"They Keep Coming Back Like Zombies": Improving Software Updating Interfaces

Arunesh Mathur, Josefine Engel, Sonam Sobti, Victoria Chang, and Marshini Chetty,
University of Maryland, College Park

https://www.usenix.org/conference/soups2016/technical-sessions/presentation/mathur

This paper is included in the Proceedings of the Twelfth Symposium on Usable Privacy and Security (SOUPS 2016).

June 22–24, 2016 • Denver, CO, USA

ISBN 978-1-931971-31-7

"They Keep Coming Back Like Zombies": Improving Software Updating Interfaces

Arunesh Mathur
amathur@umd.edu

Josefine Engel
jme4yg@virginia.edu

Sonam Sobti
sonam.sobti9@gmail.com

Victoria Chang
vchang7190@gmail.com

Marshini Chetty
marshini@umd.edu

Human–Computer Interaction Lab, College of Information Studies
University of Maryland, College Park
College Park, MD 20742

ABSTRACT

Users often do not install security-related software updates, leaving their devices open to exploitation by attackers. We are beginning to understand what factors affect this software updating behavior but the question of how to improve current software updating interfaces however remains unanswered. In this paper, we begin tackling this question by studying software updating behaviors, designing alternative updating interfaces, and evaluating these designs. We describe a formative study of 30 users' software updating practices, describe the low fidelity prototype we developed to address the issues identified in formative work, and the evaluation of our prototype with 22 users. Our findings suggest that updates interrupt users, users lack sufficient information to decide whether or not to update, and vary in terms of how they want to be notified and provide consent for updates. Based on our study, we make four recommendations to improve desktop updating interfaces and outline socio-technical considerations around software updating that will ultimately affect end-user security.

1. INTRODUCTION

Vulnerabilities in client-side applications that run on user devices are on the rise. Typically, software vendors roll out software updates or "patches" to protect users by fixing these vulnerabilities and making changes to the software—such as adding new features, enhanced performance, or bug fixes. For this reason, the United States (US) government, various security agencies, and security experts advise end-users to download and install updates in a timely fashion to keep their systems secure [43, 32, 12]. However, recent studies have shown that non-expert end-users report delaying updates because they lack awareness on the importance of installing these security patches [28] or possess incorrect mental models of how updating systems work [49]. Moreover, even users identified as "professionals", "software developers", and "security analysts" only install updates about half the time more than non-experts [34]. Despite this evidence that there are factors influencing updating behaviors, the majority of research on software updates focuses on the network and systems aspect of delivering updates to end users [17, 47, 24, 13, 23].

However, a growing number of studies have begun to explore the human side of software updates for Microsoft (MS) Windows users [46, 49] including the reasons why users avoid updates in the first place. While these studies uncover users issues with software updates, they do not answer the question of how to make practical improvements to current software updating interfaces that could enhance security or whether these findings hold for users of other operating systems. To answer these questions, we further examined what prevents users from applying software updates across different operating systems and used this evidence to explore how to improve desktop software updating interfaces.

To achieve this goal, we conducted a three-phased research study. First, we conducted a qualitative formative study of 30 US Internet users' desktop software updating practices to complement previous work that focused solely on MS Windows users [49, 46]. Second, we distilled our findings into the design of a *minimally-intrusive*, *information-rich*, and *user-centric*, low-fidelity prototype of an alternative desktop software updating interface. Third, we conducted a think-aloud study with 22 Mac OS X users to evaluate our designs and draw recommendations to improve desktop software updating interfaces.

We make the following contributions. First, we confirm the findings of previous studies [46] and show these findings also hold for desktop users of operating systems other than MS Windows. Specifically, our findings reveal that users avoid updates that interrupt them, they lack sufficient information to decide whether or not to perform an update, and that users vary on how they want to provide consent and be notified of updates. Second, our study newly identifies additional reasons that users avoid updates such as whether or not users trust the software vendor providing the update, obscure change logs, and unknown installation times. Third, we contribute the design of an alternative updating desktop interface for Mac OS X that addresses these issues.

Finally, based on the positive reaction to our design con-

Symposium on Usable Privacy and Security (SOUPS) 2016, June 22–24, 2016, Denver, Colorado.

cepts in our think-aloud study, we contribute four validated recommendations for improving current updating interfaces for the desktop: personalizing update interfaces, minimizing update interruptions, improving update information, and centralizing update management across a device. We also discuss the socio-technical aspects of the updating process, namely around trust, consent, and control for making changes to in the wild software. We believe enhancing usability through improved desktop updating interfaces and further research to address the socio-technical aspects of software updating will ultimately lead to more secure systems.

2. BACKGROUND AND RELATED WORK

In this section, we explain the software updating process and touch upon the related work on software updates.

2.1 Software Updates and Automation

The usable security community has long recognized that human beings are the "weakest link" in the security chain [16, 39], attributing many security failures to human factors. Moreover, the community has recognized that most security decision making cannot be fully automated because humans often have to perform a part of the task—such as responding to security warnings (e.g., SSL [42], [3]) and identifying phishing emails [50]. Additionally, automation is often context dependent and limited by failure cases [18]. To compensate for the human element in security systems, Cranor [11] developed a framework to help designers fully consider all the factors to integrate users in the loop for security decisions in various systems. Software updates are no exception.

In fact, software updates typically involve users at various stages of the update process. An update process often varies based on the device type, operating system and application [31, 4], and the degree of automation and user involvement in each step can result in significantly different update experiences. Generally, a software update involves [13, 17]:

1. *Discovering the Update*: Users can either search for updates manually on websites or app stores, or set updating preferences for a specific application or the operating system to automatically notify them when updates become available.
2. *Downloading the Update*: Users can choose to either download available updates manually or set preferences for the system to automatically download them on their behalf.
3. *Installing the Update*: Users can manually install or have their system automatically install updates. Installation may involve closing applications affected by the update and often, an update is only applied after an application restart or machine reboot.
4. *Using the System Post-Update*: Once applied, updates may notify users that they have completed.

Depending on the degree to which the update system notifies and involves users, software update preferences are often referred to as [49, 17]:

- *Manual*: Users initiate and complete all the steps of the updating process, e.g., software drivers for input and output computer peripherals.
- *Automatic*: The update system automates one or more steps of the updating process such as downloading, installing, and notifying users. Users may have to briefly discontinue using the application to complete the installation or perform a restart of their machine or application, e.g., MS Windows patches and MS Office updates.
- *Silent*: The update system automates the entire update process and in addition, does not notify users explicitly at any step. Typically, in a silent update, the system installs the update without interrupting the user and applies it when users restart or re-open the application. Often, users fail to notice such updates, and lack the provision to disable or prevent them [34], e.g., Google Chrome updates [17].

In terms of reaching users' machines soonest after release, recent studies suggest that silent updating mechanisms may be the most effective in patching machines after an exploit is disclosed when compared to methods requiring a user's consent to download, install, or apply an update [34, 17]. Most software vendors and the US government recommend automatic updates for users to keep their systems secure instead of manual updates for this reason [43, 32, 12]. In our work, we examine user reactions to a low-fidelity prototype that conceptually makes all updates silent.

2.2 Deploying Software Updates

Numerous studies have explored ways to develop and deploy software updates, and compared the effectiveness of different mechanisms. For instance, Duebendorger and Frei studied the effectiveness of silent updates [17], Vojnovic *et al.* studied automatic patches [47], and Gkantsidis and Karagiannis studied the Windows patching system for distributing patches on a planet scale [24]. These studies comment on each patching mechanism's strengths and weaknesses and make suggestions for improving the creation and distribution of patches at scale but with no focus on how users will appropriate these updates. Another set of studies focuses on improving the deployment of patches in large organizations [13, 23]. For instance, Oberheide *et al.* help network administrators infer the impact of patches before deployment [36]. Others have investigated how to improve the deployment of patches via USB drives in regions with sporadic connectivity [9]. While these studies focus on improving the software patches themselves, they do not study the end-users who apply these patches, why they avoid patches, or how to improve patching interfaces as we do in our study.

2.3 User Experience with Software Updates

There is a growing body of work focused on understanding users' general online security behaviors and on user barriers to software updates. For example, Ion *et al.* [28] compared the capability of expert and non-expert users to process security advice. They found that non-experts updated their software less frequently compared to experts, lacked awareness about the effectiveness of software updates, and avoided updates that they felt introduced software bugs. Similarly, Wash and Rader surveyed almost 2000 US Internet users and found only 24% used protective security behaviors such as downloading patches [48]. Other studies show that users often disable or only perform updates on WiFi networks when

Phase	No of Users	Timeline
1. Formative Study	30	Jun '14–Sep '14
2. Prototype Design	–	Oct '14–Feb '15
3. Prototype Evaluation	22	Feb '15–May '15

Table 1: Research Timeline Overview.

they have limited and expensive Internet data plans [7, 29]. These studies provide evidence that users infrequently apply updates and touch on a few barriers in the process but they are not solely focused on users and software updates.

Several researchers have studied users and software updates in more depth. For instance, Fagan *et al.* [21] surveyed 250 users about attitudes toward software updating notifications. They found that users were reluctant to apply updates because they disliked being interrupted by notifications which were often perceived as obscure and unclear. In complementary work, Vaniea *et al.* [46] studied 37 non-expert MS Windows users and found that past negative updating experiences, such as dealing with user interface changes that required re-learning how to use an application, affect future updating behaviors. In another study of the same Windows users, Wash *et al.* [49] found that users' updating behaviors and intentions with their updating preferences are mismatched, often resulting in less secure systems. The authors conclude that there is a tension between automation and control in the updating process which may be difficult to resolve through improved usability alone. These studies focus on understanding users' software updating behaviors but not on how to improve updating interfaces as we do in our study.

Thus far, only two studies have sought to improve updating interfaces. First, Sankarpandian *et al.* [38] developed a desktop graffiti system TALC, which reminded users to install updates by painting their desktop with graffiti when their machines were left un-patched for a long amount of time. These researchers focused more on how to improve the process of gently notifying and nudging users to pay attention to install updates rather than with improving the overall experience of updating. Second, Tian *et al.* [45] developed a novel updating notification that used user generated reviews to help mobile users make privacy conscious decisions about which updates to apply based on what permissions were asked for by the updates. In contrast our study deals not only with notifications but updating as a whole on desktops, where privacy issues manifest differently because users do not explicitly grant permissions to applications.

3. PHASE ONE: FORMATIVE STUDY

We conducted a three-phased research process over the timeline shown in Table 1. In Phase One, we investigated users' current software updating behaviors and preferences through a qualitative interview-based study.

3.1 Method

3.1.1 Procedure

In mid to late 2014, we recruited 30 participants to take part in semi-structured interviews about their overall experience with software updates, including their likes and dislikes about software updates and their current software updating behaviors. We recruited participants through advertisements on university and affiliated mailing lists around the US, and social media (Facebook, Twitter) posts. We focused on finding adult Internet users that used Internet-enabled devices such as a desktop, laptop, tablet, or smartphone since they were likely to encounter software updates frequently. All interviews were conducted over the phone or Skype, audio-taped, and lasted between 45–60 minutes each. Participants were compensated with USD 15 gift cards for their time. The study was approved by the Institutional Review Board (IRB) of our institution.

The interview guide was developed after a survey of the existing literature on software updating and usable security at the time and informed by Cranor's human in the loop framework [17, 11] to ensure we covered all aspects of the software updating process. Cranor's model describes the human factors that affect secure systems, namely: *Communication* (informing the human that an action is necessary), *Communication impediments* (what might prevent the human from taking the action?), *Characteristics of the human receiver* (demographics, intentions, comprehension, and knowledge retention), and finally, *Behavior*. We used the framework to tease out the various human elements in the software updating process we discussed in Section 2. Concretely, we walked the participants through the specifics of the update process—discovering, downloading, and installing updates—and asked them the following questions for their applications and operating systems:

1. How do users learn about updates on their machines? Do they manually seek updates or wait for notifications?
2. Do users feel updates are important to security? What motivates them to either install or avoid updates?
3. How do users navigate the update process and how do they make decisions about security vs non-security related updates?
4. Do updates interrupt users' workflow? How does this affect their behavior?
5. Do users understand software update change logs and more generally, what action updates ask of them?

We also asked participants whether they ever changed, or sought help to change, the default update preferences for their operating systems and applications. In addition, we collected our participants' demographics (age, gender, education, income), their security management practices on their Internet-enabled devices such as installing anti-virus or enabling firewalls, their online security knowledge/actions when downloading software and dealing with suspicious emails, and past experiences with security incidents. The interview guide in its entirety is available in the Appendix.

3.1.2 Analysis

Once the interviews were transcribed, three researchers independently analyzed the transcripts. We inductively looked for patterns and threads in the data, marking them with labels, and then grouped and organized these labels into

Demographic	Phase One (N = 30)	Phase Three (N = 22)
Age		
18–34	66.7%	95.5%
35–54	26.7%	0%
>55	6.6%	4.5%
Gender		
Male	53.3%	36.4%
Female	46.7%	63.6%
Education		
College	6.7%	45.5%
Bachelor's	30.0%	40.9%
Master's	36.7%	9.1%
Other	26.6%	4.6%

Table 2: Demographic Information: Phase One and Phase Three.

themes [40]. The research team held regular meetings to discuss the initial results during this time, and arrived at the final set of themes shown in Table 3 after multiple rounds of discussions and consensus building. Example themes included "Updates interrupt users" and "Users need information for decision making". In the following section, we use the prefix *P* to indicate an interview participant.

3.1.3 Participants

Table 2 summarizes Phase One's demographics. Participants were predominantly between 18–34 years old, with a fairly even gender split. Most were educated, having college degrees, lived in the District of Columbia, Georgia, and Maryland, and earned a median annual income of USD 60,000. All participants owned desktops and 24/30 owned laptops as well. Two thirds of our participants used a single operating system: MS Windows (15/30), Mac OS X (3/30), and Linux (2/30). The remaining third used a combination of two operating systems: both Mac OS X and MS Windows (8/30) or Linux and MS Windows (2/30).

A large portion of our participants were aware of security breaches that were heavily publicized in the media. For instance, 18/30 were aware of the Heartbleed bug [8] and 11/30 were aware of the 2013 Target breach [44]. One-third of our participants had been victims of online breaches and malware including credit card frauds and computer viruses, and 8/30 participants stated they went above and beyond their e-mail providers' services to maintain their security. For instance, one participant reported using text-only mode for reading messages, and another reported scanning all downloaded attachments. Overall, while our participants were gender balanced, they represented a younger, more educated, and as a result, more technology savvy sample.

3.2 Findings

Our formative study showed that software updates interrupt users and their computing activities, supporting findings of previous studies [21, 46], for users of operating systems other than MS Windows . Our study also illuminates new evidence of information barriers to updates namely: trust in vendors, obscure change logs, and unknown installation times. Information barriers extend beyond the wording of unclear and obscure notifications [21] to the update's purpose, possible consequences of applying an update, and information to plan when to do an update. We also noted that unlike in constrained settings [7, 29], our participants were less concerned about updates using up Internet data. Finally, our participants varied on how they wanted to be notified or provide consent for updates based on the frequency of application use and the changes the update was going to perform. Additionally, they wished to manage and control all the updates on their devices centrally.

3.2.1 Interrupting Users

While our participants appreciated the importance of software updates for maintaining security, enhancing performance, and adding new features to their software, they felt that updates disrupted their computing activities in two ways: Interruptive update notifications and reminders diverted their attention while unwanted reboots and context switches lowered their productivity.

Notifications and Reminders: 22/30 participants reported that inopportune update notifications caused the largest disruption because they appeared during regular computing activities such as watching a video, doing a presentation, or during work times. These notifications were also hard to dismiss completely, so the same update could interrupt a user multiple times with reminder prompts. In a typical example, P17 remarked: *"I tend to let the update notifications go away but these days it looks like people keep forcing it so it comes back and back like a zombie."* Our participants prioritized dismissing update notifications and opted for reminders. Yet, these intrusive messages led to them ignoring many software updates because frequent interruptions were annoying and required active attention.

Rebooting and Context Switch: 19/30 participants reported that they delayed updates if they thought the update would require them to reboot machines, restart applications, and save their work. P9's example captures participants' feelings: *"I absolutely put them off until later, because the update requires me to stop what I'm doing, restart the program and computer, and then completely try to reconstruct where I left off."* Even if participants went through with updates, they became frustrated at having to recreate the context of their activities from which they were interrupted. This caused a negative perception of updates as a disruptive force. In another illustrative example, P12 expressed displeasure about restarts losing the context of open tabs in a browser: *"Usually when it tells me I have to shut down my browser, that's when I'm not happy. That and restarting, especially if I know I have a lot of windows or programs or something open, having to restart."*

3.2.2 Information for Decision Making

We asked participants what information they actively sought or wanted for informing decisions about applying software updates. They reported the following factors:

Update Categories: Vendor-specified update categories influenced our participants' decision making. 24/30 participants said they prioritized performing "major updates" including operating system and security related updates over others. P5's quote exemplifies the reasoning: *"I think if I saw the words security or something along those lines, I would be more apt to do the updates than if it said, you*

know, this improves usability."

Other participants revealed that existing vendor categories for updates inadequately captured an update's purpose and often led to them ignoring an update as P20's quote highlights: *"Just being told that it is critical does not really make me feel like it is critical. I need to feel the urgency and feel like there could be a consequence if I don't update it."* Concretely, they mentioned not knowing whether the update improved performance, fixed bugs, or enhanced security upfront and that these factors helped make decisions about going forward with an update or not.

Update Change Logs: Two thirds of our participants reported they glanced through the update change logs. In one telling example, a participant elaborated: *"I read almost all update notes and if it does not have any then I am going to disregard it. I take it pretty seriously. I have to know what the update is going to do on my software." (P17)* However when asked to reflect more deeply, 6 of those admitted the change log was unlikely to influence their decision to update. Close to half of the participants on the other hand, felt logs either presented too little or too much information, remarking that change logs could use less technical language or visual cues to better interpret the update information.

Trust in Vendors: Our participants' opinion of the software vendor influenced their decision to go forward with an update. 19/30 mentioned that they preferred updates from sources they trusted such as the app store or through the vendor's official website. This trust, they further explained, was amplified either through the reputation of the vendor (e.g., a large software company such as Microsoft or Apple), or through positive past experiences with applying updates from that vendor. P2's example quote captures this sentiment: *"I'm pretty good about—just when I see an update request, if it's from a source I know and trust—running it right then."* For our participants, trusting an update's creator was crucial for making a decision to go forward with a suggested update.

Even though trust was important to our participants, they often had trouble finding reputable sources for updates as P8 explains: *"Sometimes finding reputable sources for updates can be challenging and some software packages put their software out on all sorts of different sites. And, you know, it's like which one of these guys do I really trust to download from?"*. Participants tried to ensure that the updates they installed were legitimate but found it difficult to easily determine the authenticity of an update in current updating interfaces.

Compatibility Issues: 16/30 participants struggled with updates that caused unexpected consequences such as removing certain features they used or that led to compatibility issues with other software. Other participants felt that they were forced to install updates to ensure that software they used frequently would not stop working. In an example illustrating this theme, P13 said: *"Typically it's because I have no other choice. If the program that I want won't run on the version of Windows that I have, and I really want to run that program, I'll do the operating system update."* In another instance, P7 complained that their computer crashed after an update and they had to perform a system restore to get things working again. Overall, compatibility issues made participants reluctant to apply updates especially since they could not predict these interactions in advance.

User Interface Changes: 16/30 participants were dissatisfied with updates that changed the user interface because they had to re-acquaint themselves with the application. This is captured by P9's quote: *"For example, one of the updates on one of my frequently used programs switched around the confirm and cancel buttons."* This change caused him to inadvertently erase documents that he needed following the update. Participants thus became averse to updates with user interface changes but more importantly, they could not always predict which updates would have these changes.

Social Influences: Nearly a third of the participants discovered security flaws and updates through social influences such as online media blogs, the news, or through family and friends. While in some cases these social cues pushed participants to actively seek out updates, in other cases—especially with large and critical updates—they made users cautious about performing updates they had been warned against by the media or social networks. For example, P3 said: *"Usually when I hear about updates from things like PC Magazine or those people start talking about it, and then I would just read about it—just to get an idea before I even consider whether to do it."* In another instance, P6, a frequent Mac OS X user explained: *"There are some that I don't do at least until I go online and research it."* Participants therefore depended not only on vendor-specified information but also on social networks and the media to inform them about whether or not they should apply a particular update.

Results Post-Update: A little over one-third of our participants mentioned that they could not always discern the changes an update made post-update, because they received little, if any, feedback about the update's actions. This made them question the overall benefits of updates especially when they had invested considerable time and effort in applying the update (e.g., interrupting their primary activity and rebooting their machines).

Infrastructure Constraints: 8/30 participants told us they avoided updates because of infrastructure constraints such as insufficient disk space and slow Internet connections or because they believed updates slowed down their machines. In a typical example, P23 explained: *"Some software updates take a lot of additional space because it always comes with that extra storage amount that I need. So it is like all the updates that I am doing are only making my computer slower so it's an annoyance."* Participants also avoided updates to save data but to a lesser extent than suggested by findings in settings where Internet constraints are more prevalent [7, 29]. For this reason, participants told us they wanted the update size in advance to more easily weigh the costs of doing an update.

Installation Time: Because software updates disrupted our participants' workflow, they sought information to organize their activities such as the time required for updates to complete. 7/30 participants indicated a willingness to perform updates if they had access to this information in advance. P13 summed this up as: *"But if they actually tell me how long it'll take, that will make me more willing to start an update."* Knowing how long an update process would take, participants told us, would allow them to perform updates

at convenient times.

3.2.3 *Users in the Update Loop*

We asked our participants how they wanted to be notified and provide consent for downloading and installing updates.

Frequently Used Applications Matter: 17/30 participants mentioned that the frequency of use and their perception of an application's importance determined the degree of care they expressed towards keeping software and devices updated. In one example, P15 explained: *"An Evernote plug-in was not up to date and it asked me to update it. And I just deleted it because I don't want to deal with going through an update for a program that I don't use all that much."* Participants were most concerned about applications that mattered to them in some way; either by frequency of use or if it served some crucial function for them.

Tracking Updates: Just over a third of our participants found it difficult to track update downloads and installs because update settings and notifications were spread over multiple locations for the operating system and third party applications. 11/30 talked about needing a central update manager to review updates for all the software installed on their devices. Specifically, participants found it difficult to easily tell what needed to be updated and when or how to change the update settings for applications and the operating system. Overall, participants desired a central location on their devices to track all updates.

Phase One Themes	Participants (N = 30)
Interrupting Users	
Notifications & Reminders	22
Rebooting and Context Switch	19
Information for Decision Making	
Update Categories	24
Update Change Logs	20
Trust in Vendors	19
Compatibility Issues	16
User Interface Changes	16
Social Influences	12
Results Post-Update	12
Infrastructure Constraints	8
Installation Time	7
Users in the Update Loop	
Frequently Used Applications Matter	17
Tracking Updates	11

Table 3: Themes from Phase One.

At the end of Phase One, the research team decided to focus on addressing three main areas of concern stemming from the formative work as show in Table 3, namely updates interrupting users, the lack of adequate updating information, and finding ways to keep users in the update loop. We describe the part of our user-centered design process next.

4. PHASE TWO: PROTOTYPE DESIGN

In Phase Two, we created a low-fidelity, interactive prototype using MS PowerPoint to improve how users receive updates on their machines and how an update system could scaffold users' decision making. The prototype contains images of our mocked up updating interfaces linked together to create the illusion of a working system. Users can navigate through the mock system using the links to explore a limited set of predefined features with a system concept for each feature. Our goal with the prototype was to elicit user reactions to the different design concepts it embodies as described in this section. As such, we focused less on the implementation details such as how the information presented could be acquired ahead of time.

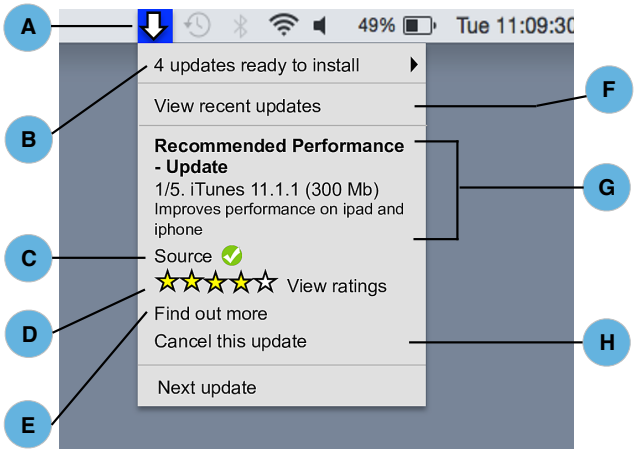

Figure 1: Update DropDown Menu. A: Update Icon. B: List of Remaining Updates. C: Source Verification. D: Update Ratings. E: Additional Information In Central Update Manager. F: List of Recent Updates In Central Update Manager. G: Summary Information For Current Update. H: Cancel Current Update.

Given that both desktop and mobile are heading towards an app-based model of software distribution [5, 27, 35], we chose to create the prototype for Apple's Mac OS X, a major operating system player [41], that has been using this model since 2010 [33]. For future work, we will extend this research to other operating systems and platforms.

4.1 Method

After identifying three areas of concern to users, the research team explored the design space for improved updating interfaces. We began with a lightweight sketching, brainstorming, and ideation phase over these themes. After several iterations of sketches, mockups, and designs were refined by feedback sessions with the research team, we settled on the following issues to alter in updating interfaces. First, we modified how users are interrupted about updates and the manner in which they provide consent to updates. Second, we augmented the information users need for making decisions about updates such as providing clear and concise update logs and the type of the update. Third, we consolidated the updates across a device into a single update manager for users to keep track of updates. We then sketched multiple designs of improved interfaces, discussing and validating the decisions we took at each point, and condensed these into a prototype we describe in the following paragraphs.

4.2 Altering Update Interruptions

Our *minimally-intrusive* low-fidelity interactive prototype alters how end-users are interrupted by either update or reboot notifications in two ways: first, in the concept behind our design, we assume that all update notifications are pushed through a single channel icon, and second, we assume we can piggyback updates requiring restarts to other times when users restart their applications or devices. We designed the following features in the interface mock-ups to reflect these concepts:

4.2.1 Single Update Notification Icon

All update notifications are reduced to a minimal visual cue, the subtle animation of a single system tray icon (Figure 1A). This inverted arrow icon animates only when an update is being downloaded or installed. Users can click on the icon to display a list of impending updates (Figure 1B).

4.2.2 Silent Updates

Conceptually, our design forces all updates to download and install automatically by default without a user's consent. When available, we envisioned that an update lives in the list of updates for a buffer period (e.g., 24 hours) to allow users to intervene. If needed, users can cancel it via the "Cancel this update" option (Figure 1H). When this buffer period expires, in our design concept the update is automatically downloaded and installed but users can continue using their applications without any reminders to restart. In cases where the restart is fundamental for an update to function, in our design concept, users' systems may remain unpatched until the next restart. Our design does not eliminate disruption from unwanted update changes but shifts the onus onto the user to decide whether or not to proceed with an impending update based on additional information. We made this design decision to be provocative to evaluate if users prefer a universally "silent" update mechanism across all their operating system.

4.3 Addressing Lack of Information

Our *information-rich* design adds to existing update information to help users accept or ignore updates via an update summary and post-update feedback.

4.3.1 Update Summary

Each impending update (Figure 1G) contains four important details we learned were lacking in the formative study and a "Find out more" (Figure 1E) link to more details in the centralized update manager:

1. *Source Verification:* Our design displays a green "tick-mark" (Figure 1C) next to the name of the software vendor to indicate a verified and trusted source.
2. *Update Type:* We tag each update with one of five categories (Figure 1G): UI fix (user interface changes), Bug fix (fixes software bugs), Security fix (fixes a major security flaw), Performance (performance enhancements), Compatibility (could cause compatibility issues with other software) to help users learn the update's purpose at a glance.
3. *Update Size:* To inform users about the data and disk space an update might consume, we display the size of the update (Figure 1G) in the summary.
4. *User Ratings:* Each update displays a five star rating (see Figure 1D) based on other users' experiences with it; with one star being poor and five stars being excellent.

4.3.2 Post Update Feedback

Participants in the formative study could not easily identify when an update was installed or what changes were made by the update. Our design shows a pop-up message when a user closes a newly updated application or the operating system for the first time post-update as shown in Figure 4. This message shows the changes made to the application and the date on which the most recent update was installed. The pop up also forces the user to rate the update before they can close the application. In our design concept, update ratings are mandatory to ensure they are eventually populated with information and to force participants to comment on this feature.

4.4 Centralizing Update Management

In the formative study, users desired a way to track *all* the updates across their device. Our *user-centric* prototype conceptually houses and controls the information and settings for all software updates through a central software update manager on the device. A "Pending" tab (see Figure 2) shows the updates that have been downloaded but not yet been installed, or that have been canceled by the user; the "Installed" tab shows recently installed updates to be viewed by last week, last 30 days, or all time; and finally, the "Ratings" tab shown in Figure 3 shows review comments, update ratings, and allow users to add their own reviews. Updates display a:

1. *Change Log:* Each update has a change log in bullet-point form clearly listing the changes the update makes as seen in Figure 2C.
2. *Time to Install:* We show the estimated time (Figure 2E) to install an update to help users plan when to do an update.
3. *Compatibility Report:* All the known possible disruptions an update might cause are listed in a report (Not shown).
4. *Update Settings:* Users can reconfigure updating settings—silent, automatic, or manual—for every application or the operating system from the central update manager (Figure 2D).

Comparison to current Mac OS X Updating System: The current Mac OS X operating system notifies users about an incoming update by means of two notifications (a pop-up, and an red call-out on the App Store icon) when updates are requested manually. No explicit notifications are provided when updates are downloaded and installed automatically. Our prototype switches all updates to silent, provides ambient notifications via an update notification icon, and does not seek users' consent to update by default.

We based our central manager design on the current Mac OS X App Store interface, which handles updates for all App Store applications and the operating system only but not third party applications. Our version of the manager adds

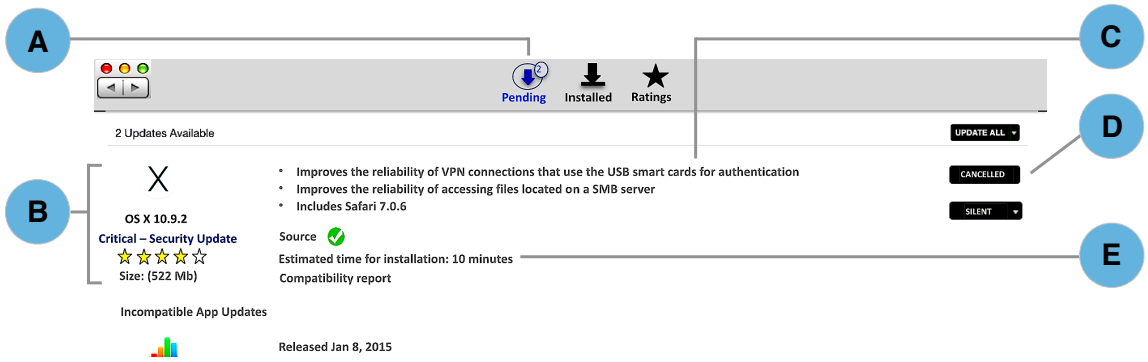

Figure 2: The Central Update Manager's "Pending Updates" Tab. A: Icon Showing No Of Pending Updates. B: Summary of the Update Information. C: Change Log. D: Update Preference Settings. E: Estimated Installation Time.

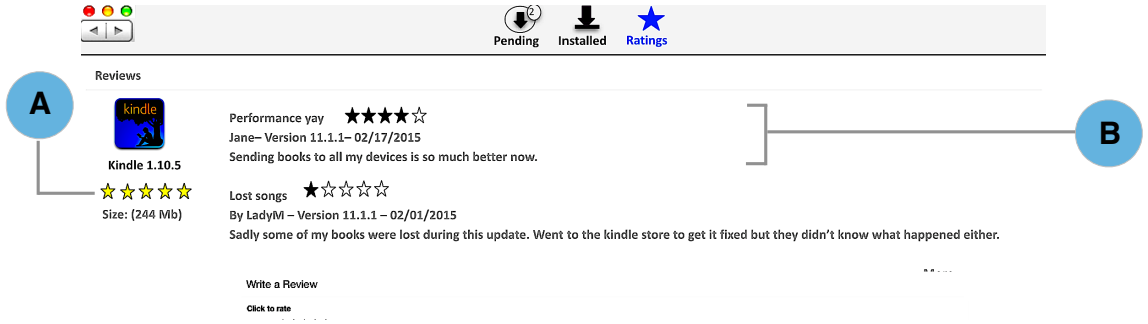

Figure 3: The Central Update Manager's "Ratings" Tab. A: Update Overall Rating. B: Update Rating and Reviews.

to the information the current App Store displays, with a clear and cohesive description of the change log in bulleted form, and the update's type, size and ratings, along with an estimate of the installation time and compatibility report. Unlike the current App Store, our version of the update manager includes an update configuration—manual, automatic or silent—for each application.

Post-update, the current Mac OS X operating system notifies users about an installed update by means of placing a tiny blue dot next to the application icon in the app "Launcher" menu. Our prototype, on the other hand, presents users with a dialog to notify them an update has taken place and to solicit a rating.

To sum up, we designed our proof of concept prototype to minimize interruptions, augment the update information available to users, and to centralize update management across a device. Our design was purposefully extreme in nature—in this case, having *all* updates as silent and having *all* applications solicit feedback post update, providing users with *all* the necessary information—much like a breaching experiment [10] to elicit user reactions and feedback.

5. PHASE THREE: EVALUATION

In Phase Three, we evaluated our low fidelity prototype. Although software updating, much like other security tasks, is a secondary task, our objective with evaluating the prototype was to elicit users' reactions and feedback on our design concepts and features.

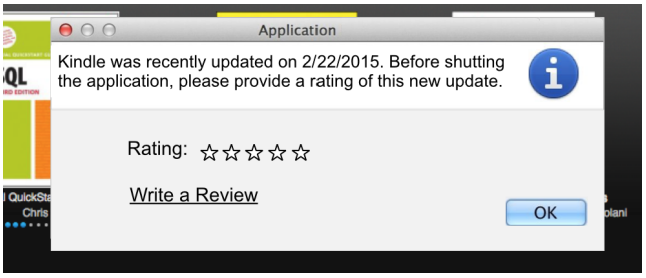

Figure 4: In-Application Post-Update Feedback Dialog.

5.1 Method

5.1.1 Procedure

To evaluate the prototype, we recruited 22 adult Mac OS X users via advertisements on our institutional mailing lists, social media (Facebook, Twitter), and from users who participated in Phase One of the study. Each participant completed a pre-study demographic survey before participating in the think-aloud session, a technique employed regularly in usability testing to gather feedback on proof of concept designs. Specifically, we employed the "speech communication" think-aloud protocol by Boren and Ramey [6], using verbal communication only as a means to acknowledge user response and keep the thought process alive [37].

Each participant performed a series of 11 tasks with our low-fidelity interactive prototype on a laptop provided by the research team. The tasks were designed to elicit user interaction with the single update icon, the central manager, and an application post-update. The first task was to search for updates on the machine to observe whether participants could find the update notification icon on their task bar. When a participant located and clicked on the icon, they were shown a list of updates. Next, participants were asked to cancel an update and following this task, they were asked to check for other pending updates on the machine and to verify whether the source of an update was genuine.

Participants were then introduced to the central update manager through a task to find out more about a particular update which linked them to the manager. In this view, they had tasks that asked them to view the specific additional information provided about updates such as the time to install and the compatibility report. Participants were also asked to do tasks that allowed them to see an empty list of pending updates and the installed and pending tabs in the central update manager. Finally, participants were tasked with rating and reviewing an update as well as changing the update setting for the operating system. The list of think-aloud tasks available as part of the Appendix.

Participants were instructed at the beginning of the session how to provide feedback, how the session would be recorded, and how they were to proceed through the tasks with the guidance of two facilitators from the research team. They were then presented with a paper list of tasks and two researchers observed the participants as they interacted with the prototype, recording their thoughts and actions as they completed the tasks. The researchers used probes such as "keep talking" and "um hmm" to remind participants to verbalize their thoughts when they stopped talking. When the participants failed to speak for more than 30 seconds, one of the researchers asked a stronger probing question relating to the task at hand. For instance, when participants failed to locate cancelled updates, the researcher asked "Where would you expect to see these updates and why?". For each participant, task responses that did not require these strong probing questions were marked "successfully completed".

Once the think-aloud session was complete, the researchers conducted an semi-structured exit interview with each participant. In this interview, we first explained the silent updating mechanism to our participants, i.e., how updates would install on their devices and how restarts would function. We then asked them about both their positive and negative reactions to the prototype, how they performed each task, and the design concepts embodied by the prototype. We also sought feedback for those tasks that the participants were unable to complete noting down how they expected the system to function. Each session lasted between 45 minutes to 1 hour and both the think-aloud session and exit interview were audio and video-taped. Participants were compensated with a USD 15 gift card for the entire session. The study was approved by the IRB of our institution.

5.1.2 Data Analysis

Once the think-aloud sessions and exit interviews were transcribed, two researchers—including one from Phase One—independently analyzed the data. We created profiles for each participant and noted their completion of the various think-aloud tasks. We also analyzed the transcripts using the same process as for Phase One—specifically seeking for differences in participants' reactions to the various design elements. In the following section, we use the prefix T to indicate a think-aloud participant.

5.1.3 Participants

Table 2 summarizes the demographics of the think-aloud session participants. Almost all of our participants were aged between 18 and 35. About 85% of them had completed their bachelor's degree or college, and either worked at, or near, or studied at our institution. Since many of our participants were students, the median annual income reported was <USD 25,000, Our participants owned a median number of 2 Internet-enabled devices and went online more frequently on laptops and smartphones than desktops, tablets, and gaming devices.

Similar to Phase One, our participants were aware of security breaches that were heavily publicized in the media—17/22 knew about the Heartbleed bug and 16/22 about the 2013 Target breach. Again, one-third of our participants had experienced either viruses or malware on their systems. Unlike Phase One, none of our participants reported extra measures to enhance their email security but claimed to ignore email attachments if the email was sent from an unknown source (15/22) or if it looked suspicious (18/22). Overall, our participants were younger, less educated, and less technical than our participants from Phase One.

5.2 Findings

Our prototype elicited varying reactions from our participants in the think-aloud session. Our participants wanted to be notified and actively consent to some but not all updates, they were positive about augmented update information as input for update related decision making, and appreciated the control of centralizing software update management.

5.2.1 Interruptions Sometimes Required For Control

Participants struggled to find the update icon—half failed to notice the system tray icon entirely. However, once participants discovered the icon, they were able to complete the remaining think-aloud tasks with ease. About half preferred our design to current software updating interfaces because they believed it would interrupt them less. T20 explained: *"It prompts me the least. I don't have to worry about it, I don't have to think about it."* The other half of our participants reacted negatively to the prototype's silent update mechanism and told us they wanted to be in the update process more actively. These participants' wanted updates with notifications at download and install time, particularly for applications they used frequently or depended on in some way, to prevent undesired changes. In an example quote, T8 said: *"I want to know how frequently the updates are, how frequently they're occurring and if there's something new or there's a bug. If there any changes, I want to know when and how they happened."*

When asked for further feedback, participants revealed they were willing to adopt a silent updating mechanism for a few of their applications. For instance, participants were amenable to silent updates from trusted vendors and for applications that did not impact their workflow significantly.

T9 said: *"I definitely prefer the silent so I wouldn't really have to do anything and to constantly be the best experience that I could have."* They believed a silent update process would keep their system secure and up to date since they often ignored updates otherwise.

5.2.2 Users Desire Improved Update Information

When completing tasks to read and interact with the additional update information we provided, participants were generally positive that the information would help them to make decisions about updates more so than current interfaces.

Update Type: 15/22 participants reacted positively to the update type, telling us these labels would clarify the purpose of various updates. However, about a third said the update type would not influence their decision to apply an update. Others preferred the binary classification of "critical" and "non-critical" to multiple labels, which they felt could be overwhelming and possibly confusing. Poor labels, participants told us, could backfire and cause a user to avoid an update if they obscured the update's purpose. In all, participants appreciated the concept of more informative labels for the update type to help in their updating decision making process.

Compatibility Report: 19/22 participants were able to find and interpret the compatibility report and 12/22 told us it would be helpful for singling out potentially problematic updates. In a typical example, T8 said: *"If I'm going to be using those apps, I wouldn't go ahead with the update because it will cause problems."* One suggested improvement was to only display compatible updates and only one participant worried how this report would be implemented. Overall, participants desired this predictive capability to help them prevent updates having an unwanted ripple effect.

Ratings: 15/22 reacted positively to the concept of update ratings with 12/22 saying that ratings would potentially influence their decision in going ahead with an update (especially with large updates). T9 explained: *"If there were mostly good reviews and there was nothing standing out as 'oh, this is a problem for my computer' it would help me make the decision to go ahead."* At least five participants wanted to see number of ratings for an update to better contextualize the information. Almost a third of participants felt that ratings would not add any value to their updating experience because they paid less attention to others' opinions. Most of our participants resonated with the idea of leveraging social networks for information to help them decide about performing updates.

Post-Update Feedback: All the participants were unanimous in disliking the concept of providing a mandatory rating post-update, seeing this as a nuisance in the long run. However, they were willing to provide feedback for updates that made visible changes (e.g. user interface modifications), and for applications that they frequently used or were important to them.

Time to Install: 13/22 found that having the information about how long was needed to install an update prior to beginning the process was useful for deciding when to do an update. T8 said: *"I think that's very important because if you're in a hurry or if you have some other work to do and sometimes you should know if you can finish the update by then or not."* 1 participant wanted aggregated view of the time required for all the pending updates and another wanted the time to install to be visible in the list of updates not just the central manager. For participants, having an estimate of the time involved for the update process was crucial for planning so as to minimize interruption to their activities.

Installation Size: Four participants reacted positively to the size of the update being displayed upfront. These participants desired some warning if the update would consume their remaining disk space. Two participants felt this information was less useful as they cared less about disk space on their devices.

Source Verification: Nine participants reacted positively to having the ability to easily identify an authentic update source. In an illustrative example, T17 remarked: *"If it's not from the verified source, then that will be the one thing that will stop me from installing the update."* However, several said they would probably not pay attention to this cue and a few felt that this cue was most important for third-party applications only. It was clear, however, that visual cues indicating update authenticity can build trust with users.

5.2.3 Users Prefer Centralized Update Management

Over half of the participants reacted positively to centralized software update management, especially for non-Mac applications that currently do not push updates via the app store. For example, T13 said: *"I like it: It seems more comprehensive because it has (for e.g.) the Microsoft stuff in it so you don't have to run the Microsoft updater as well as the app store updater mechanism."* Only a few participants thought that being able to control the update preference on a per-application basis in a central update manager was useful. However, it was unclear if participants reacted this way to the additional controls because they told us they generally preferred to keep default settings. Participants suggested the central update manager could also provide a history of updates so that changes could be rolled back to a state before the update occurred if something went wrong. In summary, participants felt managing updates in a single location would reduce information overload and make the updating process more consistent across a device.

6. DISCUSSION

Contrary to Wash *et al.*'s paper [49] which suggests that increasing usability may enable those users who wish to be less secure to apply fewer updates (i.e., to switch to manual updates), our findings suggest that users may want to be less secure in the first place because they suffer from a poor updating user experience. Improving usability therefore should still be a goal for encouraging users to apply updates that are security related. This is particularly important as the Internet of Things evolves and users are faced not only with updating their personal devices but devices in their homes, office, on their bodies, and elsewhere. Our findings suggest four primary directions to improve desktop software updating interfaces: personalizing update interfaces, minimizing update interruptions, improving update information, and centralizing update management. We outline considerations about the socio-technical aspects of the software updating process specifically around trust, control, and consent.

6.1 Personalizing Updating Interfaces

Our first recommendation is to personalize updating interfaces to minimize notifications about updates that can safely be made silent and find those applications or systems that a user is likely to want to monitor for changes. In this vein, we could effectively increase the number of updates applied silently and reduce the overwhelming amount of notifications that may not be of interest to the users. This recommendation stems directly from our findings that users desire some control over what changes are made to a machine or application that they depend on in any capacity.

We envision the personalization of updating interfaces could occur in a similar way as others have suggested [22] in the domain of requesting Android application permissions. In other words, users could be shown only update requests that demand their attention and decision making, such as for applications they use actively on a day to day basis. This would give users ample opportunity to cancel potentially disruptive updates and minimize unwanted consequences. We recommend empowering users with a "cancel" option to prevent an update from ever occurring at the risk of them never applying certain updates. This stands in contrast to current automated updating systems that delay updates but install them anyway if a user has not responded in a certain period of time. Giving users more power over their systems and applications may make them more likely to trust the updating process.

We also propose that any updates that users do not actively wish to monitor could be made silent. For example, security updates and updates for infrequently used apps (which if left unpatched can still be sources of vulnerabilities [34]) could be applied without the users consent, assuming disk and data constraints are not an issue. Personalizing updating interfaces in this manner of course depends on whether a system can learn which applications or updates the user cares about and how to apply silent updates selectively. Our study highlighted several factors a system could consider for this purpose include the frequency of use of applications over time, its importance and an update's characteristics (e.g., purpose, size, or, installation time) to determine if users should be notified and prompted for consent. Users could also be unobtrusively asked at installation time to designate whether a particular application should notify them of any changes. In such a system, update notifications could also better highlight why certain updates are recommended.

Update interfaces could also be personalized by profiling users' individual personalities traits to see whether these can be correlated with various updating behavior preferences. Already, the Security Behavior and Intentions (SeBIS) scale [20] has shown that users vary in their software updating behavior intentions and how differences in risk taking, decision making, impulsiveness are correlated with security decisions (and software updating in particular) [19]. Future research could identify how automation defaults or updating interfaces could be configured as a function of these characteristics and how to better involve users in decision making about when to apply updates.

6.2 Minimizing Update Interruptions

Our second recommendation for improving desktop updating interfaces is to minimize update interruptions where possible to increase the uptake of updates including those that are security related. Update interruptions can be improved at the interface level by making update notifications less intrusive. For example, similar to our proposed design, update notifications could sit somewhere between passive and active notifications using icons that subtly and visually morph to indicate to a user that an update is available and provide more information to only those that desire it.

Future work could also consider how we can leverage Dynamic Software Updates (DSU) [26] to avoid restarts caused by updates altogether to minimize update interruptions at the back-end. Updates restarts could also be piggybacked on times that a user restarts a system or application on their own. This would also create a need for designing visual cues and nudges to gently prompt a user to restart an application or their machine to enable an update to be applied. For example, Google Chrome colors the Chrome Menu icon from green to orange to red over time to nudge users to relaunch their browsers [25].

6.3 Enhancing Update Information

Our third recommendation to improve updating interfaces on the desktop revolves around better informing users about updates and their consequences specifically by providing more information that builds trust in the update process, e.g., via compatibility reports and update ratings. Further research into how to generate compatibility reports and how update ratings can be gathered and provided will help users make informed choices about which updates to apply. For instance, "social proof" has already been shown to improve security feature adoption in Facebook and update ratings could similarly help users assess if they should move forward with an update [15, 14]. Other visual enhancements to show that update sources are verified or vetted could also instill trust in the updating process and further motivate users to perform updates.

6.4 Centralize Desktop Update Management

Our fourth recommendation for the desktop updating interface is to centralize update management where possible. We envision a large scale change to current interfaces that would require operating system vendors to provide better ways for updates to be pushed through a single channel or for the information about updates to be gathered for a central update information repository across a device. Examples of applications that are already making strides in this vein include Metaquark's AppFresh [30] for centralizing Mac OS X updates and SparkleProject, a framework for third party application developers to push Mac OS X updates [1]. This model is already manifest on mobile phones where updates and their notifications already propagate more centrally than on the desktop through app stores.

We also propose that a central update manager could provide ways for users to preview the effects of an update on their system to mitigate the fact that users are averse to unwanted changes. For instance, users could be given the option to try out the new version of an application or operating system via an interface overlay or by using a parallel version of an application installed on a system without committing to the update process or applying the update. Providing an easier way to roll back unwanted changes may also make users more amenable to apply updates without fear of

breaking their workflow, and therefore can potentially enhance security if those updates are applied as well.

6.5 Socio-Technical Updating Aspects

Our study and recommendations highlight the complex socio-technical nature of updates and the stakeholders involved. Updates involve trust between users and those seeking to make changes to their systems, gathering consent from users to make those changes, and surrendering control to external parties to make those changes. This involves a complex interplay of actors such as application and operating system vendors, developers, and users.

The question of whether these stakeholders will want to make the updating interface improvements we recommend possible remains open. For example, our recommendations depend on application developers modifying the information they provide about updates, how updates are deployed to users, how users are notified about updates, and the potential consequences of applying any update. To centralize update management, operating system vendors will have to build supporting frameworks to enable developers to push updates centrally and to have a vetting process similar to mobile systems for checking all updates and whether they comply with established guidelines for best informing users of upcoming changes.

Vendors might have large incentives to improve updating interfaces or otherwise risk alienating and losing their user base. For instance, a recent study of Tinder and Tesla updates [2], showed the backlash of unhappy users when unwanted changes were made without their consent to these apps. Yet, it is unclear who should have ultimate control over software changes, and whether there needs to be some governmental oversight for security purposes and consumer protection, particularly as some user bases grow as large as the population of several countries.

Furthermore, if update interfaces are indeed personalized, the question of transparency and accountability for selectively applying silent updates is also open i.e., how would such systems explain their decision making to users in a comprehensible way and provide meaningful controls to users so that they can still provide consent for changes being made to their applications and systems. We will only know more about whether a personalized interface would make or break users' mental models of updates by testing these recommendations out in the wild.

In all, these socio-technical issues around software updating will require the usable security community to closely examine the practice of software updating from all viewpoints. We can certainly empower users through improved software updating interfaces but we need to better understand all the nuances of control, consent, and trust in updates. We should also be asking ourselves the question of who should be allowed to make changes to systems to properly address the issue of improving security through updates.

6.6 Limitations and Future Work

Our studies have several limitations. First, the use of self-reported data in Phase One is subject to recall bias, meaning there may exist errors and differences in the recollections recalled by our participants. Second, because of the low fidelity nature of our prototype, its evaluation in Phase Three is based on participants' opinions rather than their actual behaviors. The evaluation also required participants to work with updates as a primary task and thus, the results may not generalize to real-world settings where software updating is considered a secondary task. Third, our participant pools were dominated by a younger set of participants, with many participants in Phase One having advanced degrees, and many students in Phase Three. Therefore, the results from both our studies may not generalize to the entire population, and hold limited validity.

To address these limitations, future work could examine updating behaviors more deeply with a higher fidelity prototype that can be deployed in the field, and validated against actual user behaviors. We also recommend testing future updating interface designs against a more representative group of Mac OS X users and extending the work to other operating systems and devices. Future work could explore how to improve updating interfaces beyond the desktop space, i.e. to mobile and the Internet of Things. Finally, given that updating involves an ecology of stakeholders, future work could examine updating practices from other perspectives such as how network administrators manage updates for large groups of users or how developers create updates in the first place.

7. CONCLUSION

We used a three phased research process to investigate current user barriers to software updates and determine how to improve desktop software updating interfaces. We found that users avoid updates primarily because they interrupt their computing activities, they lack information that enables them to decide whether to apply an update or not, and they notify and involve users in ways that are undesirable. Users responded positively to our *minimally-intrusive*, *information-rich*, and *user-centric* low fidelity prototype designed to minimize how updates interrupt the user, to augment current updating information to help users make a decision to update or not, and to centralize update management across a device. Based on the evaluation, we suggest that updates can be improved by personalizing software updating interfaces, minimizing update interruptions, improving information for update decision making, and by centralizing update management across a device. We also believe that the socio-technical aspects of updating are complex and need to be explored in more depth for future work. Ultimately, improving update interfaces and addressing these open questions will enhance the security overall.

8. ACKNOWLEDGEMENTS

We would like to thank Susan Wyche, Michelle Mazurek, Katie Shilton, and members of the Human–Computer Interaction Lab at the University of Maryland for their feedback on drafts of this paper. Our research is based upon work supported by the Maryland Procurement Office under contract H98230-14-C-0137. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Maryland Procurement Office.

9. REFERENCES

[1] Sparkle Project. `http://sparkle-project.org`, 2015.

[2] A. Acker and B. Beaton. Software Update Unrest: The Recent Happenings Around Tinder and Tesla. In

Proceedings of the 49 Hawaii International Conference System Sciences, HICSS '16. IEEE, 2016.

[3] D. Akhawe and A. P. Felt. Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 257–272, Washington, D.C., 2013. USENIX.

[4] Apple. Get Software Updates for Your Mac. `https://support.apple.com/en-us/HT201541`, 2015.

[5] D. Barrera and P. Van Oorschot. Secure Software Installation on Smartphones. *IEEE Security & Privacy*, 9(3):42–48, May 2011.

[6] T. Boren and J. Ramey. Thinking Aloud: Reconciling Theory And Practice. *Professional Communication, IEEE Transactions on*, 43(3):261–278, Sep 2000.

[7] M. Chetty, R. Banks, A. Brush, J. Donner, and R. Grinter. You're Capped: Understanding the Effects of Bandwidth Caps on Broadband Use in the Home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 3021–3030, New York, NY, USA, 2012. ACM.

[8] Codenomicon. The Heartbleed Bug. `http://heartbleed.com`, April 2014.

[9] H. Corrigan-Gibbs and J. Chen. Flashpatch: Spreading Software Updates over Flash Drives in Under-connected Regions. In *Proceedings of the Fifth ACM Symposium on Computing for Development*, ACM DEV-5 '14, pages 1–10, New York, NY, USA, 2014. ACM.

[10] A. Crabtree. Design in the Absence of Practice: Breaching Experiments. In *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, DIS '04, pages 59–68, New York, NY, USA, 2004. ACM.

[11] L. F. Cranor. A Framework for Reasoning About the Human in the Loop. In *Proceedings of the 1st Conference on Usability, Psychology, and Security*, UPSEC'08, pages 1:1–1:15, Berkeley, CA, USA, 2008. USENIX Association.

[12] N. Cyber Security Alliance. Stay Safe Online. `https://www.staysafeonline.org/`, 2015.

[13] J. Dadzie. Understanding Software Patching. *Queue*, 3(2):24–30, Mar. 2005.

[14] S. Das, A. D. Kramer, L. A. Dabbish, and J. I. Hong. Increasing Security Sensitivity With Social Proof: A Large-Scale Experimental Confirmation. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 739–749, New York, NY, USA, 2014. ACM.

[15] S. Das, A. D. Kramer, L. A. Dabbish, and J. I. Hong. The Role of Social Influence in Security Feature Adoption. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 1416–1426, New York, NY, USA, 2015. ACM.

[16] P. Dourish, E. Grinter, J. Delgado de la Flor, and M. Joseph. Security in the Wild: User Strategies for Managing Security As an Everyday, Practical Problem. *Personal Ubiquitous Comput.*, 8(6):391–401, Nov. 2004.

[17] T. Duebendorfer and S. Frei. Why Silent Updates Boost Security. *TIK, ETH Zurich, Tech. Rep*, 302, 2009.

[18] W. K. Edwards, E. S. Poole, and J. Stoll. Security Automation Considered Harmful? In *Proceedings of the 2007 Workshop on New Security Paradigms*, NSPW '07, pages 33–42, New York, NY, USA, 2008. ACM.

[19] S. Egelman and E. Peer. The myth of the average user: Improving privacy and security systems through individualization. In *Proceedings of the 2015 New Security Paradigms Workshop*, NSPW '15, pages 16–28, New York, NY, USA, 2015. ACM.

[20] S. Egelman and E. Peer. Scaling the Security Wall: Developing a Security Behavior Intentions Scale (SeBIS). In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 2873–2882, New York, NY, USA, 2015. ACM.

[21] M. Fagan, M. M. H. Khan, and R. Buck. A Study of Users' Experiences and Beliefs about Software Update Messages. *Computers in Human Behavior*, 51, Part A:504 – 519, 2015.

[22] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android Permissions: User Attention, Comprehension, and Behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 3:1–3:14, New York, NY, USA, 2012. ACM.

[23] T. Gerace and H. Cavusoglu. The Critical Elements of the Patch Management Process. *Commun. ACM*, 52(8):117–121, Aug. 2009.

[24] C. Gkantsidis, T. Karagiannis, and M. VojnoviC. Planet Scale Software Updates. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 423–434, New York, NY, USA, 2006. ACM.

[25] Google. Update Google Chrome. `https://support.google.com/chrome/answer/95414?hl=en-GB`, 2015.

[26] M. Hicks and S. Nettles. Dynamic Software Updating. *ACM Trans. Program. Lang. Syst.*, 27(6):1049–1096, Nov. 2005.

[27] D. Hinchcliffe. The App Store: The New "Must-Have" Digital Business Model. `http://www.zdnet.com/article/the-app-store-the-new-must-have-digital-\newlinebusiness-model`, January 2010.

[28] I. Ion, R. Reeder, and S. Consolvo. "...No one Can Hack My Mind": Comparing Expert and Non-Expert Security Practices. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 327–346, Ottawa, July 2015. USENIX Association.

[29] A. Mathur, B. Schlotfeldt, and M. Chetty. A Mixed-methods Study of Mobile Users' Data Usage Practices in South Africa. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 1209–1220, New York, NY, USA, 2015. ACM.

[30] Metaquark. Appfresh. `http://metaquark.de/appfresh/mac`, 2015.

[31] Microsoft. Install Windows Updates. `http://windows.microsoft.com/en-us/windows-vista/install-windows-updates`, 2015.

[32] Microsoft. Microsoft Security Intelligence Report Volume 18: July through December 2014. http://download.microsoft.com/download/7/1/A/71ABB4EC-E255-4DAF-9496-A46D67D875CD/Microsoft_Security_Intelligence_Report_Volume_18_English.pdf, 2015.

[33] D. Murph. Apple mac app store: open for business starting January 6th. http://www.engadget.com/2010/12/16/apple-mac-app-store-open-for-business-\newlinestarting-january-6th/, 2010.

[34] A. Nappa, R. Johnson, L. Bilge, J. Caballero, and T. Dumitras. The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 692–708, May 2015.

[35] J. Newman. How Mobile Apps are Changing Desktop Software. http://www.pcworld.com/article/259110/app_invasion_coming_soon_to_your_pc.html, July 2012.

[36] J. Oberheide, E. Cooke, and F. Jahanian. If It Ain't Broke, Don't Fix It: Challenges and New Directions for Inferring the Impact of Software Patches. In *Proceedings of the 12th Conference on Hot Topics in Operating Systems*, HotOS'09, pages 17–17, Berkeley, CA, USA, 2009. USENIX Association.

[37] E. L. Olmsted-Hawala, E. D. Murphy, S. Hawala, and K. T. Ashenfelter. Think-aloud Protocols: A Comparison of Three Think-aloud Protocols for Use in Testing Data-dissemination Web Sites for Usability. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2381–2390, New York, NY, USA, 2010. ACM.

[38] K. Sankarpandian, T. Little, and W. K. Edwards. Talc: Using Desktop Graffiti to Fight Software Vulnerability. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1055–1064, New York, NY, USA, 2008. ACM.

[39] M. A. Sasse, S. Brostoff, and D. Weirich. Transforming the "Weakest Link"–a Human/Computer Interaction Approach to Usable and Effective Security. *BT Technology Journal*, 19(3):122–131, July 2001.

[40] I. Seidman. *Interviewing As Qualitative Research: A Guide for Researchers in Education and the Social Sciences*. Teachers college press, 2013.

[41] Statista. Global market share held by operating systems Desktop PCs from January 2012 to June 2015. http://www.statista.com/statistics/218089/global-market-share-of-windows-7, 2015.

[42] 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 Conference on USENIX Security Symposium*, SSYM'09, pages 399–416, Berkeley, CA, USA, 2009. USENIX Association.

[43] Symantec. 2015 Internet Security Threat Report, Volume 20. https://know.elq.symantec.com/LP=1542, 2015.

[44] Target Corporation. Data Breach FAQ. https://corporate.target.com/about/shopping-experience/payment-card-issue-faq, April 2014.

[45] Y. Tian, B. Liu, W. Dai, B. Ur, P. Tague, and L. F. Cranor. Supporting Privacy-Conscious App Update Decisions with User Reviews. In *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '15, pages 51–61, New York, NY, USA, 2015. ACM.

[46] K. E. Vaniea, E. Rader, and R. Wash. Betrayed by Updates: How Negative Experiences Affect Future Security. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 2671–2674, New York, NY, USA, 2014. ACM.

[47] M. Vojnovic and A. Ganesh. On the Effectiveness of Automatic Patching. In *Proceedings of the 2005 ACM Workshop on Rapid Malcode*, WORM '05, pages 41–50, New York, NY, USA, 2005. ACM.

[48] R. Wash and E. Rader. Too Much Knowledge? Security Beliefs and Protective Behaviors Among United States Internet Users. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 309–325, Ottawa, July 2015. USENIX Association.

[49] R. Wash, E. Rader, K. Vaniea, and M. Rizor. Out of the Loop: How Automated Software Updates Cause Unintended Security Consequences. pages 89–104. USENIX Association, 2014.

[50] M. Wu, R. C. Miller, and S. L. Garfinkel. Do Security Toolbars Actually Prevent Phishing Attacks? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 601–610, New York, NY, USA, 2006. ACM.

APPENDIX

A. PHASE ONE: INTERVIEW GUIDE

Thank you for participating in today's study. As you read in the consent form, we will be recording the session so we can review it to make sure that we don't miss any part of our conversation. Your name will not be associated with the recording or with any data I collect. Your comments and opinions will only be used in combination with the feedback gathered from other people participating in this study. Your individual comments will remain confidential. Do you have any questions regarding the consent form? Do I have your permission to start the recording?

Session Introduction

Today, I'm going to be talking with you about security issues and discussing your software updating habits. The interview should last around 30 to 45 minutes. Before we begin, there are a few things I would like to mention:

1. During our discussion, I will ask you to share with me your thoughts and opinions on the different topics that we cover. I would like you to be honest and straightforward about your knowledge and habits regarding software security. I might ask you to expand on anything you mention if the information could be useful to the study. Please keep in mind that there are no right or wrong answers.
2. We are not here to test you or your knowledge about security issues or software technology. We want to get

a better understanding of how people really think and act when it comes to these issues.

3. Please feel free to comment on any thoughts or ideas you have as we talk. Your feedback is important in that it helps us get a better picture of real user behavior. Do you have any questions before we begin? Okay, let's get started.

Security Awareness

1. Are you aware of any major breaches to personal online security?
2. Have you heard about the following security breaches: Heartbleed, Adobe 2013, Target 2013
3. How do you find about online security issues? From which sources?
4. How do you determine integrity of the source of software update? Have you ever avoided installing an update because it was not from an authentic source?
5. Have you ever been a victim of an online security breach?

Security Habits

1. Do you secure your physical electronics (like laptop, tablet, phone)? How?
2. Are you concerned about email security? Do you take any measures to protect your security on email?
3. Are you concerned about software security? Do you take any measures to protect your security in regards to software?

Software Updates

1. What comes to mind when you think of software updates? How do you feel about them? Do you usually have a positive or negative feeling? What is your motivation for acting on software updates? How do you decide whether to go through the update or not?
2. How comfortable are you in installing software updates? If not, what would make you more comfortable?
3. Do you treat all software updates the same? Or do you think or act differently depending on the type, say whether they are app updates, or operating system updates?
4. Do you feel that it's necessary to update software every time an update is available? Are there some updates that you always do and some that you usually ignore?
5. Does the process of software updating feel like a necessity or more of an interruption? Why?
6. Do you typically understand what the update is going to do to the software before you download and install it? Do you read the information presented in the update notice? Do you understand the information? Does this information have an impact on your decision to go through with the update or not?

Software Update Process

Discovering the update

1. How do you find out about a software update? Do you usually wait to be notified or if you hear about an update from an external source, do you seek it out?
2. Are there any barriers that get in the way of you discovering updates?
3. What makes it easy to discover updates?

Downloading the update

1. How do you download updates? Where do you go or what do you do? Do you seek them out?
2. Are there any barriers to downloading updates? For example, connectivity issues like data usage, Wi-Fi availability?
3. What makes it easy to download updates?

Installing the update

1. How do you typically install updates? Automatically or manually?
2. When do you typically install updates? After download or later (why if later)?
3. Are there any barriers to installing updates (restarting, downloading an installer, interrupting current activity)?
4. What makes it easy to install updates?

Applying the Update

1. What do you typically expect to have happen after you install an update and begin using the software again?
2. Does your interaction with the software typically match your expectations after
3. Do you usually have a positive or negative experience after installing an update?
4. Do any of these factors have an effect on how you feel about future updates?

Software Updating Preferences

1. Do you configure any of your systems to do updates in any of silent, automatic, manual ways? Do you prefer any of these mechanisms? Why or why not? Do you know where to change these settings in your operating system? And the settings for each piece of software.
2. What are the reasons you go through with software updates? Does it depend on the device? Piece of software? Operating system? How do you determine if an update is critical or not? How do you feel about whether an update might address a security issue vs. a feature change?
3. What are the reasons why you might avoid software updates? Frequency of patches (Do you avoid if they're released more frequently?), cost of updates, connection speed, incompatibility with other software, fear of breaking existing software/changing interface.

Current Software Updating Interfaces

1. How frequently do you update your software?
2. Which device do you most frequently update and why? (Does the update behavior depend on the device being used?)
3. What software do you update more often?
4. Have you installed updates both on Windows PC and Mac OS? Which one did you prefer?
5. What browsers do you use to access Internet?
6. Which one do you prefer most in terms of update?
7. Did you ever avoid installing an update to avoid incurring additional charges when your Internet was not free? (wifi/3G/broadband)
8. How would you want to improve the updating process?

B. PHASE THREE: THINK-ALOUD TASKS

Locating updates

1. Find out if there are software updates for your machine
2. Find the list of available updates
3. Cancel the OS X update from taking place

Authenticating the Source

1. Verify that the source of iTunes 11.1.1. update is authentic
2. How would you get additional information about what the iTunes 11.1.1 update does?

Installed Updates

1. Close the update manager and find out if there are any more updates
2. Find out which updates were recently installed

Information About the Update

1. How much time will the next update will take to download and install?
2. Change the update preferences for iTunes updates
3. Check for applications the OS X 10.9.2 update is incompatible with
4. Read update ratings and comments from users