

Smart Cookies: The Unstealable Authentication Cookie

Andrew Miklas, Shvetank Jain

October 1, 2006

The web has become a new, highly interactive medium. Many modern websites provide their users with the opportunity to alter their content in some way. This trend is especially evident in Wikis, where users are encouraged to collaboratively edit pages if they have something to add. Blogs and online forums are another example of sites which encourage their readers to contribute content. Finally, web-based e-mail providers such as GMail and Yahoo can be considered collaborative web-pages in some respects, as ordinary end-users are able to directly influence the content that is rendered on the mailbox owner's browser.

The collaborative web has made a large number new applications possible. However, like many other advances in Internet development, it has also created a number of security issues, including "cross-site cooking". This attack, itself a special case of "cross-site scripting", allows a malicious user to retrieve cookies from another user's browser set by a domain the he does not control. In order to capture the cookies, the attacker simply has to ensure that his script is rendered by the client's browser when he visits the domain for which the cookies are desired. The attack does not involve exploiting a bug in the user's browser; in fact, the attack can be as simple as the one shown in Figure 1. Since websites will frequently accept a cookie containing a session ID as proof that the user is authenticated, this attack allows a malicious user to gain unauthorized access to a service.

An obvious fix would be to require interactive websites to carefully escape all user-provided content so that client browsers do not execute untrusted scripts. However, even though many popular web program-

```
<a href="#" onclick="window.location=
'http://example.com/stole.cgi?
text='+escape(document.cookie);
return false;">Click here!</a>
```

Figure 1: Sample cross-site cooking attack code [3]

ming environments provide the necessary functions to escape scripts, there have been a number of high-profile cases of cookie theft [1, 4]. Another approach explores the effect that each cookie has on the rendering of the page, and concludes that if there are no effects, the cookie is unnecessary [2]. While the authors don't describe their solution as a method of preventing cookie theft, it is possible that some of their techniques could be adapted in order to detect and prevent cookie theft. However, this system is relatively heavy-weight, and does not guarantee that cookies cannot be stolen.

We propose a new type of cookie to be used for authentication. These "smart cookies" use asymmetric encryption as the basis for authentication, rather than the shared-secret approach employed by traditional cookies. The enrolment phase, corresponding to the "set-cookie" HTTP header with ordinary cookies, proceeds as follows:

1. **Server:** Request Smart Cookie
2. **Client:** Generate public/private key pair, and send the server the public half.
3. **Server:** Store the public half as the session ID

The verification phase, corresponding to the "cookie" HTTP header, proceeds as follows:

1. **Server:** Generate a nonce
2. **Client:** Encrypt the nonce with the private key, and send the result back to the server.
3. **Server:** Decrypt the signed nonce with the public key on file, and verify that it matches the transmitted nonce.

Smart cookies have the same domain-based access restrictions and expiration semantics as ordinary cookies. However, note that the private key component of a smart cookie never leaves the client's browser. Therefore, while the browser may wish to provide indirect access to the private key via a "sign" Javascript method, it should not provide scripts the ability to read the private key directly. In fact, no component of the browser that can be influenced by downloaded code should be allowed to access the private key. Since the key cannot be directly read by scripts, it cannot be stolen via cross-site scripting attacks. A useful side-benefit of smart cookies is that the authentication tokens cannot be intercepted in transit for later use, as is possible with traditional cookies over unencrypted HTTP sessions.

We are still working out the details of the smart cookie protocol. In particular, we are striving to ensure that clients and servers can fallback gracefully to traditional cookies if either side does not support smart cookies. We are also exploring several options that will not necessitate the server remembering a nonce value across HTTP operations, although we don't believe that holding this state across multiple HTTP operations will pose a problem if HTTP/1.1 persistent connections are used.

References

- [1] EWeek. Google plugs cookie theft data leak, 2005. <http://www.eweek.com/article2/0,1895,1751689,00.asp>.
- [2] C. K. Umesh Shankar. Doppelganger: Better browser privacy without the bother. In *Proceedings of ACM Computer and Communications Security*, 2006.
- [3] Wikipedia. HTTP cookie, 2006. http://en.wikipedia.org/wiki/HTTP_cookie.
- [4] S. B. Youssef. Hotmail/MSN cookie theft advisory, 2006. <http://lists.grok.org.uk/pipermail/full-disclosure/2006-February/042518.html>.