



Helping users to manage the information they disclose to websites

Dave Raggett,
W3C/ERCIM
dsr@w3.org

This paper describes ideas for helping users to manage the personal data they disclose when interacting with websites. This is supported by a user agent that either runs as a browser add-on or as a trusted 3rd party service. The user agent allows users to select what identity to use with each website, along with what choice of credentials to provide. Users can set privacy preferences and review what personal data they have disclosed to each website.

Introduction

As more and more aspects of our lives are recorded online, there are real dangers of information leaks between the different faces we present when interacting with others: one's private self, family, close friends, work colleagues and members of the public. Further risks come from potential abuse by unscrupulous or insecure websites. This paper describes ideas for helping users to manage the information they disclose to websites, using either a browser add-on, or a trusted 3rd party service. This builds upon work in the [PrimeLife research project](#) using funds provided by the European Commission's 7th Framework Programme.

The current situation

It is common for websites to track successive visits with HTTP cookies. Web servers can also track information provided in HTTP requests, e.g. the IP address, time of day, the user agent, and language preferences. The IP address can be readily mapped to a rough geolocation, something that is used to limit access to media resources such as recordings of television programs (e.g. the BBC iPlayer limits access to viewers in the UK).

Search engines save query strings for extended periods of time, and also make use of HTTP redirection to track which links user's click on in the search result pages. The information so collected can in principle be analysed to make a detailed profile of users, particularly when combined

with other sources of information. Most users are unlikely to turn to an anonymizing proxy server as a means to protect their privacy. As yet there is no way for users to clear their personal data held by search engines.

A common way for websites to collect personal data is by inviting users to fill out web page forms. Modern browsers can remember what values you entered into single line text fields. When you re-visit the page and start typing into a field, the browser displays a drop down menu of what you have typed before. You can remove a specific entry by selecting a value and pressing the delete key. You can get the browser to forget all of your previous form entries on all pages. You can even instruct the browser to disable remembering of form field values altogether. Websites can force you to type a field's value by setting the autocomplete attribute to "no" (part of HTML5).

Another technique is to save field values in web page cookies. A web page script extracts values from the cookie and fills out the field. When the user types a new value for a field, the cookie is updated by an event handler set by the script on that field or on the form. Cookies can be used for authenticating users, session tracking and remembering specific information about users including site preferences and the contents of shopping carts.

Users are often required to sign into a website with a user id and password. This is hard for users to remember, and it is commonplace to use the same user id and password on multiple sites as people trade security for usability. Increasingly websites ask for an email address for the user id. This is used to check for a valid email address — users have to read an email sent to them when they sign up, and to click on a link to activate the user account. Email addresses are global ids and as such make it easy to track user activity across multiple co-operating websites. Another way to achieve that is through a (single-pixel) image loaded from a tracker site.

[OpenID](#) is an increasingly popular means for users to sign on to websites using a web page address. The technique supports single sign-on, so that users only need to enter their password once per session, rather than once per website, although this only applies to websites supporting OpenID. The use of a web page address (a URI) suffers from the same drawback as email addresses in that both are globally unique, facilitating tracking user activity across sites.

There is little support for helping users to manage their privacy. Browsers provide the means for disabling the use of cookies, but many websites rely on these, so that users find it hard to do without them in practice. Whilst you can get the browser to clear the cookies, you

typically can't clear the personal data that a given website holds on you server-side.

With the culture of everyone expecting services for free, many websites rely on advertising and co-marketing with other sites that offer complementary services. The website gains from maximising the personal data it collects on its users. This is offset by data protection laws (at least in Europe), and the negative effects on the website's popularity from privacy related scandals.

New directions

Browsers such as Firefox provide a means for users to easily install add-ons that extend the browser's capabilities. This can be used to implement a trusted assistant that helps users to manage the identity they use with each website, as well as to track what data they have disclosed over a prolonged period of time.

The assistant can further allow users to describe their preferences for how websites should handle their personal data. These can be matched with privacy policies provided by websites, resulting in an agreement whereby the website is authorized to perform certain operations and required to do certain others. This paper describes possible extensions to HTTP to support this, along with alternatives based upon annotating web pages, and conventions for web forms.

Credentials are statements that attest to properties, e.g. that you are a resident of a given city, and are over 21 years of age. If the issuer of the credential is trusted by a website, and the authenticity can be established satisfactorily, then credentials can supplement or even replace the need for disclosure of your user id and password. If websites are to use credentials, we need to consider the implications for the user interfaces and over the wire protocols.

In many cases, users are not in a position to make an informed decision as to whether to trust a website with their personal data. One approach to this is for the website to display a seal of approval from a well known entity, e.g. the "better business bureau", who only issue seals to websites that conform to established codes of business. Another approach is for the user to contact an authority or a group of friends to see what they think. This allows for an aggregation of experience over many people and many sources of information. This approach can be implemented as a delegation mechanism and is described below in terms of trusted intermediaries.

A further challenge is to support cross-website handling of personal data, e.g. where you want to use a service on one site that needs to access your data on another. [OAuth](#) is a new means to support this wherein

the user is invited to sign into the second site to create a credential that authorizes the first site to access the data for a given purpose and for a limited time. This paper will describe a means for trusted intermediaries to issue such credentials on the user's behalf, something that is likely to be of increasing importance for cloud-based applications that run 24x7 on behalf of users, and deals with the case where the user isn't immediately available to handle a given authorization request.

Privacy Assistant as a browser add-on

Firefox add-ons are browser extensions that can be installed over the Web. The add-on typically consists of scripts, style sheets, images and UI components, and can be designed to add a menu option to the browser's menu or status bar, allowing the user to invoke the add-on as if it were built into the browser. The add-on can also inspect each web page as it is loaded into the browser, hook into user interface events, save and retrieve information from local storage, communicate with remote servers, and make use of additional capabilities exposed by the browser.

A browser add-on could work in parallel with the browser to track what data you have given to each site (and when), and to allow you to determine which sites a given value has been disclosed to. One challenge is the lack of standards for form field names.

A further idea is for the assistant to propose user names and passwords rather than burdening users with having to come up with these. The user wouldn't need to remember them, since the assistant will safely store them in an encrypted form.

Privacy preferences, authorizations and obligations

An introduction and definitions relating to the protection of personal data can be found on the [website of the Data Protection Office of the European Commission](#). Further information can be found on the [European Commission's Data Protection website](#), which states the following data protection principles:

- Fairly and lawfully processed;
- processed for limited and explicit purposes;
- Adequate, relevant and not excessive;
- Accurate;
- Not kept longer than necessary;
- Processed in accordance with the user's rights;
- Secure;
- Not transferred to third parties without adequate precautions.

Users should be able to access their personal data and require websites to correct without delay any inaccurate or incomplete personal data.

Users should be able to have their personal data erased if the website is found to have been processing it unlawfully.

The PrimeLife project is working on ideas for comparing user preferences with privacy policies provided by websites. The policy describes authorizations as things that the user permits the site to do, and obligations as things that the user requires the site to do. Here are some examples:

- The site may use the data for some set of purposes, e.g. marketing, payment, delivery, etc.
- The site may pass the data onto 3rd parties for specific purposes and subject to the same obligations
- The site must delete the data after 12 weeks
- The site must notify the user when the data is accessed for specific purposes

These are in addition to any legal requirements imposed by data protection laws.

The site's privacy policy could be conveyed to the browser as part of the web page, either via a link to an external representation, or embedded in the page, e.g. within the value of an HTML meta element. Eitherway, a browser add-on could extract the policy and compare it with the user's preferences, and alert the user if the site's policies aren't compatible. The user would then have the choice of ignoring the warning or "walking away" from the current page.

Notifications could be provided to users in a variety of ways, e.g. on paper as letters sent through the post, as emails, or some other means. The convenience to the user is a major factor. In many cases it will be appropriate to allow users to view notifications via a pull mechanism, analogous to RSS. The browser add-on acting as a privacy assistant could provide a means to retrieve and present such notifications. The assistant could also help users to visit the appropriate web page for each site for reviewing what personal data is held on them. The URIs for these functions should be included in the website's privacy policy.

Privacy safe zones and search engines

The very act of visiting a website discloses personal information through the data carried in the HTTP request. W3C's Platform for Privacy Preferences (P3P) defines a special set of ["safe zone" practices](#) for accesses to policy reference files at well known locations. These practices require websites to avoid using in an identifiable way any information collected while serving a policy file or policy reference file. In principle, before downloading a web page from a website, a browser can

first visit the policy reference file at the well known location and warn the user if the policy is incompatible with the user's privacy preferences.

One issue for P3P is the potential for policy reference files to be very large if different pages on the site have different policies. A possible solution is for the browser to pass the privacy preferences to the website to match against its policy. If it is incompatible, the site follows the safe zone practices, and returns an error response including the site's policy. This assumes that the browser has first determined that the website conforms to the privacy protocol and as such won't cheat!

If visiting the site (even a well known location or through a special request on the root URI) risks data being collected and used in inappropriate ways, what can a browser do? One solution is to involve a 3rd party, such as a search engine. If a website identifies itself as adhering to a privacy standard in a declarative way (e.g. a link or meta element), then this could be picked up by search engines when indexing the site. This could be exposed in the markup for search results or through a query API provided by search engines. The approach allows for users to restrict searches to sites that adhere to privacy standards. Browsers could store this information as part of bookmarks.

Credential-based access control

Credentials provide an alternative to having to fill out forms. The website provides information as to what kinds of credentials are needed to access a given web page. This information could be provided as part of the HTTP 401 Unauthorized response via the WWW-Authenticate header. The browser would then check to see what credentials it has available, asking the user if there is a choice, and retry the request, this time with the appropriate credentials using the HTTP Authorization header.

The above assumes that the browser, or browser add-on acting as a privacy assistant, provides the user interface for presenting the information on required credentials and privacy policy, for assisting the user in selecting which credentials to disclose, helping with getting additional credentials when needed, and warning the user when his or her preferences are incompatible with the privacy policy provided by the website for the requested web page.

This is not as simple as it sounds, and there are opportunities for using natural language generation to convert policies into plain language that users can readily understand. For instance, a cryptographic credential could be used to selectively reveal some but not all attributes. The user needs to understand when selecting this credential that only the stated attributes will be disclosed.

This process starts with a representation of the policy or communication intent and transforms it into one or more sentences via a sequence of steps. This can be supplemented by graphical icons as appropriate. The number and kinds of icons, and their purpose is an active area of study within the PrimeLife project.

Requesting information on required credentials and privacy policies

It may be worth considering how to request information on required credentials and the proposed privacy policy for a given URI, without requesting the associated resource. One possibility is the [HTTP OPTIONS method](#). This allows for message bodies in the request and response. The "*" request as described in RFC2616 would seem appropriate for querying for the site's generic policy.

Responses to this method are not cacheable. A new link header (Privacy-Policy) could be used to provide a cacheable URI for the policy. You could thus use the HEAD method to obtain the URI for a policy. When requesting the policy, you would use the Referrer header to give the resource URI (the one used for the HEAD request). [RFC3864](#) defines the registration procedure for new headers. However, this link header may be redundant since as explained below, mechanisms for including policies in HTTP headers are valuable for negotiating privacy preferences.

The Link header was proposed in [RFC2068](#) for HTTP 1.1 but removed in [RFC2616](#) due to lack of implementation support. Mark Nottingham covers this in his Internet Draft [draft-nottingham-http-link-header](#). The Link header corresponds to the HTML link element, and provides a means for describing a relationship between two resources, generally between the requested resource and some other resource.

A relevant example could be:

```
Link: <http://www.example.com/Policy>; rel="PrivacyPolicy"
```

The rel value names the semantics of the link and is either a well known name or a URI. The Link header seems well suited to static policies, but may be inappropriate for a negotiation of privacy preferences between a data subject and a data controller, where the disclosed preferences are subject to change in successive rounds.

Note: this paper doesn't define the specific syntax and semantics for credentials and privacy policies, which are subject to ongoing investigation.

Avoiding the need to extend HTTP

An alternative to extending HTTP would be to convey the information on required credentials as part of the web page rather than as part of an HTTP 401 response. In this approach the browser is redirected to a page that asks the user to provide the necessary credentials. For this to work the browser has to be able to recognize what credentials are being asked for, and to provide the corresponding user interface where needed. A privacy assistant implemented as a browser add-on could dynamically extend the page to allow the user to select credentials from drop down menus associated with form fields. This assumes some conventions for binding the fields to the credentials.

Privacy Assistant as a trusted intermediary

End users are often in a position where they aren't really able to make an informed decision. They make like the sound of the services on offer from a website, but not have any clear idea as to the risks. This is where it makes sense to ask an independent expert authority or to ask friends and other people for a group reaction, perhaps based on some kind of reputation based scheme.

Another perspective relating to this is that using a browser add-on as a privacy assistant introduces problems. What happens if a drink spills into the computer, or if it is dropped, lost or stolen? That sounds like an issue of providing effective backups. However, what about accessing the web from multiple devices? I may be traveling and have stepped into an Internet Cafe, or I may be on the train and using a iPhone or similar mobile device. More and more users want to have a presence on the Web that isn't tied to a single machine. Cloud-based services are springing up to address this demand. It therefore makes sense to consider what possibilities a cloud based privacy assistant could offer.

Single sign-on using a trusted intermediary

This is similar to OpenID, except that you only need to disclose the URI for your privacy assistant and not your user id. The web page for the website that you want to use can then direct your browser to the personal assistant for you to sign on. If you have previously done so in the current session, then the assistant can immediately redirect the browser back to the appropriate page on the website you are trying to use. This is passed as a parameter in the requested URI or via the HTTP Referrer header.

An alternative to browser redirection is to use an iframe within the web page for the personal assistant, as the browser will run the iframe in a different security context. This approach is at risk from screen hijacks where a floating div element is placed above the iframe and used

to fool you into supplying your user id and password to the hijacker. The redirection technique by contrast allows you to check the browser location field to see if the URI is correct for your privacy assistant.

The personal assistant can help you to select what identity you use with a given website, e.g. you could pick from a number of identities corresponding to different faces (family, friends, work). It could even provide a temporary id that only lasts for a single session.

The website creates a temporary id that it will use to identify you for the duration of the session. This is passed to your privacy assistant, which uses it to create credentials attesting to attributes of your identity, e.g. that you are over 21 years old and legally entitled to order a crate of wine. The redirection/iframe technique can be used when your personal assistant needs to present you with choices e.g. of which credential to disclose.

Cross-site authorization

The trusted intermediary can also be used when you want to give a service on one website access to your personal data on another site. This starts with your browser pointing to the first site. You are invited to click on a button to authorize access. The browser is directed to the second site, which in turn directs the browser to your privacy assistant where you grant access. The browser is then redirected back to the original page. This process involves your assistant generating one credential to attest that you are entitled to access your personal data on the second site, and generating a second credential authorizing the second site to provide data to the first for a named purpose, but only for a limited duration.

Offline Authorization

We are not always online, and yet we increasingly want to establish our presence on the Web. This is leading to new kinds of cloud-based web run-times that execute applications on our behalf 24x7. A web run-time is an execution platform for running applications defined by a combination of markup, scripts, and other resources. Such applications need ways to establish their rights to access other services, and likewise, to verify the rights of other applications to access the services these applications provide.

Trusted intermediaries can meet this need through policies that express your trust relationships, i.e. who is allowed to access what resources for which purposes. Your web agent takes on your role when signing in with your privacy assistant. The assistant then creates credentials attesting to attributes of your identity and passes them to the service the agent wants to access.

Reality or Illusion?

It is worth playing the role of the devil's advocat and asking if we build this technology, whether developers and users will want to adopt it? Is there a simpler alternative that would meet enough of the needs so that the more advanced approach doesn't take off?

End users focus on the services they want to access and play down the importance of privacy concerns. Is the flexibility offered by negotiable privacy policies really needed? What is the demand for anonymous credentials? Perhaps it is more a question of enabling users to recover from problems when things go wrong? In this view, a strong legal framework that safeguards users' interests is the key to preventing abuses. This still needs a framework for users to verify the accuracy of the personal data that an organization holds on them, and to request corrections as needed. A notification scheme that meets the majority of needs would probably be okay, and avoid the need for users to set their preferences.

Further Reading

to be added before the workshop