How BrowserID Works

2 weeks ago

(a special thanks to Mike Hanson and Ben Adida for their careful review of this post)

BrowserID is a decentralized identity system that makes it possible for users to prove ownership of email addresses in a secure manner, without requiring per-site passwords. BrowserID is hoped to ultimately become an alternative to the tradition of ad-hoc application-level authentication based on site-specific user-names and passwords. BrowserID is built by Mozilla, and implements a variant of the verified email protocol (originally proposed by Mike Hanson, and refined by Dan Mills and others).

Before learning the technical details of BrowserID, it's recommended you experience a user's perspective of BrowserID with the myfavoritebeer.org demo, and then work through the integration tutorial for a website developer's perspective.

This post aims to provide a readable technical overview of the system. First it will summarize the key design elements of BrowserID. Next, it will explore the various actors in the system and their inter-relationships. Finally, we'll walk through several of the most important flows, including certificate provisioning (where the user obtains authentication material from an identity provider), assertion generation (where the user uses that material to tell a website who they are), and assertion verification (where the website being logged into verifies the user's email address).

BrowserID Features

Perhaps the best way to begin understanding BrowserID is to walk through its key design features:

- **An email is an identity** - There are no *usernames* in BrowserID, it uses emails addresses instead. Users identify with emails quite naturally, and no new infrastructure is needed to reliably verify ownership of them.
- **Decentralized** - A user's authentication to a website occurs in relative isolation. No network transactions with third parties are needed, so it is efficient and privacy-protecting. Additionally, any email address may be used, and any email provider may provide first class BrowserID support for their users.
- **Ownership-Based Authentication** - In BrowserID, the browser manages authentication material which can be used without a password - making authentication with BrowserID more reliant on ownership factors, and less on knowledge factors.
- **Usable today, and better tomorrow** - An HTML5 implementation provides a functional system today, and BrowserID is designed with adoption by browser vendors in mind. Native support in browsers will afford improvements in both user experience and security.

Mechanism

BrowserID uses asymmetric cryptography and digital signatures to allow browsers to create signed assertions about the user's identity, and by identity providers to vouch (via signing of a key-email pair) for a user's identity in a disconnected fashion. BrowserID uses cross document messaging to communicate between documents served from different domains, which makes a usable implementation of BrowserID possible *right now* without modifications to existing browsers.

Actors

As said above, BrowserID is *decentralized*, which results in several actors interacting under a healthy mutual distrust. These actors include:

- **Primary Identity Authorities** (or often just *primary*) - Services that directly provide the user with an identity in the form of an email address. Example *primaries* include Yahoo! mail or gmail. A primary can build "BrowserID support" and directly vouch for its users' identities.

- **Relying Parties** (or *RPs*) - Sites that use BrowserID for authentication (relying on the claims of identity authorities). The demonstration site myfavoritebeer.org, is an example RP.
- The **Implementation Provider** (or *IP*) - This may be the user's web browser if native support for BrowserID exists, otherwise browserid.org fills this role by serving web resources that implement the client portion of the system. In addition to key management and implementations of the required algorithms, the *IP* also serves as a *Secondary Identity Authority*. The secondary authority - that is, browserid.org, or in the case of native support, a set of servers selected and vetted by the browser vendor - will verify and vouch for email addresses issued by providers who have not yet implemented BrowserID support.

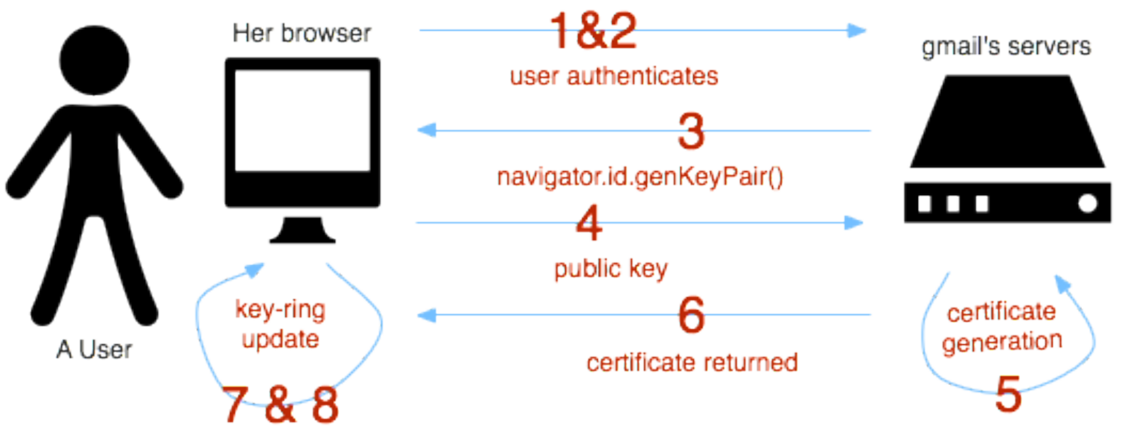
Important Flows

A solid understanding of how BrowserID works can be attained by working through the main flows of the system, in terms of the interactions between the actors defined above. This section will walk through the three most important flows:

Certificate Provisioning

Certificate Provisioning is the process by which a primary (like gmail.com) verifies that a user owns one of her email addresses (like lloydhilaiel@gmail.com) and provides them with a signed certificate that vouches for their ownership of that email.

Visually, the flow looks like this:



The actors involved in this flow include the user, her browser (which happens to have BrowserID support built in), and her email provider gmail (who in this case happens to be a BrowserID primary identity authority).

1. Some event occurs whereby the user indicates that they'd like to log into an RP using their gmail address, and the user is directed to a web-page on gmail.com designed for the purposes of key provisioning.
2. The user authenticates to gmail using their user-name and password. (or, perhaps, a stronger authentication scheme - the strength of this login is entirely up to the authority)
3. gmail-hosted javascript invokes navigator.id.genKeyPair(), a client side function implemented by the browser, that causes a key-pair to be generated. Upon the completion, the public key is returned to the callee (gmail's javascript), and the key-pair is cached by the browser for the duration of the session.
4. gmail's javascript code on the client sends the public key up to a gmail server over SSL.
5. gmail signs the user's email address, the public key, and a validity interval generating a JWT (which is just a means of encoding a signed JSON object).
6. gmail returns this bundle to the client as a response to the request in step 3.
7. JavaScript code served from gmail invokes navigator.id.registerVerifiedEmail() on the client passing in the certificate.
8. The browser locates the private key generated in step #2 and moves it and the certificate from temporary session storage into the user's BrowserID key-ring. The user now has a valid certificate from gmail stored in their browser which they can use to generate assertions proving their identity.

Users will encounter certificate provisioning anytime they wish to use an email to log into a site that they haven't used recently on their current browser.

The flow above assumes that the primary (gmail) has built custom BrowserID support. In practice, BrowserID must handle the case where an email provider hasn't (yet) built such support. In this case browserid.org manually verifies email addresses and acts as a *secondary authority* (itself issuing certificates for email addresses which it does not control).

Finally, in the flow above the browser has *native* support for BrowserID, exposing functions to generate key-pairs and store certificates. In the absence of such support, BrowserID provides a small JavaScript shim that implements the missing functionality using standard HTML5 techniques and cryptographic routines implemented in JavaScript.

Assertion Generation

Assertion Generation is the process by which a user's browser produces an *assertion* that proves that a user owns a given email address.

1. During the process of logging into a website, the user clicks on a "sign in" button on the RP's site, causing the RP to invoke `navigator.id.getVerifiedEmail()`.
2. The user selects an email address that they would like to use to log in from a list rendered by the browser.
3. The browser combines the domain requesting the identity (the *audience*), a validity period, and the certificate associated with the identity in to a bundle (the certificate includes a public key, and the email address being shared).
4. That bundle is signed using the private key associated with the identity, encoded into a JWT, and returned to the web page.

The result of assertion generation is a JSON structure which looks like this:

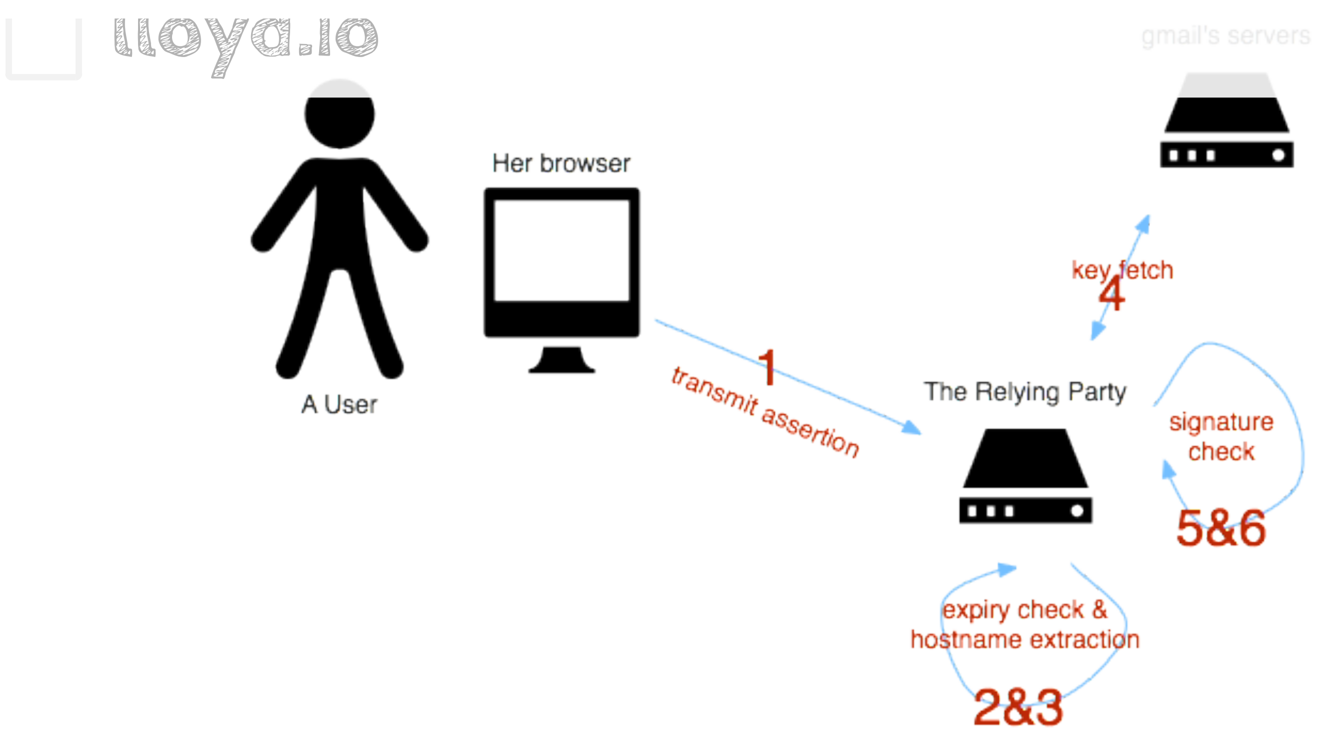
```
{
  "assertion": {
    "audience": "myfavoritebeer.org",
    "valid-until": 1308859352261,
  }, // signed using lloyd's key for lloydhilaiel@gmail.com
  "certificate": {
     "email": "lloydhilaiel@gmail.com",
     "public-key": "<lloyds-public-key>",
     "valid-until": 1308860561861,
  } // certificate is signed by gmail.com
}
```

At the completion of this flow, the browser has provided the RP with an email address that they can verify is owned by the user. See the next section for how the verification process works.

While the flow above describes the case where native browser support exists for BrowserID, the flow is identical (except for the user interface) when the browser does not have native support: In this case all of the required functionality can be supplied by a JavaScript shim.

Assertion Verification

Assertion Verification is the process by which a *Relying Party* can verify that an assertion of a user's ownership of a certain email is valid. Verification looks like this:



1. The RP (securely) transmits the assertion from the client up to her servers.
2. Validity periods are checked on both the certificate and the assertion.
3. The RP extracts the host-name of the email within the assertion; this is the primary identity authority for the email address. In our example above, it's gmail.com.
4. Public key(s) for gmail.com are attained from a well-known location on their servers (specifics TBD).
5. The certificate signature is verified; success proves to the RP that the embedded user's public key is valid.
6. The assertion signature is verified using the embedded user's public key, after which point the RP knows the assertion is valid and the user owns the specified email address.

At the conclusion of the *assertion verification* flow, the RP has a verified email address for the user.

The above flow assumes that the primary identity authority supports BrowserID; specifically, that the authority provisions certificates and publishes their public keys on their site. In the case that the email that is the subject of the assertion is not from a domain where BrowserID support is present, then the assertion certificate will include an issued-by property that is the domain of *secondary authority*: the entity that has vouched for the validity of the email address. The common case today is that this will be browserid.org, but in the future there may be a small number of secondary authorities run by browser vendors or trusted organizations. RPs are explicitly asked to trust these authorities for email verification, so their processes and operational security would need to be transparent and of the highest quality.

In a future where BrowserID is widely adopted, secondary authorities are the exception rather than the rule. Identity issuers would be directly responsible for the security of their own users.

Implementation Status

At the time of writing browserid.org is a partial implementation of the system described here. The key differences between what is described and what exists are:

- **certification** - BrowserID today requires that authorities host all public keys associated with all users. It will move to certificates in the coming weeks.
- **primary support** - BrowserID doesn't currently support primary identity authorities as described above, as there aren't any. In the coming months it will defer to 3rd parties properly and gain support for primary authorities.

Differences from the Verified Email Protocol

This post exists to provide a clear description of how BrowserID works, and also to precisely express the ways that it is different from various different implementations of the same theme. BrowserID is a simplification of the protocol originally proposed by Mike Hanson, having two key differences:

Secondaries de-emphasized.

The original proposal emphasized the distribution of secondary identity authorities more than BrowserID does. There are significant UX and administrative challenges in supporting distributed secondary authorities, and with BrowserID the thinking is that it is better to focus on encouraging email providers to include BrowserID support than it is to create a new ecosystem of secondaries, which may ultimately be detrimental to the safety and usability of the system.

No webfinger based assertion verification

The original proposal included two different ways for an identity authority to vouch for a user's identity. The first method was as in BrowserID, via a cryptographic signature. The second method was for the authority to publish the user's current keys via webfinger and in this way vouch for them.

The latter approach is omitted from BrowserID because it is perceived as both reducing the privacy of the system (RPs would ultimately leak more information back to identity providers about the user's activities), and because it increases total system complexity.

Showing 48 comments

Sort by oldest first

Ar

In the case of "Assertion Verification" it is clear how one can detect where to check: the IP or the primary. For the "Assertion Generation" this is not. How does the generation know if the primary is browserID enabled?

1 day ago Like Reply

haloman

I'd argue that it's only partially decentralised since secondary providers are centralised. I am a bit concerned that browserid.org is a Mozilla landgrab at a time when they are rapidly losing marketshare, and I'm not convinced that other browser manufacturers will, or should, trust Mozilla as the central authority.

1 day ago Like Reply

Lloyd Hilaiel

@Ar for assertion verification, in the case that the IP (a secondary identity authority) has generated the assertion there will be an 'issued-by' property in the assertion -- From the post: "In the case that the email that is the subject of the assertion is not from a domain where BrowserID support is present, then the assertion certificate will include an issued-by property".

During generation, there will be a way that the implementation (the browser) can check if the primary has support and can fallback to a secondary. Now how that will work precisely isn't completely nailed down. Perhaps something in well-known. Here's the github milestone where we'll figure that out:

https://github.com/mozilla/bro...

1 day ago Like Reply

Lloyd Hilaiel

@haloman It's still unclear to me whether 3rd party (non RP, non browser affiliated) secondaries can be implemented in a way that's not confusing all around, but I agree that there are significant benefits.

With respect to the central service, as you know, it's a bootstrapping tool. My best first guess is that mozilla runs it with the level of transparency that you'd expect - to *earn* folks trust. As the experiment evolves though, I think we'll have a better understanding of how much we should re-emphasize distributed secondaries. Some specific concerns about a plethora of secondaries is that there is then a burden placed on RP's (who should they trust?), and also on users (an email address alone is no longer enough information to use the system, you need to specify who the secondary will be as well).

We've got a good size mountain to climb yet!

1 day ago Like Reply

Harshavarma

So anyone who has access to my computer can actually get in to any site where i use BrowserId?

1 day ago Like Reply

Scott Elcomb

I'm concerned about this as well; what happens if a user looses control of their registered email account? (TOS violation, provider shutdown, account gets hacked?) While an interesting concept, I'll need to be more thoroughly convinced before I use or recommend BrowserID.

1 day ago in reply to Harshavarma Like Reply

nottledim

Hmm, I don't know. When I tried to register through myfreebeer.org I was unsuccessful. It seems the javascript tries to open new windows which are not permitted by my browser (FF v5) config. All it does is open a new blank window, download some stuff and close the window again. Interacting is impossible.

It works a bit better with Chromium. After logging out if you try to log in again it only offers my registered email addresses. It doesn't log out of browserid.org. There needs to be an easy way to do that.

1 day ago Like Reply

Michael Roitzsch

Looks like a nice improvement over OpenID, but they way I understand it, the primary I choose to use can completely impersonate me against the relying party. Did I miss anything that would prevent that?

1 day ago 2 Likes Like Reply

Pascal Sartoretti

It's a shame that I could not login with BrowserID to post this comment :-)

1 day ago Like Reply

Nate Silva

Michael: Interesting. By my reading the primary could verify a fake public key for you, and a malicious party could use the corresponding fake private key. If this is the case, couldn’t a malicious party (or law enforcement, or oppressive government) gain access to an account by forcing Gmail to provide them with a fake public/private key combination?

1 day ago 1 Like Like Reply

Cryptosmith

I like the idea of using an email address as an identity - it makes personalized identities a bit easier. However, the examples are misleading because they use "gmail" but as far as I can tell, gmail doesn't really support BrowserID.
If I understand correctly, "BrowserID.org" is the only primary at the moment.

1 day ago Like Reply

Lloyd Hilaiel

I agree. It would have been clearer perhaps if I used a fictional name instead of gmail.com.

1 hour ago in reply to Cryptosmith Like Reply

Benjamin Krämer

@Cryptosmith: As far as I understand the verified email protocoll, BrowserID.org serves as secondary identity authority. Therefore you find an "issuer" element in the reply from the verifier.

1 day ago Like Reply

Cryptosmith

Weird. I just did the myfavoritebeer demo and created a BrowserID.Org login. I'm not sure where Firefox stashed the private key. It isn't under "Your Certificates," where PGP would have left it. Perhaps it's considered a "Security Device," but there's nothing that appears related to Browser ID.

1 day ago Like Reply

Benjamin Krämer

the key is saved in the local storage for the domain https://browserid.org. when you open the site https://browserid.org/manage and look in the source code you will find it. The method display_saved_ids() parses window.localStorage.emails and get's the public key. you can also view your public key on that site ;)

1 day ago Like Reply

Cryptosmith

OK, so my PRIVATE key resides on a separate site. Oof. That's a poor design choice. It makes the key holder is the ultimate authority on identity and not the individual.

1 day ago Like Reply

darkskiez

Cryptosmith, I _think_ your private keys remain in your browser storage (with the html5 js code at present but could be natively implemented), but they are signed by your email provider, (or in this case browserid.org instead, for the moment)

1 day ago in reply to Cryptosmith Like Reply

Michael Hanson

No, the private key resides in browser-local storage - it is KEYED on the browserid.org domain but the storage is your local hard drive. The public key is signed by browserid.org (or, eventually, a primary authority), but the private key never leaves your machine.

You are correct that this is not the safest place to keep it - a hack at the SSL level, or an attacker that compromises the server, could inject code to manipulate the client-side storage. A browser-native implementation of the protocol (coming) will store it in your browser profile, where it can only be accessed by trusted (i.e. addon or built-in-to-the-browser) methods.

1 day ago in reply to Cryptosmith Like Reply

darkskiez

Except the private key is worthless to the primary authority as they can generate their own private key pair that authenticates as you :)

I think its still very valuable as a lightweight auth or two-factor additional step.

23 hours ago in reply to Michael Hanson Like Reply

Benjamin Krämer

yes, and that was also stated in the verified email protocol. i think the secondaries are supposed to offer service as temporary authorities for the time till primary authorities exists. therefore the issuer is shown in the verification reply. you are free to trust that issuer or disaccept the assertion. if the mail was verified by a primary authority, the verification reply contains a signed certificate from the primary authority as shown above. those replies are more trustworthy. but since there are no primaries yet, we have to trust in secodaries to get the thing running.

right now i'm doing a test implementation. i want to support BrowserID and integrate it as alternative login in a discussion board. i allow no registration using BrowserID, so it's only for single sign-on purpose for now. so i can use the email of the already registered user and compare it with the mail in the assertions reply.

i think if BrowserID is implemented in firefox and maybe some other browsers, this could be a great deal!

1 day ago Like Reply

Benjamin Krämer

I have implemented it now and it works great! But untill it's stable and the security concerns are more better discussed i made it activatable by choice per user. on default it's disabled for all users. only users that want to use it and now that it's experimental can already use it.

1 day ago Like Reply

darkskiez

Its probably worth mentioning, but the concerns about impersonations by people who control the primaries, well, anyone who has access to your email account can 'reset' your password. Particularly sensitive sites could also use a password, in conjunction to browserid, now you have two factor authentication!

If you lose control of your email address, well, I would assume sites would/could allow you to link multiple [backup] email accounts together into one identity. Many sites already do this.

1 day ago Like Reply

Adam Crabtree

This may sound ridiculous, and while I think this is a great idea in theory, but it seems strange to take email, an unofficial form of identification, and making it official at a time when the future of email is belayed on all sides by Facebook, Google+, Twitter, etc... (granted, these are sites not standards, but I think they forebode email's eventual replacement).

This feels a lot like using Social Security #s for identification, which they were never built for, albeit with added security.

1 day ago Like Reply

Michael Hanson

Adam - you'll note that all of those sites still log you in through an email address. The essential idea of "username" AT "somewhere on the Internet" is the cryptographic basis of BrowserID. A Facebook ID is really "facebookID" AT "facebook.com" and a Twitter ID is really "twitterID" AT "twitter.com" - even if they don't participate in today's email-based message exchanges. If all of the email infrastructure we've built up over the last 40 years goes away, the core idea of site-scoped identities will still be with us, and the ideas contained in this system will still work.

1 day ago in reply to Adam Crabtree Like Reply

Adam Crabtree

Although I will say, aside from my comment, this is pretty cool. =D

1 day ago Like Reply

mati_pl

Is there a way to display all GUI in other language than English? Polish for example? What about translations?

1 day ago Like Reply

Lloyd Hilaiel

@mati_pl In it's experimental form, we're english only for now. We will localize it in the near future, here's the placeholder milestone: https://github.com/mozilla/bro...

1 day ago Like Reply

HedgeMage (Susan Stewart)

Sorry, you lost me at "email as identity" -- I get more than enough spam as it is. Logging in to sites without revealing my email address is one of the key selling points of OpenID.

19 hours ago 1 Like Like Reply

Lloyd Hilaiel

Hi HedgeMage. Here's a thread for you: http://groups.google.com/group...

I don't know if you'll be swayed, but it's worth a shot!

1 hour ago in reply to HedgeMage (Susan Stewart) Like Reply

Edchick

This is all very technical and from a user perceptive i dont see how this would benefits me. Does that means every sites who uses BrowserID as Login system would no longer require a user to login every time once they have their "key" stored in their browser? (Like info stored in cookies).

19 hours ago Like Reply

Lloyd Hilaiel

Does the video help make the case? https://browserid.org/users

This post is *supposed* to be techhnical, it's the technical primer! :) Please respond if you feel like the video could be changed to make it more clear why users should care

1 hour ago in reply to Edchick Like Reply

Richard Pitt

It appears to me that all this facility does is lower the latency for email verification - it removes the smtp link and substitutes a server lookup to the public key host and some CPU cycles to do the verification.

16 hours ago Like Reply

darkskiez

Its does more than email verification, because you don't need to tell sites a password for re-authentication. That would be a pain to have to send a new email or find an old one to get back into a site. Its better that lightweight websites can delegate your security to your email provider than store your password.

10 hours ago in reply to Richard Pitt Like Reply

Richard Pitt

To quote what I just got - which would be done away with if this verification process were in place...

"You've just posted a comment on "trickyco.de - lloyd's blog." and have selected to receive email notifications whenever new comments are posted. Please click the following link to confirm that you would like to receive these notifications:

http://disqus.com/anonverify/?...

Thanks,
trickyco.de - lloyd's blog."

16 hours ago Like Reply

Ronald

Delegation of authority to anybody other than yourself does not suffice as a secure authentication in my books. Further more there are better Public Key Cryptography based login systems already out there that goes with the anonymity of the web. Except Firefox doesn't implement it very well.

Please see: http://wiki.cacert.org/Technol...

Furthermore the method I linked supports using Security Tokens as a key and certificate store (at least on Windows at the moment). Not to mention an additional down point is when (not if) the e-mail provider is down for whatever reason, the user won't be able to login to anything.

I've already listed 2 primary problems the architecture above will never be able to solve, and it's also the same issues with OpenID. All in all, the direction of authentication systems like OpenID and BrowserID is vulnerable on so many key structural levels that unfortunately I deem it no better than a simple username and password system.

In fact, a username and password system coupled with a user's password manager that generates 200 bit passwords which is encrypted using security tokens is far better (from the point of view of security and anonymity) than this.

The challenge then should become how to make it easier for technology like ClientCerts to gain public adoption. We as developers ought to take out the technical elements so every day users can use it and not by adopting what is essentially a centralised authentication system; the underlying point in "Assertion Verification" makes BrowserID a centralised system. Claiming to be decentralised is misleading and inappropriate.

Bottom line, there is no point in giving public key to service providers like "gmail" or browserid.org. You might as well skip that part and upload the public key directly to the website you would like to access so that there is only 2 point of contact of your login information.

16 hours ago Like Reply

Tony Mechelynck

"Sign in" → An empty popup comes up, and goes away after a few seconds, but apart from that, nothing happens. In particular, the original page still says "Sign in", not "Hellow, someone_or_other".

"My account" → Manage your email addresses (none listed). You can, at any time, cancel your account (If I click that, nothing happens).

:-??? How do I *create* an account :-???

15 hours ago Like Reply

Marcel

To get some attention for the BrowserID experiment I wrote a little WordPress plugin to login with BrowserID to any WordPress powered site.

10 hours ago Like Reply

Lloyd Hilaiel

Awesome! We should reference this and the drupal plugin on browserid.org: https://github.com/mozilla/bro...

1 hour ago in reply to Marcel Like Reply

Crucing Stin

You need to promote that system, do that and everything goes

6 hours ago Like Reply

nottledim

@Tony Mechelynck I've been trying to find where the problem is. It's quite tricky because the window closes quite quickly. However there seems to be 2 pieces of information which might be a clue:

1) in storage.js getEmails calls JSON.parse on window.localStorage.emails. That value is being reported as null.

2) jquery.js line 6058 is reporting: uncaught exception: steal.js Could not load /jquery/event/event.js. Are you sure you have the right path?

Any ideas?

4 hours ago Like Reply

nottledim

@Tony Mechelynck I've fixed this for my browser (I think): about:config search for dom.storage.enabled. This should be set to true. In my case it was false.

4 hours ago Like Reply

Grok Grokem

I run a site with over 2M unique active users, 500K of which pay for our service. I could never implement this system because it uses email as the identity. A system that truly wants to solve the authentication problem needs to support sites that need to tie an account to an email address and those that can't require it. Because of the demographic I serve, older non-computer savvy, I would lose upwards of half my non-paying base and a third of my paying customers. I base these guesses of customer base lose on calls for account access where they state they "never read email and don't know how to access it" or "That was my old email address".

I believe that there are a large number of sites like mine that are looking for a better way to authenticate users but can't jump on board with system based on email.

3 hours ago Like Reply

Lloyd Hilaiel

There is nothing preventing a traditional login as a fallback. This is what's done in practice with existing id systems. So more than half of your base gets simpler login and the remainder is unchanged.

That said, if you're targeting a demographic where half of your users don't have email, it suggests to me that maybe you should wait to adopt browserid until we have gotten through several iterations of user testing and UX refinement. We should be learning a *lot* in the coming months.

1 hour ago in reply to Grok Grokem Like Reply

Ben Clifford

Protocolwise, the ID you use (that happens to be able to receive over SMTP) doesn't have to be "your email address" (i.e. the thing you read in gmail or pine) - it can be, like openid, some "identity-only" service.

Except that sites will presumably assume that they can and should send messages to that ID, which doesn't happen with openid because the IDs don't look like email addresses...

3 hours ago Like Reply

Bai

Missing revocation

OpenID has an active request to the identity provider every time you do an authentication.
BrowserID does not seem to have this, the identity is proven by a public key infrastructure.

So my concern is:
If someone steals your private keys: Even if you find out, you have no chance to revoke login ability, because there is no certificate revocation list. On openID you simply change your password, if you think someone found out your password.
But with BrowserID you will have to actively go to all service providers, to register a new public key, to immediatly block access to your data.

If you compare this to certificate based email encryption: you are missing a central certificate revocation list with BrowserID

3 hours ago Like Reply

Lloyd Hilaiel

The current thinking is that certs will be extremely short lived, like valid only for some number of hours. This design choice replaces revocation. Tradeoffs abound!

1 hour ago in reply to Bai Like Reply

Vincent Bernat

A post comparing OpenID and BrowserID would be welcome. Even after reading the comments, I still conclude that BrowserID is like OpenID with email address instead of URL.

1 hour ago Like Reply

Lloyd Hilaiel

Ben Adida wrote this recently: http://identity.mozilla.com/po...

1 hour ago in reply to Vincent Bernat Like Reply

Add New Comment

Login

Please wait…

blog comments powered by DISQUS