Building an authentication system under strict real-world constraints

Jan Schejbal

Abstract

This article presents thoughts and concepts for building an authentication system under the constraints of a real-world scenario. It gives an overview and comparison of the various ways such a system can be constructed and the advantages and drawbacks of the different methods. A part of this proposal shows a way to allow reliable authentication without revealing the full identity of the participants to the authentication system. The contents of this article are based on an actual scenario and the system described will be proposed for actual deployment. The proposed system enables users to authenticate to third parties which are neither known to nor trusted by the organization operating the system. Furthermore, it protects the privacy of the users and allows reliable identification of members of the deploying organization without requiring direct access to the membership database.

1 Introduction

In the past, authentication realms did seldomly span multiple organizations. Today, web applications make it necessary to allow users to authenticate to third parties, often unknown to the organization operating the authentication system.

In this case, we attempt to build an authentication system for a small political party in Germany. The system is supposed to allow members to prove their membership status to services operated by third parties unknown to the operators of the authentication system, and to provide single sign-on both for internal services and services operated by third parties. Ideally, third parties should receive only the data necessary to verify certain access rights, for example the information that the authenticated person is a member, without revealing any additional information.

Most of the data involved in the authentication process (including the fact that a certain person is a member) are highly sensitive. To protect membership information, the membership database is kept on a separate, closed system with very strict access control. The department responsible for implementing the authentication system does not have a direct need to know and hence has no access to that system.

A pseudonymous solution, where only minimal information about users is known to the authentication system, has to be considered.

As the organization implementing the system is still relatively small and relies mostly on volunteers, both manpower and financial resources are limited, but highly skilled and dedicated IT volunteers are available. The design of the system needs to reflect that.

The central membership database contains for each member (among other things) the membership number (unique ID), the name, the postal address, an e-mail address, and information about membership in the regional and local chapters. Using the postal address for verification purposes is not possible due to the prohibitive cost associated with mailing each member.

For privacy reaons, any registration or account creation in the authentication system needs to be voluntary and initiated by the user (opt-in). Simply creating an account for every member and sending an initial password to the registered e-mail address is therefore not an option.

In building an authentication system that fits these constraints, two separate problems need to be solved: First, the identity of users wishing to participate in the system needs to be verified. Then, secure authentication to the third parties has to be handled. As the identity checking step is very specific to the organization, a custom solution is required. For the authentication step, widely used and accepted protocols allow third parties to easily implement the authentication in their applications. Using a (perhaps modified) single-sign-on or decentralized authentication solution is a good way to achieve this.

The following section presents a possible approach to the identity checking problem. In the next section, the authentication protocols that could be used for the authentication step are listed, analyzed and compared. After a short section giving an overview of possible further research topics, a conclusion summarizes the findings in this work and proposes a design for the authentication system.

2 Identity checking

As explained in the introduction, on-line verification of user identities is not possible. Personal verification of identities, either trough trusted volunteers or based on a web of trust similar to the one used by the CAcert certification authority [1] requires too much effort and is not required for the desired security level.

The easiest way to verify users is using the e-mail addresses stored in the membership database. One way of doing this would be generating a random unique token for each member and e-mailing it to the registered address. Knowledge of this token could then be used as a proof of ownership of that e-mail address and thus as a proof of being the member with which that address is associated.

Another way would be to require users to enter their membership ID and e-mail-address into a registration form, *then* e-mailing an confirmation link containing a token to that address, and checking the address against the membership database after the user proved his ownership by clicking the confirmation link.

The first method requires the use of all e-mail addresses from the membership database and would mean that everyone would get a token. As this would affect even members who did not opt in, this could lead to some controversy. For this reason, this could probaly only be done together with a regular mailing (like an invitation to the annual party convention). This method would allow simple user verification by entering the token. This would automatically identify the user and could then serve to trigger the export of the required data about that member from the membership database to the authentification system.

The second method avoids any usage of membership data before opt-in, but poses several other risks: Entering an e-mail address of another person would cause that person to get a verification mail. Altough this risk exists in most on-line registration systems, it still may be wise to avoid it. This could be done using a multi-step approach, i.e. requiring the user to enter some additional data which is available in the membership database but not to the general public and checking those against the database before sending out the confirmation mail - then exporting the membership data after the e-mail address was verified using the confirmation link. This would cause additional delay and workload, as each operation on the membership database needs to be performed by a person with appropriate access rights. An additional issue would be members forgetting which e-mail address is their membership address: If a different address is used,

membership verification will fail.

For the reasons mentioned above, the first method is preferable if sending tokens to all users can be accepted. If this is not possible, the second method can be used to still perform a secure identity verification.

The first method would also allow the creation of pseudonymous accounts, while maintaining the limit of one account per member and allowing to disable accounts when the member leaves the organization: For this, the list of tokens (or hashes of the tokens) combined with the corresponding information about the chapters in which the token owner is a member could be sent to the authentication system operators. As this list contains no personally identifying information, it could be provided before the user presents the token, which would ensure that user registrations can be handled without delay. The token could be used to inform the authentication system operators about leaving members. Building the system this way would make sure that no critical data has to be stored outside the security database, but the system would be unable to allow members to verify their exact identity (e.g. real name) to third parties. The advantages and disadvantages should be considered carefully, as most applications for which the system is expected to be used (e.g. survey systems, discussion plattforms) do not require strong identity verification.

For the non-pseudonymous variant, once the user identity is verified, the required information would be transferred to the authentication system. The tokens (if used and not given to the authentication system operators) could serve as a proof of user consent, so that exporting the data without user consent would not be possible. A combined approach, where by default accounts are pseudonymous but can be linked to a full identity, could be considered.

Another verification method would be the usage of hashes over the relevant data - giving a list of hashes to the operators of the authentication system would allow them to verify data entered by users, without having access to plain text data of non-participating users. However the hash list would still require protection, as dictionary-style lookup attacks are possible (for example, given the name and address of a person, checking if they are a member would be possible with access to the hashes). As this method also creates problems (users have to enter exactly matching information) and provides limited protection, it can not be recommended.

3 Analysis of authentication standards and technologies

This section gives an overview, analysis and comparision of possible standards and technologies that could be used to build the authentication part of the suggested authentication system.

3.1 X.509 client certificates

X.509 client certificates [2] could be issued by a self-signed CA. This requires no significant cost, limited effort, and allows storing the most critical parts of the system (certificate databases, private keys) on offline machines. Using a self-signed CA makes the system independent of policy requirements by third parties (CAs) and ensures that service providers using the system have to set it up specifically for accepting (only) certificates issued by this organization, avoiding problems if one of the over thousand [3] of regular CAs is compromised.

As not all members can be expected to be skilled and experienced computer users, the enrollment process needs to be as simple as possibe. Using the <KEYGEN> tag specified in HTML5 and implemented in all major browsers except Internet Explorer [4], key generation and certificate enrollment can be

made very user-friendly. Additionaly step-by-step guides are necessary to educate users about private key security practices, backing up their key and usage. An important aspect of using X.509 certificates is that it makes phishing attacks impossible.

However, X.509 certificates make it difficult for users to use the system when not working from their home computer. As some members may want to be able to access services from work, cyber cafes or their mobile devices, this could pose a problem. Setup guides for transferring the certificates to common mobile devices would need to be created.

The certificates can be used directly for authentication to applications or to authenticate to a trusted server which is part of the authentication system. The server could then filter and further pseudonymize the data before forwarding them to the application. Authentication directly to applications would not only disclose all data inside the certificates, but also allow different application operators to link user profiles, for example using the serial number of the certificate. Issuing multiple certificates to the same user (for example with different levels of information) would reduce usability too much for less experienced users.

Direct authentication would allow users to authenticate to applications without a central authentication system being able to observe which applications users are using. However, using OCSP [5] to verify certificate validity would compromise this, as the OCSP responder could observe the queries. For this reason, the certificate validity must be ensured using Certificate Revocation Lists (CRLs).

Certificates can contain multiple OU (organizational unit) entries, which would allow specifying all chapter memberships. The membership ID, if supposed to be included, or a corresponding pseudonym, should be placed in the UID entry, which is also multi-valued.

Due to the mentioned privacy and usability issues, X.509 certificates should not be used for direct authentication to applications, especially not when they would contain full names. They could be considered for login to an authentication server, at least as one of multiple options.

3.2 OpenID

OpenID [6] is an open protocol meant to allow using a central identity to login on any web site or application. It fits the scenario presented here reasonably well. With OpenID, a user wishing to log into an application provides his OpenID identity URL. The OpenID consumer library at the application then performs a discovery process and redirects the browser to the OpenID provider. The user then logs in to his OpenID provider and confirms that he wants to log in to the requesting web site. The user is then redirected back to the web site or application, which (in the simple variant of the protocol)[6] then verifies if the authentication was correctly performed by asking the OpenID provider.

OpenID 2.0 supports the usage of pseudonymous identifiers [6]. To operate in this mode (called *identifier_select*), an anonymous identity URL specifying only the OpenID provider is used, together with a 'magic value' indicating this mode instead of a real user identifier. The OpenID provider then generates a pseudonymous identifier and returns it when confirming the successful authentication. To obtain a service-specific pseudonymous identifier, a user-unique secret can be hashed with the realm value, which is supplied by the application requesting authentication and usually consists of the applications domain. As the applications would be limited to a single OpenID provider, the identity URL could be hardcoded so no additional user interaction would be required.

Additional information about the user can be transferred using the OpenID Attribute Exchange protocol [7], which is also part of OpenID 2.0. This can be used to transmit further information like chapter membership.

OpenID is widely supported in a number of existing web applications. There are numerous free and open libraries that allow application operators to implement OpenID very quickly. The OpenID protocol is very complex and requires the usage of a large and complex library. This complexity increases the probability of security issues inside the library.

A disadvantage of using OpenID is that the authorization server knows when a user logged on to which service. OpenID is prone to phishing attacks, as the user is redirected from an untrusted site to the authentication server. To mitigate this risk, users need to be educated to check the identity of the OpenID server (for example using the SSL indicator in the browser UI).

Despite these problems, OpenID is a good choice for the authentication part of the problem due to its features and widespread support.

3.3 SAML

SAML [8] shares many properties with OpenID. Although it is an older standard (defined in 2002, while OpenID was defined in 2005), it seems less wide-spread than OpenID. SAML is a framework allowing for a wide variety of uses, while OpenID is tailored to web application single-sign-on. SAML is thus more complex than OpenID, allowing for more flexibility but requiring even more complex libraries. To illustrate the complexity, it is important to note that SAML is based on XML and SOAP, two already highly complex standards, extended with additional standards like XML Signatures. When used in a specific scenario, some of the complexity can be removed by limiting the features used, either by selection of one of the existing profiles or by creating an own profile.

Due to the higher flexibility, anonymous and pseudonymous logins are easier to achieve (using SAML assertions) than in OpenID, as SAML allows subjects to be identified with transient (anonymous) identifiers, while OpenID is more fixed on the identity-based approach. OpenID seems better for a scenario where many different authentication providers are supposed to be used, while SAML seems better for a scenario like this, where only a single provider exists.

SAML is preferable to OpenID in terms of features, however, limited library support, additional complexity and the fact that OpenID is more well-known are strong disadvantages. Still, it is a protocol well-suited for this scenario and on par with OpenID.

3.4 Proprietary solution

Creating a proprietary protocol should generally be avoided, as it is very easy to make security-relevant mistakes. Using a non-standard protocol would require creation of custom client libraries for a number of languages. On the other hand, the protocol could be tailored perfectly to the needs of the given scenario and avoid much complexity. No complex third-party libraries (with potential security issues due to the complexity) would be required. The additional effort for development and creation of client libraries may be less than the effort needed to understand and use a highly complex library. Even if using existing libraries was easy, potential implementors (relying parties) could be scared away from using the system by seemingly complex standards like SAML.

Due to the risks and disadvantages of a proprietary, self-developed solution, this should only be considered if easy-to-use library support for existing protocols is insuffcient either on the side of relying party or the authentication provider. If this way is chosen, the methods used in existing protocols should be simplified but re-used and both protocol and code have to be extremely well reviewed by experts. The limited complexity would make this possible, but still difficult.

3.5 OAuth

OAuth [9] is an open protocol meant for web application authorization. Its primary purpose is to enable users to allow one web application to access another (web) application in the name of the user. This could be used for the purpose of user identification: The user would allow the application to access his identity data on the authorization system. However, this would be a workaround. OpenID is a protocol which has very similar properties, fits the given purpose better and should thus be preferred.

3.6 LDAP, Kerberos

LDAP [10] is a well-established protocol and supported by many existing web applications for authentication. However, LDAP is not an authentication system, it is a way to access a directory (database). While it can be easily used to store and retrieve authentication information, it does not provide single-sign-on or a useful way to authenticate to third parties. Authentication using LDAP usually means that the user gives his password to the application which then checks it against the LDAP server. This is not suitable for this scenario, especially as the user needs to have a comfortable way to select which information should be disclosed to the application and the application is not supposed to know the password of the user.

Kerberos [11] is designed as a single-sign-on solution inside organizations and is more focused on client-server applications as opposed to web applications. It is not really suitable for this scenario, either. Using Kerberos is difficult for application providers.

3.7 Exotic cryptographic identity management standards

There are special protocols using advanced cryptographic schemes like blind signatures and zero-knowledge proofs which are built to fulfill exactly the required purpose, i.e. anonymous or pseudonymous authentication while proving some properties like membership.

An example of this would be Microsofts U-Prove [12] or IBMs Identity Mixer [13]. Libraries for those systems are released under a license allowing free public usage. These systems would avoid some of the problems present in the systems suggested above, like leaking linkable pseudonyms to services (with X.509) or the authentication server being able to observe user activity (for example with OpenID). However, these highly complex systems are not supported by common web applications and few libraries for them exist. This would make it very difficult for application providers to implement such an authentication scheme. For this reason, using any of these protocols is not advisable.

4 Further research

It is necessary to review and compare existing libaries and select ones that contain the required features and could be used in such a scenario. The code of the libraries should be reviewed to ensure security was kept in mind by the creators of the libraries.

For SAML, a suitable profile needs to be selected or designed. If a proprietary protocol is chosen, it has to be designed, implemented and reviewed.

Legal aspects of the proposed system (especially privacy/data protection laws) need to be checked and may require changes to the design.

5 Conclusion

Even with the strict constraints and requirements, it is possible to build a secure authentication system using open standards. The proposed system is easy to set up and maintain, preserves user privacy and can be easily implemented by application providers.

The best method to securely identify users in the given scenario is generating a token for each user inside the secure membership database and sending those tokens to the users. Hashes of those tokens (or parts of it), linked to anonymous information about chapter membership, should be provided to the authentication system operators to allow direct verification and creation of pseudonymous accounts. (Providing chapter membership information after user registration would be possible, but this would remove the ability of instant account creation.) Name-based accounts can be offered as an option, possibly by requiring the user to enter a second (part of the) token before the corresponding data is transferred from the membership database to the authentication system.

For privacy reasons, client certificates should not be used for direct authentication to applications. (If they are used this way, CRLs should be used instead of OCSP to check for revoked certificates.) Client certificates can be used for authentication to an authentication server, either as the only way or as an optional alternative to password-based authentication. Information about the user (name, chapter memberships) could be stored exclusively in the certificate to ensure that no personal data is stored on the authentication server. The authentication server would then retrieve this information from the certificate when it is presented by the user.

The authentication server should use either OpenID with application-specific random pseudonyms, or SAML using transient anonymous user identifiers with all identifying information inside the assertion. This ensures user privacy while still allowing the application to know all required information (i.e. the membership status of the user). Anonymous usage, i.e. pseudonyms changing for each login, could be offered as an optional service for applications that only want to limit access to members, but do not need to distinguish users (or recognize returning ones). In case of OpenID, the membership information is exchanged using OpenID Attribute Exchange. In case of SAML, it is contained in the assertion. For OpenID, the usage of pseudonyms and OpenID Attribute Exchange limits the possible configuration choices and requires the usage of version 2.0 of the OpenID protocol.

Users need to be educated about secure usage of the authentication system, i.e. verifying the identity of the login server before entering their password. The server will be able to observe which user uses which service at which time. To lessen this problem, logging should be restricted to the minimum needed for secure operation. Additionally, applications could implement their own independent login systems for regular usage and require only one authentication against the authentication system to confirm membership status after the account was created, and at regular intervals to verify that the membership information is still current.

Applications can use one of the existing libraries available for most programming languages to easily verify membership information using this system. They need to restrict their implementation to only accept identities from the authentication server provided by the system, and check for the specified attributes.

The decision between OpenID and SAML should be based on available library support. Only if no easy-to-use libraries for OpenID and SAML exist should an approach using a simple but proprietary protocol be considered. This protocol would take the place of OpenID/SAML in the system. Relying party libraries would need to be provided in a number of languages and a thorough review of both protocol and implementation would be necessary.

References

[1] CAcert. *Assurance Policy*, 2008-2009.
http://www.cacert.org/policy/AssurancePolicy.php.

[2] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.
http://www.ietf.org/rfc/rfc5280.txt.

[3] Peter Eckersley and Jesse Burns. Is the SSLiverse a Safe Place? 27C3, 2010.
https://www.eff.org/files/ccc2010.pdf.

[4] MDC Doc Center. *keygen*.
https://developer.mozilla.org/En/HTML/Element/keygen.

[5] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560 (Proposed Standard), June 1999.
http://www.ietf.org/rfc/rfc2560.txt.

[6] OpenID Authentication 2.0 - Final. OpenID Specifications.
http://openid.net/specs/openid-authentication-2_0.html.

[7] OpenID Attribute Exchange 1.0 - Final. OpenID Specifications.
http://openid.net/specs/openid-attribute-exchange-1_0.html.

[8] SAML Specifications. OASIS Standard, 2003-2009.
http://saml.xml.org/saml-specifications.

[9] E. Hammer-Lahav. The OAuth 1.0 Protocol. RFC 5849 (Informational), April 2010.
http://www.ietf.org/rfc/rfc5849.txt.

[10] K. Zeilenga. Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map. RFC 4510 (Proposed Standard), June 2006.
http://www.ietf.org/rfc/rfc4510.txt.

[11] B. Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications 32(9) 33-38*, september 1994.
http://nii.isi.edu/publications/kerberos-neuman-tso.html.

[12] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy*. The MIT Press, August 2000.
http://www.credentica.com/the_mit_pressbook.html.

[13] IBM Research - Zurich. Specification of the Identity Mixer Cryptographic Library.
http://www.zurich.ibm.com/~pbi/identityMixer_gettingStarted/ProtocolSpecification_2-3-2.pdf.