Addressing Identity on the Web

Blaine Cook

Submitted to the IAB workshop on Internet Privacy, Boston, December 2010

The development of truly open, federated services on the web has been hampered by the lack of a simple, effective addressing scheme and corresponding security mechanism. This paper discusses solutions for these two problems that emerge from the webfinger protocol.

Webfinger

The webfinger approach to addressing emerges from a realization that HTTP URLs are too complex and too varied in form for most users (both casual and professional) to truly understand and use effectively. Webfinger provides a mechanism with which to translate an email-compatible identifier into a profile consisting of links to services used by the owner of the identifier. This approach combines the simple semantics of the email address with the powerful infrastructure of the web and expressiveness of HTTP URLs.

The webfinger protocol is really just a trivial composition of host-meta, URI templates, and XRD documents. The domain part of the address is used to construct a well-known URI. This document contains a template URI that can be used to look up profile data for the user in question.

For example, the address `bob@example.com` is first translated to

```
http://example.com/.well-known/host-meta
```

This URI must return an XRD document that contains a link element with a `rel` attribute with a value of `lrdd`, and a template attribute that conforms to ((uri templating)). For example we'll use

```
http://example.com/profiles/{uri}
```

but the LRDD URI could also point to other domains or paths, or include query strings. The original email address is then substituted into the template URI. The substitution for our example is:

```
http://example.com/profiles/bob@example.com
```

which is a valid HTTP URI (before escaping) that contains the user's profile information in XRD format.

With this simple transform, we can transition from an email address to a rich set of data about a person. Now, all of the profile data that we've represented is public, and we still don't have a generic approach to exhanging private data with others on the web. However, if we leverage the public profile data, we can bootstrap into an HTTP-based web that incorporates rich concepts of privacy while embracing the "light touch" philosophy of HTML and the web.

Implementing Privacy

Commensurate with this light touch philosophy, there are a number of possible approches to implementing webfinger-based privacy. All of the approaches, however, involve signalling the identity of the requestor to the server handling the request. In order to do this, the client sends an additional HTTP header, `From`, whose content is set to the bare address of the requestor. A minimal example looks as follows:

```
GET / HTTP/1.1
Host: example.org
From: bob@example.com
```

Based on this request, the content may be delivered to the user depending on example.org's policy. If `bob@example.com` is allowed to see the content, then the content is returned with a standard HTTP 200 OK response. If not, differential content, or an alternate HTTP response may be used (e.g., HTTP 401 Authorization Required).

Of course, based on the request above alone, the server has no way to verify that `bob@example.com` actually made the request. Any HTTP client is capable of trivially adding the From header, so we need to authenticate the request. As mentiomed above, there are many ways to approach this problem. In this paper, we will discuss two possible approaches, one cryptographic another non-cryptographic approach.

Public-Key Authentication

The cryptographic approach is in some ways conceptually simpler than the non-cryptographic approach, but carries a higher implementation cost. In order to authenticate the request, we provide a public key in the requestor's webfinger profile and sign the request using it. The exact mechanics of signing requests is beyond the scope of this discussion, but two very well-understood approaches include TLS (usong client-side certificates) and "magic signatures" using signed JSON packets to encode the request body.

Dial-back Authentication

The second approach is to use server dial-back; the client, acting on behalf of the user identified by the `From` header, provides an HTTP-based mechanism by which the server can validate the domain of the requestor. The server can then authorize the requesting domain to make requests on behalf of the user by checking the Webfinger profile of the user identified in the from header for a link with a `rel` attribute equal to `delegate` and an href that points to the (previously authenticated) requesting domain.

There are a number of ways to authenticate a client that is making an HTTP request. The most widely deployed approach at the moment is part of the PubSubHubbub protocol, which incorporates a callback URL and a shared secret to authenticate subsequent content delivery requests. In PubSubHubbub, the callback URL enables the server to verify that the subscriber actually intended to make the subscription request. For the purposes of this discussion, the callback URL enables us to verify that a given domain intended to make any given request.

Conclusion

The net result of webfinger plus the strategies to authenticating the identity of actors on the web is that we can compose any social interaction that is presently encoded by closed-wall social networks, but in a way that is interoperable and open to any participants.

Most importantly, the user experience of creating relationships online remains largely the same as the user experience of Facebook, Twitter, or any other "traditional" social networking platform. In addition to a proven user experience, we add a rich set of possibilities in the privacy space. New networks can form where privacy is an utmost concern, and individuals can make decisions regarding information sharing in more specific contexts, rather than the share-all versus share-nothing dichotomy that exists today.