Please Fix Your Interfaces: The Ease of *Basic* Usability

Sumeet Gujrati and Eugene Vasserman
Kansas State University

This paper does not propose a tool per se, but rather encourages the use of best practices and methods, well-known "tools" in the Human-Computer Interaction (HCI) community, when building new privacy-enhancing tools or evaluating existing ones. As security professionals, we strive to develop tools which provide strong protection against vulnerabilities, but the tools fail in practice if they are unusable. We show that the amount of work required to drastically improve the usability of a piece of software, even third-party code, is far from prohibitive. We synthesize the lessons learned from applying known HCI methodology, hopefully smoothing the process for future readers. These lessons allow developers to quickly evaluate their software for common usability flaws, which would allow significant usability improvements in only a few hours of developer time. As a case study, we evaluated the usability of TrueCrypt for two main tasks which *must be performed by every new user* of the software: *creating* an encrypted file container and *opening* that container. Unlike whole disk encryption which generally requires a password once at boot time, containers (and associated passwords) are used as needed, potentially creating a conflict between security and usability, in which the latter usually triumphs [2, 3, 8].

The main "tool" in our arsenal is the **cognitive walkthrough**, through which we found TrueCrypt to be very difficult to learn and understand for non-expert users. Cognitive walkthroughs are a usability evaluation and improvement method in which an evaluator, knowledgeable in the domain of the software to be tested, uses the software like a novice user would, and identifies interface issues [6, 7, 10]. At each step of the walkthrough, the evaluator is mindful that the user should be able to:

- achieve the right effect,
- notice that the correct action is available,
- associate the correct action with the desired effect, and
- see that progress is being made towards a solution (assuming correct actions are performed).

Lewis et al. find that this method has the advantages of low cost and speedy results (almost 50% issues can be revealed) [4], but is not a replacement for a full user study, as some straightforward changes lead to surprising outcomes. However, the ease and accuracy of the cognitive walkthrough and the scale of measurable usability improvements suggest that all security software would benefit from this inexpensive and quick exercise.

Based on our findings in the cognitive walkthrough, we suspected that TrueCrypt's user interface is unnecessarily complex and has serious usability issues. We validated our hypothesis through a **user study** of a modified interface, incorporating a post-installation quick-start wizard and a streamlined process for initial container creation and use. We also overhauled the visual appearance and rewrote the terminology used throughout the application.[1] We ensured that all security functionality is retained — the security-sensitive operations of the back-end logic are unaffected. User study results showed statistically significant improvements in average time to complete common tasks (see Fig. 1). Surprisingly, some tasks were found to be *more difficult* in the new interface, but these misguided changes can be reverted back to their original design without sacrificing other, clearly beneficial changes. We therefore show, at a high granularity, which changes to which TrueCrypt interface components are beneficial, and which are detrimental to usability.

Through this experience we demonstrate that a few hours of user interface analysis at the design or development phases, or even immediately before release, can vastly improve the usability of the software. The payoff is high even when using external, never-before-seen code, as well as feature-frozen code. We find that usability can be improved **without sacrificing features** of the final product. We further note that developers do not need to be experts in user interface design to carry out this exercise. We distill the following lessons learned from our evaluation of TrueCrypt's interface and usability:

[1] Our redesigned interface is available as a patch to the TrueCrypt source code (http://www.cis.ksu.edu/~eyv/projects/truecrypt/). We contacted TrueCrypt's authors, offering our improvements for incorporation into the official distribution.

1. **Avoid unknown terminology:** Our results highlight the important relationship between terminology and usability of security software, for example changing the terms "Mount," "Dismount," and "Volume" to "Open," "Close," and "File Container," respectively. We suggest removing unknown terms or minimizing their use, and using intuitive terms instead, even when they may not be the correct technical terms. If a more intuitive term can not be found, such as the name of an encryption algorithm, then developers should separate that term from the main workflow of the task (see Suggestion 3 below).
2. **Do not misuse native UI controls:** Users are most likely to be familiar with behaviors of common native controls. For example, clicking a "Command Button" control, such as "OK" or "Cancel," the user can initiate an immediate action. The label and/or icon on it indicates the action it performs when clicked, and the user should be able to determine the action without any external reference. The Microsoft user experience interaction guidelines for command buttons [1] further states that adding an ellipsis (...) at the end of the button label indicates that the command needs additional information (including confirmation). The command button is not designed as a substitute for a pop-up menu, or to indicate an action which it does not perform. For example, the behaviors of "Volume Tools..." button which displays a pop up menu, and "Select File..." button which requires entering a new file name (or it will overwrite the selected file) are not typical command button behaviors, and confused users.
   We ourselves inadvertently violated this suggestion by replacing the drive letter and opened container association with a command button marked with an open folder icon. The command button icon suggested the action of opening a folder, but failed to convey to the user "which drive" or "which path" it is opening. In this case, the icon on the button does not indicate the complete action it performs. Since two concepts are being conveyed: 1) the drive letter of the TrueCrypt container, and 2) the path to the TrueCrypt file, a single control/label proved insufficient. Developers should refer to user interface design guidelines of specific controls if unsure about their correct usage.
3. **Separate required and optional parameters:** Optional input parameters may be unknown to novice users (e.g. names of encryption algorithms), and displaying these choices may impede successful completion of tasks. Those parameters should be assigned reasonable default values (as TrueCrypt does) and be separated from the main workflow (which TrueCrypt does *not* do — see Figs. 2(b) and 2(e)). Advanced users can change them by launching an optional input screen. There are occasions when even required parameters may be unknown to the user. In those cases, we suggest providing sufficient but small timely help messages, such as tool tips, as we have observed that users do not read lengthy help messages and/or do not understand them. It is also important to gather required input ahead of time, using as few screens as possible. The effects of this are evident from the modified file container creation wizard (Fig. 3), which reduced the number of steps to create a container from 5 to 1.
4. **Display the consequences of an action immediately (Immediacy of consequences):** In the context of user interfaces, the consequences of an action such as clicking a button, if displayed immediately after performing the action, affect the user's behavior (in terms of ability to rectify the action if performed incorrectly) more positively than if displayed at a later point of time. In one such study, Meson and Redmon examine the effects of immediate versus delayed visual feedback on the accuracy of identifying errors (signals) in sample stimuli and suggest that the accuracy of signal detection is greater under immediate feedback than delayed feedback [5]. Vaniea et al. aim to understand spatial proximity of access control information and the item that the information describes, and reveal that placing the access-control policy directly under the object that the policy describes (such as photo and album thumbnail) improves users' ability to notice errors in setting access-control policies without negatively impacting other non-access-control tasks than placing it on the sidebar, or on a separate settings page [9]. TrueCrypt's original file container creation wizard performs poorly in this regard: if an existing file is selected and "Next" button is pressed in step 1 (Fig. 2(a)), the wizard displays an overly lengthy warning message stating that the file already exists and will be replaced. If this warning is ignored, the wizard presents another warning when "Format" is pressed in step 5 (Fig. 2(e)). It is unlikely that users expect this behavior or know to return to step 1 to fix it. We suggest the developers to thoroughly analyze the action performed by each control and its consequences, and provide options to the users to rectify a potential problem (if needed) immediately rather than delaying.

References

[1] Command Button Guidelines, accessed on August 29th, 2012. [online] http://msdn.microsoft.com/en-us/library/windows/desktop/aa511453.aspx.

[2] Ross Anderson. Why cryptosystems fail. In *Proceedings of ACM CCS*, December 1993.

[3] Don Davis. Compliance defects in public key cryptography. In *Proceedings of the USENIX Security Symposium*, July 1996.

[4] Clayton Lewis, Peter G. Polson, Cathleen Wharton, and John Rieman. Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, pages 235–242, 1990.

[5] Matthew A. Mason and William K. Redmon. Effects of immediate versus delayed feedback on error detection accuracy in a quality control simulation. *Journal of Organizational Behavior Management*, 13(1):49–83, 1993.

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

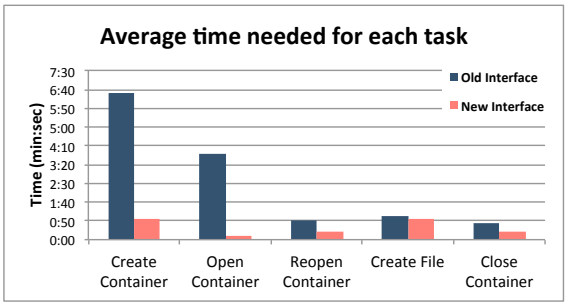
[7] John Rieman, Marita Franzke, and David Redmiles. Usability evaluation with the cognitive walkthrough. In *Conference companion on Human factors in computing systems*, pages 387–388, 1995.

[8] M. Angela Sasse and Ivan Flechais. *Security and Usability*, chapter Usable Security: Why do we need it? How do we get it?, pages 13–30. O'Reilly Media, 2005.

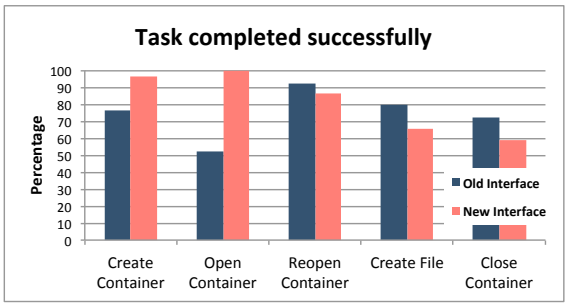
[9] Kami Vaniea, Lujo Bauer, Lorrie F. Cranor, and Michael K. Reiter. Out of sight, out of mind: Effects of displaying access-control information near the item it controls. In *Proceedings of the Conference on Privacy, Security and Trust*, July 2012.

[10] Cathleen Wharton, John Rieman, Clayton Lewis, and Peter Polson. *The cognitive walkthrough method: A practitioners guide*. John Wiley & Sons, Inc., 1994.

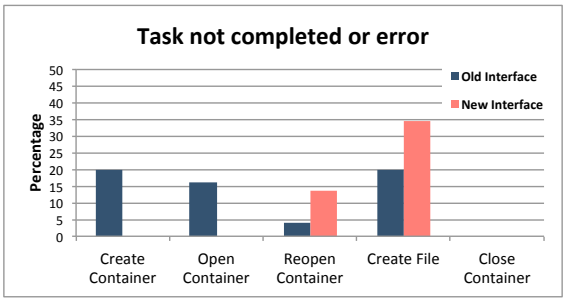
A Interface improvements and performance results



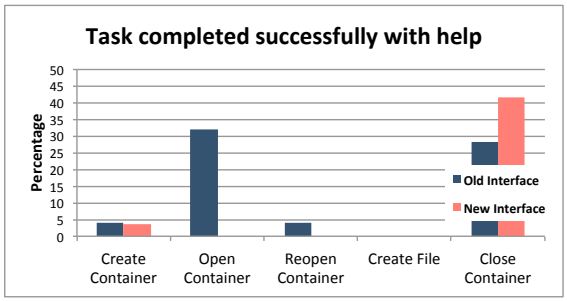
(a) Time required to perform various tasks. All differences statistically significant except "Reopen Container" and "Create File"



(b) Percentage of successfully-completed tasks



(c) Percentage of tasks not completed or completed with errors



(d) Percentage of tasks successfully completed with hints

Figure 1: Comparison of user performance when using the old and new interfaces in terms of task time (a) and success rates (b)–(d).

(a) Step 1: File selection screen

(b) Step 2: Encryption options screen

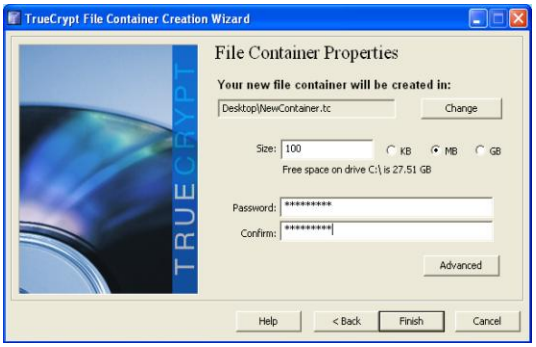
(c) Step 3: Container size screen

(d) Step 4: Container password screen

(e) Step 5: Container filesystem options screen

(f) Results: Finish screen

Figure 2: The five-step TrueCrypt file container creation wizard (steps shown in order).



(a) Combined required options screen, incorporating steps 1–5 of the original interface (Fig. 2(a)–2(e))

(b) Finish screen incorporating parts of step 5 and all of step 6 of the original interface (Fig. 2(e)–2(f))

Figure 3: The *single-step* file container creation wizard in our modified interface. Selecting "Finish" transitions from (a) to (b).