Please Continue to Hold

An empirical study on user tolerance of security delays

Serge Egelman
Brown University
egelman@cs.brown.edu

David Molnar
Microsoft Research
dmolnar@microsoft.com

Nicolas Christin
Carnegie Mellon University
nicolasc@andrew.cmu.edu

Alessandro Acquisti
Carnegie Mellon University
acquisti@andrew.cmu.edu

Cormac Herley
Microsoft Research
cormac@microsoft.com

Shriram Krishnamurthi
Brown University
sk@cs.brown.edu

ABSTRACT

We present the results of an experiment examining the extent to which individuals will tolerate delays when told that such delays are for security purposes.[1] In our experiment, we asked 800 Amazon Mechanical Turk users to count the total number of times a certain term was repeated in a multi-page document. The task was designed to be conducive to cheating. We assigned subjects to eight between-subjects conditions: one of these offered a concrete security reason (virus-scanning) for the delay, another offered only a vague security explanation, while the remaining conditions either offered non-security explanations for the delay or no delay at all—in the case of the control condition.

We found that subjects were significantly more likely to cheat or abandon the task when provided with non-security explanations or a vague security explanation for the delay. However, when subjects were provided more explanation about the threat model and the protection ensured by the delay, they were not more likely to cheat than subjects in the control condition who faced no such delay. Our results thus contribute to the nascent literature on soft paternalistic solutions to security and privacy problems by suggesting that, when security mitigations cannot be made "free" for users, designers may incentivize compliant users' behavior by intentionally drawing attention to the mitigation itself.

1. INTRODUCTION

The computer security community has proposed various approaches to thwarting security breaches. Such *security mitigations* can be divided into three different categories: mitigations that are invisible to the user, mitigations that do something noticeable on the user's behalf, and mitigations that suggest the user take a particular action. The latter category, in particular, can come at a substantial cost for the user. If this cost is too large, users may choose to forgo the security mitigation, with detrimental individual and collective effects: negative externalities may be created when users fall victim to security breaches [15], and revenues for a firm producing more secure products may be lost if users choose less burdensome (even though less secure) software.

In this paper we explore the cost of security mitigations from the perspective of the end user, and how manipulating – and offering explanations for – those costs may impact users' acceptance of security mitigations. We are specifically interested in quantifying the inconveniences users will accept in the name of security. Previous work on security mitigations investigated costs [6,7,9,13,18,24,28,29,36], but focused on application compatibility and the speed impact of the mitigation on a set of benchmarks. In contrast, we attempt to measure what makes costs of mitigations acceptable to users. We posit that not only should acceptability of security mitigations be evaluated directly through user studies, but that non-normative mechanisms should be devised to increase the acceptability of those mitigations. By "non-normative," we refer to mechanisms that do not affect the technical performance of the security mitigation, but may influence the way users react to it: for instance, providing explanations with varying degrees of detail, making certain types of information more or less salient, or artificially manipulating speed and delays in the product's performance.

As a first step in this line of inquiry, we designed an experiment in which we asked subjects to count the number of times a specific word appeared in a PDF document using a custom Flash-based viewer. We purposefully designed this task to be conducive to *cheating*: the user could submit a response without actually reading the entire document. The rate of cheating with a "mitigation" in place compared to "no mitigation" then gives us a quantitative measure of the acceptability of a security mitigation. That is, if it takes an unacceptable amount of time to complete the task in one condition, we would expect to see a disproportionate amount of cheating in that condition.

We controlled for two common user-facing aspects of a security mitigation: the presence of a delay and the presence of a notice explaining the reason for the delay. We designed our study as a four condition between-subjects experiment: one control condition, with no delay and no notice; one "loading" condition, with a delay but no notice; one "security" condition with a delay and a vague notice; and one "security" condition that incorporated a delay along with a detailed explanation of the threat model and its mitigation. Based on results from the behavioral economics and social psychology literature, we hypothesized that subjects would be more tolerant of the delay when told that the delay was for security purposes, and when primed with more detailed explanations of the threat model and its mitigation. Thus, we expected to see a disproportionate amount of cheating in only the condition with the delay and no notice.

[1]An unpublished version of this work was presented at the 2010 Workshop on the Economics of Information Security (WEIS).

We conducted our experiment with 400 subjects recruited from Amazon's Mechanical Turk. Mechanical Turk is a service for advertising "tasks" that can be completed by human workers for a set price per task. We found that subjects were significantly more likely to cheat or abandon the task when provided no explanation or a vague security explanation for the delay. However, when subjects were provided more explanation about the threat model and the protection ensured by the delay, they were not more likely to cheat than subjects in the control condition who faced no such delay. Thus, when subjects were primed for security and the delays supported the security priming, subjects were compliant.

After performing our initial experiment, several open questions remained regarding how participants would behave when given an unambiguous non-security explanation for the delay versus the vague explanation. Likewise, we wanted to examine the extent to which priming our participants for security influenced our results. To answer these questions, we recruited an additional 400 subjects who were assigned to four new conditions. Thus, this paper reports on a user study with 800 total subjects.

Our findings suggest that if a delay is necessary due to a security mitigation, then the mitigation may be more acceptable to users if they are told of the threat model and how the mitigation protects them. This advice may sound counterintuitive, given that most security mitigations intentionally do not make themselves visible to the user unless an attack occurs, and perhaps not even then. For example, Windows programs do not usually employ pop-ups to inform users that address space randomization is being used or that stack canaries are being inserted. On the other hand, many anti-virus programs do explicitly warn the user when a scan is in progress, which may incur significant delays to user operations. There is therefore a tension between security which is "invisible," and measures made explicit "for security reasons." In this paper, we attempt to better characterize these tensions by providing empirical evidence. Our work shows that user studies of security mitigations shed important light on the acceptability of mitigations in contexts not well served by previous approaches. In this regard, our results contribute to the nascent literature on soft paternalistic solutions to security and privacy problems.

Our approach side-steps the question of "is $X\%$ overhead on this benchmark a lot or a little?" by placing the user-facing aspects of the mitigation directly in the context in which they are experienced by users. Because the technical aspects of the actual mitigation can be abstracted away, these types of user studies can be generalized to yield actionable findings for multiple types of security mitigations. Therefore, we believe user studies are an important addition to traditional benchmarking and application compatibility analysis for evaluating security mitigations. Our experience with Mechanical Turk shows that these studies can be carried out at modest cost even with hundreds of users. We hope this will encourage others to perform such studies as part of the process for evaluating future security mitigations.

2. BACKGROUND

Our work responds to and is informed by traditions from computer security, behavioral economics, human-computer interaction/computer-supported collaborative work, and psychology. We now discuss background from each of these communities in detail.

Computer Security. *Security mitigations* are features of an application or operating system that make it more difficult for an adversary to take control of a victim's computer, even when the victim's software has a bug such as a buffer overflow. The value of a security mitigation is that it trades speed and application compatibility for increased attack difficulty, *without requiring the defender to have detailed, specific knowledge of the attack in advance.* This tradeoff is attractive because finding all bugs in computer software or enumerating specific attacks ahead of time is incredibly difficult.

The computer security community has a long list of proposed mitigations stretching over more than fifteen years. Classic examples include stack canaries [7,13], address space layout randomization [28], automatic bounds checking [6, 18], and non-executable memory [24,28]. More recent examples include software changes such as Nozzle [29], or efficient software fault isolation for x86 and x64 architectures [23,36], and hardware changes such as those found in Raksha [9] or SmashGuard [27]. Anti-virus software can also be viewed in this category, and it has been undeniably successful commercially.

In the evaluation section of every proposal of which we are aware, the cost of a mitigation is evaluated along two axes: the loss in speed on benchmarks and the impact on application compatibility. Application compatibility is an important consideration, but one which we do not address in this work. Another increasingly important phenomenon is that adversaries can develop reliable methods for bypassing mitigations [11]; once such bypasses are found, users pay the cost of the mitigation but receive no benefit against sophisticated adversaries.

Even so, a major part of discussions on adoption from the earliest mitigations to the present centers on whether a specific speed impact is "too much," as measured on a set of benchmarks. The key problem we address is that measuring speed on benchmarks is merely a proxy for measuring *user acceptance* of a mitigation. Clearly, if there is no speed impact from adopting a mitigation, the experience of the user with the mitigation is indistinguishable from the original experience in the common case where no attack is present. Therefore the mitigation will be acceptable.

Unfortunately, in most cases security mitigations cannot be made "free." The computer security community has then historically proceeded to the difficult question: "is the speed impact of the mitigation on these benchmarks acceptable?" Complicating the question is the fact that "acceptance" means different things in different scenarios. For example, an additional 50 milliseconds to load a web page may lead to a significant loss in revenue for a web site. An additional hour added to a batch job, which normally takes a year, may not be noticed. While research has been done to examine the wait times that most Internet users will tolerate [14,26], none of these studies specifically addressed security explanations for these delays.

Behavioral Economics, Usability, and Soft Paternalism. Our contribution can be related to the nascent literature on the application of soft paternalistic approaches [22,34] to privacy [2,35] and security [5] problems. In recent years, there has been growing interest in understanding the psychological motives, as well as the possible cognitive and behavioral biases, that affect privacy [1] and security [3,31] decision making. In parallel, the computer science community has started investigating how to make privacy and security systems more usable [8]. These streams of research converge when lessons derived from the behavioral economics, decision research, or psychological literatures are incorporated into the design of systems that take into account systematic biases that affect our decision making in information security. These approaches do not merely aim at making systems more usable, but to actually anticipate known and costly biases – and sometimes even exploit those biases in manners that nudge users towards certain choices, without limiting their freedom [22, 34]. Some specific examples include providing salient information [35], creating interactive audited dialogs [5], and better conveying risks to non-experts [3].

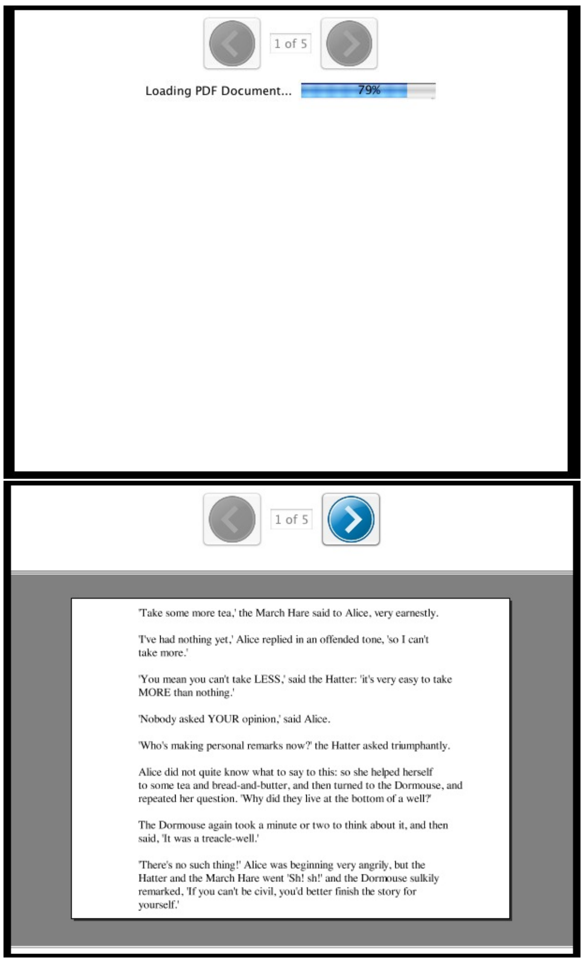
Psychology. Milgram explored the role of obedience to authority in his seminal 1963 experiment. He concluded that people are generally compliant with requests—however bizarre or nonsensical—when those requests come from people in positions of authority [25]. While work on obedience to computer security mitigations has some parallels, our work differs in that our requests did not come from a human being in a position of authority. Our experiment tests a hypothesis related, in part, to the results of a famous experiment by Langer, Blank, and Chanowitz [21]. In their series of experiments, they showed that even "placebic" information provided in the form of explanation, or reason for a request, was sufficient to generate compliance with a request, even though the reason itself conveyed no actual information. In our experiments, we tested whether providing an explanation for the security delays in loading pages would, in fact, increase the likelihood that subjects would comply with the security mitigation.

3. METHODOLOGY

We conducted an experiment using Mechanical Turk to examine whether people would put up with an inconvenience if they were told it was for security purposes. We conducted our experiment with two cohorts of 400 participants each, for a total of 800 participants. In this section we describe the study environment and the initial four conditions that we created. In Section 4 we present our results and our rationale for adding the second cohort of 400 participants, including the motivations for the additional four experimental conditions that we created.

Researchers have recently begun using Mechanical Turk as a way to quickly perform large-scale human subjects experiments for very little cost [19]. In 2009, Ross et al. performed a series of surveys using Mechanical Turk and concluded that the demographics do not significantly differ from the population of U.S. Internet users [30]. In 2009, Jakobsson performed an experiment to study the quality of work produced by Mechanical Turk users. He commissioned a survey using Mechanical Turk as well as an identical one using an "established, independent survey company" and found no significant differences between the two participant pools [17].

We created a task wherein we told study subjects that they were beta testing a new web-based document viewer, *SuperViewer* (Figure 1). When first launching SuperViewer, all subjects saw a progress bar that took ten seconds to load before displaying the first page of the document. Those in the *Control* condition never saw this progress bar again, while those in the three experimental conditions repeatedly saw this progress bar each time they viewed a new page



'Take some more tea,' the March Hare said to Alice, very earnestly.

'I've had nothing yet,' Alice replied in an offended tone, 'so I can't take more.'

'You mean you can't take LESS,' said the Hatter: 'it's very easy to take MORE than nothing.'

'Nobody asked YOUR opinion,' said Alice.

'Who's making personal remarks now?' the Hatter asked triumphantly.

Alice did not quite know what to say to this: so she helped herself to some tea and bread-and-butter, and then turned to the Dormouse, and repeated her question. 'Why did they live at the bottom of a well?'

The Dormouse again took a minute or two to think about it, and then said, 'It was a treacle-well.'

'There's no such thing!' Alice was beginning very angrily, but the Hatter and the March Hare went 'Sh! sh!' and the Dormouse sulkily remarked, 'If you can't be civil, you'd better finish the story for yourself.'

Figure 1: Screenshots of the SuperViewer interface. Subjects were able to change pages using the arrow buttons. The above image shows the loading screen that all subjects saw when first launching SuperViewer; subjects in the *control* condition never saw this screen again, while those in the experimental conditions repeatedly saw it on every page. The page number indicator was subsequently removed after our pilot studies.

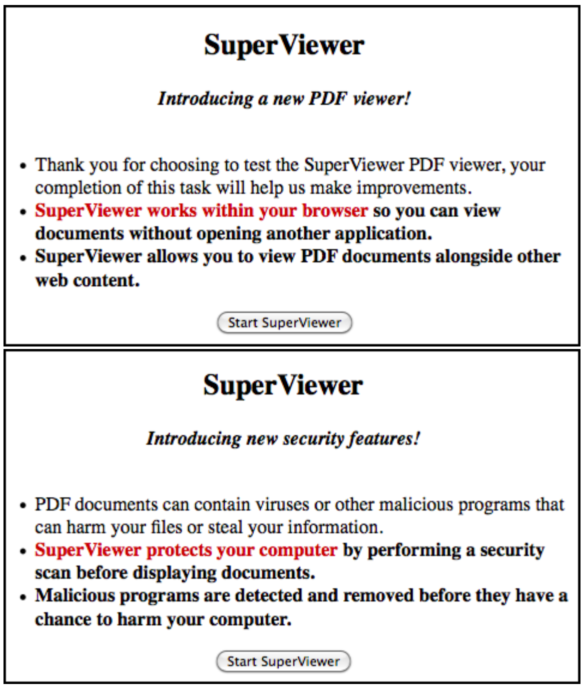
of the document (i.e., they had to wait ten seconds each time they turned to a new page). The three experimental conditions differed based on the text used to explain the reason for the progress bar.

For the task itself, we asked subjects to read a document using SuperViewer. We told subjects that to receive payment, they must report the frequency that a particular word occurred in that document. Thus, they would have to read the entire document in order to accurately answer the question. SuperViewer features a very basic interface: two buttons for navigating forward and backward in the document, which forced subjects to view the document pages in order. Likewise, there was no "search" functionality, otherwise completing the task would have been trivial. One goal of this task was to make it appear indistinguishable from other non-research Mechanical Turk tasks (e.g., product categorization, image labeling, etc.). Thus, if subjects did not believe they were engaged in a research study for the public good, they may have been more inclined to "cheat." We defined cheating as submitting a response to receive credit without using SuperViewer to read the entire document (e.g., reading part or none of the document). Our main interest was to examine whether subjects' cheating varied based on the four between-group conditions we created:

- **Control** — SuperViewer was launched when subjects clicked a button. Immediately after launching, a progress bar was displayed for ten seconds with the label, "Loading." After this ten second period, the first page of the document was displayed. Subjects could change pages in the document by clicking one of two arrow buttons. Thus, subjects were forced to view pages in order.
- **Loading** — This condition was identical to the *Control* condition, with one exception: when advancing to a subsequent page after the first, an additional ten second progress bar—also labeled "loading"—was displayed before subjects were allowed to view the next page. Subjects only saw these progress bars once for each new page; subjects would not see a progress bar again when flipping to a previously viewed page. The purpose of this condition was to examine whether study subjects would tolerate an unknown delay or whether they would cheat by quitting the task early and reporting an incorrect word frequency.
- **Security** — This condition was identical to the *Loading* condition, with one exception: the label on the progress bar was changed from "Loading" to "Performing security scan." The purpose of this condition was to examine whether study subjects would tolerate a delay when they were told it was for security purposes or if they would cheat by quitting the task early and reporting an incorrect word frequency.
- **SecPrimed** — This condition was identical to the *Security* condition, with one exception: prior to launching SuperViewer, subjects were informed of the danger of viruses embedded in online documents and that SuperViewer scans documents for their protection. The purpose of this condition was to examine whether subjects were any less likely to cheat if they understood why the "security scan" was being performed, or if simply performing an ambiguous security function was reason enough (as was the case in the *Security* condition).

Prior to launching the SuperViewer applet, subjects were shown a page of information about the software (Figure 2). The main purpose of this page was to prime those in the *SecPrimed* condition to security concerns and to convince them that the software was protecting them against a legitimate threat model. In order to balance all of the conditions in terms of total workload required of our subjects—the total amount of text to read—we added a placebo text page for those in the other three conditions.

In order to determine whether our subjects cheated during the experiment, we recorded the number of unique document pages they viewed, the total number of pages viewed, the time it took them between opening the document and submitting their response, and the numerical response that they submitted.


SuperViewer

Introducing a new PDF viewer!

- Thank you for choosing to test the SuperViewer PDF viewer, your completion of this task will help us make improvements.
- **SuperViewer works within your browser so you can view documents without opening another application.**
- **SuperViewer allows you to view PDF documents alongside other web content.**

SuperViewer

Introducing new security features!

- PDF documents can contain viruses or other malicious programs that can harm your files or steal your information.
- **SuperViewer protects your computer by performing a security scan before displaying documents.**
- **Malicious programs are detected and removed before they have a chance to harm your computer.**

Figure 2: Screenshot of the instructions subjects saw before launching SuperViewer. The image above was seen by those in the *Control*, *Loading*, and *Security* conditions. The image below was seen by those in the *SecPrimed* condition.

Upon completing the initial experiment, we invited subjects to complete a survey based on their experiences using SuperViewer in exchange for a bonus payment. The first page of the survey asked subjects about their overall opinions of SuperViewer and the factors that influenced those opinions: color, look and feel, ease of use, speed, and security features, each rated using a 5-point Likert scale. The second page of the survey contained questions about subjects' risk perceptions, both when they used SuperViewer during the experiment, as well as when performing other activities on their computers (e.g., browsing the web, reading email, downloading files, etc.). The third page of the survey contained questions about what anti-virus software the subjects currently used, as well as the types of threats they believed said software guarded against. Finally, the fourth page of the survey featured demographic questions.

3.1 Pilot Studies

We decided to pilot our experiment using five pages from *Alice in Wonderland* as the document, and we used a version of SuperViewer written in Java. We offered to pay each participant $0.05 to complete the task and we targeted 100 subjects, who were randomly assigned to the four conditions. Overall, we were underwhelmed at the rate of response to this task; it took us fifteen days to recruit 100 subjects. We decided that we needed to pay our subjects more, and so we created another 100 tasks, but this time paying $0.11 per participant. This time it took us only six days to recruit 100 subjects.

While we had decided that we must pay subjects at least \$0.11 to complete this experiment in a timely manner, we discovered another potential caveat: several subjects had emailed us indicating that they could not load the viewer. In fact, while 200 people completed these pilot experiments, 355 others attempted to complete the task but were unsuccessful. Given that almost two thirds were unable to complete the task, we assumed that this was due to Java incompatibilities. Thus, we decided to rewrite SuperViewer in Flash.

We created a third pilot study to evaluate our Flash implementation. We recruited another 100 subjects and decided to pay them \$0.05, since we reasoned that with fewer technical incompatibilities from using Flash, we may receive an adequate participation rate with our original payment amount. Indeed, instead of taking fifteen days, using Flash allowed us to gather data from 100 subjects in just eight days. During the task itself, we asked subjects how many times the word "Hatter" occurred in the text, the correct answer being eight. Of our 100 subjects, only two clearly cheated, and each was in a different condition. We concluded that the task was too easy to perform, and therefore it would require an inordinate number of total subjects to get enough cheaters. Thus, we needed to make the task both longer and more frustrating; we changed the text from five pages of *Alice in Wonderland* to ten pages of *Ulysses*. The tenth page of the document only filled half a page, though we added a blank 11th page to indicate the end of the document. Because of this, we considered anyone who reached either page ten or eleven to have viewed the entire document.

The pilot versions of SuperViewer all displayed the current page number and the total number of pages in the document (Figure 1). By removing this status indicator, we reasoned that subjects will become more frustrated when they have no indication of when the task will end, and therefore, the observed effect size would be greater. Finally, we increased the payment to \$0.11 again, since we expected many more subjects to abandon the task without submitting any data, and therefore we needed a larger population.

4. ANALYSIS

A total of 800 Mechanical Turk users participated in our experiment between February 1st and April 5th, 2010. These subjects were randomly assigned to eight conditions. The first cohort of 400 subjects were assigned to the four experimental conditions outlined in Section 3. Three weeks later, we extended the experiment by recruiting a second cohort of 400 subjects and assigning them to four new conditions, which we describe and analyze in Section 4.2. The distribution of cheating across all eight conditions is presented in Figures 3 and 4. We observed several significant differences between the conditions. In this section we analyze these differences, both in terms of the number of people who cheated, as well as subjects' task performance.

4.1 Cheating

Subjects were required to read ten pages of *Ulysses* in order to answer the question, "how many times did the word *said* appear?" The correct answer was 23. We considered it cheating if a participant submitted an answer to this question without reading the entire ten pages. We hypothesized that subjects who had to wait for the progress bars to load before viewing subsequent pages would be significantly more

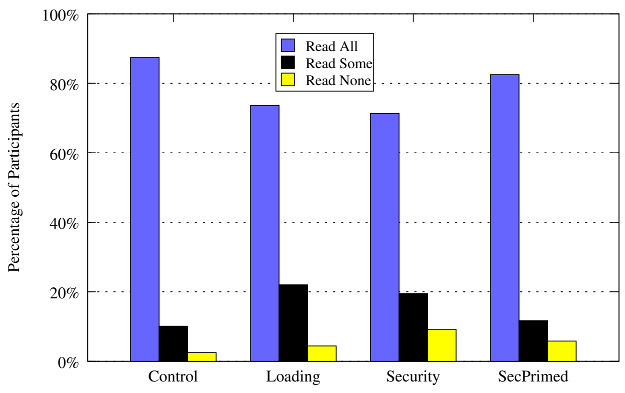


Figure 3: The number of participants who cheated in each of the four initial conditions. Because conditions were randomly assigned, the number of participants in each condition were not equal, which is why this data is graphed as a percentage. Overall, participants in the *Control* and *SecPrimed* conditions were significantly less likely to cheat than participants in the other two conditions.

likely to cheat than those in the *Control* condition. We further hypothesized that subjects who were told that this delay was for "security purposes" would be less likely to cheat than those who were not given an explanation for the delay (i.e., those in the *Loading* condition). Finally, we hypothesized that subjects who were given details of the threat model and the mitigation (i.e., the *SecPrimed* condition) would be just as likely to cheat as those who did not receive this information (i.e., the *Security* condition).

Overall, we found that our hypotheses were partially corroborated: subjects in the *Control* and *SecPrimed* conditions were significantly less likely to cheat than those in the *Loading* and *Security* conditions ($\chi^2_3 = 10.676$, $p < 0.014$). This indicates that subjects were more likely to cheat when they had to wait, except when they were told exactly why they had to wait; the label on the progress bar made no observable difference, except when subjects were informed of the danger of PDF viruses and that they were being protected by our software.

We further hypothesized that we would observe two types of cheating: subjects who submit answers before viewing any of the document and subjects who submit answers before reaching the last page (but after opening the document). The former type of cheating happens before subjects experience any types of delays, and therefore should be equally distributed across all of the conditions. Indeed that was the case: a chi-square test indicated no significant differences between the groups with regard to subjects who submitted an answer without ever viewing the document. We therefore decided to remove these subjects from the rest of our analysis, since they did not provide us with any data relevant to our hypotheses. Our results are robust to the point of yielding significance even with these subjects.

We examined the second type of cheating, subjects who submitted answers after only partially reading the document, and found significant differences between the conditions ($\chi^2_3 = 8.619$, $p < 0.035$). Furthermore, we believe that this effect was diminished by the ability to "return" a Mechanical Turk task without receiving credit. Subjects who did not wish to complete the task—but who also did not wish to cheat by entering an arbitrary answer without reading the entire document—had the ability to return the task. Unfortunately, Mechanical Turk does not give us the ability to view the data from participants who chose to return the task (nor did we think to instrument SuperViewer to collect this data), so it is unclear if a disproportionate number of participants returned the task in one condition over another.

4.2 Additional Conditions

Examining our first four conditions, we found that when participants were provided with a detailed security explanation for the delay, they were significantly more tolerant than participants who did not see a detailed explanation. This in and of itself does not prove that participants were more tolerant because we displayed a security explanation. It merely shows that without a specific explanation, participants were more likely to cheat. It is possible that providing a different detailed explanation unrelated to security may have yielded a similar effect. Likewise, it is entirely possible that if we changed the "Loading" label to represent something more concrete, participants may have behaved differently.

We also had concerns about the full effect of priming subjects in the *SecPrimed* condition. It can be argued that from the onset, we divided participants into two groups: security-primed and not security-primed. The other conditions could then be interpreted as subgroups of the latter group. That is, from our initial experiment, it is unclear to what extent participants in the *SecPrimed* condition were influenced by the security priming versus the label on the progress bar. Thus, the effect of the priming information cannot be separated from the effect of the progress bar.

Finally, participants in the *SecPrimed* condition were exposed to progress bar labels that directly supported the priming information; the priming information informed them of new security features, while the delay was directly attributed to these security features through the progress bar labels. Thus, the effect may have been attributable to this link. We decided to further examine these open questions by creating four additional experimental conditions. Each of these new conditions was identical, with the exception of the introductory priming information (Figure 2) and the labels on the progress bars:

- **Adjusting** — This condition was identical to the *Loading* condition (i.e., no priming information), however, we changed the labels on the progress bars from the ambiguous "Loading" to "Adjusting document width." The purpose of this condition was to examine whether participants were as likely to cheat with a unambiguous non-security delay explanation versus an ambiguous non-security explanation.

- **AdjPrimed** — This condition was identical to the *Adjusting* condition, however, we added priming information similar to that shown in Figure 2. In this condition, we examined whether participants would be as likely to cheat when the non-security delay supports the non-security priming. In the introductory screen, participants were informed that SuperViewer includes a new feature to dynamically resize documents to better fit their screens. Thus, when the delay occurred, the labels that read "Adjusting document width" clearly supported this feature.

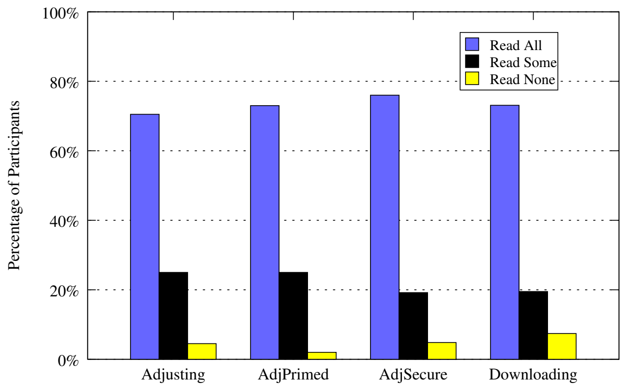


Figure 4: The number of participants who cheated in each of the additional four conditions. We observed no significant differences between these conditions. Overall, when compared with our initial four conditions, participants in the *Control* and *SecPrimed* conditions were significantly less likely to cheat than participants in any of the other six conditions.

- **AdjSecure** — This condition was identical to the *Adjusting* condition, however, we added the same security priming information that we used in the *SecPrimed* condition. The purpose of this condition was to separate the effects of the security priming with the effect of the excuse for the delay. That is, from the *SecPrimed* condition, we only learned that the combination of the security priming and security label on the progress bar were effective; we did not previously determine the effect of the security priming by itself.

- **Downloading** — This condition was identical to the *Adjusting* condition, however, we changed the wording on the progress bar to a second unambiguous non-security explanation. The purpose of this condition was to confirm that there was nothing unique about the *Adjusting* condition, and that any similar unambiguous non-security explanation would have a similar effect.

The results from these conditions can be seen in Figure 4. As can be seen, we observed no statistically significant differences between these four conditions with regard to cheating. More importantly, when we compared all eight conditions (Table 1), we still observed significantly less cheating in the *Control* and *SecPrimed* conditions ($\chi^2_7 = 14.782$, $p < 0.039$). When we removed the participants who viewed none of the document before submitting an answer, our results were similarly robust ($\chi^2_7 = 14.972$, $p < 0.036$). Thus, our experiment indicates that participants were significantly less likely to cheat only when they were primed for security and only when the explanation for the delay supported the security priming.

4.3 Task Performance

We examined the accuracy of subjects' responses, as well as the amount of time they spent performing the task to determine if there were any differences in their performance based on the conditions. We first examined the accuracy of subjects' answers, and found no significant differences between the four conditions. However, we did observe that

Condition	N	Total Time (s)	Time Per Page (s)	Unique Pages	Total Pages	Cheaters
Control	119	447	50.9	9.68	19.26	15 (12.6%)
Loading	91	485	56.6	8.71	13.33	24 (26.4%)
Security	87	514	63.4	8.47	12.20	25 (28.7%)
SecPrimed	103	517	56.9	9.43	14.22	18 (17.5%)
Adjusting	88	427	55.8	8.44	11.59	26 (29.5%)
AdjPrimed	100	474	56.8	8.98	12.90	27 (27.0%)
AdjSecure	104	513	66.1	8.71	14.21	25 (24.0%)
Downloading	108	504	70.8	8.62	15.93	29 (26.9%)

Table 2: The number of subjects in each of the eight randomly-assigned conditions, the average time spent performing the task, the average reading time per page, the average number of unique pages viewed (out of a maximum of 11), the average number of total pages viewed, and finally the number of people who cheated by not reading the entire document before submitting a response.

Condition	N	0 Pages	1-9 Pages	10-11 Pages
Control	119	3 (2.5%)	12 (10.1%)	104 (87.4%)
Loading	91	4 (4.4%)	20 (22.0%)	67 (73.6%)
Security	87	8 (9.2%)	17 (19.5%)	62 (71.3%)
SecPrimed	103	6 (5.8%)	12 (11.7%)	85 (82.5%)
Adjusting	88	4 (4.5%)	22 (25.0%)	62 (70.5%)
AdjPrimed	100	2 (2.0%)	25 (25.0%)	73 (73.0%)
AdjSecure	104	5 (4.8%)	20 (19.2%)	79 (76.0%)
Downloading	108	8 (7.4%)	21 (19.5%)	79 (73.1%)

Table 1: The number of subjects in each condition who read none of the document, some of the document, or all of the document before submitting a response. We considered subjects in the first two categories to have cheated. Recall that we considered those who reached either page ten or eleven as completing the document, since the tenth page was half empty, while the eleventh page explicitly stated it was the end of the document.

Condition	N	Speed Rating	Speed Concerns	Security Concerns
Control	63	4.38	6 (9.5%)	5 (7.9%)
Loading	40	3.53	13 (32.5%)	5 (12.5%)
Security	38	3.71	10 (26.3%)	6 (15.8%)
SecPrimed	46	3.65	12 (26.1%)	12 (26.1%)
Adjusting	56	3.63	24 (42.9%)	7 (12.5%)
AdjPrimed	48	3.23	21 (43.8%)	6 (12.5%)
AdjSecure	55	3.33	23 (41.8%)	15 (27.3%)
Downloading	64	3.45	30 (46.9%)	3 (4.7%)

Table 3: The differences in survey responses with regards to perceptions of speed and security during the experiment. The columns show the number of respondents in each experimental condition, their average response using a 5-point Likert scale to rate the speed of SuperViewer, and the number of respondents in each condition who explicitly mentioned performance or security concerns.

across every condition, cheaters reported significantly different results from those who read the entire document (t_{758}, $p < 0.0005$). Likewise, we observed that the distance of subjects' answers from the correct answer was inversely correlated with the number of unique pages they visited ($r = -0.543$, $p < 0.0005$). That is, the more pages the subjects read, the closer their answers were to the correct answer.

We found no significant differences between the experimental conditions when we examined how long subjects spent reading each page. Across all conditions, those who cheated spent significantly less time on the task: $t_{758} = 10.931$, $p < 0.0005$. However, we did notice significant differences based on experimental condition when we examined how many pages subjects viewed. Table 2 depicts the average number of unique pages each participant viewed, as well as the average number of total pages (i.e., counting the same page multiple times if a participant revisited that page). When we examined the total number of pages subjects visited on average, we found that those in the *Control* condition revisited significantly more pages than those in the other conditions ($F_{7,752} = 2.800$, $p < 0.007$).

4.4 Exit Survey

After subjects completed the experimental portion of this study, we sent them an email offering them an additional \$0.50 in exchange for completing a survey on their opinions of SuperViewer. We received a total of 419 valid responses. After filtering out subjects who cheated by submitting an answer without ever opening the document, we were left with 410 responses. These respondents claimed to be 273 men and 137 women, with 79% of our respondents holding a college degree or higher. It should be noted that all demographic data was self-reported and unverified, and therefore it may not representative of reality. Likewise, survey respondents were self-selected from our population of experimental subjects, and therefore may not be representative of the entire population. However, we observed no significant differences with regard to demographics between any of the conditions. Additionally, the proportion of cheaters who responded to our survey did not differ significantly from the proportion of cheaters who participated in our experiment. Of these 410 respondents, 82 of them (20.0%) cheated during the experiment by submitting a response after only partially reading the document.

On the first page of our survey, we asked respondents to report their overall impressions of SuperViewer using a 5-point Likert scale. We also asked respondents to rate several factors that contributed to this impression: ease of use, color, look and feel, speed, and security features. Using an ANOVA, we observed a significant difference between the conditions when it came to perceptions of speed ($F_{7,402} = 7.285$, $p < 0.001$). Upon performing post-hoc analysis using Tukey's adjustment for multiple testing, we found that people in the *Control* condition rated speed significantly higher than those in the *Loading* ($p < 0.048$), *AdjPrimed* ($p < 0.0005$), and *AdjSecure* ($p < 0.001$) conditions. These findings were expected, since those in the *Control* condition were not subjected to additional waiting times. After our initial experiment, we also found significant differences between the *Control*, *Security*, and *SecPrimed* conditions, but these differences disappeared after we doubled the number of conditions. The average ratings are displayed in Table 3. Despite the differences in speed, we noticed no difference between the conditions regarding the impact of security features on respondents' overall opinions of SuperViewer. Over 50% of respondents said that "ease of use" was the primary factor that influenced their overall opinion of SuperViewer, which was consistent across the eight conditions.

When we asked subjects what they disliked most about SuperViewer, 139 of them (33.9%) said something about performance or the time it took to load each page:

- *It was very slow.*
- *Pages a little slow to load.*
- *Loading a page with "security features" took an obscene amount of time.*
- *The loading time of the security features.*

We performed a chi-square test to examine whether a dislike for the speed was predominant in any of the experimental conditions. Significantly fewer people in the *Control* condition reported speed as being the source of their dislike ($\chi^2_7 = 29.399$, $p < 0.005$). Thus, it is likely that this effect was in response to the presence of the progress bars in all seven experimental conditions. The breakdown of these responses are displayed in Table 3.

We asked subjects whether they felt there was a danger viewing the document with SuperViewer and to rate that danger using a 5-point Likert scale. We noticed that 64 of the respondents (8%) said that they had no idea and therefore could not rate the danger. We therefore removed these respondents when we analyzed this question. We observed significant differences between the eight conditions with regard to subjects' perceived dangers ($F_{7,338} = 3.692$, $p < 0.001$). Upon performing post-hoc analysis using Tukey's adjustment for multiple testing, we found that this difference was due to respondents in the *SecPrimed* condition rating the danger significantly higher than respondents in the *Control* condition ($p < 0.030$). As a follow-up question, we asked respondents to describe any concerns that they had during the experiment. We observed that those who were primed for security concerns—the *SecPrime* and *AdjSecure* conditions—were more than twice as likely to raise security concerns than those in the other conditions ($\chi^2_7 = 20.016$, $p < 0.006$). Some of these concerns included:

- *Security is my major concern here. Is it really safe to view PDF?*
- *Wasn't sure if it contained a virus.*
- *I was a bit concerned when it ran a (sort of) virus scan before displaying the text.*
- *What if there is a virus attached to the PDF file?*

Our results indicate that the security priming prompted participants' security concerns. However, those in the *AdjSecure* condition were almost twice as likely to cheat than those in the *SecPrime* condition. This indicates that once security concerns were raised, participants were more likely to accept the delay when believed it was designed to address their security concerns. Likewise, participants in the *Security* condition cheated because they were not primed for security; they did not have increased security concerns and therefore did not care about the mitigation.

5. DISCUSSION

Our experimental findings were slightly different than what we expected. In the physical world, people must undergo various security mitigations of questionable effectiveness, all the while remaining fairly complaisant. Schneier has written at length about "security theater," security measures that have no security value other than demonstrating to the public that *something* is being done [32]. Photo identification is checked at office buildings to compare visitors to non-existent watch lists, liquids are banned from airplanes despite evidence that attacks using liquid explosives are impractical, and soldiers with unloaded weapons are placed in prominent public places. Yet the public in general is fairly tolerant of these ineffective security measures, even though they are inconvenient both in terms of time and cost. While the TSA arguably causes more visible inconvenience and delay than most U.S. government agencies, elected representatives do not receive enough constituent complaints for them to actually change policy. In fact, it is likely that investment in technologies such as full body scanners is done mainly to ease perceptions of security, rather than to increase actual security [4]. This lead us to hypothesize that humans may be tolerant of these inconveniences at the mere mention of "security," whether rational or not.

In this section we explore how our hypotheses compared with the data we collected. We discuss several possible explanations for participants' behaviors and explain the greater applications for research in this area. Finally, we discuss some of this study's shortcomings and outline future work in this area.

5.1 Explanations

Our initial motivation for the contrast between the *Security* and *SecPrimed* conditions was to examine the role of bounded rationality when people tolerate the security delays. Those in the *Security* condition had no rational reason to tolerate the delay, since no explanation was given other than the ambiguous "security scan" label on the progress bar. Whereas those in the *SecPrimed* condition were given a plausible explanation for the security scan. If no significant differences were detected, we hypothesized that this would be due to bounded rationality: participants would not need to understand the security explanation to comply. Based on the parallels between our experiment and previous work on soft paternalism and bounded rationality, we did not expect to observe a statistically significant difference in behaviors between these two conditions. We were surprised that this was not the case.

Taken at face value, our results indicate that when given a valid explanation for a security delay, people will tolerate it. While at the same time, without a plausible explanation or without an understanding of the threat model, people will not tolerate the same delay regardless of whether they are told it is for "security purposes." We believe there are several possible explanations for these results.

5.1.1 Habituation

Computer security concerns have grown to prominence in recent years, such that even casual users must interact with security mitigations. Users are told to keep antivirus software up to date, to visit only secure websites denoted by a lock icon, and to think critically about the software they install. Yet from users' perspectives, they see a large quantity of computer security mitigations, but a fairly low attack rate. This calls into question whether these mitigations are worth the cost to users [16]. In the case of SSL warnings, one of the most noticeable user-facing computer security mitigations, the false positives dwarf the actual positives. Users become habituated to ignoring security warnings because the warnings either do not explain the risks and the threat model, or they use jargon such that users do not comprehend them [12,33]. Thus, users become habituated to many computer security mitigations because they see them so frequently and rarely associate them with consequences.

Due to habituation, users ignore security mitigations when they do not understand risks. This may explain the lack of differences when comparing the *Security* condition with the other non-security conditions. When users were forced to wait for an arbitrary security check that they did not understand, they did not believe they were in any danger. In the exit survey, 26% of the participants in the *SecPrimed* and *AdjSecure* conditions mentioned security concerns, over 65% more than those in the *Security* condition. Thus, it is possible that without highlighting a specific threat, users are habituated to computer security messaging. It is not clear whether this is actually different from behaviors in the physical world; humans are much better at conceptualizing physical world threat models than online threat models.

5.1.2 Reciprocity

Another possible explanation for the lack of cheating among participants in the *SecPrimed* condition is that they felt more obliged to complete the task than participants in the other conditions. Since these participants were told about the threat model and how our software was protecting them, they may have felt like they owed us something in return, since we were performing a service on their behalf. We observed that participants in the seven experimental conditions all ranked the speed of the program as significantly worse than those in the *Control* condition. This correlates with seeing the progress bar before viewing each page. However, participants in the *SecPrimed* condition, while just as annoyed as those in the other experimental conditions, were more likely to read through to the end of the document.

5.1.3 Sunk Costs

In Section 4, when we observed no significant differences with regard to the amount of time taken, we adjusted participants' completion times to account for the amount of time they had to wait in each condition. That is, we were measuring the amount of time participants spent reading the documents, which did not include the amount of time they spent waiting. We did this because the waiting time was artificially created by us. However, when we factor this waiting time back into each condition, we found significant differences ($F_{7,752} = 2.234$, $p < 0.030$). Those in the *SecPrimed* condition spent significantly more time in total than those in the *Control* condition ($p < 0.046$, post-hoc analysis using Tukey's correction for multiple testing).

This may explain why those in the *SecPrimed* condition did not cheat any more than those in the *Control*: participants in the former condition had invested almost 40% more time to complete the task! Thus, the sunk time cost may have dissuaded them from abandoning the task. However, this explanation is less plausible since those in the *SecPrimed* condition did not spend significantly more time than those in the other experimental conditions where cheating was significantly more abundant.

5.2 Applications

The study presented in this paper and its results have several immediate applications, both for computer security and public policy.

5.2.1 Security Messaging in Software Systems

Previous work [33] has shown the importance of security warnings in software systems in leading people to adopt secure behavior. The present study shows that, in addition to making systems more secure, good advance warning systems that clearly explain the rationale for a design choice, also render the system considerably more psychologically acceptable, and make people more likely to tolerate the security choices made for them "under the hood" when these come at a cost.

5.2.2 Alpha Testing Security Features

Security features are best tested by a large number of users, as the multiplication of different use cases across a varied user pool can uncover a number of vulnerabilities that would be harder to observe with a limited amount of testing. The novel contribution of our study is to show that alpha testing (similar to pilot studies in a usability context) is also of utmost importance to gauge the psychological acceptability of security mitigation mechanisms. Performing these types of studies is also extremely cost effective: between our pilots, the experiment, the bonus survey payments, and the fees to Amazon, this study cost under $350 (beyond the researchers' time). In this manner, scientific results can be used to guide engineering decisions.

As a case in point, consider the relatively complex set of warnings that have to be bypassed to accept a self-signed certificate in Firefox 3. The original beta versions of Firefox 3 contained a 11-step bypass mechanism, which, while marginally increasing security, also utterly annoyed users. Eventually, the bypass mechanism was reduced to a more manageable 4-step process, which is still perceived as too lengthy and impractical given the number of self-signed certificates in circulation [33]. Alpha testing could have helped to spot the problem before Firefox 3 was beta-released, which would have avoided public embarrassment, while leading the Firefox developers to focus on designing a better SSL warning system.

5.2.3 Scareware Defenses

An unfortunate consequence of the results we have obtained is that scareware – the tactic of coercing victims into installing fake anti-virus or anti-spyware mechanisms that in fact contain attack code – appears like a very viable strategy for malicious entities. While this result is not surprising, our current work helps quantify the strength of the psychological bias we have to combat when devising defenses against scareware. The more convincing the messaging chosen by the attacker, the more likely the user is to tolerate “odd” behavior from the software installed, so long as the user believes that the behavior is the cost of being protected from a potential risk. Figuring out how to counter this bias opens a whole new avenue of research.

5.2.4 Public Policy Applications

Beyond the software realm, the study seems to confirm that, by calling on people’s fears, one may improve the psychological acceptability of any action typically considered as annoying. This shows why Schneier’s “security theater” [32] may actually indirectly contribute to security, albeit in a different realm than what it is supposed to originally protect. While the security measures deployed in airports may be very ineffective against people managing to smuggle dangerous materials onboard an airplane, they help pacify the vast majority of the population that has to stand in line, often times in uncomfortable positions. In other words, these security measures, however questionable they may be in terms of actual protection provided, are likely effective at performing crowd control, which in turn improves the overall security of the environment. It is likely that similarly long lines and intrusive procedures within airports may not garner the same level of compliance. Within the context of security, our results corroborate Langer et al.’s results showing that people tend to be compliant with requests when given a rationale, rational or not. However, our results diverge from Langer et al.’s when security is not related to the request [21].

5.3 Caveats

We collected data from a total of 800 participants in our experiment. Of these, a total of 189 cheated by submitting a response without reaching the end of our document. Forty of these cheaters submitted responses without ever reaching the first page of the document. During our first cohort of 400 participants, where we examined the security conditions, we examined our web server logs and discovered that 55 additional participants had begun our task but chose not to submit a response. Unfortunately, we cannot determine the conditions to which these 55 were assigned. We do know that all 455 were randomly assigned to one of the the four initial conditions. However, it is possible that participants in one of these conditions was more likely to return the task incomplete than participants in the other conditions. But given that there were no significant differences regarding how the remaining 400 participants were split between the four groups, we find this explanation unlikely. At the same time, given the lopsided demographic data that we gathered from the exit survey, it is likely that some self-selection bias impacted our study.

We recruited our second cohort of 400 participants almost three weeks after finishing collecting data from our first cohort. Because all eight conditions were not assigned in parallel, and because we did not think to run a second control condition in the second cohort, it is possible that our two cohorts came from two exogenous populations. While this is possible, we find it highly improbable given that the demographic data did not differ significantly between the two cohorts.

5.4 Future Work

In this paper we highlighted some interesting initial findings with regard to users’ tolerance of security delays. We believe that studies in this area are a crucial missing step in the software development process as well as in the computer security community as whole. A security mitigation may solve a specific security problem, but if users are unwilling to accept the time cost associated with it, it has not solved the problem. However, our study was not without its caveats. We have several future experiments planned to refine our results and to pursue new questions in this area.

5.4.1 Determining Maximum Tolerance

In this experiment we showed that those in the *primed* condition were no more likely to cheat than those in the *control* condition. However, differences may still exist with regard to how much of a delay participants will tolerate. We have several experiments planned to examine these upper bounds. In one experiment, we expect to randomly vary the amount of time it takes for the pages to load (though remaining constant on a per-participant basis). This will allow us to use a regression to calculate the upper limit of how long participants are willing to wait in each condition.

In another experiment, we expect to increase the number of pages in the document by several orders of magnitude. By making the document unbearably long, everyone will be forced to abandon the task at some point. We will measure the point at which participants abandon the task in reference to their randomly assigned condition.

5.4.2 Latency vs. Bandwidth

The security mitigations that we modeled in this experiment are high latency in nature: participants were interrupted and had to pause their current task until the mitigation had completed running. Once complete, participants were free to resume their task at full speed until they were interrupted again. Other types of security mitigations are high bandwidth in nature: users are not interrupted, but the speed at which they can perform their tasks is noticeably decreased. We expect to perform experiments in this area as well, in order to further explore the types of slowdowns that are likely to be tolerated.

One particular example is the Tor project. Tor is an anonymous proxy network that uses onion routing [10]. One disadvantage of onion routing is that the level of privacy is directly proportional to the number of hops that packets must traverse. Obviously, this means that privacy comes at a time cost. This creates profound design decisions for Tor’s designers when it comes to specifying default configurations. However, these decisions can be made easier with a better understanding of how much delay users are willing to tolerate in the name of increased privacy. Köpsell performed a similar study on user tolerance of delays when using the AN.ON anonymous network and found a linear relationship between participant fall-off and increases in delay times [20]. However, in his experiment, it is unclear whether the study participants understood that the delay was correlated with an increase in privacy.

6. ACKNOWLEDGMENTS

This work was supported by NSF grant CT-0830945. The authors would also like to thank Lorrie Cranor and Erin Krupka for their input during the early stages of this work.

7. REFERENCES

[1] A. Acquisti. Privacy in Electronic Commerce and The Economics of Immediate Gratification. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 21–29. ACM, 2004.

[2] A. Acquisti. Nudging Privacy: The Behavioral Economics of Personal Information. *IEEE Security and Privacy*, 7(6):82–85, 2009.

[3] F. Asgapour, D. Liu, and L. J. Camp. Risk communication in computer security using mental models. In *the 2007 Workshop on the Economics of Information Security (WEIS)*, Pittsburgh, PA, 5-6 June 2007.

[4] E. Berman and L. Heger. Scanners Help Economy by Warding Off Fear of Flying. `http://www.cnn.com/2010/OPINION/02/08/Berman.terrorism.scanners/index.html`, February 8 2010.

[5] J. Brustoloni and R. Villamarín-Salomón. Improving Security Decisions with Polymorphic and Audited Dialogs. In *Proceedings of the 3rd Symposium on Usable Privacy and Security*, pages 76–85. ACM, 2007.

[6] J. Condit, M. Harren, S. McPeak, G. C. Necula, and W. Weimer. CCured in The Real World. In *PLDI '03: Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, pages 232–244, New York, NY, USA, 2003. ACM.

[7] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, Q. Zhang, and H. Hinton. StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *Proc. 7th USENIX Security Conference*, pages 63–78, San Antonio, Texas, January 1998.

[8] L. Cranor and S. Garfinkel. *Security and Usability: Designing secure systems that people can use.* O'Reilly Media, Inc., 2005.

[9] M. Dalton, H. Kannan, and C. Kozyrakis. Raksha: A flexible information flow architecture for software security. In *ISCA '07: Proceedings of The 34th Annual International Symposium on Computer Architecture*, pages 482–493, New York, NY, USA, 2007. ACM.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[11] M. Dowd and A. Sotirov. How to impress girls with browser memory protection bypasses. In *BlackHat Briefings Las Vegas*, 2008.

[12] S. Egelman, L. F. Cranor, and J. Hong. You've been warned: An empirical study of the effectiveness of web browser phishing warnings. In *CHI '08: Proceeding of The 26th SIGCHI Conference on Human Factors in Computing Systems*, pages 1065–1074, New York, NY, USA, 2008. ACM.

[13] H. Etoh. GCC Extension for protecting applications from stack-smashing attacks (ProPolice). `http://www.trl.ibm.com/projects/security/ssp/`, 2003.

[14] D. F. Galletta, R. Henry, S. McCoy, and P. Polak. Web site delays: How tolerant are users? *Journal of the Association for Information Systems*, 5(1), 2004.

[15] J. Grossklags, N. Christin, and J. Chuang. Secure or insure? A game-theoretic analysis of information security games. In *Proceedings of the 2008 World Wide Web Conference (WWW'08)*, pages 209–218, Beijing, China, Apr. 2008.

[16] C. Herley. So Long, and No Thanks for The Externalities: The rational rejection of security advice by users. In *NSPW '09: Proceedings of The 2009 New Security Paradigms Workshop*, pages 133–144, New York, NY, USA, 2009. ACM.

[17] M. Jakobsson. Experimenting on Mechanical Turk: 5 How Tos. `http://blogs.parc.com/blog/2009/07/experimenting-on-mechanical-turk-5-how-tos/`, July 2009.

[18] R. W. M. Jones and P. H. J. Kelly. Backwards-Compatible Bounds Checking for Arrays and Pointers in C Programs. In *Distributed Enterprise Applications*, pages 255–283, 1997.

[19] A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing User Studies with Mechanical Turk. In *CHI '08: Proceeding of The Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems*, pages 453–456, New York, NY, USA, 2008. ACM.

[20] S. Köpsell. Low latency anonymous communication - how long are users willing to wait? In G. Müller, editor, *Emerging Trends in Information and Communication Security (ETRICS)*, volume 3995, pages 221–237, 2006.

[21] E. Langer, A. Blank, and B. Chanowitz. The Mindlessness of Ostensibly Thoughtful Action: The Role of "Placebic" Information in Interpersonal Interaction. *Journal of Personality and Social Psychology*, 36(6):635–642, 1978.

[22] G. Loewenstein and E. Haisley. The Economist as Therapist: Methodological ramifications of 'light' paternalism. In A. Caplin and A. Schotter, editors, *The Foundations of Positive and Normative Economics: A Handbook*, pages 210–245. Oxford University Press, USA, 2008.

[23] S. McCamant and G. Morrisett. Evaluating SFI for a CISC architecture. In *15th USENIX Security Symposium*, pages 209–224, Vancouver, BC, Canada, August 2–4, 2006.

[24] Microsoft Corporation. IE8 Security Part 1: DEP/NX Memory Protection. `http://blogs.msdn.com/ie/archive/2008/04/08/ie8-security-part-I_3A00_-dep-nx-memory-protection.aspx`, 2008.

[25] S. Milgram. Behavioral Study of Obedience. *Journal of Abnormal and Social Psychology*, 67:371–378, 1963.

[26] F. F.-H. Nah. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163, 2004.

[27] H. Ozdoganoglu, T. Vijaykumar, C. E. Brodley, B. A. Kuperman, and A. Jalote. SmashGuard: A Hardware Solution to Prevent Security Attacks on the Function Return Address. *IEEE Transactions on Computers*, 55:1271–1285, 2006.

[28] PaX Project. Address space layout randomization. `http://pageexec.virtualave.net/docs/aslr.txt`, Mar 2003.

[29] P. Ratanaworabhan, B. Livshits, and B. Zorn. Nozzle: A defense against heap-spraying code injection attacks. In *Proceedings of the Usenix Security*

Symposium, August 2009.

[30] J. Ross, A. Zaldivar, L. Irani, and B. Tomlinson. Who are the Turkers? Worker Demographics in Amazon Mechanical Turk. Technical Report SocialCode-2009-01, University of California, Irvine, 2009.

[31] B. Schneier. The Psychology of Security. *Communications of the ACM*, 50(5):128, 2007.

[32] B. Schneier. Is Aviation Security Mostly for Show? `http://www.cnn.com/2009/OPINION/12/29/schneier.air.travel.security.theater/index.html`, December 2009.

[33] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor. Crying Wolf: An Empirical Study of SSL Warning Effectiveness. In *Proceedings of the 18th USENIX Security Symposium*, 2009.

[34] R. Thaler and C. Sunstein. *Nudge: Improving decisions about health, wealth, and happiness.* Yale University Press, New Haven and London, 2008.

[35] J. Tsai, S. Egelman, L. Cranor, and A. Acquisti. The Effect of Online Privacy Information on Purchasing Behavior: An experimental study. *Information Systems Research*, 2010, Forthcoming.

[36] R. Wahbe, S. Lucco, T. E. Anderson, and S. L. Graham. Efficient software-based fault isolation. In *SOSP '93: Proceedings of The Fourteenth ACM Symposium on Operating Systems Principles*, pages 203–216, New York, NY, USA, 1993. ACM.