Improving Security Decisions with Polymorphic and Audited Dialogs

José Carlos Brustoloni and
Ricardo Villamarín-Salomón
Department of Computer Science
University of Pittsburgh
210 S. Bouquet St. #6111
Pittsburgh, PA 15260

{jcb,rvillsal}@cs.pitt.edu

ABSTRACT

Context-sensitive guidance (CSG) can help users make better security decisions. Applications with CSG ask the user to provide relevant context information. Based on such information, these applications then decide or suggest an appropriate course of action. However, users often deem security dialogs irrelevant to the tasks they are performing and try to evade them. This paper contributes two new techniques for hardening CSG against automatic and false user answers. Polymorphic dialogs continuously change the form of required user inputs and intentionally delay the latter, forcing users to pay attention to security decisions. Audited dialogs thwart false user answers by (1) warning users that their answers will be forwarded to auditors, and (2) allowing auditors to quarantine users who provide unjustified answers. We implemented CSG against email-borne viruses on the Thunderbird email agent. One version, CSG-PD, includes CSG and polymorphic dialogs. Another version, CSG-PAD, includes CSG and both polymorphic and audited dialogs. In user studies, we found that untrained users accept significantly less unjustified risks with CSG-PD than with conventional dialogs. Moreover, they accept significantly less unjustified risks with CSG-PAD than with CSG-PD. CSG-PD and CSG-PAD have insignificant effect on acceptance of justified risks.

Categories and Subject Descriptors

H.5.2 [**Information Systems**]: User Interfaces—*interaction styles, screen design, evaluation*; H.1.2 [**Information Systems**]: User/Machine Systems—*human factors*; I.3.6 [**Computing Methodologies**]: Methodologies and Techniques—*interaction techniques*

General Terms

Security, Human Factors.

Keywords

E-mail client, attachment, virus propagation, context-sensitive guidance, polymorphic dialogs, audited dialogs.

Symposium On Usable Privacy and Security (SOUPS) 2007, July 18-20, 2007, Pittsburgh, PA, USA.

1. INTRODUCTION

Computer applications often need to make context-dependent security decisions. Consider, for example, an email agent, such as Mozilla Thunderbird [1]. If a user receives an email attachment containing a Microsoft Word file, should the email agent allow the user to open it or not? Many viruses propagate by exploiting Word vulnerabilities [2]. Anti-virus software may not detect new virus strains. The risk of opening the attachment may be justified if the user knows the sender and is expecting such a document from her, but not otherwise [3,4].

In such situations, an application usually needs user input, because the application cannot determine automatically all the context relevant to the security decision. For example, in the above scenario, Thunderbird would not know whether the user is expecting the attachment.

Some applications translate this need into complete delegation of the security decision to the user. The application warns the user of the risk and asks whether the user wants to accept it. We call such dialogs *warn-and-continue* (W&C). Unfortunately, users typically do not know how to make such decisions responsibly. More often than not, users click continue automatically. Users thereby accept many unjustified risks that enable successful attacks [5].

Given the futility of warning users, many applications hide security decisions by abolishing warnings or dialogs altogether. For example, Thunderbird allows the user to cancel or save an attachment instead of opening it, but does not warn if an attachment is unsafe. We call this approach *no-warning* (NW). In contrast, recent versions of Microsoft Outlook completely hide unsafe attachments. By default, users cannot open or save such attachments [6]. We call this approach *no-dialog* (ND). ND trades off usability for security, while NW makes the opposite tradeoff.

Context-sensitive guidance (CSG) can help applications achieve a better balance between security and usability than do ND, NW and W&C. Applications with CSG ask the user to provide context information necessary for a security decision. Based on such information, these applications then decide or suggest an appropriate course of action [7].

However, if CSG accepts unverified user inputs that enable users to continue, CSG can become as insecure as W&C or NW. Many users quickly learn how to circumvent security mechanisms that stop them from accomplishing what they want.

This paper contributes two new techniques for hardening CSG against automatic and false user answers. *Polymorphic dialogs*

continuously change the form of required user inputs and intentionally delay the latter, forcing users to pay attention to security decisions. *Audited dialogs* thwart false user answers by (1) warning users that their answers will be forwarded to auditors, and (2) allowing auditors to quarantine users who provide unjustified answers. Quarantined users may be subject to a variety of sanctions, such as being unable to use the application for increasing periods of time, having to pay increasing fines, or having to pass remedial training.

We implemented CSG on Thunderbird. One version, CSG-PD, includes CSG and polymorphic dialogs. Another version, CSG-PAD, includes CSG and both polymorphic and audited dialogs. In user studies, we found that untrained users accept significantly less unjustified risks with CSG-PD than with NW dialogs. Moreover, they accept significantly less unjustified risks with CSG-PAD than with CSG-PD. These effects are not due to simple risk aversion: CSG-PD and CSG-PAD had insignificant effect on acceptance of justified risks.

Experimental results also suggest that CSG-PD can provide secondary benefits by *reducing* task completion time relative to NW. Unjustified risks are typically not task-related. By reducing acceptance of such risks, CSG-PD can also reduce wasted time. On the other hand, audited dialogs can reverse such gains if audits temporarily suspend users. In our experiments, CSG-PAD had insignificant effect on task completion time relative to NW.

The rest of this paper is organized as follows. Section 2 details our threat model. Sections 3, 4, and 5 describe the application of CSG against email-borne viruses and design of polymorphic and audited dialogs, respectively. Section 6 briefly describes our prototype implementation. Sections 7 and 8 present our evaluation methodology and experimental results. Section 9 discusses those results and Section 10 comments on related work. Section 11 concludes.

2. THREAT MODEL

This section details our threat model.

CSG enables an organization to embed in its members' computer applications the organization's policies for classifying risks. Members should accept only risks that the organization's policies consider *justified*, and avoid *unjustified* ones. The organization may be, e.g., a governmental, military, or commercial entity. Risks may originate from outside or inside the organization.

CSG handles primarily risks whose evaluation requires inputs that computers cannot obtain automatically. CSG obtains such inputs from users. CSG therefore complements more automated defenses, such as firewalls, anti-virus software, and intrusion detection systems. Given that none of these defenses is infallible, they often need to be combined in a layered security approach.

The primary threat against CSG is that users may not provide legitimate inputs. Users often deem security dialogs irrelevant to the tasks they are performing and try to evade them [5,8]. Polymorphic and audited dialogs seek to mitigate this risk.

CSG and polymorphic and audited dialogs assume that neither attackers nor users can disable or spoof them, e.g., by reconfiguring, modifying, substituting, or extending applications. Audited dialogs also assume that the respective applications have access to a private or secret key that attackers and users cannot access. An organization may, e.g., use operating system protection mechanisms to reserve such privileges to system administrators. Additionally, an organization may use mechanisms such as Trusted Network Connect [9,10] to verify the configuration of a member's computer whenever the latter attempts to connect to the organization's network.

This paper illustrates the use of CSG and polymorphic and audited dialogs against email-borne virus propagation. We assume that attackers may wish to infiltrate an organization's computers with malware that firewalls and anti-virus software do not detect. Such malware may be new and specially targeted against the organization, thus thwarting signature-based detection.

Audited dialogs require an organization's privacy policies to grant the organization's auditors the right to read members' answers and context information relevant to security decisions (e.g., email messages and attachments). An organization's members might attempt to evade auditing by using instead external (e.g., Web-based) email accounts. We assume that an organization's firewalls block direct communication between members' computers and external email servers. Such privacy policies and blocking are quite common in corporate environments.

3. CONTEXT-SENSITIVE GUIDANCE AGAINST EMAIL-BORNE VIRUSES

This section describes our application of CSG against email-borne viruses.

CSG modifies an email agent so that an organization can define certain attachment types to be risky. For example, an organization may consider Word files risky because many viruses exploit Word vulnerabilities to propagate. For each risky attachment type, CSG allows the organization to define a decision tree for classifying the risk as justified or not. The email agent transforms this decision tree into dialogs that are presented on a sidebar when the user clicks on a risky attachment. The email agent allows the user to open or save the attachment only if, according to the organization's decision tree and the user's answers, the risk is justified.

Typically, an organization would implement its policies by modifying a CSG template that comes with the application. We implemented such a template for Thunderbird. Our template policy condenses advice from several sources [3,4].

According to our template policy, when a user clicks on an attachment that is a risky document (e.g., Word file), Thunderbird notifies the user that the attachment might contain a virus. The dialog asks whether the user: (a) does not wish to open the attachment; (b) finds the attachment suspicious but is curious about it; or (c) has reasons to expect a message and attachment like those from that sender to that account (see Fig. 1).

If the user selects (a), our template policy aborts the operation immediately. If the user selects (b), the template also eventually aborts the operation. However, it first asks context information that enables it to suggest what the user can do to better evaluate the risk, while reinforcing alignment between the user's and the organization's understanding of unjustified risks. The template asks the user whether: (b1) he does not use that account to communicate with that sender (e.g., it's usually a good idea to

maintain separate accounts for business and personal email); (b2) the message refers to something, such as a meeting, that the user does not remember; (b3) the message is unusually short or contains errors that the user would not expect the sender to make; (b4) the message does not convincingly explain the purpose of the attachment; (b5) the attachment seems out of character for the sender; (b6) the sender is technical or customer support; or (b7) other (see Fig. 2). According to the user's answer, the template explains why the organization considers the risk unjustified and suggests what the user can do to better evaluate the risk. For example, if the user does not remember a reference in a message (option b2), the template explains in simple language that this is a common ploy that attackers use, and asks the user to verify the reference by other means (see Fig. 3). After user confirmation, the template aborts the operation. The user can later retry the operation, hopefully after following the received guidance.

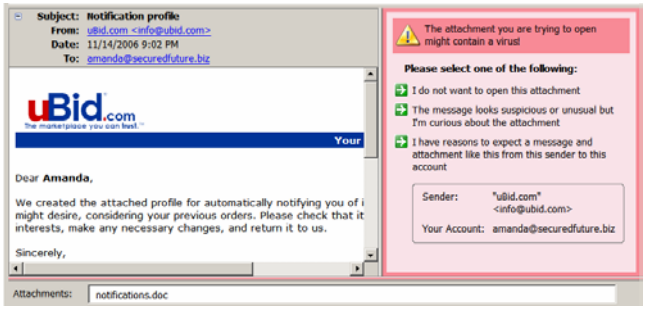


Figure 1. CSG alerts user that an attachment might be infected

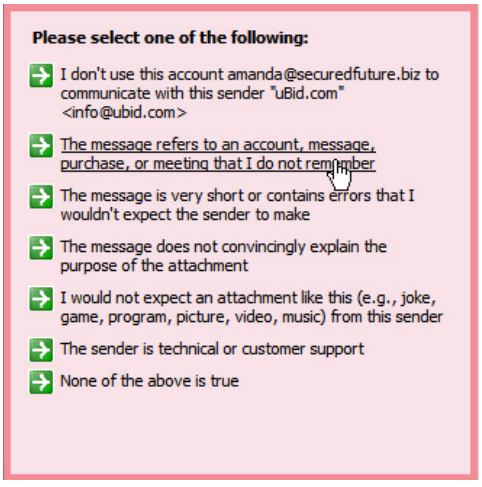


Figure 2. CSG options when user is curious about the attachment

Figure 3. CSG dialog when message references something that user does not remember

If the user selected instead (c), our template policy still attempts to ensure that the user did not make a mistake. The template asks if: (c1) the user does not wish to open the attachment; (c2) does not know the sender; (c3) the sender is technical or customer support; (c4) the message refers to something, such as a meeting, that the user does not remember; or (c5) the user does wish to open the attachment (see Fig. 4). If the user selects (c1), the template aborts the operation immediately. If the user selects instead (c2), the template asks whether the user: (c2a) does not wish to open the attachment, or (c2b) would like the application to ask the sender to retransmit the attachment in a safer document type (e.g., ASCII text – see Fig. 5). If the user selects instead (c3), the template asks whether the user: (c3a) does not wish to open the attachment, or (c3b) has verified by other means (e.g., phone) that the sender did send an attachment like that (attackers often impersonate technical or customer support, even though the latter usually avoid sending risky attachments – see Fig. 6). If the user selects instead (c4) or (c5), the template respectively presents the dialog in Fig. 3 or opens the attachment.

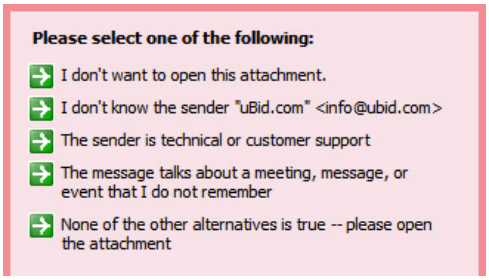


Figure 4. CSG options when user expected attachment from sender to account

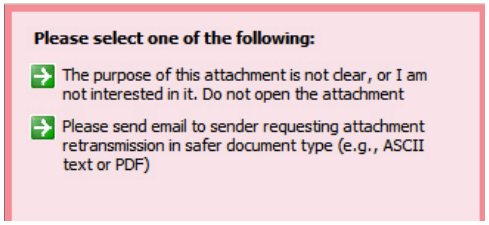


Figure 5. CSG options when user does not know sender

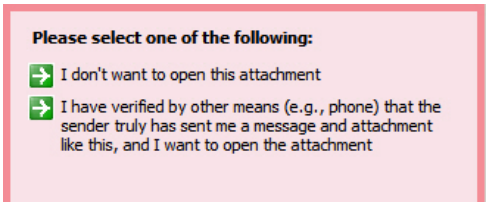


Figure 6. CSG options when sender is technical or customer support

The entire decision tree of our template policy is presented in Appendix A. Different organizations would modify such a template to implement their own policies.

4. POLYMORPHIC DIALOGS

This section discusses the design of polymorphic dialogs.

CSG's security depends on the truthfulness of users' answers. Two factors may conspire against such truthfulness. First, after a hyperlink arouses a user's interest, the user often regards any intervening dialogs as meaningless formalities. The user will often give any responses that seem necessary to get the target object, even false ones, as long as the effort required for those responses does not exceed the user's interest in the object. Second, many dialogs are such that users almost always need to give the same answer. Repetition can condition users to give that answer automatically, even when it is false. Automatic answers reinforce perceptions of dialogs as mere formalities and reduce effort for getting objects.

Polymorphic dialogs attempt to improve the truthfulness of users' answers by combating automatic answers. In a polymorphic dialog, each answer's form continuously changes. These changes can make automatic answers' effects less predictable and force users to respond more attentively. Polymorphic dialogs also delay and increase effort necessary for response. Greater effort may moderate users' interest and propensity to give false answers.

The design space for polymorphic dialogs is vast. This paper considers only two simple dialog changes. First, when a dialog includes two or more options, they are displayed in random order, as illustrated in Fig. 7. Users cannot automatically find a certain option always at the same place in the dialog. Second, the final option that confirms an operation (e.g., option c5 in Fig. 4, which opens the attachment) becomes active only after the respective dialog has been displayed for some time. This delay encourages users to consider the dialog's other options. Users can identify a polymorphic dialog by its use of distinctive arrows for selecting options, as shown in Figs. 1 to 7.

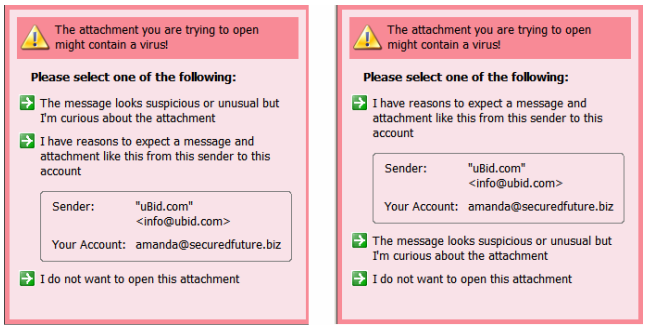


Figure 7. A polymorphic dialog varies the order of its options each time the dialog is displayed

5. AUDITED DIALOGS

This section describes the design of audited dialogs.

Audited dialogs seek to hold users accountable for the truthfulness of their answers. It makes three types of modification in an application. First, each dialog that accepts user input is modified to notify users that their answers may be audited. For example, the dialog in Fig. 8 is the audited version of the dialog in Fig. 6. Second, a final confirmation dialog is added. This dialog notifies the user that confirmation of the operation will cause the user's answers and its context (e.g., message and attachment) to be forwarded to the organization's auditors. This dialog also summarizes possible consequences to the user if auditors find that the user's answers are unjustified in the respective context (see Fig. 9). For example, auditors may suspend the user, require the user to pay a fine, or require the user to pass remedial training. Third, the application is modified to enable auditors to suspend the user for a specified amount of time. While a user is suspended, the user cannot use the application normally (see Fig. 10). The application will only display the auditors' notice and explanation of failed audit and penalties (see Fig. 11).

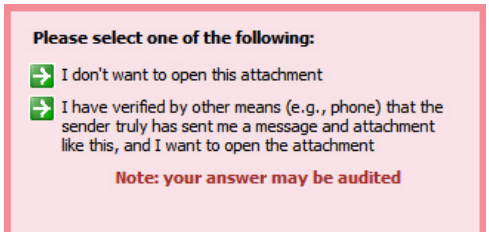


Figure 8. Audited version of the dialog in Fig. 6

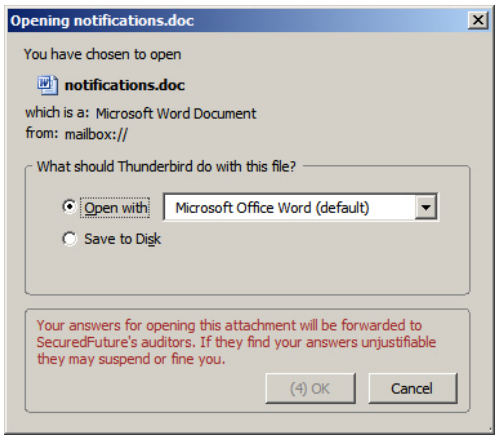


Figure 9. Final confirmation dialog for operation and forwarding user's answers to organization's auditors

Audited dialogs require an authenticated channel between each member's application and the organization's auditors. An email agent can implement such a channel by automatically adding or verifying a signature (if using public-key cryptography) or message authentication code (if using shared secrets) to the messages sent between the application and auditors.

Penalties for accepting unjustified risks may monotonically increase with each subsequent violation. For example, the user may be suspended for increasing periods, and after a certain number of violations may also need to pay increasing fines.

Auditing members' answers can be labor-intensive. It typically can be performed only on small samples. Auditors can reduce effort by sending members *training messages* containing attachments that auditors *a priori* consider unjustified risks. Judging members' responses to training messages can be automated and therefore may be easier than evaluating responses to other messages. Training messages also encroach less on users' privacy.

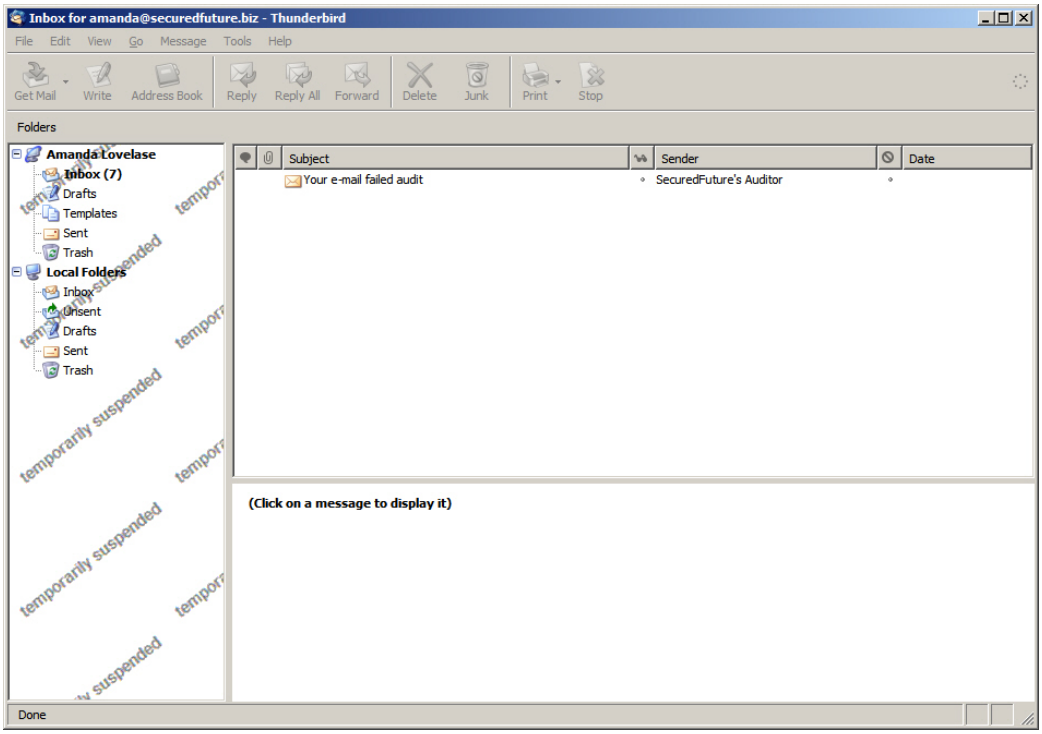


Figure 10. Thunderbird's screen while user is suspended. The user can access the auditors' notice of failed audit and penalties, but no other messages.

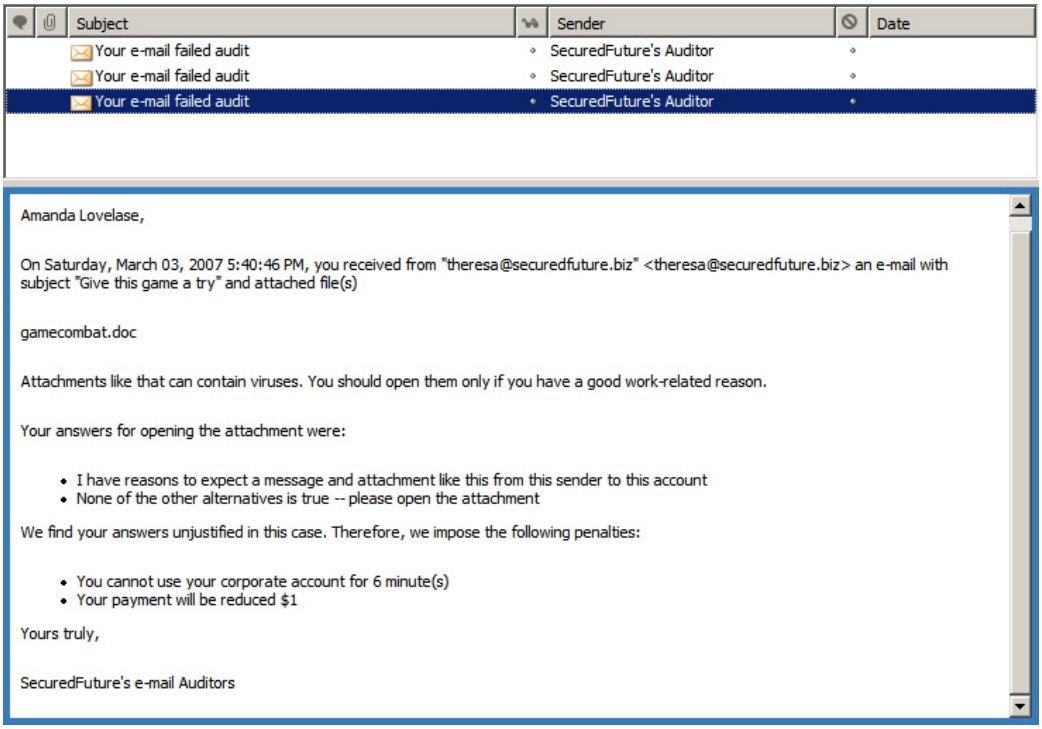


Figure 11. Notice and explanation of failed audit and penalties in Thunderbird, after user's acceptance of unjustified risk for a third time

6. PROTOTYPE IMPLEMENTATION

This section briefly describes our prototype implementation.

We prototyped the defenses described in Sections 3, 4, and 5 (CSG against email-borne viruses and polymorphic and audited dialogs) as a Mozilla extension for Thunderbird on a PC running Windows XP. A configuration option selects NW (Thunderbird's default dialogs), CSG-PD (CSG with polymorphic dialogs) or CSG-PAD (CSG with polymorphic and audited dialogs). We implemented user interfaces in XUL, Mozilla's XML User Interface Language, and processing logic in JavaScript. These choices facilitate porting our extension to other platforms where Thunderbird runs.

7. EVALUATION METHODOLOGY

This section explains how we evaluated our techniques.

7.1 Experimental design

We performed a user study to compare NW, CSG-PD, and CSG-PAD dialogs. The user study involved two similar scenarios, A and B. Each scenario comprises the same counts of justified and unjustified risks. We divided participants into two groups, CSG-PD and CSG-PAD. Participants in the CSG-PD group performed one of the scenarios (randomly selected) using NW dialogs, and then the other scenario using CSG-PD. Similarly, participants in the CSG-PAD group performed one of the scenarios (randomly selected) using NW dialogs, and then the other scenario using CSG-PAD.

We measured: (1) the counts of justified and unjustified risks each participant accepted with each type of dialog, and (2) the time each participant took for completing a scenario's tasks with each type of dialog. We performed paired t-tests to test the significance of differences between measurements for: (1) NW and CSG-PD and (2) NW and CSG-PAD. To test significance of differences between measurements for CSG-PD and CSG-PAD, we performed an unpaired t-test.

We randomly selected the order in which participants performed the two scenarios to avoid order-induced biases. On the other hand, participants always used NW first to avoid learning effects. Participants were already familiar with NW at the beginning of the study. Consequently, there is nothing new that participants might have learned from NW and applied to CSG-PD or CSG-PAD. The converse, however, would be true: CSG-PD and CSG-PAD do provide guidance that participants might learn and apply to NW. For the same reason, no participant used both CSG-PD and CSG-PAD.

Both CSG-PD and CSG-PAD impose higher costs for accepting an unjustified risk than does NW. Users need to learn these costs. After repeated use, users might also learn how to reduce these costs. To shed light on these learning processes, we grouped unjustified risks by their order of appearance in each scenario. We calculated the frequency with which risks appearing in a certain order were accepted by participants using each type of dialog. We define *net acceptance frequency* as the difference between corresponding acceptance frequencies with CSG-PD or CSG-PAD and NW. Net acceptance frequency can be used as a proxy of users' cost estimate. We plot the net acceptance frequencies vs. order of unjustified risk to investigate how cost estimates evolve with continued use of a type of dialog.

7.2 Scenarios

In each scenario, the participant is asked to role-play an employee of a company. The participant receives initially a handout briefly describing the employee and coworkers (including some personal details), the company, and tasks the employee is involved in. In scenario A, the employee is selecting applicants for a job at the company. In scenario B, the employee needs to process customers' insurance claims. After the participant has read a scenario's handout, we ask the participant to process the respective employee's email messages at the company's email server. In each scenario, the employee's inbox contains 10 unread messages, each containing a Word attachment. The first and sixth messages' attachments are needed in work-related tasks and therefore pose justified risks. The remaining messages' attachments pose unjustified risks. We consider that a participant accepts or rejects an attachment's risk by respectively opening the attachment or not.

Appendix B contains the handouts for the scenarios.

7.3 Participants

Participants in the user study were at least 20 years old and had previous work experience where they needed to use an email agent, such as Outlook or Thunderbird. We considered such experience necessary for participants to be able play the assigned roles faithfully. We required participants to be native English speakers or have similar proficiency, so as to rule out linguistic difficulties that might, e.g., cause a participant to miss nuances or errors that suggest that a message is spoofed. We excluded from the study Computer Science or Engineering students or graduates, whose greater familiarity with computers might cause them to process email differently from the general population.

We recruited participants by distributing flyers around the University of Pittsburgh's campus, posting in Pittsburgh jobs newsgroups, posting in http://pittsburgh.craigslist.org and http://pittsburgh.backpage.com volunteering sections, and publishing a printed ad in Pittsburgh's City Paper.

After recruitment, we excluded from the study participants who accepted less than half of the unjustified risks in the scenario they performed with NW dialogs. These participants did not perform a second scenario with CSG-PD or CSG-PAD. These participants' performance suggests that they were well-trained on email security before the user study. CSG is not intended for such users. Instead, CSG is designed to help *untrained* users make better decisions. This criterion excluded 6 of 26 participants recruited for the study (23%). Excluded participants accepted on average 31% of unjustified risks (σ= 10%, min=12%, max=38%).

Table 1 summarizes characteristics of the resulting participant groups (where "SR" denotes self-reported). Different numbers of participants were necessary in each group to reach statistically significant results. A majority of participants were female. This fact was unplanned and we do not assign any particular

significance to it. The table suggests that the groups were roughly similar.

Table 1. Characteristics of the participant groups

	CSG-PD	CSG-PAD
# Participants	13	7
# Female	10	6
# Male	3	1
Familiarity with email agents (SR)	4.1 / 5	3.9 / 5
Ease of user study tasks (SR)	4.5 / 5	4.3 / 5
# Unjustified risks accepted w/ NW	79%	66%

7.4 Laboratory sessions

Each participant role-played the two scenarios described in Section 7.2 in an individually scheduled laboratory session using the prototype described in Section 6. Each participant's session lasted between 40 and 110 minutes. Participants received between $15 and $22 compensation for their time.

We took notes and recorded the participant's computer screen, face, and voice. These recordings helped us debug the scenarios and prototype before the user study, and thereafter helped us confirm counts and tasks completion times. We did not record participant names or other personal information. We report only aggregate results.

We tested CSG-PAD with the following penalty policy. On the first violation, the participant was suspended for 3 minutes. On the second violation, the participant was suspended for 6 minutes. For each subsequent violation, the participant was suspended for 6 minutes and $1 was subtracted from the participant's compensation. For consistent testing conditions, we programmed the prototype to automatically detect acceptance of an unjustified risk and generate the corresponding user suspension message 7 seconds thereafter. We included in the prototype mechanisms for preventing suspension circumvention by restarting Thunderbird.

8. EXPERIMENTAL RESULTS

This section presents our experimental results.

Tables 2, 3, and 4 show the main results of our user study. The noted effect sizes are Cohen's d; values of 0.2, 0.5, and 0.8 are indicative of small, medium, and large effects, respectively [11].

Table 2 shows that, compared to conventional NW dialogs, CSG-PD provides a statistically significant and large reduction in the number of unjustified risks accepted ($d = 1.08$, $p = 0.0022$) and a medium reduction in tasks completion time ($d = 0.59$, $p = 0.053$). The former result is due to CSG and polymorphic dialogs. The latter result is due to the former: because participants using CSG-PD accept fewer unjustified risks (typically task-unrelated), they can also complete assigned tasks earlier. There was insignificant difference in the number of justified risks accepted.

Table 3 shows that, compared to conventional NW dialogs, CSG-PAD provides an even larger and statistically significant reduction in the number of unjustified risks accepted ($d = 2.6$, $p < 0.0005$). CSG-PAD had no effect on the number of justified risks accepted and had insignificant effect on tasks completion time.

Table 2. Comparison between CSG-PD and NW (paired t-test, n = 13)

	mean	std. dev.	eff. size	p-value
# unjustified risks accepted				
CSG-PD	4.15	1.86		
NW	6.31	1.11		
difference	-2.15	1.99	1.08	0.0022
# justified risks accepted				
CSG-PD	1.92	0.28		
NW	2.00	0.00		
difference	-0.08	0.28		not sign.
tasks completion time (minutes)				
CSG-PD	20.35	8.75		
NW	25.34	12.02		
difference	-4.98	8.38	0.59	0.053

Table 3. Comparison between CSG-PAD and NW (paired t-test, n=7)

	mean	std. dev.	eff. size	p-value
# unjustified risks accepted				
CSG-PAD	2.14	1.46		
NW	5.29	0.76		
difference	-3.14	1.21	2.60	< 0.0005
# justified risks accepted				
CSG-PAD	2.00	0.00		
NW	2.00	0.00		
difference	0.00	0.00		not sign.
tasks completion time (minutes)				
CSG-PAD	27.60	9.94		
NW	29.72	19.21		
difference	-2.12	19.83		not sign.

Table 4 compares CSG-PAD with CSG-PD. Pooled standard deviations are noted for differences. We calculated p-values assuming that the two groups have equal or unequal variance, and report the higher result. The table shows that the reduction in the number of unjustified risks accepted with CSG-PAD relative to CSG-PD is statistically significant and large ($d = 1.16$, $p = 0.024$). This difference is due to audited dialogs. There was insignificant difference in justified risks accepted and tasks completion time. The latter result, however, seems to be an artifact of insufficient sample size. CSG-PAD can be more time-consuming than CSG-PD because audits may suspend the user, delaying completion of tasks.

Table 4. Comparison between CSG-PAD and CSG-PD (unpaired t-test, n_1=7, n_2=13)

	mean	std. dev.	eff. size	p-value
# unjustified risks accepted				
CSG-PAD	2.14	1.46		
CSG-PD	4.15	1.86		
difference	-2.01	1.74	1.16	0.024
# justified risks accepted				
CSG-PAD	2.00	0.00		
CSG-PD	1.92	0.28		
difference	0.08	0.28		not sign.
tasks completion time (minutes)				
CSG-PAD	27.60	9.94		
CSG-PD	20.35	8.75		
difference	7.26	9.16		not sign.

Fig. 12 shows how the net acceptance frequency of unjustified risks evolved with continued use of each type of dialog. The graphs show that when users first encountered CSG, the dialog had negligible effect on acceptance of an unjustified risk. Users supplied incorrect answers to get the target object at about the same rate as users got the object with the NW dialog. Effectively, users initially treated the guidance as meaningless. After having accepted 3 unjustified risks with CSG-PD or 2 unjustified risks with CSG-PAD, however, users apparently learned that unjustified risks have higher costs. CSG-PD's higher cost decreases net acceptance frequency for the remaining unjustified risks on average by 36%. CSG-PAD's still higher cost decreases net acceptance frequency for the remaining unjustified risks on average by 58%.

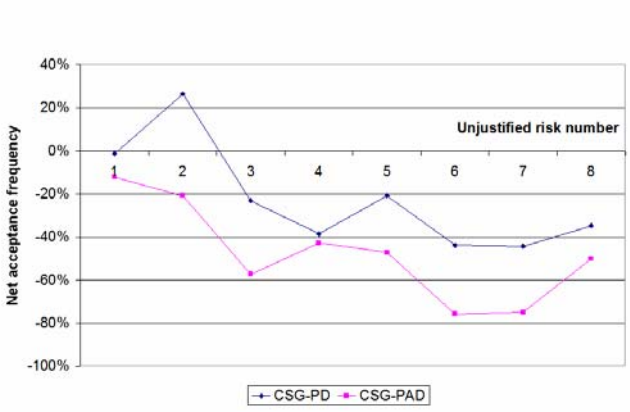


Figure 12. Unjustified-risk net acceptance frequency decreases after user learns costs imposed by CSG-PD or CSG-PAD

Table 5 shows the results of a survey completed by participants at the end of their sessions. Participants found CSG-PD easy to understand and that it provides fairly good guidance, but they didn't really find the questions helpful or follow guidance. They would be fairly comfortable with CSG-PD in the future, but would give friends a neutral recommendation. Although objective measurements show that CSG-PAD is more secure, participants rated its guidance much lower than CSG-PD's. They'd be neutral about receiving CSG-PAD's guidance in the future, but would give friends a negative recommendation. Further questioning revealed that some participants disliked CSG-PAD's penalties or found that they had been applied unfairly. For example, some participants automatically trusted messages received from a coworker and found it hard to conceive that viruses might forge such messages. The auditors' messages did not explain sufficiently well to these participants why they failed audit.

Table 5. Participant perceptions of CSG-PD and CSG-PAD (worst = 1, best = 5)

	CSG-PD	CSG-PAD
Dialogs are easy to understand	3.9	3.7
Questions are helpful	2.4	2.1
Interface provides good guidance	3.6	2.6
Participant followed guidance	2.5	2.4
Would feel comfortable receiving such guidance in future	3.7	3.0
Would recommend to friend	3.1	1.9

9. DISCUSSION

This section discusses the previous section's results, including limitations and pointers for future work.

Because of the difficulty of recruiting participants and time limitations, our user study did not isolate the effects of CSG and polymorphic dialogs. We measured only their aggregate effect. We also explored only very simple dialog polymorphism. Overcoming these limitations would be interesting future work.

CSG-PD significantly reduced time for completing tasks in our scenarios. However, the frequency of unjustified risks in our scenarios was 80%. If unjustified risks were less common, this benefit of CSG-PD would probably disappear. Further studies are needed to characterize this relationship.

We tested CSG-PAD only with a single penalty schedule that was inspired mostly by limitations of our laboratory environment. A possibly less disruptive penalty would be to suspend the user's ability to open attachments while preserving the user's ability to receive message bodies and send messages without attachments. It would be interesting to determine in real-world trials what types of sanctions work best and respective dosage effects.

Security dialogs can become ineffective after repeated use, as users find ways to circumvent them or start dismissing them automatically. Fig. 12 does not suggest such a loss for CSG-PD or CSG-PAD. However, longer studies would be desirable for ascertaining their continued effectiveness after prolonged use.

In Fig. 7, the box indicating the sender and user accounts always follows the option for opening the attachment. Such layout may

facilitate pattern-matching and automatic answers. A possible improvement would be to anchor that box at the bottom of the dialog window.

Table 5 suggests that CSG-PD and CSG-PAD are not technologies that users would individually seek or disseminate. However, it appears that users would accept and be able to use them, if they were adopted by an organization. Audit notifications should be more informative than they were in our study. Audit decisions that users do not understand can generate resentment. The security concepts underlying an audit decision need to be explained in plain language, so that users can learn from the notification itself.

10. RELATED WORK

Some dialogs in previously existing software can be considered polymorphic. For example, the Firefox browser [12] imposes a delay before the user can confirm the command for installing an extension. However, polymorphic dialogs do not seem to have been enunciated before as a general design principle. This paper is possibly also the first to evaluate this technique's impact on security (Table 2), habituation effects (Fig. 12), and user acceptance (Table 5).

In an earlier version of this work, we tested: (1) CSG (without polymorphic dialogs) and (2) CSG with audited dialogs (but no audits) (CSG+A). An initial pilot study (n = 16) did not suggest significant benefits, but it had problems in experimental design, scenarios, and participant selection, as discussed in [7]. Another small pilot study (n = 4) with the present methodology also suggested insignificant benefits. However, analysis of session recordings suggested that users were quickly memorizing the positions of the dialog options that allowed them to continue, and then responding automatically. This observation led to the CSG-PD technique described in this paper. Further analysis suggested that users ignored threats of auditing, and probably would continue to do so unless they actually experienced audits and penalties. This hypothesis led to CSG-PAD in the current form.

Xia and Brustoloni proposed CSG classes called *guidance without override* (GWO) and *guidance with override* (G+O) [5]. In their user studies, GWO provided very high security, while G+O gave significantly less, but still significant, security. GWO is appropriate only for security decisions that an application can itself make and enforce, based on inputs that users find easier to provide legitimately than by forgery (e.g., certificate verification). G+O is used in security decisions that an application can merely suggest or are based on inputs that users can easily forge (e.g., whether to send a password in plaintext). This paper is concerned with the latter class. We found it much harder than Xia and Brustoloni did to obtain significant benefits from G+O. This discrepancy may be due to application differences: the decision of whether to send passwords in plaintext is much easier for users to understand and process than are policies for attachments.

Wu et al. proposed Web Wallet, a browser sidebar for helping users decide whether to send passwords and other credentials to a Web site [13]. Web Wallet's dialogs provide G+O and are highly effective against normal phishing attacks. However, they are specialized to that goal and do not readily apply to decisions about email attachments.

GWO and G+O are subclasses of *just-in-time instruction* (JITI): dialogs explain security concepts only when the user needs to apply them. Whitten and Tygar advocate an opposite approach, *safe staging*: dialogs encourage the user to learn security concepts *before* the user progresses to a stage where those concepts are needed [14]. Xia and Brustoloni's user studies found safe staging as effective as G+O in decisions about sending passwords in plaintext, but less effective than GWO in decisions about accepting unverified certificates [5]. It would be interesting to perform a similar comparison in decisions about email attachments.

JITI and safe staging are subclasses of *embedded training* (ET). Kumaraguru et al. have investigated ET against phishing [15]. In their studies, system administrators intentionally send phishing email to users. When a user clicks on a link in such a message, the email agent automatically displays a message explaining why the user should avoid such links. They found that including graphics in such messages, and particularly presenting them in the format of a comic strip, is much more effective than using text only. We believe that applying similar techniques in auditors' notifications could significantly improve the effectiveness and user acceptance of CSG-PAD.

11. CONCLUSIONS

Policies that do not depend on user input may be secure but inflexible (e.g., Outlook's ND), or flexible but insecure (e.g., Thunderbird's NW). To be both secure and flexible, policies often need to consider context information that can be obtained only from the user. However, designing effective dialogs for eliciting such information can be a formidable challenge. Many users view such dialogs as meaningless obstacles and do not hesitate to give false answers. This paper contributes two new techniques for improving the truthfulness of user answers and the consequent quality of security decisions. Polymorphic dialogs continuously change the form of required user inputs, preventing automatic answers. Audited dialogs hold users accountable by forwarding users' answers to auditors. To illustrate the use of these techniques, we designed a policy and corresponding context-sensitive guidance (CSG) for avoiding virus infection from email attachments. We implemented CSG with polymorphic dialogs (CSG-PD) or with polymorphic and audited dialogs (CSG-PAD) on Thunderbird. Results from a user study show that users accept significantly less unjustified risks with CSG-PAD than with CSG-PD, and significantly less unjustified risks with CSG-PD than with conventional dialogs. Moreover, CSG-PD and CSG-PAD have insignificant effect on acceptance of justified risks. Users quickly adapted to the new dialogs, and we found no evidence of loss of effectiveness after continued use. Users' perception of the new dialogs was lukewarm and it appears unlikely that users would adopt them spontaneously. However, it appears that users would accept them if adopted by an organization. To improve acceptance of CSG-PAD, auditors should proactively explain their decisions in plain language (e.g., on notification messages themselves).

12. ACKNOWLEDGMENTS

Matthew DeSantis from CERT and Ashley Brooks from CMU's Heinz School collaborated in an earlier version of this work [7]. Jason Hong and Lorrie Cranor provided valuable feedback on that version. We also thank comments from Thomas Carey and anonymous referees. This research was partly supported by NSF grant ANI-0325353.

13. REFERENCES

[1] Mozilla. "Thunderbird – Reclaim your inbox," http://www.mozilla.com/en-US/thunderbird/

[2] US-CERT. "Microsoft Word Vulnerability," Technical Cyber Security Alert TA06-139A, May, 2006, http://www.us-cert.gov/cas/techalerts/TA06-139A.html

[3] L. Rogers. "Use Care When Reading Email with Attachments," news@sei, vol. 6, no. 3, SEI, CMU, 2003, http://www.sei.cmu.edu/news-at-sei/columns/security_matters/2003/3q03/security-matters-3q03.htm

[4] US-CERT. "Using Caution with Email Attachments," Cyber Security Tip ST04-010, 2004, http://www.us-cert.gov/cas/tips/ST04-010.html

[5] H. Xia and J. Brustoloni. "Hardening Web Browsers Against Man-in-the-Middle and Eavesdropping Attacks," in *Proc. 14th International World Wide Web Conference (WWW2005)*, ACM, pp. 489-497, May 2005.

[6] W. Kennedy. "Blocked Attachments: The Outlook Feature You Love to Hate," Microsoft, http://office.microsoft.com/en-us/outlook/HA011894211033.aspx

[7] R. Villamarín-Salomón, J. Brustoloni, M. DeSantis and A. Brooks. "Improving User Decisions About Opening Potentially Dangerous Attachments In E-Mail Clients," Poster, Symposium on Usable Privacy and Security, CMU, July 2006.

[8] L. Cranor and S. Garfinkel (eds.). "Security and Usability – Designing Secure Systems That People Can Use." O'Reilly, 2005.

[9] Trusted Computing Group. "Trusted Network Connect." https://www.trustedcomputinggroup.org/groups/network/

[10] H. Xia, J. Kanchana and J. Brustoloni. "Using Secure Coprocessors to Protect Access to Enterprise Networks," in *Proceedings of the Networking'2005 Conference*, IFIP, Lecture Notes in Computer Science, 3462:154-165, Springer-Verlag, May 2005.

[11] J. Cohen. "Statistical Power Analysis for the Behavioral Sciences," Lawrence Erlbaum, Hillsdale, NJ, 1988.

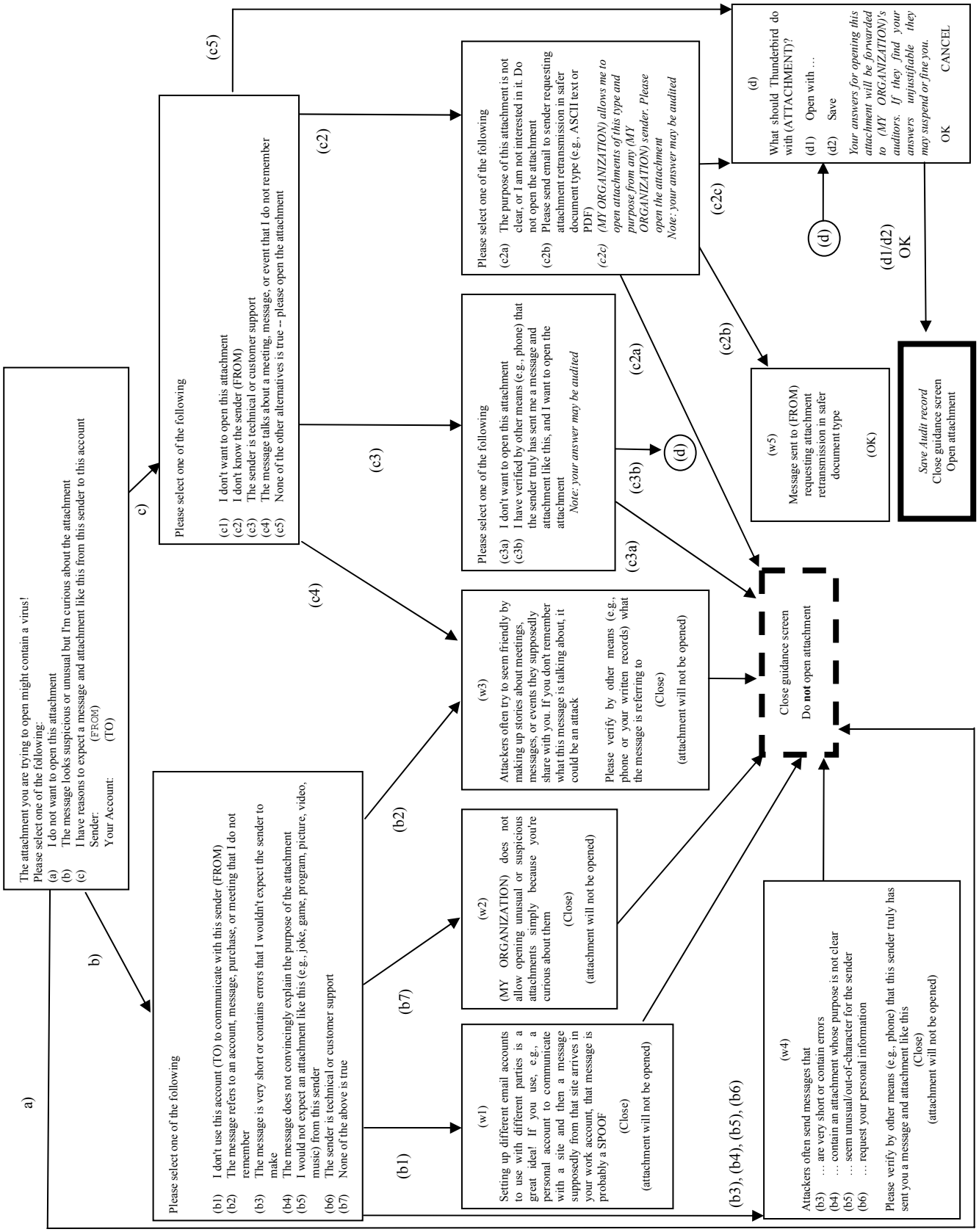
[12] Mozilla. "Firefox – Rediscover the Web," http://www.mozilla.com/en-US/firefox/

[13] M.Wu, R. Miller and G. Little. "Web Wallet: Preventing Phishing Attacks by Revealing User Intentions," in *Proc. Symposium on Usable Privacy and Security*, CMU, July 2006.

[14] A. Whitten and J. D. Tygar. "Safe Staging for Computer Security," in *Proc. Workshop on Human-Computer Interaction and Security Systems*, CHI'2003, ACM, April 2003.

[15] P. Kumaraguru, Y. Rhee, A. Acquisti, L. Cranor, J. Hong and E. Nunge. "Protecting People from Phishing: The Design and Evaluation of an Embedded Training Email System," in *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI'07)*, ACM, April 2007.

Appendix A. Template policy decision tree



Appendix B. User Study Handouts

Scenario #1

You will be role-playing Chris, an office worker at a company called ACME

Chris works in a group dedicated to evaluation of credit card applicants. The other members of his group are:

- Alex: always meticulous and precise in her writing
- Bob: Always serious
- Frank: happy and carefree

Chris has two email accounts, **chris@acmecorp.biz**, which he uses for work-related messages, and **chris679@gmail.com**, which he uses for private messages.

You are to check Chris' inbox and do the following tasks:

Task 1

Chris wants to hire another worker. He advertised the position in work websites and is expecting resumes from applicants (whom he does not know). Chris needs to pick the applicant with most years of experience and write down her/his name.

Task 2

Finish processing (delete right away, read, answer, etc. messages) Chris' inbox's messages.

Additional information

If Chris needs help with his computer, he can send a message to techsupport@acmecorp.biz or contact Tech Support by phone. Chris always uses GMail for his account at priceline.com and PNCBank and for any other private communication. Chris recently ordered a getaway weekend from priceline.com. He travels next week.

Scenario #2

You are going to role-play Amanda Lovelase, an accountant working for SecuredFuture (SF), an insurance company that accepts claims in electronic format.

Amanda's only known people at SF are:

- Henry Buffett, an insurance specialist, who communicates verbosely.
- Theresa Goodrich, a nice old lady (although pretty busy), who works at payroll.

You are to check Amanda's inbox and do the following tasks:

Task 1

SF offers forms on its website that a claimant must download and then send as attachments to your email address (amanda@securedfuture.biz). You have to review the forms and check if all the required fields contain the proper information and if so, you acknowledge receipt to the sender and forward the forms to Henry (henry@ securedfuture.biz). Otherwise you ask the sender to retransmit with corrections.

Task 2

Finish processing (delete right away, read, answer, etc.) messages in Amanda's inbox.

Background information

Before joining SF, Amanda volunteered for free a charitable organization, since she always has been a goodhearted person. She managed her own website (lovelase.org) where she advertised volunteering opportunities.

She is paying less attention now to her website, but she still uses her email address (amanda@lovelase.org) for all kind of personal matters, like to manage her accounts at uBid.com and barnesandnoble.com