THE PRIVACY PRACTICES OF WEB BROWSER EXTENSIONS

Whatever they are, the full terms of privacy these products offer are seldom made clear.

David M. Martin Jr., Richard M. Smith, Michael Brittain, Ivan Fetch, and Hailin Wu

DO TODAY'S COMMERCIAL free-of-charge software downloads show appropriate respect for the privacy of the computer user? We believe the answer to this question is "no."

Our research group at the University of Denver Privacy Center addressed the question by investigating Microsoft's Internet Explorer (IE) browser extensions. These downloadable pieces of software improve IE by giving it the ability to automatically fill out Web forms, perform price comparisons while shopping online, and liven up the interface with thematic images and sounds. A recent summary of browser extension products is available in [4].

Browser extensions are usually free of charge—in exchange for clickstream and profile information about the user and access to the user's display for advertising. The spectrum of information practices among products is broad. While some require full user address information and formal registration, others only request a zip code or age range at download time. Some products create a complete record of every Web site the user visits (for targeted marketing purposes), while other products avoid performing any actions that could leave an audit trail hinting at the user's Internet activities and personal interests.

Whatever they are, the full terms of exchange are seldom made clear to users. Although advertising components and requests for personal information are apparent, tracking a user's Internet activities is an inherently invisible act. Users enticed to download software described as "totally free" have no a priori reason to suspect the software will report on their Internet usage.

The difference between "free gift" and "free of charge" is significant. A gift is given with no expectation of compensation. A haircut at the local beauty school may be free of charge, but it is no gift; the school extracts training value from the exercise, and the customer suffers increased risks. Public domain software is usually a gift. Software that tracks users for business purposes certainly isn't. Vendors and users must understand that tracking records are potential subpoena and mining targets as long as they exist, whether in primary, backup, system log, or debug form.

We downloaded 16 IE browser extensions and watched them at work. A number were well behaved. But some extensions seemed to outright exploit our hospitality, watching and reporting our every move in the browser, some intercepting competitor-bound data and one reporting back to headquarters on pages that we "securely" downloaded using Secure Socket Layers. We tried to reconcile the observed behavior with the corresponding software's privacy policy and licensing agreement and noted any discrepancies. We then com-

TERRY MIURA

❀ VENDORS AND USERS must understand that tracking records are potential subpoena and mining targets as long as they exist, whether in primary, backup, system log, or debug form. ❀

pared the products against the well-established standards of Fair Information Practices (FIPs) [8].

The browser extensions we tested were designed for IE, version 5.0. IE is broadly programmable, and many browser extensions are available for it [4] and can be kept track of by the user (see Figure 1). By trapping certain IE events, it is not difficult for a browser extension to observe what the user does within IE and then transmit the audit trail to a remote site. However, we would not characterize the problems we saw as problems with IE; rather, they are misuses of it.

This article concentrates on the privacy disclosure and data flow practices of browser extension software, and we freely summarize conclusions from our study. The full report (www.privacyfoundation.org/advisories/advbrowserext.htm) also contains product privacy ratings and lab notes.

Fair Information Practices

FIPs are a widely recognized set of topics to address when an entity manipulates data about an individual. First described in a 1973 report of the U.S. Department of Health, Education, and Welfare [7], FIPs are defined more carefully for the Internet in a 1998 U.S. Federal Trade Commission report [8]. In our investigation, we compared stated and observed information practices against FIPs standards. The following are descriptions of the FIPs, along with our interpretation of how they should apply to downloaded browser extensions.

Notice/awareness. Notice says that individuals should be informed of an entity's information practices such as:

- who collects the information,
- how the information will be used, and
- what will happen to the information after the proposed relationship with the entity is terminated.

Notice merely describes the business process and software's operation. Making it more accurate should not require any reengineering. And despite profound divergence regarding the practicality of the other FIPs, most practitioners agree that notice is both the most important and the easiest FIP to implement.

The FTC report states that in the context of a Web page, notice is easily achieved by posting a privacy policy in a conspicuous place. However, an optional Web page disclosure is inadequate for browser extensions. Instead, the moment of downloading is a perfect time to explain the product's information practices.

Our guiding principle is that software should only communicate the data necessary to perform the desired function as described to the user. This means that software should not monitor and report on a user's computer use unless its monitoring function is disclosed. Furthermore, explanations of monitoring should never be hidden away in optional side documents (such as most privacy policies). In general, the less the monitoring function itself benefits the consumer, the more prominent the disclosure should be. Proper notice requires effective communication with all users, not just those curious enough to search for it.

In our investigations, we found rampant problems with notice. This is both good and bad news; good news because notice problems can be fixed in many cases by simply updating a privacy policy and making it more prominent, bad news because poor notice undermines the effectiveness of all FIPs.

Choice/consent. Once informed, individuals should be given a choice as to whether they agree to the stated practices. This can be accomplished in downloaded software by following the prominent privacy disclosure with a brief dialogue asking whether the installation should proceed. All secondary uses of the software—uses beyond those implicit in the request to download—should also be put to user consent. For example, if joining a mailing list is optional, then the user should be given the choice clearly.

Once downloaded software has been installed, users should be given the choice to stop using the software. Every product we reviewed made this possible, usually through the Windows control panel "add/remove programs" applet. A user should be able also to temporarily disable a monitoring product without fully uninstalling it; not all of the products we examined made this possible.

Access/participation. People move, change their address, and make mistakes. To the extent possible, an entity gathering information about people should provide the opportunity for people to view their records in order to contest, correct, or remove information obtained about them that is inaccurate.

Security/integrity. Data cannot be held privately without applying security measures. At minimum, sites maintaining personal information should have their practices reviewed by a security specialist, and their network transactions should be designed with privacy and data integrity in mind.

Problems Observed with Notice/Awareness

When analyzing shortcomings in the notice/awareness practices of Web browser extensions, we assigned each problem to one of eight problem categories. The list of categories follows, along with the approximate percentage of products in each category.

Lack of candor (100% of products tested). Software that "phones home" more data than is required for the functionality as offered and disclosed to the end user at download time is insufficiently forthcoming. Free-of-charge monitoring products should not masquerade as free gifts.

Loopholes (69%). Many sites instruct their users to "return often" to the privacy page to check for policy updates, even when they require an email address as part of the registration procedure, and even when their agents routinely "phone home" for tracking or auto-upgrade purposes. Surely these mechanisms could be used to communicate refinements to privacy statements as well. Other sites reserve the right to release personal information in order to protect their property and rights. This could be construed to cover the sale of customer lists due to bankruptcy, appease an aggressive or powerful business partner, or practically any situation in which the company has an economic interest.

Poor placement (85%). Products that put their disclosure in a poor location undermine the entire user agreement, since users can't really agree to unknown terms of use. Notice should be unavoidable, particularly in products with unexpected monitoring behavior.

Technical jargon and legalese (69%). "Clickstream," "URL," "cookie," and "IP address" do not have entries in standard dictionaries and should be defined if used in a privacy statement. Similarly, putting the privacy statement after 18 paragraphs of legal prose (four of them fully capitalized) makes the statement inscrutable to the average user.

Perspective mismatch (31%). A vendor who only addresses data exposure due to a product's ordinary operation within their ordinary business process only partially addresses users' concerns. In particular, user data that is transmitted to the vendor must be disclosed even if the vendor does not intentionally store it for long-term access.

Incorrectness (69%). Sometimes the privacy policy is simply incorrect, and the described behavior is not the behavior we observed. These slips appear to be mostly unintentional, but they do point to a disconnect between policymakers and developers.

Vagueness (46%). A privacy statement that raises more questions than it addresses is probably too vague. For example, one policy states that personal information is held in confidence "except as outlined below," whereupon 17 paragraphs follow, with one of them mentioning the use of the information in order to "make e-commerce faster and easier." Is the personal information held in confidence or not?

Domain confusion (20%). Web site privacy seals such as TRUSTe [5] and BBBOnLine [2] cover Web site practices only, but placing these seals near agent descriptions can mislead users into thinking they apply to the agents downloaded from the site as well.

Note that while the Platform for Privacy Preferences Project (P3P) [5] specification addresses many of these shortcomings in the context of Web site privacy practices, there is no current specification for communicating the privacy practices of downloaded software to P3P user agents.

Methodology: Spying on the Double Agents

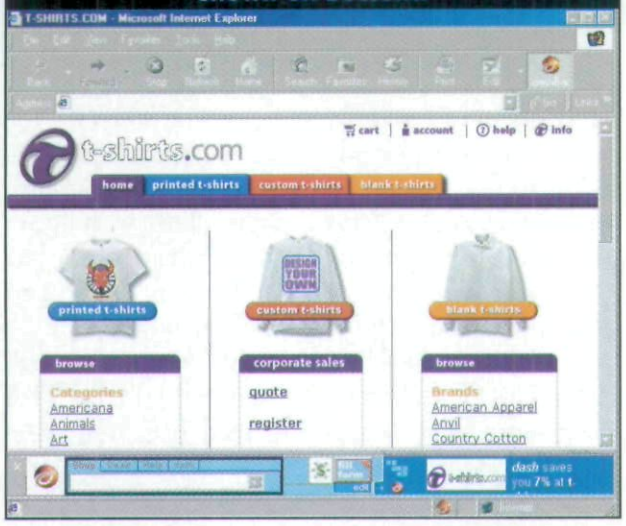
After evaluating adherence to the FIPs, we turned our attention to data flow, asking the question: "Is the product's data flow consistent with the service offered to the user?"

Uncovering a browser extension's behavior is conceptually a simple task demanding few extraordinary measures. The software's marketing pitch and privacy policy usually gave us a good idea of what to expect. We then selectively applied tools such as tcpdump, ethereal, windump, a custom logging proxy, a custom TCP stream reassembler, SST Inc.'s TracePlus32/Web Detective (which can produce cleartext versions of SSL transactions), regedit, the Visual Studio OLE viewer, the disk imaging and recovery tool Norton Ghost, and a file viewer for simple cache diving. We did not attempt to reverse-engineer compiled code; if we could not answer important questions on our own, we asked the product's manufacturer.

Inappropriate URL Monitoring

The most common problem in the data flow area (in roughly 50% of products reviewed) is the inappropriate monitoring of URLs. URLs can contain a wealth of sensitive parameters. Consider the URL www.google.com/search?q=AIDS+treatment&btnG=Google+Search. Clearly, this URL suggests not only that the user searches the Web using Google, but also that the user is interested in AIDS treatment.

Figure 1. IE with a browser extension bar shown on bottom.



The part of the URL following the '?' delimiter is called a query string and is usually handed over to a program running on the Web server for further processing. At various Web sites we have observed query strings containing usernames, email addresses, physical addresses, telephone numbers, flight numbers, and so forth. Users have no reason to think that a browser extension that simply tracks "the Web sites you visit" would retransmit this information to a remote server, but many extensions do.

The browser extensions themselves are not responsible for personal information appearing in URLs. Nonetheless, their designers should recognize the invasiveness of monitoring this class of URLs and ensure that personal data is stripped away before transmitting the data to remote servers.

We believe many companies actually have no interest in receiving sensitive data and simply did not anticipate this issue when designing their products. For example, companies whose products we reviewed do not appear to be in the spamming business, so their servers probably just ignore any email addresses embedded in URLs they receive. But by then, the URLs and email addresses have already been inappropriately exposed and possibly logged.

Some browser extensions intentionally examine query strings in the hopes they will have shopping or profiling value. For example, after noticing a user's Web search for portable MP3 players, for example, a browser extension might present an advertisement or coupon for a specific player. This can be done in a privacy-friendly way by reacting to known keywords on the monitored user's PC—without transmitting the URLs to a remote site and thereby exposing potentially sensitive data. A little defensive programming would make a big difference here.

Data Flow Case Study: Three Browser Extensions That Fill Out Forms

Three of the browser extensions we investigated have the ability to fill out shopping-oriented Web forms. The extensions are Dash, Gator, and Obongo. In this section, we investigate the data flow of these competing products in order to illustrate the impact that design decisions can have on privacy. Keep in mind that while we have the luxury of analyzing data flow in isolation, these firms had to balance many business and technical requirements in their designs. In addition, we neglect other facets of privacy (such as the FIPs) in this section. Therefore, we caution against inferring overly broad conclusions about the products or companies discussed here.

To the user, the products all behave in roughly the same way. After installation, the user configures the software with assorted personally identifiable information (PII) such as name, phone number, credit card number, and username/password pairs for gaining access to restricted Web sites. The software attaches itself to the Web browser and waits until the user reaches a page containing a Web form, at which point the software announces its ability to complete the form for the user. If the user clicks on the software's window appropriately, it will fill out the form fields from the stored PII. For users who do more than a little Web shopping, this is a very welcome piece of automation.

The first design decision is where to store the user's PII. Dash and Gator employ a strong privacy design by storing the PII encrypted on the user's PC only, while Obongo stores it on Obongo's own remote server.

Obongo protects the data from eavesdroppers by encrypting it during network transactions; the main threat is the PII may be mishandled at Obongo. This threat could be soundly mitigated by encrypting the PII on Obongo's server under a key known only to the user, but Obongo does not do this. A representative explained the company did not find it to be a viable option, since users often forget their passwords. Instead, Obongo relies on business practices and strong internal security procedures to safeguard the data. The advantage of Obongo's remote storage strategy is that a registered user can access the PII from any PC; the other extensions do not support this mobility.

But by maintaining access to the PII, Obongo is in a more precarious position than if it was technically unable to produce the data. For example, a plaintiff pursuing an Obongo user's Hotmail username and password could seek a subpoena asking Obongo to divulge the information. The same action would be

Figure 2. A sample Web form.

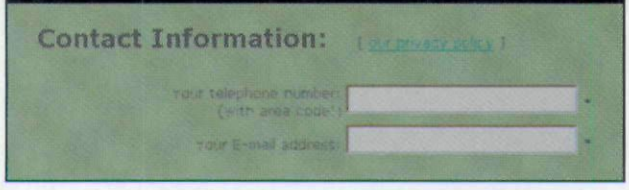


Figure 3. Obongo form analysis.

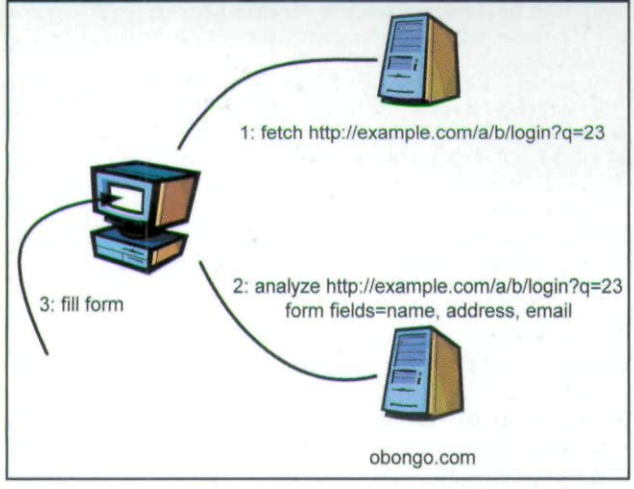
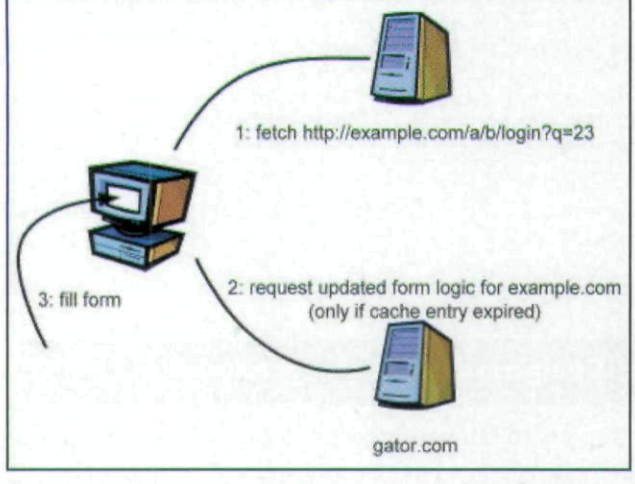


Figure 4. Gator form analysis.



absolutely futile against Dash and Gator users, since these companies simply do not have any means to produce the requested information. Note that while ToySmart.com had a privacy policy protecting its customer list, that did not stop it from later trying to sell the list (eventually, the U.S. Federal Trade Commission did [8]). In other words, strong technical protections can be far more enduring than guarantees based on legal obligations and business practices.

Another important design decision is where to put the Web form analysis program. In Figure 2, a user observing the positioning of "email address" next to a blank field will conclude that an email address belongs in the field. But form-filling software has to reach the same conclusion by looking only at HTML code. Programming this logic can be challenging because Web designers only ensure that a field's meaning is apparent to humans and not necessarily to programs.

Dash stores the form-decoding logic on the local PC; Gator keeps a local cache of remotely supplied logic; Obongo uses a hybrid local/remote design. By storing the logic on the user's PC, Dash, and Gator do not have to rely on a remote server for form field detection, analysis, and completion. (Dash actually does transmit the user's clickstream back to its own server, but not as part of its form-filling function.) Keeping the logic local automatically protects those products from the inappropriate URL monitoring problem previously discussed. In the case of Obongo, once the user asks it to fill out a form, Obongo sends the URL and small parts of the blank form to its remote server for analysis (see Figure 3). The response to this query will indicate how to complete the form fields from the PII.

Since their server must be consulted whenever an Obongo user requests form completion, the clickstream of completed forms could be stored inappropriately on the Obongo server. But a more dramatic problem is that by sending excerpts from the blank form back to its own server, Obongo rides on the user's credentials to deliver data the Obongo server may not have been able to obtain on its own. For example, consider a corporate personnel Web site protected by a firewall and authentication dialogue. Assuming these measures adequately protect access, the Web site designers carelessly encode employee names and social security numbers into URLs for their time-billing system. The Obongo server would never be able to reach these pages by itself, but a user requesting Obongo's form-filling help would cause the Obongo server to receive some very sensitive information.

A user should not have to fear that software designed to assist the user would transmit excerpts of securely downloaded pages to a remote site, regardless of that site's legal obligations to the user. Although the other form-filling extensions also scan and interpret securely downloaded pages, they do not rely on a remote server for analysis, so they never expose the data. By analogy, few people would want to use a client/server spelling checker with the server based in Redmond, but most of us routinely use the spelling checker monolithically contained in Microsoft Word without privacy worries.

Of course, there is a point to Obongo's strategy—recognizing that the form analysis problem is difficult, its use of a remote server allows it to adapt to new

Web form design styles as they emerge. But the online use of a remote server is not the only solution to the problem. For instance, the Gator server sends form-filling logic updates to its extensions when its cached logic becomes out of date, and it does so with a mechanism that does not require its extensions to report the user's URL stream (see Figure 4). When the Gator extension requests a logic update, it only gives the domain name of the Web site containing the form, which usually contains significantly less information than the full URL. The response contains logic for all of the forms on the site.

Conclusion

We believe the number of unreported but significant privacy problems in Internet software far exceeds the number of high-profile, privacy-related cases reported. We also believe that most of these unreported problems are best explained by oversight on the part of developers and entrepreneurs unfamiliar with common privacy pitfalls. The Internet is still a relatively new deployment environment, and little guidance is available to those who want to do the right thing. By showing some privacy consequences of early decisions, we hope to help minimize future lapses. Software designers and entrepreneurs may wish to keep our recommendations in mind (see the sidebar).

There is a legal sentiment in the U.S.—dating to Justice Harlan's 1967 Supreme Court opinion on wiretaps—stating roughly that privacy protections not codified in law can only exist when society expects them to exist. In other words, by blithely looking away when Web browser extensions surreptitiously send their observations of home life back to headquarters, we run the risk they will actually accrue them correctly. It is therefore important to keep society (and professional societies) aware of current privacy practice and developments.

It is time to bring privacy practices in Web software out of the turgid realm of the legal agreement page, elevating them instead to first-class criteria that discerning consumers will count along with speed, memory consumption, and ease of use in the search for the perfect tool for the job. C

References

1. ACM Code of Ethics; www.acm.org/constitution/code.html.
2. BBBOnLine privacy seal program; www.bbbonline.org.
3. FTC action against ToySmart.com, July 21,2000; www.ftc.gov/os/index.htm.
4. Mendelson, E. 30 ways to browse better. *PC Magazine 19*, 18 (Oct. 2000), 180–203.
5. Platform for Privacy Preferences Project (P3P); www.w3c.org/P3P.
6. TRUSTe privacy seal program; www.truste.org.
7. U.S. Department of Health, Education, and Welfare. Records, computers and the rights of citizens, 1973.
8. U.S. Federal Trade Commission. Privacy online: A report to Congress. June 1998; www.ftc.gov/reports/privacy3/index.htm.

David Martin (dm@cs.du.edu) is an assistant professor in the Department of Mathematics and Computer Science at the University of Denver.
Richard M. Smith (rms@privacyfoundation.org) is the chief technology officer of the Privacy Foundation.
Michael Brittain (mbrittai@cs.du.edu) is an M.A. student in Digital Media Studies at the University of Denver.
Ivan Fetch (ifetch@cs.du.edu) is a B.S. student in Computer Science at the University of Denver.
Hailin Wu (hwu@cs.du.edu) is a Ph.D. student in Computer Science at the University of Denver.

This research is a project of the University of Denver Privacy Center. Collaborating with the Privacy Foundation, the Pravacy Center addresses electronic pravy topics from technical, legal, business, and socian persectives.

Recommendations to Software Designers and Entrepreneurs

1. Tell the truth. If advertisers or others are paying for access to your users' screens or clickstreams, don't tell the users that your software is "free," and don't bury the details of the economic exchange in a legal document. Refer to the ACM Code of Ethics [1].
2. Ask lawyers to review your practices and statements in consultation with engineers familiar with the product's data flow.
3. Design with privacy in mind; reduce data transmissions to a minimum.
4. Don't force users to opt-out of questionable practices. Instead, invite them to opt-in by explaining how it would help them (or your company). Luring customers into an unwanted situation is not only unethical, it's also bad for business.
5. If you monitor URLs, choose those that specifically interest you and ignore the rest. In particular, remove text following the query string delimiter '?' and ignore 'https:', 'file:', and 'mailto:' URLs.
6. If you monitor URLs, design and maintain a mechanism to allow users to access the data collected about them.