Selected issues with web identity mechanisms and a possible way forward

Background

It has been demonstrated, repeatedly, over the past years that the approaches to safeguard the identities of websites, users and user sessions currently widely in use on the web are flawed. Improvement is needed, but, to identify it, we need to scope out the most important issues. We will describe the areas we have seen as issues that have developed in the context of identity management, the opportunities we see for it to develop – both in the current webapp contexts and in future – and present a straw man solution to illustrate how it may evolve.

Current Practices on Identity Management

Identity management is the process of creating an identity, using an identity to establish an authenticated and authorized session, terminating the session, and terminating the identity. While the creation and termination of the identity is not standardised, as we note later, it is vital that we take into account how these procedures are applied for any potential solution.

Currently, most of the web's identity management pertains to users. Later, we will explore identity management for others as well, but, for now, we will discuss it with regards to users foremost. There are two rough categories of user handling that we can generalise.

The first one is where the contents of the site are private, and one must be logged in to gain any form of access. Typically, this can be a corporate intranet website. Commonly, these sites present anonymous users with a login screen, but also an explanation of what the site is, with links to get help/information, copyright/legal, and, depending on the site, how to acquire/create an account.

The second type of site is where the contents are visible to both anonymous and identified users, although the presentation may be customised for the identified users, giving them more information, or additional access. A typical example is an online storefront, which allows anonymous users to shop, maintain a shopping cart, and even check out without ever registering. If identified, however, the users can skip the step of filling in their address; have saved shopping carts, or other features.

Handling of Credentials

In order to accommodate the usage patterns above, sites have en masse opted out from using HTTP-based authentication, discussed below, and instead presented their own authentication based on *<form>* fields. Although better-secured alternatives are appearing[1], the prevalent method still involves sending the actual password over the network. Thus, even though server-side the

[1] For example http://www.peej.co.uk/articles/http-auth-with-html-forms.html, which adds Digest authentication on the JavaScript level

passwords are usually hashed with salts, they still need to traverse the Internet for each login.

Security, at this level, has generally been added by virtue of TLS, which websites have been using as a catchall solution. Using *https* instead of *http* has been seen as sufficient, although this amounts to obscuring the problem instead of solving it. Attacks on TLS will still provide raw, unfettered access to the password being transmitted, allowing the same kinds of hijack methods as those on unencrypted authentication.

Session Management

The by far most prevalent method of session management on the web is the venerable session cookie, a static, commonly long-lived, text string that is transmitted for every single page request. Infamously, this allowed the Firesheepplugin to function and hijack Facebook and Twitter sessions, although this was merely a demonstration of the known weakness.

Most commonly, session cookies are tied to a specific IP address or range, although this only mitigates the vulnerability. As an acknowledgement of the weakness of cookies, some websites and site software ask users to re-authenticate in order to access privileged areas[2]. The *HttpOnly* cookie, and explicit domain restrictions, also attempt to mitigate the ability of scripts to hijack cookie sessions, but are still vulnerable to MITM attacks using DNS cache poisoning.

Finally, as with passwords, session cookies are also considered *secure* if sent via TLS[3]. The same advantages, and restrictions, apply. Additionally, in order for these cookies to be truly secure, sites should opt for having all traffic encrypted, including the loading of static, non-privileged material, such as logos, stylesheets, or common page assets.

Issues with security solutions

The security issues surveyed above are not new. They are fairly well known, and have been around for a very long time. The reason they remain the preferred authentication solution is because of the lacking user experience of more secure options.

User Experience of HTTP-based Authentication

HTTP-based authentication and session management, such as HTTP Digest, was intended to alleviate or completely eliminate the issues with the current web session model. Despite being available for over a decade, however, it has consistently failed to gain traction, primarily due to the user experience when it is employed. There are four issues that contribute to this.

First is the login window, which is browser specific, which lacks any form of integration with the website. Apart from being a modal window, something that, in the authors' opinion, is a transgression of the web model, it provides no reference to how a user may register for an account, retrieve a lost password, or

---

[2] For example, phpBB will require an administrator to re-authenticate in order to perform administrative actions.

[3] https://developer.mozilla.org/en/DOM/document.cookie

otherwise find help. In order to compel site authors to use HTTP authentication, the end-user experience of using it must be able to meet-or-exceed the form-based authentication.

Second is the poor session management. When creating an account, the user would be prompted once for username/password to register, and then once more for username/password to log in, even though they just typed them in. Additionally, there is no clean way to log out a session, short of triggering another "please log in" dialog, or asking the user to quit the browser.

Third is the inability to, without additional hacks, provide for "guest" access, where a user can access the site both logged in and anonymous. A cookie can be used to indicate the user is logged in, and thus trigger a 401, but then they are logged in and cannot log out.

Finally, it was not designed to be used in an environment that supports identity federation. It is possible to generate one-time logins for users to support federated scenarios, but it is an awkward solution, at best. Federated identity, however, stands to become a cornerstone going forward, if done well.

User Experience of OpenID/oAuth Identity Federation

The other major open authentication- and session management solution is OpenID/oAuth. While these differ in many ways, the basic premise of the two is that they provide a secure way for a site to verify an identity and access the resources of that identity, respectively. OpenID allows for an arbitrary identity, whereas oAuth only works for identities from services that have a pre-existing agreement.

In both cases, the basic procedure is the same. First, the user must either specify the identity she wishes to use, i.e. OpenID URL, or select the service she wishes to use, e.g. Facebook. Then, she is taken to the identity provider site, thus removed from the site she was at. There, she may be asked to log in, and, finally, approve the identity federation/authorization procedure. Finally, she is redirected back.

A weakness we see here is that, when using OpenID, a website cannot help the user on how to access their account. It cannot help them remember their username, as it does not curate the profile. Even if the user remembers the URLs they use for OpenID, the site cannot give them a hint on which identity was used and associated with their profile.

Web Identity Candidate Areas for Improvement

We believe that, if the browser provided identity management for webapps to easily and seamlessly integrate with, it would stand to improve both the overall security of webapps and also the ease of use.

Improved Identity- and Session Management

The immediate, clear-cut candidate for improvement by improving the browser's identity handling is providing a better means of authentication and session management than what is available today, in order for sites to use the browser-based authentication mechanism instead of using their own.

Single-site authentication

Browser-based authentication should address the weaknesses of the HTTP Digest user experience, noted above. We believe that the reliance on session cookies is almost exclusively caused by the ease-of-use and -integration that session cookies provide. If the browser can match, or preferably exceed, it, there should be no reason for the continued reliance on session cookies. The key aspect here is how to integrate the user experience.

Identity Federation

As the curator of the user's site-specific sessions, the browser is already aware of them, either via a session cookie or, as is increasingly more common, by having the actual usernames and passwords stored with user consent. In both cases, since the browser already knows which identities the user has, it could provide a selection field for identity federation purposes. This would allow users to, when they need to use a federated identity, immediately select one of their stored ones, instead of having to locate- or remember an identity URL, or select from a list of providers they may not even have a relationship with.

Multi-Faceted Web Application Authentication

As more websites provide integration tools, for example embedding a Twitter feed, it looks probable that sites/webapps move from a combined whole into an assortment of parts, collected and integrated in a parent. In such a scenario, you can have, for example, a hospital health application, wherein a doctor can view a patient journal, speak to the patient, with both audio and video support, as well as take live sensor readings from the patient. Such an application would be mixing multiple security zones: hospital records for the journal, sensor readings, and audio/video. Each of these zones would, normally, require an authentication procedure.

A browser-provided identity management system that keeps track of user identities and sessions could potentially help secure such an application. For example, the browser could ease or automate identity handling for each of the application components. Another example could be the authentication towards a TURN server for the audio/video[4] component. In this case, it would be desirable to protect the TURN server against unauthorized use; the TURN server, however, is accessed only indirectly via a browser API accepting a TURN server address as a parameter.

Possible Approach

Suppose that, instead of dealing with usernames and passwords, browsers would handle identities and sessions. We would assign an identity to each entity in such a system; a browser would have one identity, a user another, a webapp a third, and so forth. In such a system, we could identify every actor, although they do not trust each other. These identities could be in the form of certificates, or some other equivalent mechanism, whereby everyone can individually assert an identity, but with the ability for that identity to be verified.

---

[4] http://www.whatwg.org/specs/web-apps/current-work/multipage/dnd.html#peer-to-peer-connections

We hypothesize that such a system could be used for trust delegation. For example, take the scenario of Bob logging in to his bank. He would use his mobile browser to go to the bank website. The browser can now identify the website, and the website can identify the browser. For the purposes of this illustration, we have chosen to disregard the implications this has on tracking. These can be mitigated by the browser, for example, creating a new or temporary identity for each site, session, or some other form of delimiter. For now, we consider it out of scope.

When Bob logs in, he will use some form of secure access token, such as a one-time password, or other, future, form of authentication; we do not wish to restrict the authentication type. What Bob will receive from the website, however, is a session token in the form of a session certificate specifically bound to the browser identity provided earlier. The browser can then use that session certificate for the purposes of having, generating or otherwise computing an authorization token, protecting against hijacking, replay attacks and content modification.

Furthermore, now that the browser has this token stored, it can offer it to the user as a federated identity, for example when using a gmail account to log into Livejournal. It can do so without exposing the gmail identity to livejournal – at least until the user affirms it – as one possible realisation is having a form field simply stating *<select type="identity">*, with possible parameters to limit which identities are accepted.

With the identities of browser, user, session and web application known, this could also be used to reduce and/or eliminate some of the notifications and prompts displayed to a user, such as access to location. The browser can also maintain a proper list of the active and stored sessions, displaying them similarly to how mail accounts are displayed in an email application, as opposed to directing users to check (or clear) their cookies.

Summary

In the above, we have described the challenges with existing security mechanisms in the browser. These, together with the new functionality that is being added to the web platform, underscore the need to solidify the identities in the browser. Furthermore, we believe the addition of capabilities such as device resource access, for example camera and location, may necessitate a more rigid security framework in the browser in order to be able to save user consent for access to these resources. We believe the web community should work on these issues, to ensure that the security of the web platform can fully support the capabilities of it.

Vladimir Katardjiev
LM Ericsson AB, Sweden
Vladimir.Katardjiev@ericsson.com

Göran AP Eriksson
LM Ericsson AB, Sweden
goran.ap.eriksson@ericsson.com