Usability of Anonymous Web Browsing: An Examination of Tor Interfaces and Deployability

Jeremy Clark
School of Information and Technology (SITE)
University of Ottawa
jclar037@site.uottawa.ca

P.C. van Oorschot
School of Computer Science
Carleton University
paulv@scs.carleton.ca

Carlisle Adams
School of Information and Technology (SITE)
University of Ottawa
cadams@site.uottawa.ca

ABSTRACT

Tor is a popular privacy tool designed to help achieve online anonymity by anonymising web traffic. Employing cognitive walkthrough as the primary method, this paper evaluates four competing methods of deploying Tor clients, and a number of software tools designed to be used in conjunction with Tor: Vidalia, Privoxy, Torbutton, and FoxyProxy. It also considers the standalone anonymous browser TorPark. Our results show that none of the deployment options are fully satisfactory from a usability perspective, but we offer suggestions on how to incorporate the best aspects of each tool. As a framework for our usability evaluation, we also provide a set of guidelines for Tor usability compiled and adapted from existing work on usable security and human-computer interaction.

Categories and Subject Descriptors

H.1.2 [**Models and Principles**]: User/Machine Systems—*Human factors, Software psychology*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Evaluation or methodology, Graphical user interfaces*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Human Factors, Security

Keywords

Anonymity, browsing, onion routing, privacy, Privoxy, Tor, usable security

1. INTRODUCTION

Tor is an important privacy tool that provides anonymous web-browsing capabilities by sending users' traffic through a network of specialized proxy servers designed to unlink the sender's identity from her traffic [4, 14]. Like any application, Tor must be usable in order for it to be widely adopted. To this end, a number of tools have been developed to assist a user in deploying and using Tor. By examining the usability of these tools and offering suggestions for their improvement, the motivation of this paper is to help increase Tor's affability among novice users with hopes of expanding its user base. In this paper, we examine the usability of Tor and evaluate how easy it is for novice users to install, configure, and use Tor to anonymise the Firefox web-browser.

In a Tor network, anonymity is achieved by mixing the sender's internet traffic with traffic from other users such that the true sender of a particular message is indistinguishable from the set of all users. In other words, the sender is only anonymous within a crowd [13]. As such, the sender's anonymity is contingent on other users being untraceable. If the other users commit errors that allow them to be traced, the number of users covering for the sender decreases, having a direct effect on her own anonymity. It is thus in the sender's individualistic interest that not only she, but the other Tor users, can properly deploy and use the software—a factor that differentiates Tor from many other security applications and underscores the importance of examining the usability of Tor.

As critical as usability is to the successful and wide adoption of Tor, it appears that no extensive usability study has been presented in the literature. Our first contribution is to compile a set of Tor-relevant usability evaluation guidelines from a variety of sources, eliminate the redundancies, and offer justifications—in some cases, based on research in cognitive psychology not yet applied to usable security and privacy. Our guidelines build on the earlier guidelines proposed to date, including Whitten and Tygar [26] and others, however our guidelines are appropriately shifted in focus from usable security to usable privacy. Using our guidelines, we perform a cognitive walkthrough of the core tasks of installing, configuring, and running Tor. We examine manually configuring Firefox for use with Tor, Privoxy (a filtering proxy) [3], and Vidalia (a GUI for Tor) [7]. We also examine two Firefox extensions, Torbutton [5] and FoxyProxy [2], designed to assist the user in performing the key tasks. And finally we inspect Torpark [6]—a standalone Firefox variant with built-in Tor support. We uncover numerous usability issues with each deployment option but find that the extensions and Torpark offer some improvement in important areas.

The remainder of this paper is organized as follows. In Section 2, we review the preliminaries of anonymous communication and onion routing, and examine the relevant threat models. Section 3 will provide an overview of the cogni-

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

tive walkthrough methodology, and present our guidelines for usable privacy. In Section 4, we commence the cognitive walkthrough with the installation of Tor. Section 5, 6, 7, and 8 will evaluate the manual configuration of Tor, Privoxy, and Vidalia, and then Torbutton, FoxyProxy, and Torpark respectively. We provide a comparison and summary of configurations in Section 9, related work is reviewed in Section 10, and concluding remarks are made in Section 11.

2. ANONYMOUS BROWSING

Anonymity means different things in different contexts. As has been pointed out by others [17], the traditional definition of anonymity is the 'state of namelessness.' Namelessness implies moving through society without an identifier—nothing to tie your actions to your identity. In the online world, identity is typically defined by one or more digital pseudonyms [13]. A *self-volunteered* identifier is a digital pseudonym used to access a webservice (i.e., a screen-name, user-name, or email address). A *server-assigned* identifier is a unique identifier used by a webservice to monitor their users (i.e., a cookie). The anonymity afforded by Tor does not extend to either of these two categories of identifiers.

In the context of Tor, anonymity means preventing the dissemination of a user's internet protocol address. There are reasons to be concerned about the privacy of an IP address. An internet service provider (ISP) can link its addresses to the owner of the account paying for it—information that is obtainable for a jurisdictional variety of reasons. Furthermore when an IP address is augmented with elements from the other categories of pseudonymous identifiers or additional information, a user's true identity may be uncovered—for example using geo-location [23, 20].

The general approach to controlling the dissemination of IP addresses online is through proxy servers. Conceptually, a proxy is someone who acts on another entity's behalf. Internet traffic may be forwarded to a 'person-in-the-middle' who then forwards source and destination traffic between the user and the webservice. A proxy server segments the connection between the user and the server into the user-proxy link and the proxy-server link. An eavesdropper could realistically exist on either of these links. On the server-side, many websites keep logs of the IP addresses of the visitors to their site and the use of a proxy will result in the proxy's address being stored instead. On the client-side, an ISP can monitor and/or log the activities of their customers, something a simple proxy cannot prevent.

The *mix proxy*, proposed by Chaum [10], uses encryption on the client-side link to prevent an eavesdropper from reading the messages before they enter the proxy. Then, in order to prevent simple timing attacks—where encrypted packets entering the proxy are correlated to the decrypted packets which are immediately outputted by the proxy—a mix proxy does not process and release traffic in its received order. Rather, these proxies take a fix-sized set of incoming packets from a group of different users and randomly reorders the packets before they are sent out. In the case of services that are not time sensitive, such as email, the servers may introduce random delays or use more complicated releasing rules.

Often clients route traffic through a constantly changing path of several mix proxies. This is known as a *mix network*. While many different mix network architectures have been proposed in literature, only a few have actually been implemented and widely deployed. A popular solution for web traffic is The Onion Router (Tor), proposed by Dingledine et al. [14]. Tor routes traffic through three mix proxies by default. The sender encrypts her message three times—a layer of encryption for each mix proxy in the path. Since the encryption is layered, the packaged message is metaphorically referred to as an onion. Each proxy removes the outer layer of the onion as it moves through the network, and the centre of the onion is the plaintext packet for the receiver. This method is known as *onion routing*, and Tor is an implementation of this method. Tor operates on a fixed number of messages at a time, called a batch, and it processes the message set once the batch is full.

The most naïve analysis of Tor would conclude that a given packet could have been originated by anyone who contributed packets to any of the batches across the entire network at the same time that the packet in question was contributed. This group of potential originators is referred to as an *anonymity set*. In the best-case scenario, all the clients in an anonymity set are indistinguishable. However in reality, this is not necessarily the case [21]. As we will see in the following sections, a number of dangerous errors exist that would allow originators to be identified. If Bob is in Alice's anonymity set but commits a dangerous error, then Alice's anonymity set is smaller and she is closer to being uniquely identifiable. Thus Alice's ability to avoid dangerous errors is a necessary but not sufficient condition for remaining non-identifiable—Alice's non-identifiability is also contingent on having the other clients in her anonymity set be non-identifiable. In terms of usability, a user is reliant not only on her own ability to correctly use Tor but on the ability of other users as well. This point is discussed by Dingeldine and Mathewson [13], who conclude that usability is paramount to the anonymity provided by Tor.

The Tor application only makes the onions. The application generating the traffic—in this paper, we consider Firefox—needs to be configured to direct its traffic into Tor, which is listening on a local port. Furthermore, this traffic is often first filtered through a different application called Privoxy [3] to ensure DNS lookups get captured and anonymised, as well as scrubbing various types of identifying data contributed to the packets from the higher layer protocols in the network stack (in particular, the application layer). Tor is a SOCKS proxy [19] and so a typical configuration will direct http, https, and DNS traffic from Firefox to Privoxy, and SOCKS traffic directly from Firefox to Tor. Privoxy then filters its received traffic and passes it into Tor through a SOCKS connection. Tor currently comes bundled with a graphical user interface called Vidalia [7].

3. GUIDELINES AND METHODOLOGY

3.1 Evaluation Methodology

A common usability evaluation method is heuristic evaluation [22]. In heuristic evaluation, a double expert (someone who knows both the application domain and human-computer interaction principles) evaluates the user interface of the application against a set of heuristics. We will employ a related method: cognitive walkthrough [25], as has been used previously by others [26].

The premise behind a cognitive walkthrough is that users learn by exploring the software interface. They often do not sit down and learn the software interface *a priori* through

a user manual or by systematically visiting every nook and cranny of the user interface. Rather they attempt to perform a task and rely on the interface to intuitively guide them through that task. The user is presumed to be pragmatic and she does the minimal amount of work to satisfy herself that the task is completed.

In a cognitive walkthrough, a set of core tasks is designated. The evaluator will perform these tasks and evaluate the usability of the application against a set of evaluation guidelines as these tasks are being performed. Thus it is similar in methodology to heuristic evaluation, but differs in the approach—instead of systematically exploring the user interface, the evaluator performs core tasks.

Cognitive walkthroughs are an important first step in evaluating the usability of a software tool. Conducting studies with users, either in a laboratory or in the field, is a common follow-up. However due to time constraints and limited budgets, user studies are typically constrained to a narrow set of tasks. By contrast, the power of the cognitive walkthrough is the breadth that it makes possible. In this paper, we examine four different deployments of Tor, from installation to configuration to actual use, and perform comparative analysis. Such a wide comparison would be extremely challenging, if not impossible, in a typical user study. A user study is most appropriate when one or two predominate solutions emerge from a large solution set which, as we will see, is not the situation we currently have with Tor.

3.2 The Core Tasks

The core tasks to be performed in our cognitive walkthrough are as follows.

CT-1 Successfully install Tor and the components in question.

CT-2 Successfully configure the browser to work with Tor and the components.

CT-3 Confirm that the web-traffic is being anonymised.

CT-4 Successfully disable Tor and return to a direct connection.

The core tasks will be performed on the following four configurations:

1. Firefox with Tor [4], Vidalia [7], and Privoxy [3];
2. Firefox with Tor, Vidalia, Privoxy, and Torbutton [5] (a Firefox extension);
3. Firefox with Tor, Vidalia, and FoxyProxy [2] (a Firefox extension); and
4. Torpark [6] (a self-contained, anonymous browser).

There is considerable overlap between the first three configurations. Both Torbutton and FoxyProxy allow the user to shortcut a few of the steps in configuration 1, but most of the steps are common between them. Thus, the walkthrough of the first configuration will be significantly longer and more involved than that for the second and third.

3.3 Usability Guidelines

The set of guidelines that we use to evaluate each of the core tasks are as follows.

G1 Users should be aware of the steps they have to perform to complete a core task.

G2 Users should be able to determine how to perform these steps.

G3 Users should know when they have successfully completed a core task.

G4 Users should be able to recognize, diagnose, and recover from non-critical errors.

G5 Users should not make dangerous errors from which they cannot recover.

G6 Users should be comfortable with the terminology used in any interface dialogues or documentation.

G7 Users should be sufficiently comfortable with the interface to continue using it.

G8 Users should be aware of the application's status at all times.

These guidelines are drawn from a variety of sources [25, 26, 16, 12, 11] and are intended for evaluating Tor specifically. However they are suitably broad and may find application in other usable privacy walkthroughs. We now individually justify the inclusion of each.

G1: Users should be aware of the steps they have to perform to complete a core task.

This is a restatement of the first guideline of Whitten and Tygar [26]. Every user of a new application knows certain things before using the system and learns certain things during the use of the system. In the cognitive walkthroughs we carry out here, the presupposition is that the user knows enough to start the process for each core task—in the case of installation, the user can download the installation file and open it; in the case of configuration, the user can explore the user interface or follow cues. We are evaluating how the application cues the user to perform the intermediary steps between these broadly defined tasks.

G2: Users should be able to determine how to perform these steps.

Once the user is aware of what intermediary steps are necessary, she must be able to figure out how to perform these steps. This is the second guideline in [26]. It is assumed the user has a mental model of how the system works. It is thus important that the system model be harmonized with the user's mental model if the user is to be successful in performing the necessary steps required to complete each core task [25]. What is less obvious is why we cannot fully rely on the user to simply modify her mental model when given conflicting information.

A predominate reason is that humans have a stronger preference for confirming evidence than disconfirming evidence when evaluating their own hypotheses. This cognitive bias is well illustrated by Wason [24], who conducted a study where a set of subjects were given the following sequence of

numbers: 2,4,6. The subjects were told that the numbers followed a rule, and their task was to determine the rule by proposing their own sequence of numbers, which would be declared as matching the rule or not. The rule was any ascending sequence of numbers. However most subjects derived a more complicated rule, such as every ascending sequence of numbers differing by two. The point of this test was that the subjects, on average, had a preconceived idea of what the rule was and only proposed sequences to confirm that rule, instead of also proposing sequences that would falsify their perceived rule.

Confirmation bias is important in usability because it proposes that users are biased toward only seeking and accepting information that confirms their mental model, and thus may avoid or even ignore information that contradicts it. It cannot reasonably be expected that users will easily and quickly adapt their mental model to new information.

A second concern with how users perform these steps is that security is a secondary goal [26, 16]. If the user is given two paths to completing a core task—one that is fast but not secure, and one that is slower but more secure–it cannot be assumed that the user will take the latter approach. In fact, studies in behavioral economics demonstrate that humans often prefer smaller immediate payoffs to larger future payoffs, such as $50 today instead of $100 a year from today [18]. Using software securely has a greater (usually non-monetary) payoff for the user, but this utility has to be substantially higher than the alternative to justify the delay in achieving it.

G3: Users should know when they have successfully completed a core task.

In other words, users should be provided with ample feedback during the task to ensure they are aware of its successful completion. This principle has been proposed in the context of heuristic evaluation [25] and for a cognitive walkthrough [11]. It was also mentioned by Cranor [12]. With Tor it is critical that users be provided with a tool to determine that their traffic is actually going through the Tor network, as there is no easy way for them to determine this for themselves (short of running a packet sniffer).

G4: Users should be able to recognize, diagnose, and recover from non-critical errors.

Users will likely make errors in performing the core tasks and it is important for them to be able to recover from these errors [25]. It is important for users to be given concise error messages.

G5: Users should not make dangerous errors from which they cannot recover.

This guideline is from Whitten and Tygar [26]. Tor has a few dangerous errors associated with it. The first is a false sense of completion, where the user thinks that her traffic has been anonymised when it has not.

Second, in converting domain names into IP addresses, a web browser will often consult a domain name server (DNS) provided by the ISP. These DNS lookups do not happen over the typical port used for web-traffic and so configuring a browser to use Tor as its SOCKS proxy does not result in DNS lookups funneling through Tor. If the DNS look-ups are not anonymised, then the ISP can know when you visit each new site whose IP address is not cached locally in your hosts file if it monitors traffic to and from its DNS.

Finally, it has been shown (e.g. [20]) that allowing client-side execution of mobile code, such as downloading and executing a Java applet, could compromise the user's anonymity when browsing a website. An applet can be created to simply request the user's local IP address using a system call to the kernel, and to pass that information back to the website over regular http (or https if Privoxy is configured to filter out packets containing the user's local IP address). To avoid this dangerous error, users must disable Java. A similar exploit may be possible with ActiveX, Flash, or JavaScript as well (although Firefox does not come with ActiveX support).

G6: Users should be comfortable with the language used in any interface dialogues or documentation.

Wharton *et al.* emphasize that applications should use simple, natural, and familiar language [25]. This is also a central design goal for Privacy Bird [12], which takes natural language privacy policies and translates them from legalese into concise, human readable summaries. In the case of Tor, Privoxy, Torpark, *etc.*, we are looking at the understandability of dialogues and the documentation, and their reliance on technical language which users may have trouble understanding.

G7: Users should be comfortable with the interface.

This is the fourth principle of usable security of Whitten and Tygar [26], and is an essential part of the principal of psychological acceptability quoted by Bishop [9]. While Tor and Privoxy do not have an extensive user interface, we will be evaluating it in terms of how intuitive it is to use. We will also be analysing the interface of the Firefox extensions.

G8: Users should be aware of the system status at all times.

This principle was proposed in the context of heuristic evaluation [25] and cognitive walkthrough [11]. Cranor advocates the use of 'persistent indicators' that allow the user to see privacy information at a glance [12]. In terms of Tor, we are looking for indicators that show Tor is enabled as well as perhaps, for example, the current external IP address of the exit node, the number of users using the system, or other relevant privacy information.

4. TOR INSTALLATION

The Tor installation file may be downloaded from the Tor website [4]. The Tor download page may be confusing to a novice user unfamiliar with the open source software movement. Alongside the versions of Tor for other operating systems, there are three versions offered for Windows: a version marked only with its version number, a version marked by its version number and "alpha," and a link to "Windows packages for experts." Each installation file also has a digital signature file. At the top of the page, it is stated which version number is the "stable" release and which is a "development" release. This is a slight breech of G6 for using unfamiliar language. The intention is certainly for non-expert users to install the stable version. A statement at the top of the page designated for non-expert, Windows users and linking directly to the stable version installation file without mentioning version numbers would alleviate potential confusion over which version to download.

The installation process is a fairly straightforward wizard-style installation. The first dialogue screen informs the user that the installation bundle actually installs three distinct applications: Tor, Privoxy, and Vidalia. It is not clear from the installation dialogue what each application does or why the three are needed. The next dialogue alerts users with previously installed versions of any of these applications to exit them before proceeding. We confirmed this is not enforced and it is possible to proceed without exiting.

The next dialogue allows users to choose which components to install. It offers a tree-style list of components that can be expanded or collapsed, and each component has a checkmark that can be selected or deselected. The three top nodes are Tor, Vidalia, and Privoxy. It states that users may "position [their] mouse over a component to see its description." Given that the user has not been cued as to what exactly Vidalia and Privoxy are, this would be an excellent place to indicate their purpose. However, the description for Vidalia is simply "Install Vidalia 0.0.7," and likewise for Tor and Privoxy with their respective version numbers. The default setting is a full installation, which installs all three components. However given that the user cannot be expected to understand what all three components are, she has the potential to make an error at this step. But she may recover from the error from simply rerunning the installer if she realizes she has not installed a required component. For this reason, it does not violate G4, although it does violate G1.

After specifying a directory for installation (a default is given: the Program Files directory in the root drive), the components will install. The final dialogue announces that installation is successful and informs the user, "Please see http://tor.eff.org/docs/tor-doc-win32.html to learn how to configure your applications to use Tor."

5. TOR, VIDALIA, AND PRIVOXY

The task of configuring Firefox to use Tor and Privoxy is simply impossible without using the aforementioned documentation page. While ideally, from an HCI perspective, it would be possible for users to execute core tasks from simply exploring the application and its interface, configuration is not confined to a single application. It requires steps to be performed within Firefox itself, as well as within Tor and Privoxy, and thus will require inter-application documentation. It cannot be accomplished merely with intra-application cues.

The configuration document is concise and broken into steps, and so it is reasonable that the user will use the documentation. The first step is downloading and installing the application, which is already completed. It does however inform the user of how Vidalia and Privoxy are different from Tor: Vidalia is a graphical user interface for Tor and Privoxy is a filtering web proxy. This information is essential for the user to have a successful installation. The fact that this information is divulged after the installation process violates G1. It should be included during the installation process, as mentioned previously.

Step two is concerned with configuring Tor. Given that Tor can be used to anonymise any web application, not just a browser like Firefox, the configuration page links to a separate page ("How to Torify") devoted to a list of applications and step-by-step instructions for each. However the configuration page does offer some high-level advice. First

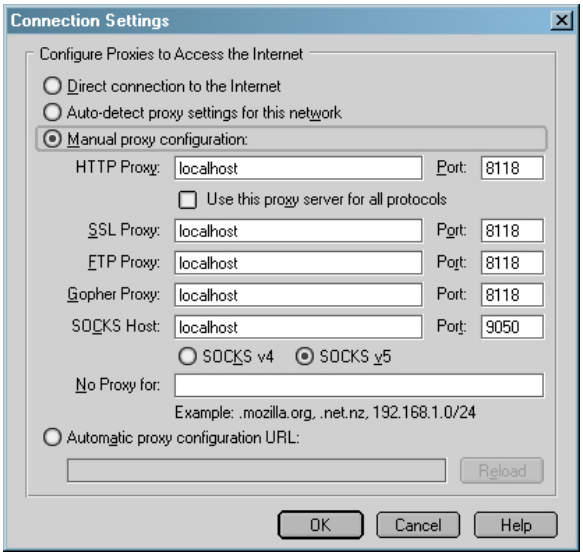

Figure 1: Connection settings in Firefox.

it states, "using Privoxy is necessary because browsers leak your DNS requests when they use a SOCKS proxy directly, which is bad for your anonymity." This statement unfortunately uses unfamiliar language (G6), however its intention is to prevent the dangerous error of DNS leaks (G5). The novice user does not need to understand what DNS is or what ports and sockets are to prevent this dangerous error—so long as they understand that the steps they need to take are preventing a dangerous error (G2).

The configuration page also offers this advice: "to Torify ... applications that support HTTP proxies, just point them at Privoxy (that is, localhost port 8118)." For the expert user, this information is concise and succinct and may suffice for them to configure their application without referencing the 'How to Torify' page. Privoxy exists on local port 8118 and Tor exists on local port 9050. However Tor only accepts SOCKS connections and thus http (port 80), https (port 443), and DNS (port 53) traffic has to be filtered through Privoxy and into Tor.

With this information in hand, it is reasonable that the user will move to the 'How to Torify' page to learn how to configure Firefox. While the configuration page is concise and fairly non-technical, the 'How to Torify' page is dense, full of technical jargon, and includes information extraneous to the task of configuring the application for Tor. The Firefox entry includes two configuration methods: one that involves changing the connection options in Firefox, and a second that involves editing the actual configuration of Firefox (a daunting task) but this second method then alleviates the need for Privoxy. This information is technical, confusing, and the second method directly contradicts the configuration page that states Privoxy is necessary (with "necessary" in bold). Furthermore, the advanced method is presented first. The documentation violates G2 because its unclear how the user should proceed, and it violates G6 for all the technical jargon. This is the first point in CT-2 where we heavily suspect users will give up or start to make errors.

The intention appears to be that novice users will use the second method to configure Firefox. Assuming they can determine this is the step they need to take to complete the

task, the instructions are rather straightforward. It gives the series of menus the users need to visit: "Tools → Options → General → Connection Settings → Manual proxy configuration" and instructs the user to "set HTTP Proxy 127.0.0.1 (or localhost), port 8118 and tick the box [X] Use for all protocols. Or you may explicitly set the Proxy information for SSL, FTP, and Gopher to localhost/8118 and then set the SOCKS Host information to localhost/9050, making sure to specify SOCKS v5." This appears complicated but referring to the configuration window in Figure 1, it is simply a matter of filling in the boxes. The problem with these instructions is that they give two options ("use the same proxy" box or individually configure) with no indication as to which the user should use. This violates G2. The options are not equivalent—the former uses Privoxy as the SOCKS host, the latter uses Tor—but will result in the same behaviour. However the user has no indication of which is better or which one they should use. They may choose the first option simply because it is given first, or they may choose the second because there is a graphic on the page illustrating what the correct settings look like—essentially the same as Figure 1.

For this configuration page to meet G2 and G6, it should pick one configuration method, place it at the top of the section, and simply give instructions on how to perform it without the technical language or extraneous information. The rest of the configuration information can be left as-is, but could be placed under an advanced configuration subheading within the section. Alternatively, the proxy settings could be encoded into a proxy auto-configuration (PAC) file. This would still require the user to find the Connection Settings in Figure 1, but the user would only have to fill in one box ('Automatic proxy configuration URL') which would result in a simpler set of instructions.

Presuming the user can configure Firefox, she can proceed to step three on the configuration page which is "making sure it's working." The first instruction is to ensure Privoxy and Vidalia are running. The correct configuration has these two applications running, however recall that the Tor application was also installed and it has an executable. It is not intuitive that Privoxy and Vidalia need to be run, but not Tor (Vidalia calls the Tor application). There are three ways to get these two applications running. By default, they both run after the installation process. Also by default, they both run when the computer boots up. If either of these defaults is changed, they can be run from the executables in their respective directories or in the start menu. By having the defaults set this way, less demand is placed on the user and so this is a commendable approach. However by default, Tor is not enabled while Privoxy is. The fact that the user needs to enable the one application to start and not the other is inconsistent. There is an option to start Tor when Vidalia starts, and this option should be made the default.

At this point, the user has Tor running but not enabled (Figure 2), Privoxy running and enabled (Figure 3), and the Firefox connection settings configured. To complete CT-2, the user simply needs select "Start" from the Vidalia menu (Figure 2) and begin browsing. If the user does not start Vidalia, or forgets to, the browser will display the error message in Figure 4 when the user attempts to visit a website. Given that Privoxy is a standalone application that can be used independently of Tor, it will not give Tor related error messages. It is not clear from this error message how

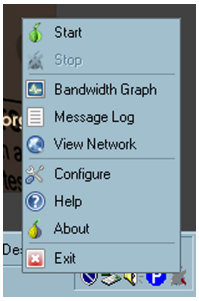

Figure 2: Vidalia menu.

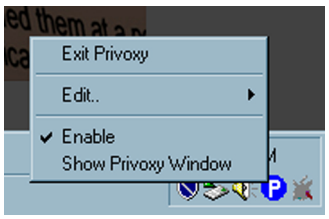

Figure 3: Privoxy menu.

the user can recover; furthermore, the links to the documentation and support are all links to webpages, and given that the error is being generated because the internet is not assessable, clicking on them will simply generate the same error. This is not a dangerous error (G5), but it is one that does not provide sufficient information for the user to recover from and thus violates G4.

By selecting Start from the Vidalia menu, the Tor taskbar icon changes from a red onion with an X through it, to a yellow onion signifying that Tor is starting, to a green onion signifying that Tor is running properly. This two-factor visual cue provides feedback (G3) that would allow most users to determine the system status at all times with a quick glance at the task bar (G8). The colours used are consistent with traffic light colours, and for users with colour vision deficiencies, the placement of the X over the icon is suitably noticeable. Overall, it is a comfortable interface to use for this core task (G7). When webpages are being accessed, the Privoxy icon spins like a radar screen. This provides feedback to the user that Privoxy is filtering traffic, and meets G3.

The third core task is to verify that traffic is being anonymised successfully. The configuration page offers a link to a "Tor detector" webpage, which checks the IP address of the packet it receives against a list of Tor servers. If it matches, it announces the configuration was successful and gives you the IP address and server name of the last mix proxy (called the exit node) in the network. This meets the feedback standard of G3 and does so without using unfamiliar language, meeting G6 as well. If Tor is not being used, the website tells the user they are not using Tor and refers them to the configuration website. This is a concise error message that provides enough information for the user to recover from the error, and thus meets G4.

While not a core task, a secondary but important task is determining the size of the anonymity set for the Tor servers the user is connected to, for reasons outlined in Section 2. This task is not possible. Vidalia does offer a View Network

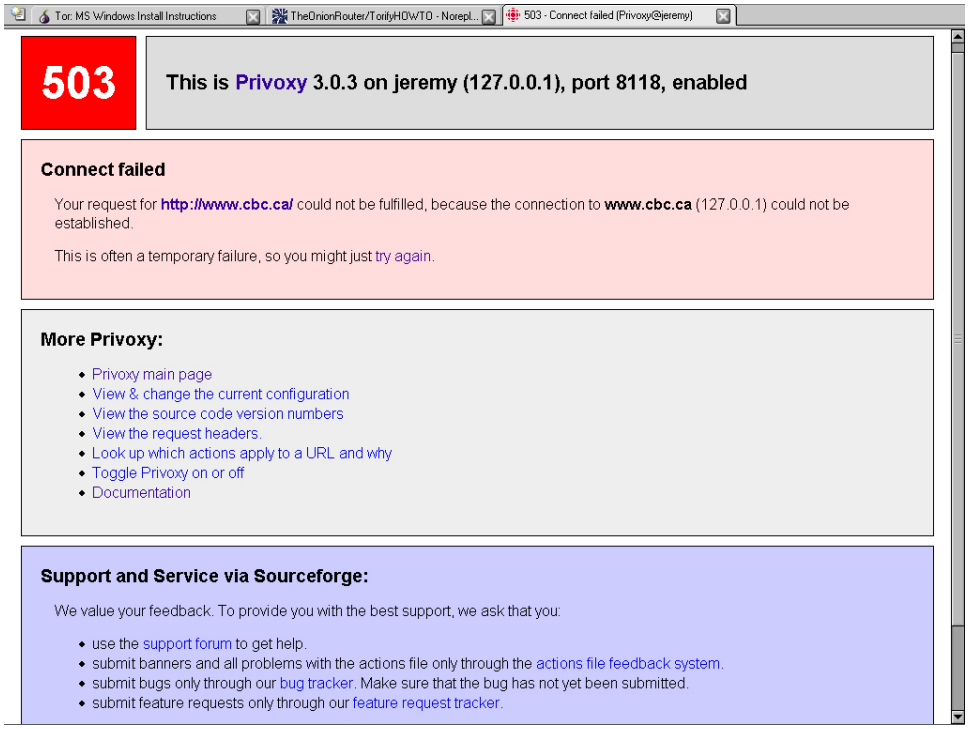

Figure 4: Privoxy error message.

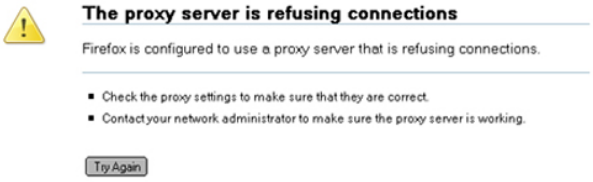

Figure 5: Firefox error message.

Figure 6: Torbutton icon in Firefox status bar.

option (see Figure 2) that presents a dialogue screen which lists the servers, displays the server locations on a world map, and gives information about the server: server name, location, IP address, platform, contact information, bandwidth, and uptime. To allow users to successfully complete this secondary task, information about the number of users connected to the servers should be displayed as well.

The final core task is disabling Tor. Unfortunately, the configuration page and the 'How to Torify' page are both silent on this issue. The correct method is to change the connection settings in Firefox (Figure 1) back to "Direct connection to the Internet." However this information is not accessible in the Tor documentation, breaking G2. Intuitively, the user may try a few alternatives to accomplish this task. The first would be to disable Tor by using the Stop option in the Vidalia menu (Figure 2). This of course, will generate the same error discussed above and shown in Figure 4. The second alternative would be to disable Privoxy by unchecking enable in the Privoxy menu (Figure 3). This generates the error shown in Figure 5 in the browser window, which is a generic Firefox error when Firefox cannot reach a proxy server. Neither provides enough information to allow the user to recover and thus violates G4. The browser is also completely useless in this stage, which will surely frustrate the user. The documentation should make the disabling process explicit, as users cannot be expected to know how to disable Tor once it is enabled.

6. TOR, VIDALIA, PRIVOXY, AND TORBUTTON

On the Tor configuration page,[1] under "Step two: Configure your applications to use Tor," the authors of the documentation offer an alternative to manually configuring Firefox for Tor (the configuration examined in the previous section). The page links to Torbutton [5], a Firefox extension, and instructs the user to visit the extension's page, install the extension, and restart Firefox. Extensions are small add-ons for Firefox that add functionality to the browser.

To perform the first core task, the user installs the Tor, Privoxy, and Vidalia bundle as in the previous section. However instead of configuring Firefox as described above, the user will click the link to Torbutton on the configuration page and be taken to the extension's listing in Mozilla's Firefox add-on directory. There is a predominant "Install

[1]http://tor.eff.org/docs/tor-doc-win32.html

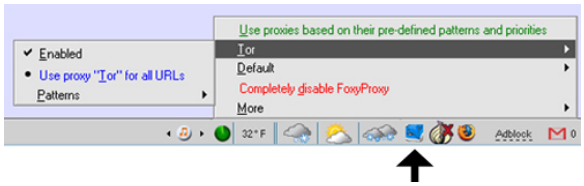

Figure 7: FoxyProxy icon in Firefox status bar.

now" button on the page and clicking it brings up a Firefox dialogue window warning the user to only install software from trusted sources. Clicking "install now" in this window brings up Firefox's list of installed extensions and informs the user to restart Firefox to use this extension. This process is straightforward, and conforms to G1, G2, and G6.

When Firefox is restarted, a cue is added to the status bar that states "Tor Disabled" in red letters, as shown in Figure 6. Many extensions install cues in this location, and so assuming the user is comfortable with Firefox, the user will likely notice the cue. Users unfamiliar with Firefox will need to read the documentation linked to from the configuration page. From a usability perspective, this has to do with Firefox's interface and is outside of the evaluation of Torbutton itself.

At this point, the first core task is complete. This allows the user to avoid the confusing configuration step criticized in the previous section. To complete CT-2, the user must be running Vidalia and Privoxy and have them both enabled. The analysis of this step in the previous section applies here. However instead of changing the configuration settings in Firefox, the user simply locates the Torbutton cue in the status bar and clicks it once. The cue will change to "Tor Enabled" in green lettering. The status can be seen at a glance, which is consistent with G8. CT-3 will proceed as in the previous section. To disable Tor and return to direct browsing, the user simply clicks the Torbutton cue a second time and it returns to "Tor Disabled." It is unclear whether users can be expected to know to do this (G2), however the change in icon is sufficient for meeting G3.

CT-2 is greatly simplified using Torbutton. The user does not have to change any settings within Firefox or wade through the confusing documentation. CT-4 is also simplified, however Torbutton still does not necessarily eliminate the non-critical errors discussed in the previous section—disabling Vidalia or Privoxy in trying to turn off anonymous browsing, instead of clicking the Torbutton cue within Firefox. However, while it is unclear in the previous section whether users will know enough to go back into the connection settings in Firefox and determine which setting to change to in order to disable Tor, we think it is reasonable to expect the user to at least stumble upon the correct solution with Torbutton even if its not the most intuitive step to completing CT-4.

7. TOR, VIDALIA, AND FOXYPROXY

FoxyProxy [2] is a Firefox extension similar to Torbutton but it provides more advanced functionality. It allows users to quickly enable and disable the use of any proxy server, not just Tor, and it allows the creation of custom access control lists to specify that certain domains always be accessed through Tor, with a different proxy, or directly. As we are testing the core tasks only, we will be testing FoxyProxy only as a method of configuring Firefox to work with Tor and we will not be evaluating its advanced functionality. The other feature that FoxyProxy prominently advertises is that it does not need Privoxy to work. Torbutton can also be configured to not use Privoxy—it is an unsung option in its preferences. However this option was not described on Tor's configuration page (nor explicitly on Torbutton's page) and the user would not likely be aware of it in performing the core tasks the walkthrough is evaluating. For this reason, we evaluated the configuration and use of Torbutton with Privoxy. However we will evaluate FoxyProxy without Privoxy.

CT-1 for FoxyProxy is similar to Torbutton. The user goes to the Firefox add-ons directory, installs the extension, and restarts the browser. However upon restart, the user enters a set-up dialogue to assist with the second core task. The first window asks if the user would like to configure FoxyProxy for use with Tor. If yes, the next window asks if the user is using Tor with or without Privoxy. Proceeding without Privoxy, the next window asks for Tor's local port number and states, "if you don't know, use the default," which is port 9050. The next window asks: "Would you like the DNS requests to go through the Tor network? If you don't understand this question, click yes." The next screen has the advanced filter settings, and the final window alerts the user to make sure Tor is running (G1).

This dialogue solves many of the usability problems encountered in the past sections. The user knows explicitly that Privoxy is not needed. The port and DNS settings both have a default setting and the dialogue clearly tells the user what to do if she does not understand the unfamiliar language (G6). The DNS setting allows the user to avoid a dangerous error (G5) even if their mental model is not sufficient to understand DNS (G2). The user is also reminded to ensure they are running Tor (G1), avoiding a non-critical error (G4), although no indication is given as to how to run Tor (G2)—in fact, the user likely wants to run Vidalia and not Tor.

The filtering options are complicated, however the user can avoid them and enable FoxyProxy to run all traffic through Tor. Enabling Tor for all URLs is more complicated than with Torbutton. The user has to click through a few menus (see Figure 7). However the user does not need to use Privoxy, which simplifies the second core task. Core task 3 is no different. Disabling Tor with Privoxy involves entering the menu and choosing "Completely disable FoxyProxy." The same pitfalls as described in the Torbutton section apply to this core task.

8. TORPARK

Torpark [6] is a standalone anonymous browser based on Firefox with built-in support for Tor. It is designed to run off a USB token so that a user can run it on a public computer that has an accessible USB port. Unlike the Tor bundle which had multiple versions, the Torpark webpage has only one version which is clearly marked (it also offers the source code, but this is in a distinct section). The installation is a self-extracting archive, so the user simply specifies the directory to which to extract the application folder. The installation process is far simpler: download the clearly marked installer, extract to a folder, and the user is done. The first core task meets all the relevant guidelines.

To run Torpark, the user opens the executable. Before opening, a warning message is displayed which states, "Torpark secures the anonymity of your connection, but not the data you send. DO NOT [sic] use identity compromising information such as your name, login, password, etc. unless you see a closed padlock icon at the bottom status bar of the browser. Torpark should not be run on untrusted computers, as they may have malware or keystroke logging software secretly installed." This warning helps the user understand the system model of Tor (G2), which they may have misconceptions about if their mental model expects, say, that the packets will be filtered as with Privoxy. The language is relatively clear and non-technical (G6), and should be understandable to an intermediate-level internet user.

A second dialogue window opens informing the user that Torpark is establishing a connection to the Tor network. After connecting, the browser itself opens. The browser comes with several preinstalled extensions. One is Torbutton, which is enabled by default and does not use Privoxy by default. This prevents the dangerous error of DNS leaks. A second extension is NoScript. This extension prevents websites from running Java applets and scripting by default. This is very important because it helps prevent the other dangerous error associated with Java applets (G5). A notification is displayed when a site attempts to run a client-side script or an applet that is blocked, with the ability for the user to add the site to a safe list (among other options). However blocking applets is a necessary but not sufficient measure if users are given the option to whitelist sites. The user may disable NoScript or whitelist everything indiscriminately if the danger is not adequately communicated. If the warning message was amended to include a warning about applets, Torpark may help prevent this dangerous error. There is, however, a flip-side to having NoScript enabled by default. It introduces a new interface that the user must be able to interact with—the usability of which has not been studied and is outside the scope of this paper. If we assume the worst case where the user simply ignores the prompts, many modern webpages will not function correctly without client-side scripting. This creates new usability problems and a negative incentive for using Torpark.

Another extension displays your external IP address (i.e., the IP of the exit node) in the status bar, which is a persistent indicator of the system status (G8). However this requires the user to know what their actual IP address is to determine that the displayed IP is different and Tor is working correctly. Thus it is not enough to complete core task 3. In fact, the user is not given any cues that would allow her to confirm that Tor is operating correctly, which violates G2 and G3. She could, of course, visit the Tor detector website mentioned above, which would provide the feedback she is seeking. Thus, if Torpark's default homepage was the Tor detector or if it offered a link to it, then CT-3 could be completed very easily.

Like in the previous sections, determining the anonymity set cannot be completed with Torpark. And given that Torpark uses Torbutton by default, CT-4 is the same as in the Torbutton evaluation with one notable difference: the Tor application is not running and so the user will not get confused and disable Tor in the application instead of the browser. Thus it prevents the non-critical errors uncovered is the previous sections.

9. COMPARISON AND SUMMARY OF RESULTS

We have performed a cognitive walkthrough for four different configurations of a Firefox-based browser with Tor. The first three configurations are largely interchangeable. The fourth, using the standalone browser Torpark, may not be as desirable to users who want to anonymise more than their browser. Tor can be used to anonymise instant messaging, file sharing, email clients, or nearly any application that uses the internet. If the user is going to anonymise more than their browser, they will need to install Tor, Privoxy, and Vidalia anyway—and therefore they would probably find it more convenient to simply configure Firefox to use these applications than to use an additional browser.

The summary of the results for the four core tasks are in Table 1. Of the first three configurations, manually configuring Firefox has the most usability problems. We found problems with the documentation and a complete lack of instructions on how to disable Tor. Furthermore, even if the user figures out how to enable and disable Tor, the options are a menu and several windows away. There is also no cue within the browser as to whether Tor is enabled or disabled at any given time, however the documentation links to a Tor detector webpage that verifies your traffic is being routed through the Tor network.

With Torbutton and FoxyProxy, it is conceptually more intuitive how to enable and disable Tor, it is a quicker process with a better interface, it does not require the user to read documentation, and there is a persistent indicator that shows the enabled/disabled status of Tor at all times. Thus Torbutton and FoxyProxy beat manual configuration on the second and fourth core tasks based on G1, G2, G3, G6, G7, and G8. Manual configuration has no advantage over Torbutton or FoxyProxy on any of the core tasks.

Between Torbutton and FoxyProxy, we found Torbutton to have the better interface. The user has to simply click it once to enable or disable Tor; the user has to go through a menu to achieve the same with FoxyProxy. Therefore Torbutton is better on the fourth core task based on G7. Both have persistent indicators in a visual cue on the status bar that changes colour according to the settings, and so they tie in terms of G8. However FoxyProxy has an advantage over Torbutton in the configuration dialogue by walking the user through the steps (instead of assuming the defaults) and imparting information to the user about the dangerous error of DNS leaks, in that Privoxy is not needed, and reminding the user to ensure Tor is running. Therefore FoxyProxy is better than Torbutton on the second core task based on G2, G4, and G5. Note however that Privoxy does offer features (i.e., packet scrubbing) that FoxyProxy does not, and so an unarticulated security/usability trade-off is at play in requiring the user to choose between using Privoxy or not.

Torpark has many advantages over using any preceding combination of Firefox, Tor, Privoxy, Torbutton, and FoxyProxy. First it does not require any documentation to install or configure. Assuming users do not like to read documentation, this is a huge advantage over using the components separately. The set-up dialogues are well written and make clear the limits of Tor in a hostile environment. None of the first three configurations do anything to thwart the dangerous error of allowing the execution of Java applets, while Torpark has Java disabled by default through the NoScript

Table 1: Partial Summary of Results

	Install Tor and Components	Configure Tor for Firefox	Verify Tor is Working	Disable Tor
Manual Configuration	Difficult	Very Difficult	Easy	Very Difficult
Torbutton	Difficult	Easy	Easy	Very Easy
FoxyProxy	Difficult	Very Easy	Easy	Easy
Torpark	Very Easy	Very Easy	Difficult	Very Easy

extension. While this is better in terms of G5, NoScript also disables JavaScript may also present new usability problems given that the many webpages which use client-side scripting will now be broken. Where Torpark fails, and the first three configurations succeed, is that it does not offer any way to verify that traffic is going through the Tor network aside from displaying the external IP address, which may be sufficient for advanced users but certainly not for novice users.

10. RELATED WORK

For an overview of the current research on anonymous communications, onion routing, and Tor, consult the anonymity bibliography [1]. Dingledine and Mathewson [13] discuss several usability issues with Tor and other mix networks. In particular, they note what our results confirm—the difficulty of the task of configuring Tor for unsophisticated users. They highlight the most important solutions to this problem as being improved documentation, solution-oriented warning messages, and having bundled Tor with the additional components it relies on.

The cognitive walkthrough methodology for evaluating usability was proposed by Wharton et al. [25] based on the observation that users tend to learn software by trying to use it and exploring its interface. They proposed a set of four criteria to help software designers conform their interfaces to the expectations of users learning their software through exploration.

A cognitive walkthrough is performed by Whitten and Tygar [26] in evaluating PGP against a set of four usability guidelines. The authors discover a number of security risks and usability issues and seek to confirm them through a 12 participant user study. Goode and Krenkelberg [15] perform a cognitive walkthrough of KaZaA based on usability guidelines they adapted for P2P filesharing applications. More recently, Chiasson et al. [11] expanded on Whitten and Tygar with two additional guidelines, one based on task completion and the other on system feedback. These guidelines are used in a user study of two password managers. Cranor [12] provides advice for software developers in the area of privacy based on lessons she learned in evaluation the usability of P3P and Privacy Bird.

11. CONCLUDING REMARKS

The two configurations that use the Firefox extensions are clearly more usable than manual configuration. However both Torbutton and FoxyProxy have their advantages and disadvantages. Our recommendation would be to add the set-up dialogue of FoxyProxy to the user interface of Torbutton. This synthesised extension would have the same one-click interface as Torbutton but would go through a small dialogue the first time it is run asking the user whether she wants to use Privoxy, giving her the option of eliminating DNS leaks (using the same style of language as FoxyProxy), and reminding her to ensure Vidalia is running. We would also recommend that it includes a warning similar to the one provided in Torpark.

These recommendations expose the difficulty of helping users prevent dangerous errors without introducing terminology and concepts that are too difficult for the novice user to understand. This consideration raises the question of who Tor's target user is. Our results show that novice users will have difficulty with at least one of the core tasks no matter which current deployment option they choose. Furthermore, the technical jargon and unfamiliar language of Tor's documentation is clearly not meant for the novice user. Is Tor simply too complicated for novice users to understand? Perhaps, but Tor might be made *more* accessible to novice users with two-tier documentation and properly structured defaults—simple instructions followed by instructions for advanced users, and configuration dialogues like FoxyProxy's, with statements that if a user does not understand a concept, to choose a default.

The extensions eliminate the need for the user to consult the 'How to Torify' page for configuring Firefox, but we would still recommend that two major changes be made to this page. The first is that one of the three different configuration options is chosen and placed at the top of the section on Firefox, and this becomes the dominant way of configuring Firefox for novice users. The second recommendation is that instructions are added to explain how to disable Tor.

None of the configurations allow the user to see the size of her anonymity set. We would recommend that Vidalia add information to its 'View Network' window about the number of users connected to each node in the network; although we recognize this metric may not be meaningful to the novice user. Java Anon Proxy (JAP) [8] has the visual cue of an anonymity meter that moves from low to medium to high, and something similar could be added for the novice user, as previously suggested in [13].

Finally we would suggest two modifications to Torpark. The first is that it makes some effort to link to the Tor detector website—either by making it the default homepage, using a custom home page that contains information about Tor and includes a link to the Tor detector, or at the very least, putting it in the browser bookmarks. The second recommendation would be that Torpark indicates to the user that running Java applets could jeopardize her anonymity. This may be preferable to simply disabling all Java applets and scripts by default, given that this disablement will introduce new usability problems with many websites no longer functioning correctly. Explicitly articulating the security concern and explaining clearly and concisely how to disable Java would make Torpark better at preventing dangerous errors, although any solution to this problem would ideally be user-tested. This information could be given in the warning

message or on a custom homepage like the one mentioned above.

With the exception of the anonymity meter, our recommendations are largely linguistic and would not require major revision to the programs themselves. Determining clear, meaningful explanations and instructions is difficult. A technique that may yield improved wording involves conducting a user study where a novice user is presented with an explanation and then asked to describe it to another novice user. Noting the use of terminology, metaphors and the level of detail across many subjects may yield commonalities allowing an improved explanation.

In conclusion, we have noted numerous usability problems that a novice user is likely to encounter in installing, configuring, and using Tor. It is our hope that the individual benefits of the examined software tools will be combined in order to help Tor achieve affinity among novice users. We also hope our guidelines will prove to be a useful compilation for future work in usable privacy.

Acknowledgments

We thank the anonymous referees for their comments which improved the paper. The first and third authors acknowledge support from SSHRC under the On the Identity Trail project. The second author acknowledges support from NSERC under a Discovery Grant, and as a Canada Research Chair in Network and Software Security.

12. REFERENCES

[1] Anonymity bibliography. In *Freehaven*. ONLINE: http://freehaven.net/anonbib/.

[2] Foxyproxy 2.2.1 http://foxyproxy.mozdev.org (accessed Nov 2006).

[3] Privoxy 3.0.3 http://www.privoxy.org (accessed Nov 2006).

[4] Tor 0.1.1.25 http://tor.eff.org (accessed Nov 2006).

[5] Torbutton 1.0.4 http://freehaven.net/s̃quires/torbutton/ (accessed Nov 2006).

[6] Torpark 1.5.0.7a http://www.torrify.com (accessed Nov 2006).

[7] Vidalia 0.0.7 http://vidalia-project.net (accessed Nov 2006).

[8] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.

[9] M. Bishop. Psychological acceptability revisited. In L. Cranor and S. Garfinkel, editors, *Security and Usability*, pages 1–12. O'Reilly, 2005.

[10] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. In *Communications of the ACM*, volume 4, February 1981.

[11] S. Chiasson, P. van Oorschot, and R. Biddle. A usability study and critique of two password managers. In *15th USENIX Security Symposium*, 2006.

[12] L. Cranor. Privacy policies and privacy preferences. In L. Cranor and S. Garfinkel, editors, *Security and Usability*, pages 447–472. O'Reilly, 2005.

[13] R. Dingledine and N. Mathewson. Anonymity loves company: usability and the network effect. In L. Cranor and S. Garfinkel, editors, *Security and Usability*, pages 547–560. O'Reilly, 2005.

[14] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[15] N. Good and A. Krekelberg. Usability and privacy: A study of KaZaA P2P file sharing. In L. Cranor and S. Garfinkel, editors, *Security and Usability*, pages 651–667. O'Reilly, 2005.

[16] C. Karat, C. Brodie, and J. Karat. Usability design and evaluation for privacy and security solutions. In L. Cranor and S. Garfinkel, editors, *Security and Usability*, pages 47–74. O'Reilly, 2005.

[17] I. Kerr. Anonymity. In progress encyclopedia article available from www.idtrail.org.

[18] K. Kirby. Bidding on the future: Evidence against normative discounting of delayed rewards. In *Quarterly Journal of Experimental Psychology*, pages 54–70. 126, 1997.

[19] D. Koblas and M. R. Koblas. Socks. In *UNIX Security III Symposium (1992 USENIX Security Symposium)*, pages 77–83, 1992.

[20] J. Muir and P. van Oorschot. Internet geolocation and evasion. In *Technical Report TR-06-05*. School of Computer Science, Carleton University, April 2006.

[21] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE CS, May 2005.

[22] J. Nielson. Heuristic evaluation. In J. Nielsen and R. L. Mack, editors, *Usability Inspection Methods*, pages 25–64. Wiley & Sons, 1994.

[23] V. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for internet hosts. In *Proceedings of SIGCOMM 2001*, pages 173–185, 2001.

[24] P. Wason. On the failure to eliminate hypotheses in a conceptual task. In *Quarterly Journal of Experimental Psychology*, pages 129–140. 12, 1960.

[25] C. Wharton, J. Rieman, C. Lewis, and P. Polson. The cognitive walkthrough method: A practitioner's guide. In J. Nielsen and R. L. Mack, editors, *Usability Inspection Methods*, pages 84–89. Wiley & Sons, 1994.

[26] A. Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *8th USENIX Security Symposium*, 1999.