Crossroads
The ACM Student Magazine

Cookies: Weaving the Web into a State

by ***Michael Nelte*** and ***Elton Saul***

The **Hypertext Transfer Protocol** (HTTP) that is the basis for the World Wide Web is stateless. There is no preservation of information across multiple HTTP requests and responses [**3**]. This design has advantages, such as simplicity and reduced overhead. However, it also has a significant disadvantage: any communication between a web browser and a web server can not occur within a known and explicit context.

Unfortunately, the ability to conduct meaningful electronic commerce (ecommerce) transactions requires stateful HTTP requests and responses. For example, vendors typically generate personalized web pages based on users' preferences and previous buying histories. Many ecommerce sites also enable users to place items of interest into "shopping carts" while they continue browsing. These items must be remembered until users end their sessions or finally pay for these products.

The Cookie Solution

Cookies were introduced by Netscape as a low-overhead mechanism for transferring and maintaining state information HTTP requests and responses [**9**]. Cookies permit a client to establish a logical session with a server even though a persistent network connection does not exist between them. A cookie is either transmitted in HTTP request/response headers or created via client-side scripts embedded in HTML. Once a cookie is received or created, it is stored on the client computer until it expires.

This article examines the use of cookie technology in Web-based ecommerce. We will first discuss how cookies are implemented and then analyze the implications this technology has on user privacy. The following technical details have been extracted from [**3**, **5**, **7**].

Set-Cookie Response Header

When a cookie is sent from a web server to a web browser, a Set-Cookie header is added to the HTTP response header. This header contains a single name-value pair followed by a semi-colon delimited list of one or more attribute-value pairs. The only required pair is the **Version** attribute that indicates which state management specification was used. In addition, there are five optional attributes or parameters that can populate this list. Note that cookies that are created via client-side

scripts use these same parameters.

The **Comment** attribute stores readable text that contains usage information for the cookie. Users can read this comment when determining whether they wish to initiate or continue a session with the given cookie. The **Domain** attribute specifies the domain for which the cookie is valid, and the **Expires** attribute (originally known as the Max-Age attribute) defines the lifetime of a cookie. If this attribute is not set explicitly, the cookie expires at the end of the current browsing session. The **Path** attribute specifies the subset of URLs to which the cookie applies. The **Secure** attribute (with no explicit value set) advises the web browser to use secure channels when returning the cookie to the originating server. Note, though, that the level of security used in subsequent HTTP requests is at the client's discretion.

When a cookie is created by a script or received in an HTTP response header, the web browser applies default values to the optional attributes that are missing. The Domain defaults to the request-host; Expires defaults to zero (i.e., delete cookie at the end of the current browser session); and Path defaults to the path of the request-URL.

An example of a Set-Cookie response header is as follows: *`Set-Cookie: customer=Wile_E_Coyote; version=1; domain=.acme.com; path=/guns`* This header entry would result in the creation of a cookie named customer with the value of "Wile_E_Coyote". The Domain is set to .acme.com, and the Path is /guns.

Cookie Request Header

If a client wishes to continue a session with the server, it returns a Cookie header in the HTTP request message. The criteria used to determine whether a cookie is returned to a server include the request-host, the Uniform Resource Identifier (URI) and the age of the cookie. For example, a cookie that has expired is deleted from the client's machine and is not transmitted back to the server.

A Cookie header includes a name-value pair followed by a semi-colon delimited list of one to three attribute-value pairs. As with the Set-Cookie header, the Version attribute is required. However, the only optional attributes permitted are Path and Domain. Each attribute is prefixed with a "$" (e.g., "$Path"). The values associated with each attribute are the same that were populated in the Set-Cookie header. A sample Cookie request header might appear as follows:

`Cookie: customer=Wile_E_Coyote; $version=1; $path=/guns; $domain=.acme.com`

From this header entry, the web server is alerted to the existence of the cookie whose attributes match those from the previous Set-Cookie example.

Accepting and Rejecting Cookies

To prevent possible security or privacy violations, a web browser rejects a cookie (contained in either an HTTP request or created by a client-side script) *if any of the following conditions hold*:

- The value for the domain attribute contains no embedded dots or does not start with a dot. Thus, an attempt to accept a cookie with a domain value of `acme.com` would be rejected because the domain does not start with a dot. An attempt to accept a cookie with a domain value of `.com` or `.com.` would also be rejected because there is no embedded dot.
- The value for the request-host does not domain-match the domain attribute of the cookie. We define two host names, A and B, as domain-matching if any of the following hold:
  - A and B map to identical IP addresses (either directly or via DNS lookup);
  - A has the form N.B, where N is a non-empty string (representing a host) and B is a FQDN string preceded by a dot. Thus `x.y.com` domain-matches `.y.com`, but not `y.com`.
- The value for the path attribute is not a prefix of the request-URI.
- The request-host is a FQDN and has the form HD, H is a string that contains one or more dots, and D is the value of the domain attribute. Thus, a cookie with a domain value of `.acme.com` from the request-host `gun.shop.acme.com` would be rejected because H, or `gun.shop`, contains a dot. However, a cookie with a domain value of `.acme.com` from the request-host `rockets.acme.com` would be accepted.

In Netscape Navigator 4.7, a user has three ways to control the acceptance or rejection cookies. The user can either choose to accept all cookies, reject all cookies, or accept only cookies that get sent back to the originating server. The latter option restricts the cookies that can be received, but does not restrict the cookies that are sent. Microsoft Internet Explorer allows a user to either enable or disable all cookies. In addition, he/she can specify that they wish to be prompted before a cookie is installed on their system.

Managing Cookies

If a web browser receives a cookie whose name, domain, and path match the same fields in a resident cookie, the old one is discarded and replaced by the new one. Note, though, that if the new cookie has an Expires attribute set to zero, it will also be discarded at the end of the current browser session [**5**].

In addition, Web browsers provide the following cookie management capabilities:

- Limit the number of cookies that can be stored on a client. For example, Netscape Navigator allows a maximum of 300 cookies [**9**].
- Discard older cookies to clear up space for newer ones using, for example, a least recently used (LRU) algorithm [**5**]. However, it is important for the user to maintain some control over cookie destruction. They may have infrequently used but valuable cookies that were time consuming to create.
- Limit the maximum size per cookie to a reasonable size. Netscape Navigator permits a maximum size of 4 KB per cookie [**9**].
- Limit the number of cookies a given host can store on a client. Netscape Navigator permits a maximum of twenty cookies per domain [**9**].

Objections to Cookies

Objections to use of cookies focus on concerns relating to user privacy. Web technologies such as cookies permit sites to record and collate observations on user browsing habits. This data is a valuable commodity with broad resale potential. As a result, personal information about users might be disseminated to a wider audience than originally intended.

Additional concerns relating to cookies are rooted in misunderstandings about how this technology works. When cookies first appeared on the web, there was considerable confusion about what information they recorded about users. A cookie cannot actively acquire any information from a user's computer. It cannot determine a user's identity, address, or income. The only way in which this type of information can end up in a cookie is if a user willingly provides it to a web site that in turn saves it in a cookie.

Nonetheless a cookie can contain an ID used to track a user's behavior [**12**]. Even so, a web site is limited in the data it can collect: it can only read cookies from its own domain. In theory, a site is unable to record user behavior on other sites.

In practice, information contained in cookies can be shared between sites. A well-publicized case involving DoubleClick revealed that the advertising agency receives cookies from users of their client sites [**10**].

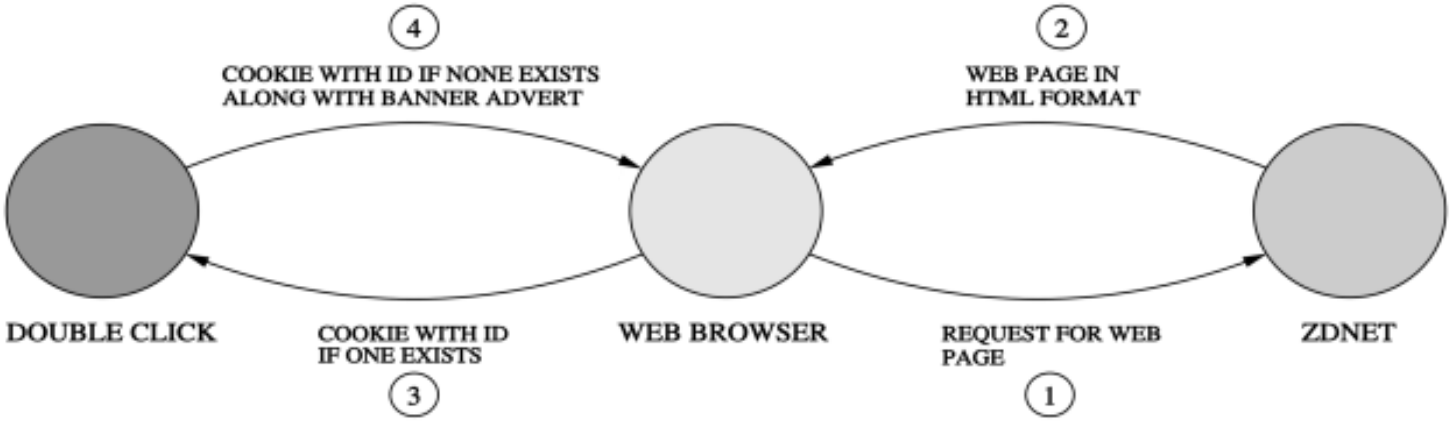

Figure 1: HTTP Transfers that Occur when Displaying Banner Advertisements.

Most commercial content providers do not store their banner advertisements locally. Instead, they subscribe to media services that select and generate advertisements on their behalf. When a user requests an HTML page from a content provider, it is downloaded along with all of its associated pictures, sounds, and plugins. An advertisement is typically embedded in an HTML page as an image that resides on a server maintained by the media service to which the content provider subscribes. In order to download this image, the browser must make a separate HTTP request to the server hosting that advertisement. In the header of this HTTP request, the content provider is identified in

the Referrer field. In addition, the header may contain any cookie(s) set by this content provider.

Traditionally, agencies have used cookies to track a users' behavior on a client site, and generate targeted advertisements. DoubleClick took this technique one step further by aggregating user activity across multiple client sites. So, a user who receives a DoubleClick advertisement while visiting a pornographic site might later find banner advertisements for pornographic sites displayed when visiting a site with no sexual content.

Any personal information provided by a user at a web site being serviced by DoubleClick may end up being shared with the advertising agency [**11**]. In addition, the agency's recent acquisition of Abacus Direct, a direct marketing service company, places a vast repository of user information at their disposal [**6**]. Using this data, DoubleClick can link a web-based ID to an actual identity in the physical world.

Another contentious practice involves using cookies in bulk mailings containing attached HTML files. Images in these HTML files can initiate requests which may lead to the setting and retrieval of cookies. It is therefore possible to link a user's e-mail address with an ID storied inside of a cookie on their machine [**12**].

Using cookies to track users has generated considerable debate and controversy [**1**, **2**]. Ironically, these problems could have been avoided if developers of web browsers had followed the guidelines for cookies presented in RFC 2019. Section 7.1 of this document states that users should be consulted *by default* before accepting a cookie:

```
"An origin server could create a Set-Cookie header to track the path of a user
through the server. Users may object to this behavior as an intrusive accumulation
of information, even if their identity is not evident. (Identity might become
evident if a user subsequently fills out a form that contains identifying
information.) This state management specification therefore requires that a user
agent give the user control over such a possible intrusion ..."
```

However, advertisers objected to this requirement because it prevents them from seamlessly tracking web surfers across multiple web sites. Web developers also raised concerns that it would be difficult to ensure consistent functionality on web sites if users were easily able to disable cookies. As a result, current browsers accept cookies by default without notifying the user.

Beyond Cookies

The recent controversy and publicity surrounding the misuse of cookies has made users wary of this technology. A number of applications that block or completely disable cookies have been created [**2**]. Lawsuits have been brought against online advertising companies such as DoubleClick for their practices involving cookies. Privacy advocacy groups are lobbying for the replacement of cookies with less invasive techniques.

In 1999, Netscape Communications Corporation, Firefly Network Inc., VeriSign, and over sixty other companies proposed the Open Profiling Standard (OPS) [**4**, **8**] . OPS permits a user to create a personal profile that stores registration information that web sites tend to require. A personal profile can contain a wide range of descriptive information about an individual, such as their name, address, zip code, phone numbers and e-mail addresses. Information such as age, marital status, interests, hobbies, user identification and passwords can also be stored.

OPS allow the user to control how much information is disclosed to a given Web site. Furthermore, a web site needs explicit permission from a user before sharing profile data with third parties. The user is also notified when their profile is requested from a particular site. Profile data can be encrypted and digitally signed for confidentiality and authentication purposes.

The Platform for Privacy Preferences Project (P3P) [**14**] is a protocol that provides web sites with a means of notifying users of their data-collection and data-use practices. Using P3P, information about these practices are stored in a machine-readable XML format that can be transmitted to browsers. When a user visits a P3P site, their browser will compare the site's profile with the user's OPS. The settings in the privacy preferences then determine what information about this user is disclosed to the site.

OPS and P3P allow a user to enter personal information once, obviating the need to enter the same information for multiple web sites. They also permit the user to set privacy preferences on a site-to-site basis. These technologies will not necessarily replace cookies. However, they may change the way in which cookies are managed by users.

Conclusion

In 1995, Netscape extended the HTTP specification to include cookies as means of preserving state information between a web server and client across multiple HTTP requests. This technology was a significant boost to the development of ecommerce web sites. By providing a web site with a mechanism for storing small amounts of data on a client, cookies make applications such as shopping carts trivial to build. However, cookies also permit sites to easily track user behavior, which raises concerns relating to user privacy.

At present, cookies are still the preferred mechanism for maintaining state information throughout and between browsing sessions. Rather than eliminating the use of cookies on the web, efforts are being made to improve the way in which they are implemented. OPS and P3P are two emerging technologies that improve the ease with which a user can control the release of personal information to web sites.

References

1

R.E. Bruner. `*Cookie' Proposal Could Hinder Online Advertising*. 31 March 1997. **http://adage.com/interactive/articles/19970331/article1.html**

2

L. Eichelberger. *The Cookie Controversy*. 8 April 1998. **http://www.cookiecentral.com/ccstory**

3

R. Fielding, U.C. Irving, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. ***RFC 2068: Hypertext Transfer Protocol***. January 1997

4

M. Kanellos. *Netscape, Partners Create Profiling Standard*. 27 May 1997. **http://www.techweb.com/se/directlink.cgi?WIR1997052706**

5

D. Kristol, L. Montulli. ***RFC 2109: HTTP State Management Mechanism***. February 1997

6

J. McCarthy. *Cookies, Ad Banners and Privacy*. 26 October 1999. **http://slashdot.org/yro/99/10/22/0249212.shtml**

7

K. Moore, N. Freed. *Use of HTTP State Management*. December 1999. **http://search.ietf.org/internet-drafts/draft-iesg-http-cookies-03.txt**

8

Netscape Communications Corporation. *Netscape, Firefly and VeriSign Propose Open Profiling Standard (OPS) to Enable Broad Personalization of Internet Services*. 27 May 1997. **http://www.netscape.com/newsref/pr/newsrelease411.html**

9

Netscape Communications Corporation. *Persistent Client State HTTP Cookies* . **http://www.netscape.com/newsref/std/cookie_spec.html**

10

W. Rodger. *Activists Charge DoubleClick Double Cross*. 21 February 2000. **http://www.usatoday.com/life/cyber/tech/cth211.htm**

11

G.R. Simpson. *Intuit Scrambles to Plug Quicken Leaks*. 2 March 2000. **http://www.zdnet.com/zdnn/stories/news/0,4586,2454429,00.html**

12

R.M. Smith. *The Cookie Leak Security Hole in E-Mail Messages*. 30 November 1999. **http://www.tiac.net/users/smiths/privacy/cookleak.htm**

13

D. Whalen. *The Unofficial Cookie FAQ (Version 2.53)*. 5 October 1999. **http://www.cookiecentral.com/faq**

14

W3C. *P3P Guiding Principles*. 21 July 1998. **http://www.w3c.org/TR/1998/NOTE-P3P10-principles.html**

Biographies

Michael Nelte is a postgraduate research student in the Data Network Architectures Laboratory at the University of Cape Town. He obtained his BSc (Honours) degree in 1998. He is working towards a Masters Degree in Computer Science with a concentration in biometrics and smartcards. His present research focuses on fingerprint comparisons on smartcards.

Elton Saul is a postgraduate research student in the Data Network Architectures Laboratory of the University of Cape Town. He obtained his BSc (Honors) degree in 1998. He is currently working towards his Masters degree in Computer Science with a concentration on security protocol design. His research focuses on making the automated GNY analysis of cryptographic protocols more accessible to engineers.