A trusted identity management system is not only essential for ensuring your customers’ safety and confidence.

With the new EU legislation, high protection of personal data has become a pressing need.
Exploding user identification

A trusted identity management system is not only essential for ensuring your customers’ safety and confidence. With the new EU legislation, high protection of personal data has become a pressing need.

Ebay, Amazon, Gmail, Facebook, Instagram, the bank, the doctor, discussion groups, online store this and online store that – the quantity of websites requiring user identification has exploded in recent years, and with this the number of passwords and the amount of personal data in the hands of others.

Simple and trustworthy systems

From a user’s perspective, it seems an almost impossible task to keep track of passwords and the whereabouts of data, often resulting in unsafe reuse of passwords for some or in general resistance against and possible withdrawal from your website for others.

As a website administrator you want simple and trustworthy systems. However, protecting the identity and data you keep from users is not only a matter of confidence. As the EU is about to introduce stricter data protection laws, placing greater demand on the handling of personal data and increasing fines significantly, it is also a matter of economic risk.

A great number of technological choices

If you get confused by the words SAML, Kerberos, WAYF, NemID, OAuth, OpenID, OpenID Connect or UMA, you are certainly not alone, and they are just a few of the acronyms of systems and protocols used to handle identity management on the web.

Furthermore, before deciding which technology to adopt, it is important to know and understand the factors that may influence this decision.

This guide presents a few of those factors and provides an overview of some of the latest technologies currently available: OpenID, OAuth, OpenID Connect and UMA.

These are non-commercial open standards, designed with the aim of making life easier and safer for users and developers at large to be freely used by anyone.

All of them have been developed with input from major Internet companies and research institutions, such as Google, Microsoft, AOL or Yahoo!, and have been adopted by major social services also including Facebook, Twitter or LinkedIn as providers. Furthermore, they are – or are expected to be – supported by several Content Management Systems (e.g. Wordpress, Joomla, Drupal, and Symphony) and by the Apache HTTP Server, to facilitate their integration.

Main objective: To limit user passwords

The main objective of the above technologies is to limit the number of user passwords by authenticating the user through an Identity Provider (for example Facebook, Instagram or Google). Thus, instead of proving the knowledge of a username and a password, the user proves access to an account elsewhere.

---

Authentication

**Authentication** is the process of proving who you are, that could be with a username and password, but also with your fingerprint, a digital certificate, or an identity token. Or it could be a proof that you have identified yourself towards a third party like Facebook or Google.

Access authorisation of user data

Another objective is to get authorization of access to a user’s data. This could be address, e-mail, birthday, list of friends, photos and so on. The advantages of this are several: The user is in control of his own data and only has a few places to update data. The advantage for the website provider is that data is usually up to date, and the provider does not have to deal with storing and maintaining user data.

Authorisation

**Authorisation** is the process of determining the level of access to a secured resource. For instance giving a photo printing shop read-only access to the exact photos you want to print.

---

2 We use Facebook and Flickr as an illustrative example. We might as well have chosen other known services.
OpenID was introduced in 2005 to reuse identities and has its emphasis on authentication. It has been adopted by many of the main social services and is still widely used, even though it is no longer recommended due to a wide range of security issues. Its latest version OpenID2.0 is now considered obsolete.

The OAuth mechanism was introduced to target authorization rather than authentication. Still it can be used for authentication and as such solves the security issues of OpenID. Despite being controversial in many senses, it is widely deployed, and thus OAuth 2.0 will be the recommended protocol for most applications at the moment.

Recently OpenID Connect has been defined as providing an authentication layer on top of OAuth. OpenID Connect offers a detailed standard that provides secure and compatible solutions among different identity providers. Furthermore, it improves the user experience by requiring the user's consent before any exchange of information as well as providing for an easy way to adopt new services. As OpenID Connect is a very recent specification, it still has to prove its full potential on a real scale. A few commercial providers are offering this protocol, the most remarkable being Google and PayPal.

UMA is an upcoming protocol extending the concepts behind OAuth and OpenID Connect by providing users with a centralized authorization management console from which to protect their cloud resources. This allows users to authorize controlled access to whomever they wish during the period of time that they decide.
The goal is to facilitate the provision of security for the user

In summary, the goal of the web-based Identity Management technologies is to make it easy for client implementers to secure and facilitate the life of the end users. This is done by:

- Providing access management mechanisms that allow for user-controlled share of resources among different providers.
- Facilitating the integration of secure authentication and authorization systems.
- Minimizing the number of passwords to be remembered by users in order to access their resources in a secured way, thereby also minimizing the risk of password reuse.

Want to know more?

For a more detailed and technical description, we refer to the document "Comparative analysis - Web-based Identity Management Systems". This is still a work in progress, and the latest public version can be found on http://www.alexandra.dk/dk/labs/Downloads/IT-sikkerhed/Authorization_Systems_Comparative_AI.pdf

Additionally, the Alexandra Institute has created a demo showing the potential of these technologies:
http://demo.rkid.alexandra.dk/oauth2-demo-php/web/
Factors to consider when choosing your protocol

In the process of selecting the Identification and Access management protocol most suitable for your business, several parameters have to be considered. We will give a short introduction to the factors we find most relevant and provide a comparison table (table 3) summarizing the current status of the protocols on each of these factors.

A more detailed description of each protocol, with examples, is given at the end of the document.

**Standard and deployment status**

As some of the technologies we present are still evolving, it is therefore important to take into account the development status of the protocol as well as how widely it is deployed.

**Maturity of the standard – need for stability?**

You have to consider whether to go for a protocol with a stable version which is not likely to change in the near future, or whether you would rather use a newer protocol that solves some of the issues in the older protocols, but is still under development.

**Complete specification – flexible integration?**

An open standard leaves a lot of decisions to the Identity Providers implementing the protocol. This facilitates their provision, but also results in the risk of security flaws as well as incompatibility issues, which tends to bind clients to specific Identity Provider implementations. In order to avoid being bound to a specific Identity Provider, client implementers can use available open source libraries to achieve more flexible solutions.

**Level of deployment – how many potential users?**

If you are not targeting a well-defined group of people, it might be of interest to look at how widely the protocol is deployed, ensuring that most of your potential users are already registered at an Identity Provider using the protocol you choose.

**Protocol integration**

One of the easiest ways for implementers to use these protocols is via their integration with other components supporting the protocol.

For web designers it can be done via the Content Management System (CMS) they use to implement their website.

For web server implementers, there are Apache modules for OAuth and OpenID Connect, with UMA on its way.

**Usage scenarios**

Do you have special desires or limitations on your application? We consider the following usage scenarios: RESTful architectures, Web Service Clients, Mobile Applications, Single Sign-On and Dynamic Identity Federation.

**RESTful architectures – need for scalability?**

A system with a RESTful architecture facilitates efficiency, scalability and ease of extension. Without going into details, a RESTful architecture can be described as a Client-Server setup with a stateless server, a cacheable client and a uniform interface enabling each part to evolve independently. Furthermore, it must be a layered system, meaning that the client cannot tell
whether it is connected directly to the end server or to an intermediary. Optionally it might make use of code on demand, for example client-side scripting.

Web service clients – non browser-based clients?
All the protocols presented are designed to be used from an Internet browser and do not support web service clients in their original form – however, several of them can be adapted to support web services.

Mobile applications – mobile clients?
As mobile applications have limitations on processing power and security features, one should consider whether the protocol has defined specific flows for these environments when choosing the protocol for such an application.

Single Sign-On capabilities – easy reuse of authentication?
Single Sign-On (SSO) enables the user to get access to several services, without requiring renewed login, whenever he has logged on to one service. If your users are likely to use other services through the same identity provider, this will ease their use of your service.

Dynamic Identity Federation – dynamic integration?
Dynamic Identity Federation is the ability to easily extend your application to integrate with new identity providers. While this is an easy way of reaching more users, the security issues this imposes should also be considered.

Security features
What context is your website working in, and what is most important to you - the privacy of your users or your confidence in who they are?
These are some of the fundamental security features to take into account when adopting an authentication/authorization technology.

Security approach – personal or organizational context?
Is your application service aimed at independent individuals or at users within an organization? We distinguish between two different security approaches: Organizational and user-centric.
In the user centric approach, the personal information of the user is managed by herself, according to her interests, and is not necessarily verified by any external party. The information is stored on the resource server.
This model is supported by all of the technologies presented in this guide.
In the organizational approach, the resource server keeps information from an organization, and the organization keeps hold of verified personal information used to identify its staff.
Users are given controlled access to the resources of the system according to their role.
This approach is only supported by UMA among the technologies presented in this document.

Data management – degree of user control
Are your customers protective about their personal information?
Are you concerned about how to handle personal data in a secure and lawful way? Do you really need to keep personal data in your systems? There are many reasons for letting users be
in control of their own data and just control their authentication at the client site. The degree of control that a user has over the information being transferred among different sites varies substantially between the protocols.

**Authentication level** – how reliable is an identity?

For some applications, it might be important to know how reliable the identity of the user is, in which case the ability of the protocol to provide this information is important.

As an example of such authentication levels, the Stork project, which aims at establishing a pan-European Identity Management System, defines four quality levels of authentication.

The level depends on several parameters, including how the identity is registered – if there has been physical appearance or other means of controlling that the person is who he claims to be, who has been validating it – Facebook or a government accredited identity provider, and which authentication protocol is used – how robust is it.

**Identity anonymity** – degree of user privacy

From a user’s point of view, the level of privacy and/or anonymity that the protocol offers might be the turning point of whether or not to use the application. It is therefore relevant to consider features such as:

- the level of control that the user has over the data that is transferred from a resource server to the client application
- at which level authentication and resource servers can trace the operations done by a user
- whether the authentication can be done without revealing the identity of the user.

**SAML 2.0** is an XML-based standard for exchanging authentication and authorization data. SAML 2.0 is widely used within enterprise contexts, and most commercial identity management software supports SAML 2.0. Furthermore, non-enterprise setups exist, with NemLog-in as one of them. However, its use does not fit well in personal contexts, since it was not designed as a privacy-enhancing protocol and since the use of XML is quite inefficient. Also, the adoption of the protocol is rather

**NemLog-in** is a SAML 2.0 Identity Provider sanctioned by the Danish government and used by many eGovernment portals. It is implemented as a web-only solution and uses the Danish national PKI (NemID) as authentication mechanism. As such, it supports most of the standard features of a SAML 2.0 federation, including Single Sign-On and Single Sign-Off.

**The ABC Technology** is based on the use of Attribute-Based Credentials. The aim is to enhance the privacy of the user by giving him control over the information to be revealed from the issued credential, allowing him to unveil only the minimum information required by the Client without needing an online interaction with the Identity Provider. The main commercial providers of this technology are U-Prove and Idemix.
Protocol comparison

The various protocols and how they perform on the parameters just described are summarized in the following table:

<table>
<thead>
<tr>
<th>Theme</th>
<th>Feature</th>
<th>OpenID 2.0</th>
<th>OAuth 2.0</th>
<th>OpenID Connect 1.0</th>
<th>UMA 1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STANDARD STATUS</strong></td>
<td>Stable version</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Complete specification</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes - but unfinished</td>
</tr>
<tr>
<td></td>
<td>Current deployment</td>
<td>Broadly spread</td>
<td>Broadly spread</td>
<td>Few sites</td>
<td>Experimental</td>
</tr>
<tr>
<td><strong>PROTOCOL INTEGRATION</strong></td>
<td>Apache HTTP server</td>
<td>Yes, Wordpress, Joomla, Drupal, Symphony, DotNetNuke a.o.</td>
<td>Yes, Wordpress, Joomla, Drupal, Symphony, a.o.</td>
<td>Yes</td>
<td>On it’s way</td>
</tr>
<tr>
<td></td>
<td>CMS integration</td>
<td></td>
<td></td>
<td></td>
<td>Drupal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>None found</td>
</tr>
<tr>
<td><strong>USAGE SCENARIOS</strong></td>
<td>RESTful client</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Web service</td>
<td>Can be adapted</td>
<td>Can be adapted 1.0a</td>
<td>Can be adapted</td>
<td>Not defined yet</td>
</tr>
<tr>
<td></td>
<td>Mobile app’s</td>
<td>No specific support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Single sign-on</td>
<td>Not recommended</td>
<td>Yes - if same provider</td>
<td>Yes - if same provider</td>
<td>Yes - if same provider</td>
</tr>
<tr>
<td></td>
<td>Dynamic federation</td>
<td>Yes - but insecure</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>SECURITY FEATURES</strong></td>
<td>User centric</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Organizational</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Data handling</td>
<td>Low</td>
<td>Unspecified</td>
<td>User consent</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>Authentication level</td>
<td>Future</td>
<td>Future</td>
<td>Future</td>
<td>Future</td>
</tr>
<tr>
<td></td>
<td>Overall security</td>
<td>Insecure</td>
<td>Implementation dependent</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
OpenID – fading out

Handles authentication

The OpenID protocol\(^3\) handles the case where a user wants to authenticate towards a relying party, for instance a discussion group, using an identity already created with an OpenID Provider. Thus, the user can show that he is who he claims to be because he can identify himself to the OpenID Provider.

Access to a site can be achieved in three main steps, which are also shown in the example below. In the first step, the user selects which provider to use. The second step handles the authentication of the user towards the OpenID Provider, and in the last step the user is allowed access to the site.

Implemented by a wide range of social media

The authentication protocol was successfully developed in 2005, and OpenID protocols have been implemented by a wide range of social media and others, including Google, Yahoo!, PayPal, BBC, AOL, LiveJournal, MySpace, IBM, Steam, Sherdog, Orange and VeriSign.

The protocol significantly simplified the lives of web developers and users, providing web developers with a simple and external Sign-On solution that could take care of user authentication while minimizing the effort of registration and password handling for the users. It was widely adopted very quickly, and by December 2009 there were over 9 million websites using OpenID\(^4\).

Flexible but insecure

The OpenID protocol fulfils the requirements of a RESTful architecture; it can be adapted to be used in web services and is supported by the Apache HTTP Server, as well as by several Content Management Systems, including Wordpess, Joomla, Drupal, Symphony and DotNetNuke. Although it allows for Single-Sign-On and Dynamic Identity Federation, the solutions present a few security vulnerabilities, particularly in terms of phishing attacks, privacy issues and lack of enforcement of basic security measures.

OpenID provides no specific support for mobile apps, but does not place any special requirements on them either.

Notes on specification

OpenID is an open standard. Its specifications are developed by the non-profit OpenID Foundation incorporated in the US, and the list of contributors includes AOL, Google, Microsoft, PayPal, PingIdentity, Symantec, Verizon and Yahoo!.

Today OpenID is still widely used; however, newer more secure protocols are rapidly taking over, and the latest specification of OpenID, namely OpenID Authentication 2.0, is now considered obsolete.

\(^3\) “OpenID Authentication 2.0 – Final”, specs@OpenID.net, 5 December 2007, http://OpenID.net/specs/OpenID-authentication-2_0.html

Example of a user who uses the OpenID protocol to log in at Meta Stack Overflow. He chooses or enters the OpenID Provider he wishes to use.

He signs in at the chosen provider – here MyOpenID.
And consents to provide the information to Meta Stack Overflow provided by the MyOpenID Provider.

Finally Meta Stack Overflow grants him access to their site.
OAuth – widely adopted

Handles Authorization

OAuth is an open standard\(^5\) maintained by the OAuth Community (oauth.net). It was originally started by a small group of people who needed an open standard for tackling the problem of giving controlled access of private resources initially stored on one site (the Resource Server) to another site (the Client)\(^6\).

While the goal of OpenID is to use a single identity to sign into many sites, OAuth has the goal of granting access to resources without exposing more information to the Client than needed, in some cases not even the identity of the user.

That is, OAuth handles the case where an application (the Client) wants access to some non-public data owned by the Resource Owner and stored on a Resource Server. Access is authorized by an Authorization Server, which in many cases is the same as the Resource Server.

Specification issues

The standard that defines this protocol is very open, leaving many decisions to the actual implementation. While this makes it easy for the server implementers to integrate with their own components and make an easy interface for client implementers, it leads to incompatibility issues locking clients to the implementations of particular Identity Providers.

Furthermore, implementers of the OAuth protocol have to make a lot of decisions themselves, some of which might lead to security flaws, and, in many cases, not providing the user with a choice of which information to reveal (see the example below).

Adopted by most important social services

Nevertheless, the OAuth protocol is already widely used and deployed on the web\(^7\) and has been adopted by the most important social services, such as Google, Twitter, Facebook, Yahoo and LinkedIn. In October 2012, version 2.0 of the protocol was published, and since this caused some controversy issues, some are still relying on their 1.0 implementation of the protocol (e.g. Yahoo).

Features

To solve some of the security issues in OpenID, OAuth requires Clients and Resource Servers to pre-register, thus preventing an OAuth Identity Provider from dynamically adopting new services.

As for OpenID, OAuth is supported by Apache HTTP Server to facilitate the development of client applications. This is also the case for Wordpress, Joomla, Drupal, Symphony and many other Content Providers.

A special flow is designed to allow for secure implementations of mobile applications. Many OAuth Identity Providers (such as Facebook, Twitter and Google) incorporate Single Sign-On

for Clients using the same OAuth Provider, and this protocol provides a much better protection than OpenID.

**Contributors**

In the very early stages of the project, Google joined in as a stakeholder, and the list of contributors includes MIT, Mitre, Salesforce, PingIdentity, AOL, Yahoo, Microsoft, Motorola, Oracle, Salesforce, RedHat, Adobe, GMX, Covata and Telstra.

**Example with Facebook Connect implementation of OAuth**

Any Person wants to log in at Slashdot.org.
Any Person chooses to login using Facebook as Identity Provider and Authorization Server.

Facebook asks for permission to give Slashdot Login access to the public profile, the friends list and the email address of the user – the user has no choice but to give access to the friends list.
Facebook asks whether the user will allow Slashdot to be able to post to Facebook for her – note that this step can be skipped without consequences for the use of Slashdot.

Now the user is logged in with her public profile fetched from Facebook.
OpenID Connect – natural authentication layer on top of OAuth

Authentication on top of OAuth

The OpenID Connect specifications are developed by the OpenID Foundation (openid.net/connect), which is also behind OpenID. It is introduced as an authentication layer on top of OAuth 2.0, merging the experiences of OpenID into OAuth.

Its final specifications were released on February 25th 2014⁸ and it is thus very recent and still has to prove its full potential on a real scale. However, a few commercial identity providers are already offering this protocol, the most remarkable being Google, Microsoft, Deutsche Telekom and PayPal.

Secure and complete standard

This is certainly a standard to be considered, because it extends the OAuth 2.0 protocol very naturally with authentication mechanisms, resulting in a more powerful and secure protocol that also allows for dynamic registration of Clients and Resource Servers.

The completeness of the standard solves the incompatibility issues and risk of security flaws that OAuth 2.0 suffered from, whereas the dynamic registration makes it easy for users to include Clients or Resource Servers in their list of trusted partners. Furthermore, the standard provides the user with a choice of which data to allow access to.

Initial support for Client implementations

As with OpenID and OAuth, there is already an Apache HTTP Server module available for OpenID Connect, whereas it seems that in the CMS world only Drupal supports it at the time of writing.

⁸⁸OpenID Connect Core 1.0", specs@OpenID.net, 25 February 2014, http://openid.net/specs/openid-connect-core-1_0.html
Example of dynamic registration

The main difference between OpenID Connect and OAuth 2.0 is the TOFU login (Trust On First Use), which allows for the dynamic registration of Clients.

The first time a user is using a given website, after having selected the Identity Provider, she will be asked to approve the new site and choose which data the site should have access to.
UMA – Adding third party delegation

Central management of cloud resources

User-Managed Access (UMA) is a project from the Kantara Initiative that extends the OAuth and OpenID Connect protocols to allow a user to centrally manage the control of his own resources placed on different cloud services. This is achieved by letting him to specify what, who, when and how does he wish to allow access to.

Evolving standard

The UMA specifications are currently evolving and being drafted with the aim of becoming a recognised standard in the future. For the moment there exist only a few implementation demos of the protocol by the Fraunhofer Institute, Cloud Identity, Gluu or the SMART project. An Apache HTTP Server module is currently on its way, whereas for the moment there seems to be no CMS support for this protocol.

This is a protocol with great potential to empower users to manage and control their web resources from a central service. But it still has to evolve and mature to become a practical reality.

UMA defines three phases to provide for controlled access to a resource:

- Protect a resource: The Resource Owner decides to protect a Resource – which is identified by an URI address – through an Authorization Manager by providing some access policies.
- Get authorization: The Relying Party – which does not have to be the same entity as the Resource Owner – requests for authorization to access the resource.
- Access a resource: The Relying Party can get access to the resource.

UMA – Resource protection

1. Alice has protected her pictures. Now she wants to give read access to Bob from 12h-14h

2. Flickr sends the resource url of the pictures that have to be protected to Facebook

3. Facebook registers the security policy linked to the pictures and sends an authorisation url to Flickr

10 This illustrative example is only provided as a theoretical approach. Facebook & Flicker have not implemented the UMA protocol.
UMA – Authorisation and access

1. Bob requests webshop to print photos from Alice stored on Flickr

2. The webshop requests access to the data stored at the resource url

3. Flickr sends the authorisation url to the webshop

4. The webshop uses the authorisation url to request authorisation from Facebook

5. Once Bob logs in at the right time, Facebook sends an access token to the webshop

6. The webshop uses the access token to retrieve the chosen pictures from Flickr
The Alexandra Institute is a privately owned, non-profit company that works with applied IT research, development and innovation with the aim of shaping growth and development in Denmark.