2011 15th IEEE International Enterprise Distributed Object Computing Conference

OpenID and the Enterprise: A Model-based Analysis of Single Sign-On Authentication

Jacob Bellamy-McIntyre, Christof Lutteroth, Gerald Weber
Department of Computer Science
University of Auckland
Auckland, New Zealand
jbel071@aucklanduni.ac.nz, {gerald,christof}@cs.auckland.ac.nz

***Abstract*—Single sign-on (SSO) protocols allow one person to use the same login credentials for several organizations. Enterprises face increasing competitive pressure to position themselves with regard to SSO, yet the ramifications of a move to SSO are not fully understood. In this paper we discuss OpenID, a relatively new SSO protocol that is gaining traction on the web. We apply enterprise application modelling techniques to OpenID in order to obtain well-founded decision aids for enterprises: we show how published modelling approaches can be used to analyse risks in OpenID, and show that these can identify security problems with common OpenID practice. Finally, we propose analysis principles that condense important general insights of authentication modelling.**

***Keywords*-SSO; Authentication; OpenID; Models**

I. Introduction

Security has long been seen as an implementation issue that is technology specific. However, in the modern Internet society, where social change influences security relevant user behavior, it is the duty of the enterprise analyst to provide high level guidance on security aspects. Single sign-on (SSO) is a class of protocols or technologies that enable users to take their web identity to a large number of sites that they are working with. For enterprises it becomes a strategic goal to be part of an SSO scheme with other sites in order to satisfy customer demands. Likewise, universities are also beginning to implement SSO for reasons of research facilitation and student satisfaction.

However, IT security is a mission-critical issue for organizations. Negligence can lead to serious consequences for users, and bring on legal ramifications that damage an organization. Hence enterprise decision makers need precise risk assessments in the language of the enterprise. If enterprise modelling wants to live up to its potential, we need a framework that can deliver strong statements on the level of the enterprise model while still being based on an understanding of the technology. The latter means again that we need models, but on a lower, platform-dependent level. A model-driven approach is needed, where models on different levels can be balanced and compared. We believe this to be a more appropriate approach for Enterprise decision makers than heavy formalisms such as BAN logic [1].

In this paper, we address both the big and the small picture. We give an exemplary analysis of OpenID [2], a protocol that was evaluated for deployment at the University of Auckland, and we derive general lessons from this analysis. The OpenID case study serves as a proof by example that model-driven security analysis is possible. It shows that in the security domain, platform-specific models of a technology have a strong impact on the validity of higher-level models of an enterprise because of the existence of common security threats.

We are framing our findings as principles that help in the identification of security risks and clarify the relationship between different modelling layers. Based on these principles we identify existing vulnerabilities in the OpenID protocol which have been discussed elsewhere [3]. In a field study we evaluate how widespread these vulnerabilities are, with the finding that they are fairly common and pose a credible risk if OpenID-based systems become a valuable target. We feed the findings from these fairly technical analyses back into recommendations for decision-makers in an organization, as well as for the community process around OpenID. The result is a balanced advice, as opposed to the excitement of many OpenID proponents or the outright rejection of many detractors.

Section II sums up requirements of SSO, and Section III gives an overview of OpenID. Sections IV and V introduce our modelling approach and discuss formal models of OpenID. Section VI discusses the applicability of the modelling approach to other authentication protocols that are used in enterprises. Section VII discusses how OpenID should be used today, and Section VIII outlines how enterprises could use OpenID in the future.

II. Motivation and Requirements of Single Sign-On

In a society that has become used to the web, each typical web user accumulates a large number of accounts on personalised sites, even if many of these sites are only used sparingly. This is because many sites use a site-centric authentication approach (SCA). An issue that has been observed is *password fatigue* where users cannot remember

1541-7719/11 $26.00 © 2011 IEEE
DOI 10.1109/EDOC.2011.26

enough distinct passwords to cover every account that they create on the web. The management of these accounts becomes a nuisance to users and they may take shortcuts in the management of this wealth of sensitive information at least for non-critical sites [4], in particular by reusing passwords. This in turn may compromise the site's security requirements.

SSO is a possible solution for password fatigue. Intranet-wide SSO is state-of-the-art today and was put forward as a research goal in the nineties. An early application that motivated this research was SSO for digital libraries. Different architectures have been proposed, and for digital libraries proxy-based solutions were mostly used [5]. These early successes have inspired research into similar SSO solutions that work for the entire web.

In the following we reformulate the motivations for SSO as a set of requirements for an SSO protocol, extending and generalizing other discussions on authentication requirements [6]. These requirements are twofold: on the one hand, there are the requirements of the users wishing to authenticate themselves, and on the other hand there are the requirements of the enterprises wishing to authenticate users.

Users require their privacy to be protected, meaning that they should only have to reveal information about themselves if they wish to do so. It should be easy for users to recognize whether it is safe to enter their credentials. Users should not have to state the same information repeatedly, especially when the site they are trying to connect to does not need it. Users also require that the way in which they manage their digital identity is straightforward and simple. Users require a way to move their identity from site to site. Finally, users need their identity to be protected properly so that no one can impersonate them.

Enterprises need a way to deliver their services reliably to as many users as possible. This needs to be done securely without inadvertently sharing confidential information with unauthorised users. Additionally, enterprises need to be able to receive certain guarantees about their users that are confirmed by a trusted party. These are called *verifiable claims*, and include, for example, belonging to the organisation they say they do, and being of the age they say they are. These also include guarantees that the communcation is with an actual human person. If enterprises allow their employees to use OpenIDs, they require that the OpenID is kept secure. If one of their employees uses a technology such as OpenID negligently, the organisation itself can be held responsible.

III. The OpenID Protocol

OpenID is a community standard for Web-wide SSO [7]. It has been in development since 2005, and is currently overseen by a committee comprising both industry and community members. During that time, OpenID has gained support from many enterprises, and a number of very large organisations such as Google, AOL, Microsoft, Yahoo, and Verisign are now OpenID providers [8], [9], [10], [11]. As such it is now feasible for enterprises to rely entirely on OpenID to authenticate users.

The OpenID protocol is complex and only specified textually in a community standard document [7]. This makes it hard to implement and prone to ambiguities, with the result that many existing implementations are partially non-compliant and contain security flaws, as our research has shown. In this paper, we describe a methodology for modelling authentication protocols by applying techniques such as formcharts and UML sequence diagrams. Looking at OpenID, we demonstrate that these models are valuable in several ways: they help to identify and clarify potential risks and possible extensions, and as such can help enterprises to mitigate security problems.

OpenID can be described as being a user-centric rather than a site centric approach to identity management. OpenID allows authentication to be done through an identity provider, which offers a service-oriented interface for authentication. What is unique about OpenID compared to other SSO standards is that the identity provider does not require any prior relationship with the web site or web service for which it is providing authentication. Users choose their own *identity providers* (in this case, OpenID providers) who provide them with a unique URL that represents their identity. They then supply this URL to a site (called a *relying party*) that supports authentication with OpenID. This means that enterprises which are acting as relying parties can leverage the authentication mechanism of other organisations and can reduce the time required for users to register for their services. We begin the modelling of OpenID with a textual use case that describes a typical way in which OpenID is used, before we move on to more formal models.

A user, Bob, decides to use OpenID. Bob must first register an identity with an OpenID identity provider. The user, Bob, picks the University of Auckland as his OpenID provider, due to its strong reputation as an identity provider. He is then given the identity, https://openid.auckland.ac.nz/bob, which he can use on relying party websites. Now Bob decides to create his own online journal and visits livejournal.com. Bob then notices a link saying 'Sign in with OpenID', and decides to use his OpenID to authenticate with the site. He enters https://openid.auckland.ac.nz/bob as his OpenID, and his browser redirects him to a login screen at the University of Auckland, his identity provider. He is then told by the University that the site livejournal.com wishes to authenticate his identity, and is prompted as to whether or not he wishes to allow the authentication. Additionally, he is asked for how long he wishes to trust verification requests from the site. Bob decides to allow the verification, and also states that he will always trust authentication requests from livejournal.com. Livejournal then decides to ask Bob for some basic profile information, called *claims*, to be

used on his journal, such as his name and date of birth. After submitting this profile information, Bob is signed into livejournal.com.

Bob then decides he wishes to ask a programming question on the site stackoverflow.com, and once again sees the option to log in with his OpenID. Because Bob is already signed into his identity provider, this time he is not prompted to enter his login details to the University of Auckland. Instead, he goes straight to the University's confirmation page. Once again Bob agrees to always accept authentication requests from stackoverflow.com. This time there are no claim requests, so Bob has now successfully signed into stackoverflow.com.

The following day, Bob decides to visit the same two sites again. He visits livejournal.com first and signs in with his OpenID. Because he is not yet logged into his OpenID provider, he is prompted to sign in. Once he has signed in, his provider forwards Bob to livejournal with a positive authentication response, and he is successfully signed into the site. He now visits stackoverflow.com and attempts to sign in. Since Bob is already signed into his identity provider and decided to trust all authentication requests from stackoverflow, all he needs to do is to provide his OpenID. The process is simple, removes registration pages, and keeps authentication information far away from new sites that have not yet earned the user's trust.

Even in this small example, there is a lot happening behind the scenes that Bob is unaware of. It takes a considerable effort for an enterprise to understand the protocol and make decisions about its deployment, and using appropriate models makes these tasks easier. In the following sections, we provide two models for the OpenID protocol using different levels of abstraction.

IV. Modelling the SSO Process from a User Perspective

In this section we provide a formal model of the sign-on dialogue from a user perspective. The model obtained will be useful for discussing specific security risks.

In the next section we will describe a model that includes the system communication. The user interface model, since it sits on a higher abstraction level, is not tied to all of the detailed mechanisms of OpenID and can therefore be applied to other SSO protocols. The system communication model, however, is already highly specific to OpenID.

We use form storyboards [12] for modelling OpenID's user interface. Form storyboarding is appropriate for modelling form-based systems, and the OpenID sign-on dialogue has a form-based interface. Form storyboards specify all possible paths that the dialogue can use. The use case above follows several such paths in the form storyboard, but a use case is not a complete specification of all possible paths.

Form storyboards are based on the observation that if a user interacts with a web form, this can be described simply as a form submission followed by a web response. For a full introduction to form storyboards and how they refer to a screen model of the dialogue, see [13]. We will only repeat the formal aspects of form storyboards that are important for structural matching with the system-oriented models, particularly sequence diagrams. In a form storyboard the system is represented as a bipartite state transition machine.

The state of the dialogue alternates between client pages represented as ovals, and server actions represented as rectangles. Client pages show content to the user in a web browser, whereas server actions correspond to information that is sent to and processed by a web server. Arrows mark the transition between states, empty squares represent blank form fields, and full squares represent pre-populated fields. The form storyboard only describes the behavior visible to the user. As mentioned above, this means that it abstracts from a substantial amount of internal system communication. In particular, the transition marked with a 2 in Figure 1 involves a complex communication between the relying party and the identity provider, which is important from an implementation perspective, but not important from the perspective of the user.

This means that two systems satisfying the user interface model in Figure 1 can have completely different implementations. For example, a system could apply some sophisticated technologies such as privacy preserving functions [14], [15] to protect a user's data from the OpenID provider. If such features do not change the user interface model, it is impossible for a user to determine whether an implementation possesses these features, and hence negligent or malicious providers go unnoticed. In the following we analyse OpenID from the user perspective, which is much easier with the user interface model as compared to the significantly larger system model in Figure 3.

A. SCA vs. SSO

Figure 1 shows the form storyboard for the sign-on dialogue with OpenID. For comparison, Figure 2 represents a traditional, SCA process, including the functionality for account registration. Both dialogues may vary a little from site to site for instance some relying parties may also require the user to agree to an end user licence agreement, or require the user to confirm their email address before they can sign in. For OpenID providers we have opted to represent claims being handled by a form with pre-populated fields. Some providers may handle this differently, but this example makes for a closer comparison with the SCA process.

OpenID's sign-on dialogue differs from SCA dialogues in that it always involves two sites: the relying party and the identity provider. The user is redirected between these two sites during the dialogue. For that reason, we have introduced an additional modelling element in this diagram: the pages and server actions executed on each of the two sites are grouped together using large dotted boxes. Transitions

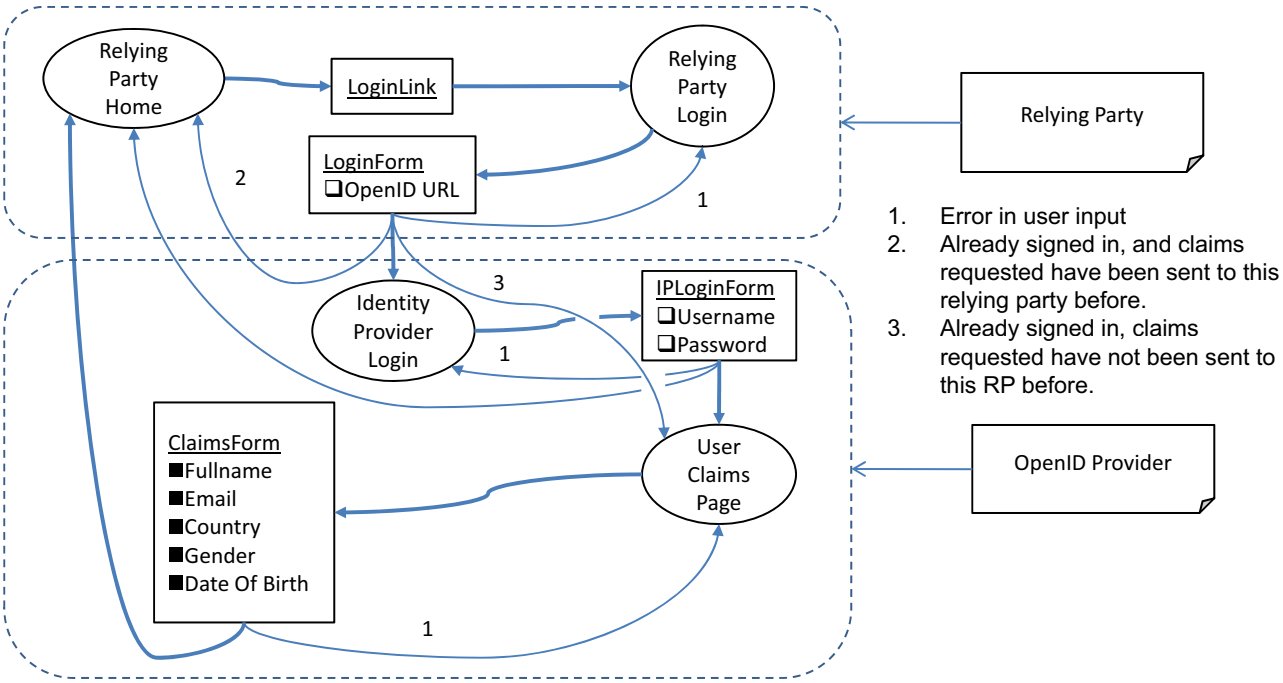

Figure 1. OpenID from the user perspective.

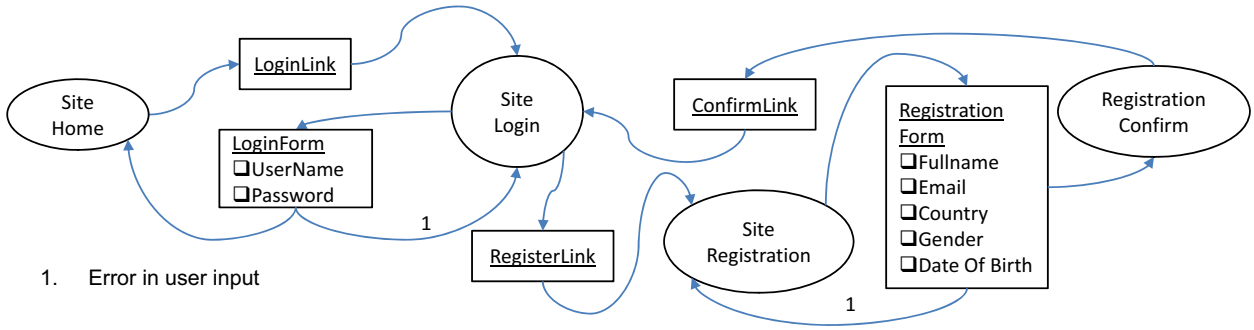

Figure 2. Site-centric authentication with account registration.

between the two boxes mean that the user is redirected from one of the sites to the other.

Usually a user will experience a sudden change to the look-and-feel of the page when they move from one site to another. For many enterprises this is not favourable, as it means that they are unable to include their brand on any of the pages of the OpenID provider during the registration process. For example, this is one cited reason why Yahoo has not become an OpenID relying party [16].

When comparing the form storyboard for normal registration and for OpenID, we can see that the processes shown in both Figures 1 and 2 require at least two server actions to complete authentication. In the case of OpenID, this occurs when the user has already signed into their OpenID provider and no claims are required. OpenID requires at most four and Figure 2 requires at most five actions. On first observation the difference between these seems minimal. Little seems to be gained outside of the registration process. When the user is not already signed into their OpenID provider, logging into a previously-visited relying party site will take three server actions, whereas it would only be two if they had used normal registration.

OpenID's advantage for sign-on is that the user does not provide a password to the relying party, thus reducing the problem of password reuse [17]. The advantage in the registration process is not that it potentially takes one less server action to complete, but rather that personal information fields during registration can be prepopulated, as shown

in the models. Forms can be prepopulated alternatively through the use of cookies and browser plugins, but then the prepopulation will only occur when the user is using a computer where they have the proper cookies and plugins. Additionally, prepopulation of this type can be a privacy concern for those using shared computers. As such, OpenID is a convenient alternative.

One more subtle advantage for users of the OpenID approach is that by engaging with the provider, relying parties must make a conscious decision to request user information. This can be contrasted with many sites based on content management systems which automatically provide claim forms as part of site registration. In a number of scenarios the information requested is needless as it simply fills profile information, which is often unreliable and hence not very useful to the relying party. By having to choose consciously which information to request, those using OpenID are more encouraged to ask for only a minimal amount of information. A number of authors have noted that to protect user privacy there should be minimum exposure of private user information [6], [4], [15], and authentication schemes such as OpenID facilitate this.

B. Phishing

Principle 1: (Phishing) For an attack that results in an identical model from a user perspective, the user is unlikely to be able to distinguish between an attack setup and a genuine SSO setup.

An important consequence is that all SSO protocols where the user is redirected from a relying party to an identity provider are under a phishing threat. The reason is that the use case follows the anti-pattern of following a link provided by an untrusted party. From an enterprise policy standpoint, the fundamental weakness of many SSO schemes can be captured as follows: the organization must allow its members to use the corporate credentials through a foreign website.

For OpenID this means that any attack which preserves the appearance of Figure 1 can elicit the participation of the user without suspicion, e.g. phishing. A phishing attack would be complete once the IPLoginForm action is performed, i.e. once a user's credentials are transmitted.

If a user opts to register then they supply the phishing site with a username and password. Because of password reuse [17] an attack on the user might already be successful at this point. If they attempt to sign in using OpenID, the relying party will instead forward them to an identity login portal that is identical to that of their own provider. Empirical evidence suggests that even experienced Internet users find it difficult to discover a phishing attack once they have arrived at the phishing site [18]. In this scenario, there is a good chance that the user will attempt to authenticate, and the attacker will then know the username and password they use for their identity provider. This means that the attacker will gain access to every site that the user uses that OpenID for. Specific methods can be used to reduce the risk of phishing attacks, e.g. Lee et al. [19] suggest a system using personalised security tokens.

V. Modelling the Single Sign-On Process from a System Perspective

The user model abstracts away from implementation. It is necessary at this point to introduce another model that gives a more in-depth description of the mechanisms described in the OpenID specification. The model we propose to use in this case is a UML sequence diagram, and the result can be seen in Figure 3. The different parties to the communication are modeled as objects along the horizontal axis. We omit type information because every object has its own type. Except for the communication between user and browser, the method call paradigm is HTTP. This means the call is an HTTP request and the return value is an HTTP response. Notes have been added to the far right to explain different points of OpenID's behaviour.

Principle 2: (Model Compatibility) A system model only complies to a form storyboard if the communication between user and browser in the system model matches one or more paths in the form storyboard.

This means that the system model in Figure 3 describes one of the paths that the user interaction can take in the form storyboard. This path is highlighted with fat arrows in Figure 1.

In the following we describe the parts of Figure 3 that are labelled on the right:

1) Delegated Identity: The URL given by the user may only point to their OpenID provider indirectly. The supplied URL may refer to a webpage, which refers to an *OpenID endpoint*, which in turn refers to the OpenID provider. Consequently, users can use any URL that they control as an OpenID without having to host their own OpenID provider. This allows for more personalised OpenIDs, and enables users to migrate between identity providers without changing their OpenIDs.

2) OpenID Discovery: The relying party must have some way of discovering whether the supplied URL is a valid OpenID. When a relying party looks at an OpenID endpoint, they will discover a link to an XRDS document and to the identity provider. The XRDS document describes which services the OpenID provider provides, such as OpenID and its extensions. Retrieving and examining this document confirms to the relying party that this is an OpenID provider, and then they can enter into communication with them.

3) Association Session: After discovering the location of the OpenID provider, the relying party contacts them directly to attempt to establish a shared secret. This is done so that later in the protocol any messages they have received indirectly through the user can be verified. The exact cryptographic procedure here is out of the scope of this paper.

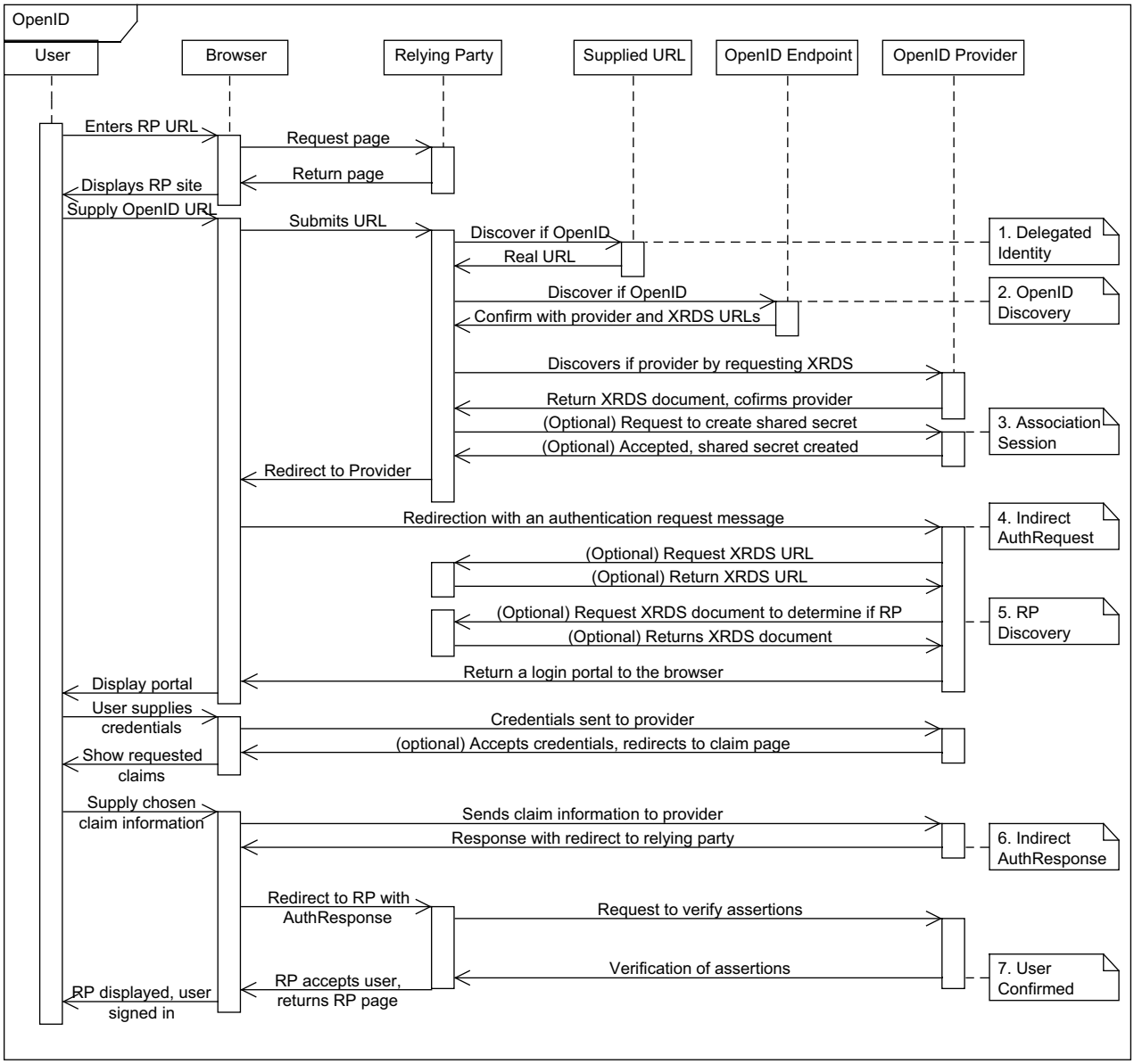

Figure 3. The OpenID process

4) Indirect Authentication Request: When the relying party redirects the user to their OpenID provider, they are also given a message to pass on. This message states that the relying party wishes the user to be authenticated with the exact OpenID the relying party was given. The message also states who the relying party is, and where the user should be returned to after they have been authenticated. It will also include any requests for claims about the user.

5) Relying Party Discovery: Having received this authentication request, the relying party should determine whether or not the URL listed in the authentication request is really a relying party. To do this they perform a discovery procedure by requesting an XRDS document, similarly to OpenID discovery. The document states that the relying party uses OpenID, and confirms where users should be returned to after being authenticated. This procedure is used to mitigate a very specific kind of phishing attack described later in this paper.

6) Indirect Authentication Response and Relying Party Confirmation: Once the user has authenticated with the OpenID provider, and submitted any claim information required, the user forwards a message from the identity

provider to the relying party which contains an authentication confirmation and any requested claims. The relying party needs to make sure that this message has not been changed by the user. Once again the relying party enters into direct communication with the OpenID provider, seeking confirmation about this.

7) Confirmation of User: Once the relying party is satisfied that the received message is the one sent by the OpenID provider, the relying party decides whether or not to accept the user based on the authentication response. They may refuse, e.g. if the user does not supply all the requested claim information. The OpenID authentication process ends once this decision is made.

A. Identifying Potential Threats

Principle 3: (Potential Threats) Every object in the system model is a potential attacker, and every arrow is a potential attack.

In the following we discuss several example attacks in the light of this principle.

1) Malicious Provider: As an example of this principle, let us consider a malicious OpenID provider, who is in the best position to mount a successful attack. The provider can easily store the user's credentials, use them whenever they wish, and record the sites on which they are used. A rogue provider of this kind can easily usurp any identities under their domain without raising any suspicion. Even if the provider does not go so far as to usurp the identity of their users, they can still record and disclose private user information.

Thefts of these kinds from rogue providers are very difficult to detect. However, a parallel can be made between the role of an identity provider and the role of an email provider. Currently, many sites rely on email verification to verify a user's ownership of a given identity. A rogue email provider can usurp the identity of its users by using a user's email account to trigger password resets, but usually not to the same degree as an identity provider. A rogue email provider could also cause gross privacy breaches by disclosing private information obtained from user emails. However, people usually use email providers whom they trust, such as those of their employers or very widely used email providers.

2) Exploiting Redirect: In another attack, a malicious relying party may attempt to phish a user's credentials by pretending to be a relying party trustedrp.com that the user trusts. They can try and gain access to a domain name which looks similar to the one they are trying to copy, e.g. using the URL http://tru*ss*tedrp.com instead of http://trustedrp.com. At this point, instead of displaying a fake login page to the user, they can take advantage of an open URL redirect available on the relying party site they are impersonating. An open redirect is a common site vulnerability that allows redirection to other URLs through a site, by misusing redirection functionality that can be controlled through a URL. For example, the malicious relying party can provide a URL like http://trustedrp.com/redirect?url=http://tru*ss*tedrp.com/ as the URL that the OpenId provider is meant to return the user to.

The provider will see that the domain is one that the user trusts, and so the user is not warned that this is a site they have never visited before. The provider may now automatically send over any claim request information previously approved by the user, or sign the user in. This is a bad state to be in: for example, the user may attempt to use paid services on the phishing site, thinking it is their trusted site.

This particular case shows why accurate modelling and strong descriptions are crucial for those implementing OpenID. The RP discovery step in the system model prevents this attack because the XRDS document will come from http://trustedrp.com and not the phisher's site. This document will describe the correct return URL, which will not have the attacker's redirect.

However, the specification merely states that RP discovery 'should' be implemented, and not why. Because this is not a necessary part of the protocol and because hosts do not understand the motivation, many do not implement it. Consequently, this kind of attack remains a real threat (see Section V-B).

3) Man-In-the-Middle: A man-in-the-middle attack is a generic attack possible at all arrows in the system model. An attacker intercepts messages and possibly modifies them before passing them onto their destination. This risk can be mitigated by encrypting the communication using transport layer security (TLS). Ideally, every part of the OpenID protocol should be protected by TLS, following the Potential Threats principle.

While it is clear to most sites that TLS should be used when accepting user login credentials, it might be less clear that the rest of the protocol should be similarly protected. In particular, a relying party may not realise the importance of RP discovery and fail to use TLS on their XRDS server. This could result in a man-in-the-middle attack that re-opens the phishing threat that RP discovery is meant to mitigate. Even the OpenID URL should be protected by TLS otherwise a man-in-the-middle attack could cause the relying party to redirect the user to a malicious identity provider. Enterprises can use the system model as a checklist for making sure they have protected every arrow with TLS.

B. Empirical analysis of Vulnerabilities in Current Implementations

We have systematically examined a number of relying party sites to determine whether they are protected against these attacks, i.e. whether they host the XRDS document and use TLS. In this study we used only analysis techniques that

Site rank range	Sites tested	RP discoverable	XRDS TLS protected	Login TLS protected
1-1000	4	3	1	1
1001-10000	2	2	0	0
10001-50000	7	5	3	5
50001-100000	5	2	1	1
100001+	14	3	3	5

Table I
VULNERABILITY STATISTICS OF 32 OPENID RELYING PARTIES.

are possible by observing the communication without performing unauthorised activities. We examined 32 sites with varying traffic rankings, listed on myopenid.com's OpenID directory, and discovered that even some very popular sites fail to protect their users properly. We used the Yahoo OpenID provider to find this information, as it notifies when a site fails to implement RP discovery, and is accepted by a number of relying parties that utilise whitelisting. To discover if the XRDS server was protected by TLS, we simply had to examine the HTML code of each site. The data, which is shown in Table I, is organised by Alexa site rankings and aggregated to protect relying parties.

Even though the sample size is small, only eight of these sites fulfilled the three requirements. This seems to indicate a gap between how OpenID should be used and how it is used in practice. Whether or not an OpenID session is secure depends strongly on the implementation. As such, security-conscious modelling as presented in this paper is a useful resource for enterprises looking to adopt OpenID.

C. Guarantees for Relying Parties

Similar statistics can be generated about the security practices of OpenID providers, and given these potential problems one might be sceptical about becoming a relying party. To address this, the relying party may use the Provider Authentication Policy Extension (PAPE) [20], which is an extension of OpenID. The purpose of this extension is to allow the relying party to make a request to the provider as to the level of security that should be used. For example, this can be used to insist that the provider use multi-factor authentication to authenticate users for this relying party. When the user returns the authentication response of the provider, it should also include whether or not it met the requirements of the relying party's PAPE request. The relying party can then use the PAPE response (or lack of it) whether or not to accept the authentication response.

However, this will only mitigate the problem partially. A badly-implemented provider may simply claim they have fulfilled all the PAPE requests so as to have their OpenIDs accepted by a wider range of relying parties. Alternatively, they may falsely think they are fulfilling the requests when actually they are not. For example, when a provider receives a request for multi-factor authentication, they may ask the user for some additional private information like their mother's maiden name. However, this would still be single factor authentication and would not fulfil the PAPE request accurately.

When using OpenID, relying parties usually have to accept that users will use a variety of different identity providers, some of which do not use good security practices, and some of which the relying party will have no knowledge about. As such, the use of OpenID can be contrary to the requirement of an enterprise that their services are delivered securely and that user privacy is preserved. This issue can be addressed through the use of white lists or black lists. By using a black list, a relying party can refuse any provider they know to be untrustworthy. This approach still allows for a great degree of freedom for users in choosing their OpenID provider and is recommended whenever a relying party knows a provider to be untrustworthy.

White listing allows only OpenID providers that the relying party trusts to authenticate its users. This helps to fulfill the requirements of security and privacy. But it also reduces the number of OpenID users who can authenticate with a site, potentially reducing the number of new users. It is advised that relying parties using white listing should explicitly state which providers they trust.

VI. APPLYING SIMILAR MODELLING TO OTHER TECHNOLOGIES

The kinds of models described above can also be applied to other SSO authentication technologies, e.g. federation-based solutions like WS-Federation [21] and Shibboleth [22]. Federated identity is different from OpenID in that the relying party and identity provider form an agreement that the identity provider will authenticate users for the relying party. Enterprises may enter into this agreement as an identity provider for their employees or customers, looking to make use of services provided by the relying party.

In a web-based scenario the user models for WS-Federation and Shibboleth would be almost identical to each other. The user model we have used for OpenID would only have to be changed slightly for it to look the same as one for WS-Federation or Shibboleth. Namely, the user model would not include an OpenID URL, and the relying party would forward the user onto the identity provider on the basis of a pre-existing authentication agreement.

The system models of WS-Federation and Shibboleth would be somewhat similar, but both would be very different from OpenID. In contrast to OpenID, the two protocols use an authentication mechanism based on security tokens. Despite these differences, the system modelling approach proposed in this paper still seems to be applicable. So overall, enterprises considering whether to use WS-Federation or Shibboleth would benefit from applying these techniques, as they make it easier to analyse the implications of using these protocols.

Authorisation protocols like OAuth [23] also lend themselves to our modelling approach. OAuth is an authorisation instead of an authentication protocol, so it addresses a different set of user requirements. It allows users to share private data (such as private RSS feeds) they have stored on one site with another site, without having to share their login credentials. The flow of the OAuth protocol from the user perspective is very similar to that of OpenID, so they have similar user models. The user model of OAuth can be used in either SCA or SSO scenarios. This means that OAuth can be modeled as an extension of OpenID or federated identity. Similar to the federated approaches mentioned above, OAuth makes use of security tokens. Similar to OpenID, OAuth can make use of HTTP redirects. Altogether, these features can be described in a system model similar to the one presented here for OpenID.

VII. When and How Enterprises Should Use OpenID

While OpenID is a useful SSO technology for enterprises to take advantage of, there are situations in which its use is not advisable. This section will detail when enterprises should and should not use OpenID.

OpenID in itself, particularly without provider whitelisting, does not fulfil the requirement stated earlier to ensure that users are who they say they are, and are therefore traceable. However, effective provider whitelisting is not mature currently, as will be explained in the next section.

OpenID is useful whenever an enterprise has a web presence that requires a user to create an account, but where the user cannot do harm to the site. This might include accounts for bug reporting systems, message boards, chat rooms, or job application sites. The risk to the user is low as the loss of these kinds of accounts is not excessively harmful, and because these are the accounts attackers are less interested in. These are also the kinds of accounts which users more frequently have to register for, and are a large source of password fatigue problems. The use of OpenID for these kinds of accounts is beneficial to the user, and if the practice were widespread would help to dramatically reduce the problem of password reuse. Consider an enterprise that offers a standard web-based registration and sign on, e.g. a webshop. Is it safe for the enterprise to switch to accepting OpenIDs? If the user only has customer privileges, and if e.g. the user does not trigger a shipment without payment, then the answer is yes. OpenID would then offer the prospect that it would become more convenient for new users to come to the site.

As the value of the service increases, enterprises should be more strict in their use of OpenID. Social networking sites, auction sites, billing sites, and bank sites are all of much higher value to the user and to attackers. In particular, any site which involves monetary transactions will become potential targets for attacks. OpenID can be used securely in these cases, but it requires both the relying party and identity provider to use good security practices.

One rule of thumb for an enterprise deciding on whether to introduce OpenID authentication is to look at how much trust is placed in the user's email account. If they currently allow for users to reset their password using an arbitrary email account, then the enterprise could also use OpenID. The trust they place in the OpenID provider is equal to that of the email provider in these situations. If enterprises do not allow for email addresses to authenticate in this manner, then they may not wish for OpenID providers to do so either.

In general, relying parties and OpenID providers should follow all recommendations in the OpenID specifications, and not follow the bare minimum required to support the protocol. In particular, section V above described several attacks that can occur when TLS is not used on certain arcs in the system model.

VIII. OpenID and the Enterprise in the Future

OpenID is evolving and is likely to become even more important for enterprises in the future. In this section we point out some of the directions that OpenID is developing towards, and how this will affect enterprises.

Currently, enterprises may be wary of using OpenID due to the threat of identity providers using insecure practices. In the future, there could be high-security identity providers that support verified claims. Enterprises could use OpenID specifically because they wish their users to utilise such high-security identity providers. For example, enterprises may wish to make use of OpenID providers which support multi-factor authentication, i.e. authentication based on additional types of credentials such as biometric security. This would allow relying parties to utilise high security authentication without having to invest in developing and integrating such mechanisms into their own systems. Relying parties could restrict the OpenIDs they accept to such providers, either if there exists an independent accreditation process for providers, or by keeping a whitelist on their own. A valid concern is that private whitelisting can lead to a monopoly of identity providers, which can in turn affect users as well as relying parties. Lastly, OpenID is in strong need of standardisation.

The OpenID specification uses the word "SHOULD" forty-eight times in describing how the OpenID protocol operates. Many of these should be changed to "MUST". Each "SHOULD" represents a point in the protocol which can be ignored in an implementation while still being compliant to the specification as a whole. This might be intended to be flexible, but the problem that results is that those using OpenID must somehow communicate with each other despite large differences in configuration [24]. One example of this is discovery. The protocol states that relying parties SHOULD use Yadis discovery, but if they do not then they can use HTML-based discovery. Those who

wish to implement the protocol well will have to support both, needlessly complicating the implementation. Those who only pick one will run the risk of discovery failing as the other party may not support both forms of discovery. This would not be an issue if the specification outlined a single compulsory method for discovery.

IX. Conclusion

SSO has the potential to make authentication a service, facilitating a cross-organisational service-oriented architecture. Currently many enterprises are facing the decision to deploy SSO, but it is hard for them to understand the full ramifications. Suitable modelling approaches can mitigate this problem, and help an enterprise to make informed decisions about SSO technologies. In this paper, we have presented

- a *general modelling approach for SSO*, on two levels of abstraction: a user interface model and a system model;
- more specifically, *models of OpenID*, which is one of the most important SSO protocols at the moment;
- model-based *analysis principles* that can be applied to assess security risks and model validity, with examples of how they apply to OpenID;
- and *findings about OpenID security problems* in many current implementations, motivating the need for better methodologies to help enterprises deal with such technologies.

SSO technologies such as OpenID will become more and more important in the future. They need time to evolve and establish themselves in the enterprise world. Modelling approaches play an important role in facilitating this process.

References

[1] M. Burrows, M. Abadi, and R. M. Needham, "A logic of authentication," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 426, no. 1871, pp. 233 –271, Dec. 1989.

[2] D. Recordon and D. Reed, "OpenID 2.0: a platform for user-centric identity management," in *Proceedings of theSsecond ACM Workshop on Digital Identity management*, Alexandria, Virginia, USA, 2006, pp. 11–16.

[3] B. van Delft and M. Oostdijk, "A security analysis of OpenID," in *IFIP Advances in Information and Communication Technology*. Oslo, Norway: Springer, Nov. 2010, pp. 73–84.

[4] A. Jøsang, M. A. Zomai, and S. Suriadi, "Usability and privacy in identity management architectures," in *Proceedings of the Fifth Australasian Symposium on ACSW Frontiers - Volume 68*. Ballarat, Australia: Australian Computer Society, Inc., 2007, pp. 143–152.

[5] D. Groenewegen and S. Huggard, "The answer to all our problems? trialling a library portal," *Library Review*, vol. 52, no. 9, pp. 452–459, 2003.

[6] K. Cameron, "The laws of identity," May 2005.

[7] B. Ferg *et al.*, "OpenID authentication 2.0 - final," Dec. 2007.

[8] E. Sachs, "Google moves towards single sign-on with OpenID," Oct. 2008. [Online]. Available: http://googlecode.blogspot.com/2008/10/google-moves-towards-single-sign-on.html

[9] D. Roy, "Yahoo! announces support for OpenID," Mar. 2008.

[10] M. Graves, "Versign, microsoft & partners to work together on OpenID + cardspace," Feb. 2007. [Online]. Available: http://blogs.verisign.com/infrablog/2007/02/verisign_microsoft-_partners_to_1.php

[11] J. Panzer, "AOL and 63 million OpenIDs," Feb. 2007. [Online]. Available: http://dev.aol.com/aol-and-63-million-openids

[12] D. Draheim, D. G. Weber, and G. Weber, *Form-oriented analysis: a new methodology to model form-based applications*. Springer, 2005.

[13] D. Draheim and G. Weber, "Modelling Form-Based Interfaces with Bipartite State Machines," *Journal Interacting with Computers*, vol. 17, no. 2, pp. 207–228, 2005.

[14] T. Nakamura, S. Inenaga, D. Ikeda, K. Baba, and H. Yasuura, "Anonymous authentication systems based on private information retrieval," in *NDT '09*, 2009, pp. 53–58.

[15] G. Elahi, Z. Lieber, and E. Yu, "Trade-off analysis of identity management systems with an untrusted identity provider," in *COMPSAC '08*, 2008, pp. 661–666.

[16] A. Tom, "What yahoo wants from OPs," 2010 OpenID Technology Summit West, Apr. 2010.

[17] B. Ives, K. R. Walsh, and H. Schneider, "The domino effect of password reuse," *Communications of the ACM*, vol. 47, pp. 75–78, Apr. 2004.

[18] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in *SIGCHI06*, ser. CHI '06. NY, USA: ACM, 2006, pp. 581–590.

[19] H. Lee, I. Jeun, K. Chun, and J. Song, "A new anti-phishing method in OpenID," in *SECURWARE '08.*, 2008, pp. 243–247.

[20] D. Recordon, M. Jones, J. Bufu, J. Daugherty, and N. Sakimura, "OpenID provider authentication policy extension 1.0," Dec. 2008.

[21] H. Lockhart *et al.*, "Web services federation language (WS-Federation)," Dec. 2006.

[22] R. L. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein, "Federated security: The shibboleth approach," *EDUCAUSE Quarterly*, vol. 27, no. 4, pp. 12–17, 2004.

[23] "OAuth core 1.0 revision a," Jun. 2009.

[24] B. Ellin, "Challenges faced implementing OpenID in RPX," OpenID Technology Summit West, Apr. 2010.