

Smart Cookies: The Restricted Access Cookie

Andrew Miklas, Shvetank Jain

October 18, 2006

The cross-site scripting attack is widely prevalent and a number of real world attacks have been reported. One way to curb scripting attack 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 programming environments provide the necessary functions to escape scripts, there have been a number of high-profile cases of cookie theft [1, 3]. 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.

Solution

We had proposed 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. However, these encryption cookies modify the typical HTTP data flow and thus, backwards compatibility is a serious issue.

However, we observed that the information stored in the cookies are generally useful to either the server, or the client, but not both. For eg., some customization preferences maybe stored inside the cookie which help the user see the page in his favourite color layout. On the other hand, mostly the cookie stores information that is needed by the server like ID numbers, authentication data, etc. This information in

the cookie is needed by the server but is not used by the client-side Javascript.

Using this we found a more intuitive approach to tackle the problem. The server can mark the cookies that it wants to be secure by prefixing the label `__NO__CS__ACCESS__` to them. The browser would make sure that it doesn't allow Javascript to have access to these labelled cookies. The rationale behind such an approach is that the authentication information is not required by the client-side Javascript. It is just required for connection initiation. So, it should not be needed by the client side Javascript.

The advantages of such an approach are two-fold:

1. **Safer** - Such a labelling scheme provides servers with a method to be able to mark sensitive cookies so that they have restricted access. This gives the server more control in the usage of the information that has been stored by it on the client.
2. **Backwards Compatible** - The browsers that do not understand such a labelling scheme can just treat it as a normal cookie. In the case of "new server/old client" the client may not be able to understand the new naming convention and thus it treats it as a normal cookie. In the case of an "old server/new client" the server would not tag the cookies according to a naming convention and thus the client would not provide the additional benefits of smart cookies. However, the server might observe unexpected behaviour in case it happens to use the prefix we use coincidentally. However, the choice of the prefix makes this as a very rare event. In the case of the "old server/old client" they would behave exactly as they interact now.

```

<?
setcookie('conf_cookie', 'fav_color=Blue');
setcookie('__NO_CS_SCRIPT__my_id',
'SUPER_SECRET');
?>
<HTML>
Cookie (probably) set.
</HTML>

```

Figure 1: Setting the cookie

```

<BODY>

<H1>Server-Side Cookies</H1>
conf_cookie:
<? print($_COOKIE["conf_cookie"]) ?>
<BR>
my_id:
<?print($_COOKIE["__NO_CS_SCRIPT__my_id"])?>

<H1>Client-Side Cookies</H1>
<script type="text/javascript">
document.write(document.cookie)
</script>

</BODY>
</HTML>

```

Figure 2: Accessing the cookie

Implementation

We tried to integrate such functionality into the browser and were successful to do so with a Firefox patch. We compiled Firefox with the patch and the modified-Firefox makes sure that the Javascript cannot access the sensitive cookies.

In the above code, once the cookie has been set, the browser is not able to access the sensitive cookie which is labelled. Thus, the client side Javascript is able to access only the non-sensitive cookie. Since the cookie cannot be directly read by scripts, it cannot be stolen via cross-site scripting attacks. We have a patched Firefox that we will be able to demo in the

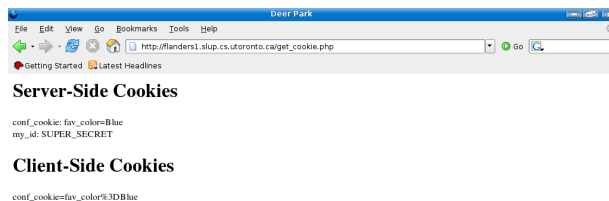


Figure 3: Rendering of code from Figure 2

presentation.

Future Work

For future work, one possibility is that we could extend this to be able to automatically separate the sensitive and non-sensitive cookies based on their content. However, we tried analyzing the nature of cookies and a lot of them are encrypted which makes it difficult to be able to understand the content. Even when not encrypted they often follow an incomprehensible format which makes it difficult to make any judgements about the sensitivity of information.

Another possibility could be to provide a more fine-grained mechanism for servers to define access policies for the use of cookies on the client side. Such a policy can be captured in the naming strategy for the cookie.

However, prefixing the name is just one possibility for tagging cookies. The P3P fields in the set-cookie header can also be used. Inside the cookie itself, we can have the policy written which could be enforced by the browser or be added as a separate field to the cookie.

Conclusion

We have proposed a solution for the cross-site scripting attacks which is backwards compatible with the current client-server model. Thus, the servers can label the cookies as 'sensitive' which then cannot be accessed by Javascript. We are analyzing the feasibility of the solution and probable directions of future work. We have a patched Firefox solution that we will be able to demo in the presentation.

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] S. B. Youssef. Hotmail/MSN cookie theft advisory, 2006. <http://lists.grok.org.uk/pipermail/full-disclosure/2006-February/042518.html>.