Hal Berghel

Caustic Cookies

They can be beneficial aids in e-commerce, but cookies are the corrosive ingredient when it comes to invading personal privacy.

The modern Webites' medium is defined, enabled, and constrained by a pair of "killer" protocols—HTML and HTTP. The former provides structure for the object-level content, metalevel indexing and description, and, regrettably for purists, control over the presentation and format. The latter is a platform-independent, client/server protocol defined for any packet-switched digital network that supports the lower-level TCP/IP protocol suite.

HTTP is an application layer protocol that sits directly atop Transmission Control Protocol (TCP), which in turn sits atop Internet Protocol (IP). HTTP is similar to File Transmission Protocol (FTP), Telnet, Simple Mail Transfer Protocol (SMTP), and other sundry application protocols that handle the business part of the Internet's work. HTTP is rather unique, however, in that it is "stateless." The logical flow of an HTTP transaction sequence is:

1. Connect a client to a server (in the case of a Web browser, this typically amounts to a mouse click);
2. Make a request of the server (get data, execute a program, write data, and so forth);
3. Fulfill the request for the client; and
4. Close the connection.

The statelessness results from step (4). Once the transaction cycle is complete, the connection between client and server is disconnected. Full stop. Dead. History.

This is like having to go to the checkout stand every time you put an item in your shopping cart. In fact, that's exactly the way the burgeoning e-commerce community looked at HTTP. There had to be a better way.

Enter Netscape. What the HTTP world needs is transaction persistence, they reasoned in the mid-1990s. If one could just retain information in a stateless setting like one retains groceries in a shopping cart, the world would be a better place. Collecting all of this information on servers would quickly bog down the servers. So the course was clear: the persistence would take the form of depositing what I affectionately refer to as "Web guano" on the client's hard disk.

Armed with a mechanism by means of which the browser software could deposit these little digital leftovers on the end user's hard disk, one could build a veritable "shopping cart" of information. One can imagine a collective "Awesome concept, dude!" as the software developers stood witness as this idea passed through the browser birth canal. How this persistent, state-object became known as a "cookie," however, remains a mystery to me (please fill me in if you know).

JASON SCHNEIDER

A Basic Cookie Mix

In any case, the technical description of a cookie is a piece of "transaction state" (connection) information left on the client before the HTTP transaction cycle is concluded. Included in this state information is a specification of a range of URLs for which that information is relevant. Subsequent HTTP requests of URLs that fall within this range will prompt the browser to transmit this state information from the client to whatever server hosts that URL. This state information can take on any number of forms—items in electronic shopping carts, persistent identifiers (user IDs and passwords, to name two), and user preferences. Whenever you see a screen prompt that reads "in order to personalize this experience…," you're about to get a cookie whether you want it or not.

Baking digital cookies is pretty simple. Here's how it works. The first stage is the cookie "ingestion." The cookie is introduced to your computer by some server-side program that includes a "Set-Cookie" header as part of a HTTP response. The cookie mix consists of two ingredients: field NAMES and field <data>. This chunk of the HTTP header might look like this:

Set-Cookie:
USER_NAME=john jones;
DATA_ELEMENT_1=pocket watch
DATA_ELEMENT_2="encoded stuff"
DATA_ELEMENT_k=....
EXPIRES=Thursday, 25-Jan-01, 12:59:59 GMT;
PATH=/gotcha
DOMAIN=whazzamatta_u.edu

NAME=value is the only essential Set-Cookie attribute. And that information would all be written on your hard disk in a pre-defined directory set by your browser (the default location for Windows is C:\Windows\Cookies). Netscape's recommended cookie parameters were 300 per client with the least-recently used deleted to make room for more, 4KB per cookie, and 20 cookies per completely specified domain.

The second stage of the basic approach is cookie "withdrawal." This happens whenever a URL is accessed by a browser that requests information from "whazzamatta_u.edu," or any other URLs with any domain tail and path matching that of the Set-Cookie header fragment. So, the URL www.dept-of-shameless-methodologies.college-of-hard-knocks.whazzamatta_u.edu/gotcha is a match with the parameters of our cookie. Thus, the cookie contents will be sent to the server irrespective of content.

Therein lies the rub. Take a look at DATA_ELEMENT's 2 through k. The contents of these fields are anyone's guess. Collecting user-profiling information, IP numbers, shopping cart contents, user IDs, user-selected preferences, serial numbers, frequencies of contact with companies, demographics, purchasing histories, credit-worthiness, and customizing interfaces, are all potentially beneficial uses of cookies. But this same information may also be abused. And the collection of such personal information as social security numbers and other personal identifiers, credit card numbers, phone numbers, and addresses is, in my view, totally unacceptable.

> It's one thing to keep track of your customers in your store, but tagging their clothing with invisible ink as they walk out the door is another matter altogether.

Web Barbarians at the Electronic Gates

While cookies can be beneficial aids in online commerce, they are not without risk to personal privacy. These digital dog tags are virtually unlimited in the range of information they can store.

There is no evidence I'm aware of that responsible online merchants are storing data within their cookies that intentionally violate the privacy interests of their customers. Also, it should be remem-

bered that when such harmful information gets stored in cookies, it is most likely the result of an end user unwisely volunteering this information in the first place. That said, real issues remain.

As our digital villages evolve, we are slowly transforming our private sanctuaries into electronic auditoriums. This trend began with email (see "Digital Village," Apr. 1997). While email started the digital assault on personal privacy, the Web accelerated it. Modern "dynamic marketers" maintain huge databases of "E_dentities" built upon easily accessible and, in some cases, public information. Phone numbers, physical addresses, email addresses, IP addresses, and social security numbers are all fair game. In the state in which I reside, for example, one is legally required to provide one's social security number when registering a boat. This all goes into the public Fish and Wildlife database. Over a dozen states still require that social security numbers be listed on driver's licenses—an open invitation for identity theft (see "Digital Village," Feb. 2000). So, there's a limit to how much of our current assaults on personal privacy we can blame on our friends in e-commerce. They may be exacerbating the problem, but they didn't cause it.

However, this doesn't diminish the potential damage of cookies. As a convenience, we'll label those who would use these tasty digital morsels to penetrate our "digital zone of privacy," Web barbarians.

I don't know to what extent I'm willing to defend the term "barbarian" in this context, but I tend to view cookie mongers as a group as a primitive civilization unto themselves, not to mention being insensitive and uncultured, so the term isn't totally inappropriate. However, I'm willing to entertain alternatives like "lowbrow," "vulgarian" and "boor" if barbarian seems too strong. The general idea, however, is that for wont of a simple technical patch to overcome the statelessness of TCP/IP, we have created a (cookie) monster. And cookies by modern technology standards tend toward the innocuous by being only moderately invasive. At least—when they are well-behaved—they are both identifiable and manageable.

Far more potential for cookie malevolence derives from ill-behaved uses. Cookie worms provide one such example. In this case, the cookie is actually infected by Web bugs. Bear with me, these things really do exist.

A Web bug is a dimensionless (for all practical purposes) point in the presentation of a Web page. Many of us used Web bugs in the early days for such things as password protection. We would sensitize a single pixel in the page, and then click on that pixel, identified by the X, Y coordinates of the cursor at the bottom-left corner of the browser window to bring up a CGI form that would allow us to modify some component of our site from any browser rather than have to log in to our account. (Remember, image anchors in HTML need not be perceptible.) In recent years, Web bugs have become buried within multi-source documents, especially within banners, without the user's detection. But one of the strengths of modern browsers is they can routinely render multi-source documents as single pages, so there's not much that can be done to deter these nasty critters.

Why Web bugs? Because every contributor to a multi-source document can potentially share all of the cookies created for the primary URL. So, if XYZ company can achieve a subtle insertion into a larger multi-source document, it can access all of the data collected in the cookies by monitoring the clickstream of the user, and also access any information recorded in a cookie by the primary Web site. In the trade, these are called third-party cookies—cookies intercepted, modified, or written by a server other than the host of the primary URL. Any alteration of the primary URL's cookies is a cookie worm. There have been successive attempts to discourage this practice by modifying the browsers so only the primary URL can access and modify cookies, but third-party cookie monsters have thwarted this protection by appending the active URL's domain tail to their own in their Set-Cookie HTTP header, so at this point no one really knows how widespread the use of Web bugs are.

The Real Issue

You may have surmised by now that I believe cookies are an exceedingly bad idea—perhaps the only fly in the Netscape ointment. I don't deny for a moment the utility of persistent transaction identifiers in facilitating e-commerce. The problem with cookies is that they're invasive, pure and simple. It's one thing to keep track of your customers in your store, but tagging their clothing with invisible ink as they walk out the door is another matter altogether.

But the potential toxicity of cookies is the symptom, not the problem. The problem society has to deal with is whether the collection of personal information about an individual without the individual's informed consent should be tolerated. Whether the means involves cookies, scripts, Java applets, Active X executables, rogue servers that dispense "pseudo cookies," or snoopware that extracts information from client browser history databases, caches, temporary Internet files, or any other means for that matter, is irrelevant. The real issue is personal privacy.

Regrettably, the community of computer scientists of which I am a part has been overly relaxed in the deployment of technology without careful scrutiny of the long-term implications of our work. Cookies are but one example of the technological shortcomings of our digital village.

Perhaps the real measure of greatness in our new millennium will be the degree to which we use our technologies to protect our individuality. While society sorts this out, I recommend you revisit your browser's security preferences and give serious consideration to disabling all cookies or at the very least, requiring a prompt. C

Hal Berghel is professor and chair of Computer Science at the University of Nevada, Las Vegas, and a frequent contributor to the literature on cyberspace; www.acm.org/hlb.

Berghel's URL Pearls

For more information on cookies, visit Cookie Central (www.cookiecentral.com) or The Limit (www.thelimit.org/cookies.html).

Effective cookie management software is widely available for Microsoft Windows platforms, and exists more narrowly for other platforms as well. All work effectively with the most popular browsers.

Examples include:

- **COOKIE CRUSHER** (www.thelimitsoft.com)—a quality cookie manager that can be accompanied with an Internet housekeeping tool, Cyber Clean.
- **COOKIE CRUNCHER** (www.-rbaworld.com/Programs/CookieCruncher)—full-functioning freeware from Mark Richter.
- **COOKIE PAL** (www.kburra.com)—a quality cookie manager.
- **VAC PAC** (www.nsclean.com/vacpack.html)—a general purpose housekeeping tool that includes separate cookie managers for Netscape and Explorer, plus other file management tools, the most interesting of which is Evidence Eliminator.
- **WINDOW WASHER** (www.webroot.com)—a general purpose housekeeping tool that includes a cookie manager component along with a cache and history manager for both Netscape and Explorer.

Final Note: Cookies are not the least of our cyber-threats. Additional privacy threats arise from the Windows 98 Registration Wizard, ill-behaved HTTP servers, public-domain utilities (one example is "Comet Cursor"), the IDENTD identification daemon, viruses, trojan horses, Java scripts, "hit logging," spyware that monitors use off-line and then reports the activity when the user reconnects, Explorer's "phone home" feature, and even innocuous productivity apps like Word and Powerpoint. The latter embed network media in just the same way as browsers and are, in principle, just as vulnerable. C