating them to more familiar concepts such as desktop-level objects and real world abstractions like geographic locations. In a controlled study, Sesame users were more likely to identify security threats accurately than users with more typical software environments. Sesame users were further able to understand many of the otherwise unfamiliar system-level concepts conveyed, suggesting viability to our fundamental interface approach. In the future, we plan to conduct further studies and explore how better to convey those concepts with which users struggled.

ACKNOWLEDGEMENTS

We thank our colleagues at Georgia Tech for their helpful feedback, in particular, John Stasko and Pixi Lab members; and we thank Symantec for their support for this work.

REFERENCES

1. Conti, G., Abdullah, K., Grizzard, J., Stasko, J., Copeland, J., Ahamad, M., Owen, H., Lee, C. Countering Security Information Overload through Alert and Pack Visualization. IEEE Computer Graphics (2006).
2. Dhamija, Rachna, Tygar, J.Doug. *The Battle Against Phishing: Dynamic Security Skins.* Symposium On Usable Privacy and Security, (2005).
3. DiGioia, P., Dourish *P. Social Navigation as a Model for Usable Security.* Symposium On Usable Privacy and Security (2005).
4. Downs, J. S., Holbrook, M. B., Cranor, L. F. *Decision Strategies and Susceptibility to Phishing.* Symposium On Usable Privacy and Security, (2005).
5. Edwards, W. K., Shehan, E., Stoll, J. *Security Automation Considered Harmful?* NSPW (2007)
6. Flinn, S.A., *Flock of Birds, Safely Staged.* DIMACS Workshop on Usable Privacy & Security Software (2005).
7. Foresti, S., Agutter, J. *Visual Correlation of Network Alerts.* IEEE Computer Graphics (2006).
8. Hutchins, E., Hollan, J., Norman, D. *Direct Manipulation Interfaces.* Human Computer Interaction, 1985. 1: p. 311-338.
9. *Know Your Enemy: Tracking Botnets.* Honeynet Project and Research Alliance. honeynet.org/papers/bots (2005).
10. Nielsen, J., Landauer, T. K., *A mathematical model of the finding of usability problems*. Proceedings of the ACM INTERCHI'93 Conference (1993).
11. Shukla, S., Nah, F., *Web Browsing and Spyware Intrusion*. Communications of the ACM.Vol. 48, No. 8 (2005).
12. Smetters, D., Grinter, R. *Moving from the Design of Usable Security Technologies to the Design of Useful Secure Applications.* NSPW (2002).
13. *Spyware.* NISCC Technical Note. National Infrastructure Security Coordination Centre. (2006).
14. Thorpe, S., Fize, D. & Marlot, C. (1996).Speed of processing in the human visual system. Nature, 381, 520-522.
15. Walker, A. *Absolute Beginner's Guide to Security, Spam, Spyware & Viruses*. Que Publishing, © 2006.
16. Whalen, T., Inkpen, K. *Techniques for Visual Feedback of Security State.* DIMACS Workshop on Usable Privacy and Security Software (2004).
17. Whitten, A., Tygar, J. *Safe Security Staging.* CHI 2003 Workshop on Human-Computer Interaction and Security Systems (2003).
18. Whitten, A., Tygar, J., *Why Johnny Can't Encrypt*. Proc. of the 8th USENIX Security Symposium (1999).
19. Wu, M., Miller, R. C., Little, G. *Web Wallet: Preventing Phishing Attacks by Revealing User Intentions*. Symposium On Usable Privacy and Security, (2006).
20. Wu, M., Miller, R. C., Garfinkel, S., *Do Security Toolbars Actually Prevent Phishing Attacks?* CHI (2006).
21. Yee, K., Sitaker, K. *Passpet: Convenient Password Management and Phishing Protection.* Symposium On Usable Privacy and Security, (2006).
22. www.sysinternals.com/Utilities/