Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 30, 2018

V. Bertola
Open-Xchange
M. Sanz
DENIC eG

October 27, 2017

An Architecture for a Public Identity Infrastructure Based on DNS and OpenID Connect
draft-bertola-dns-openid-pidi-architecture-00

Abstract

The following document describes an architecture for an open, global, federated Public Identity Infrastructure (PIDI), based on the Domain Name System (DNS) and on the OpenID Connect framework built over the OAuth protocol.

Status of This Memo

This Internet-Draft will expire on April 30, 2018.

Copyright Notice

1. Introduction

How to deal with online identities is one of the great unsolved problems of today's Internet: each Internet user has to authenticate for hundreds of different online services, all of which require some personal information that he or she has to provide and maintain separately; and this leads to severe usability and security issues.

This document describes an architecture for a Public Identity Infrastructure (PIDI), an identity management framework, building on existing protocols and on new extensions, that can provide the three basic functions of online identity management - authentication, authorization, and management of personal information - and do so in an open, global and federated manner, creating a single interoperable personal identity space that can be shared by the entire Internet, while at the same time preventing any centralized control of all online identities and empowering users rather than identity providers.

2. Requirements Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Throughout this document, values are quoted to indicate that they are to be taken literally. When using these values in protocol messages, the quotes MUST NOT be used as part of the value.

3. Key features

In the PIDI architecture described in this document, people and other entities identify themselves in their online activities by using a DNS-style label, located inside an existing and valid domain name, as an identifier. Such identifier allows users to log into any Internet service using a single account associated to their identifier.

Identifiers are jointly managed by two complementary entities, acting together as the identity provider; users are able to choose the managers of their identifier among any number of compatible providers, or to host one themselves.

Users can employ their identifier to log into any website or online service supporting this architecture, even without prior registration; on first access to that specific service, the service

can request access to the user's personal information as entered by him or her into the personal profile.

If the user consents to this access, the requested information will be made available to the service, which can thus automatically present the appropriate messages and legal information and then create a local account or profile for the user, associated to the identifier. Thus, "registering" for an online service is not necessary any more; users just self-identify themselves whenever necessary.

As the architecture is federated, like email and other public Internet standards, multiple interoperable providers of identifiers can exist, including personal providers self-hosted by their users; all of them are intrinsically supported by any online service implementing the standard, though services, like in the email environment, may implement local policies that blacklist certain providers or identifiers, or treat them differently. Users can pick any provider and, if they control the domain name to which the identifier belongs, can move their identifier to a different provider whenever they want, simply by changing a record in the domain name's zone.

By performing authentication in a single place, users are freed from the need to remember, update and protect huge numbers of passwords. Any password change, any security update and any additional authentication mechanism, such as two-factor authentication, can be implemented once and immediately apply to all the logins of the user, thus making it easier to upkeep security. The system can work with any type of authentication mechanism, even without passwords.

The focus of this architecture is to authorize and authenticate users in the online space only, i.e. to ensure that the user of a given identifier is always the same that initially acquired that identifier at registration; the architecture does not address the issue of how to actually verify his or her true identity in the real world. Accordingly, there is no requirement for an actual real-life authentication of the users, and their identity and personal information are entirely self-declared; users may also have multiple identities (e.g. a personal one, a business one etc.), as an additional protection to their privacy.

At the same time, nothing prevents specific implementations or specific identity providers to support third-party validation of the user's personal information, thus also providing proof of the user's real world identity, which may also be required by specific services to accept the identifier. However, interaction with online-offline authentication systems and providers, such as governmental e-ID

documents or certification authorities, is outside of the scope of this document, except for what may be necessary to transfer and store inside this architecture the additional information related to this interaction.

4. Technical design and motivations

To simplify implementation, the architecture discussed in this document builds over an existing and widely adopted identity management framework: OpenID Connect [OpenID.Core]. However, that framework fails short of several of the requirements and objectives set forth in the previous section.

The proposal thus uses a pillar of the Internet, the Domain Name System [RFC1034], and a few purpose-developed specifications such as [I-D.sanz-openid-dns-discovery], to complete OpenID Connect and reach the desired objectives.

4.1. Advantages and shortcomings of OpenID Connect

OpenID Connect [OpenID.Core] is an identity management framework that has been recently gaining widespread adoption; it gets nearer than others to meeting the requirements. Building on the OAuth 2.0 [RFC6749] authorization framework, OpenID Connect provides authentication, authorization, and basic management of personal information; it is currently already being used by many big Internet access and content providers to offer authentication to third-party services, and many different implementations, both commercial and free, are readily available.

However, OpenID Connect, in its current status, is aimed at the creation of individual and non-interoperable sets of identities, entirely controlled by a single identity provider. While this design suits a company willing to provide single sign-on for all its websites, or an online service willing to let other services authenticate users against the credentials it provides, there is no easy way for multiple providers to offer identities in the same interoperable set, or for online services to support identities by multiple providers without explicitly implementing separate support for each and every provider; and there is no way to create a single, public, global identity set that can be used by the entire Internet without having to rely on a single and centralized identity provider. Also, once users adopt an identifier run by a specific provider, there is no way for them to move it transparently to another provider.

In the end, the centralization of all three functions and of the ownership of the user's identifier in a single entity that cannot be

easily picked and changed by the user creates significant risks for privacy and security, and prevents competition and choice among multiple identity providers. While users should be able to centralize these functions in a single entity that they really trust, they should also have the options of owning the identifier directly, of distributing these functions among more than one entity and of changing these entities as easily as possible.

4.2. Motivations for the use of the Domain Name System

To create a single identity framework for the entire Internet, a single namespace and a lookup service for the identities are necessary; and to prevent control by a single entity, they need to be implemented in a decentralized and federated manner that must also provide the necessary security features.

The Internet already relies for its basic functioning on a single namespace and on a lookup service: the Domain Name System [RFC1034]. The ubiquitousness of the DNS, the familiarity of Internet users with DNS-style hierarchical strings, and the increasing adoption of DNSSEC [RFC4033], make the DNS the proper container for a public, secure, decentralized and hierarchical identity naming and lookup service.

By using DNS strings as identifiers for human identities, with the addition of a simple mapping mechanism such as that described in [I-D.sanz-openid-dns-discovery], it is immediate for anyone to know where and how to look up information about an identifier, without knowing any other piece of data - not even the identity provider that is managing it.

Locating identity identifiers in the DNS has the additional advantage that users, rather than identity providers, can easily become the sole owners of their identifier by acquiring a personal domain name; the controller of that zone - the user - can point the identifier towards a different provider just by changing a record in the zone, even without the current provider's consent or action, much reducing the opportunities for user lock-in by the identity provider.

4.3. Separation of roles

While the current OpenID Connect implementations concentrate authentication, authorization and personal information management all in a single entity, a separation of roles is introduced to increase the decentralization and security of the system, mimicking the roles existing in the Domain Name System industry. Two different entities - an "identity authority" and an "identity agent" - co-manage each online identity, fulfilling different functions, with the user being

free to choose and change each of them, and even to run directly one or both of these roles.

5. Elements of the architecture

5.1. User

A user of the PIDI architecture is an entity of any kind - a physical person, a juridical person, a host, an application, or anything else - that needs to authenticate itself to gain access to online services and applications, and to provide and distribute over the Internet, in a controlled manner, information about itself.

5.2. Online identity

An online identity is a collection of personal information associated to an identifier that represents it.

No assumption is made on whether distinct online identities belong to distinct physical persons, or even whether they belong to human beings at all. Users may own any number of online identities; they should not be required to disclose the correlation between their different online identities.

The personal information included in an online identity is entirely self-asserted by the controlling user; in the absence of appropriate additional mechanisms outside the purview of this document, assumptions should not be made on whether such information is "true" or "false" for any definition of these terms. The only assumption that can be safely made about the personal information included in an online identity is that the controlling user is stating that information about himself, herself or itself.

5.3. Identifier

A PIDI identifier is a string, unique on a global Internet scale, that represents a distinct online identity in the PIDI architecture.

PIDI identifiers MUST follow any syntax which is acceptable to the mapping mechanism described in [I-D.sanz-openid-dns-discovery]. Specifically, an identifier MUST be one of the following:

- A fully qualified DNS name following the conventions set forth in section 2.3.1 of [RFC1035].

- A syntactically valid internationalized DNS name (IDN) as per [RFC5890].

- An e-mail address following the syntax described in section 3.4.1 of [RFC5322].

- A syntactically valid internationalized e-mail address within the framework of [RFC6530].

In all these cases, no assumption is made on whether these strings actually correspond to existing network objects such as a host or a mailbox; they could not correspond to anything but the online identity that they represent. However, when applying the mapping mechanism to an identifier, the resulting string MUST belong to an existing and working domain name, and MUST point to an existing DNS record of the appropriate type and syntax as specified in [I-D.sanz-openid-dns-discovery].

5.4. Claim

A claim is a piece of information associated to the online identity that is represented by an identifier.

Claims are made by a standard claim name, to which a predefined claim type is associated, and by a claim value that contains the actual information. To ensure interoperability, claim names and types are publicly standardized.

Claims MUST follow the specification and format described in [OpenID.Core]; however, further claim names and types may be defined in additional specifications.

5.5. Identity authority

An identity authority is an entity responsible for the authentication and authorization functions of the PIDI identity management framework.

More specifically, the identity authority MUST perform the following activities:

- Allow identity agents, on behalf of their users, to create, update and cancel identifiers, verifying the proper set up of the required DNS records in the identifier's domain name zone, including an appropriate proof that the user has write access to that zone.

- Allow the user to set the password or the other credentials that will be used for authentication, and store them securely.

- Authenticate the user whenever necessary; to this purpose, the authority should either ask the user to provide credentials and verify them, or rely on the secure storage of the results of a previous authentication.

- Whenever the user tries to log into an online service for the first time, or whenever the service requests access to additional claims, show the user the list of claims that the service would like to access, and ask the user for specific and separate consent on the sharing of each claim; then, authorize the service to access the consented claims at the appropriate identity agent.

- Allow the user to review and change the consent that was given for access to claims, for each claim and relying party.

To perform these activities, the identity authority MUST act as the OpenID Provider defined in [OpenID.Core]; it MUST also allow third parties to retrieve its OpenID Configuration according to the specification in section 4 of [OpenID.Discovery].

Moreover, the identity authority MUST allow relying parties to perform dynamic client registration as defined in [OpenID.Registration], and it MUST NOT require any Initial Access Token or other out-of-band mechanisms, though an authority may apply policies to prevent some clients from registering if these clients can be presumed to be abusive or malicious. The identity authority MUST use the distributed claims mechanism described in section 5.6.2 of [OpenID.Core] to direct a service requesting access to claims to the identity agent managing the personal information associated with the identifier.

The identity authority, unless also acting as identity agent, should not have access to any claim associated to the online identity, except the identifier, the authentication credentials and any personal information necessary to verify them, or that the user has voluntarily shared with the identity authority to allow it to provide its services.

After each login attempt by the user, the identity authority SHOULD communicate to the identity agent that such attempt has happened and what was its outcome. Details for such communication will be specified separately.

5.6. Identity agent

An identity agent is an entity responsible for the personal information management function of the PIDI identity management

framework, as well as for the management of the relationship with final users.

More specifically, the identity agent MUST perform the following activities:

- Allow users to acquire, move and cancel their identifiers, performing the necessary technical operations.

- Allow users to enter, update and delete the value for any claim supported by the architecture and by the agent.

- Provide to any relying party that shows a valid authorization, received by the appropriate identity authority, access to the claims that the user has consented to share with that relying party.

The identity agent should also perform the following activities:

- Allow users to verify which claims have been shared with each relying party.

- Allow users to retrieve a list and history of their logins, unless such information has not been made available to the identity agent.

To perform these activities, the identity agent MUST act as a provider of distributed claims as defined in [OpenID.Core], running an appropriate UserInfo Endpoint; it MUST allow access to such endpoint to any relying party that has been successfully authorized to do so by the identity authority. The identity agent MUST also allow third parties to retrieve its OpenID Configuration according to the specification in section 4 of [OpenID.Discovery]. It SHOULD also provide to identity authorities and relying parties an endpoint to communicate user logins.

The identity agent, unless also acting as identity authority, should not have access to the user's password; it also should not have access to other credentials used for the authentication process, unless they are claims that can be legitimately used for other purposes (e.g. a mobile phone number). The identity agent should use the PIDI identifier to grant access to its PIDI services, relying on the identity authority for authentication.

5.7. Relying party

A relying party is any online service, website or application willing to accept PIDI identifiers to recognize and authenticate its users.

A relying party can accept PIDI identifiers natively, using them as the sole identification method for its accounts, or can use PIDI identifiers as a pointer towards an internal username in its own accounting system. In both cases, the relying party should allow users to register and authenticate with any valid PIDI identifier, though the relying party may apply policies to reject or discriminate against specific identity agents or identity authorities that are credibly presumed to be abusive or malicious. Moreover, a relying party may apply its own policies and requirements to determine which users should be disallowed from using its services, even if they show up with a valid PIDI identifier.

More specifically, the relying party MUST perform the following activities:

- Allow users to enter a PIDI identifier in its login and registration forms or procedures.

- Perform the mapping described in [I-D.sanz-openid-dns-discovery] and then an OpenID Connect authentication flow, to authenticate the user, to gain authorization to access the user's claims, to retrieve such claims as authorized, and optionally to use them to create or update a local account for the user.

To perform these activities, the relying party MUST act as the Relying Party defined in [OpenID.Core]; it MUST also perform a dynamic client registration as defined in [OpenID.Registration], every time it encounters an identity authority never seen before. To do so, the relying party MUST be able to retrieve the OpenID configuration of the identity authority and of the identity agent, according to the specification in section 4 of [OpenID.Discovery].

The relying party, unless also acting as identity authority, should not have access to the user's password; it also should not have access to other credentials used for the authentication process, unless they are claims that the user has authorized the relying party to access. Moreover, the relying party should never have access to any user claims different from those that the user has authorized the relying party to access, and it should honor any request by the user to update or delete the claims that he or she already provided.

After each login attempt by the user, the relying party SHOULD communicate to the identity agent that such attempt has happened and

what was its outcome. Details for such communication will be specified separately.

6. Interaction flows

The following sections provide a high-level description of the sequence of steps that has to be followed jointly by the various actors to perform some basic actions. The sequence of steps is meant to be normative, but the detailed technical specification of each step can be found in the standards referenced in the previous section, or in further standards that will be specified separately.

6.1. Identifier creation flow

To create a valid identifier, the actors have to follow this procedure:

1. The user approaches an identity agent and requests the provision of an identifier, agreeing with the agent on the identity authority that will manage it.

2. If necessary, the identity agent registers and/or sets up the domain name in which the identifier will be created.

3. The identity agent or the user, depending on who operates the domain name, sets up the appropriate DNS record for the mapping of the identifier.

4. The identity agent requests the creation of the identifier at the agreed identity authority.

5. The identity authority verifies that the DNS record has been successfully set up and that the user, either directly or through the identity agent, has write access to the domain name zone of the identifier.

6. If verifications are successful, the identity authority creates the identifier, in inactive state, and communicates success to the agent.

7. The agent communicates success to the user and redirects the user to the identity authority to set up the authentication credentials.

8. The user approaches the identity authority and sets up the authentication credentials.

9. Upon successful setup of the authentication credentials, the identity authority activates the identifier and communicates success to the user.

After the conclusion of this procedure, the identifier is ready for use for authentication.

6.2. Authentication flow

To authenticate and log the user into a relying party, the actors have to follow this procedure:

1. The relying party asks the user to provide the identifier.

2. The user provides an identifier of choice, corresponding to the online identity that the user chooses to use for this login, to the relying party.

3. The relying party performs a DNS lookup to identify the identity authority and the identity agent that manage the identifier.

4. If the relying party has never performed a login towards that specific identity authority, or if for any reason (e.g. expiration) it does not possess any valid credentials towards that identity authority, the relying party performs OpenID Connect Discovery section 4 and then OpenID Dynamic Client Registration towards the identity authority, to acquire valid OpenID Connect client credentials; the acquired credentials may be stored for future use.

5. The relying party and the user perform an OpenID Connect authentication using the Authorization Code Flow, at the end of which the relying party receives an Identity Token and, if claims need to be accessed, an Access Token. Optionally, at the end of this step, the identity authority communicates to the identity agent that a login has been made by the user towards that relying party. During the Authorization Code Flow, the identity authority must ask the user for separate and specific consent to share each claim that has been requested by the relying party, and store this consent for future reference.

6. If the relying party has been granted access to any claims, but has never performed a connection towards that specific identity agent, the relying party performs OpenID Connect Discovery section 4 towards the identity agent.

7. If the relying party has been granted access to any claims, it performs a connection to the UserInfo Endpoint of the identity

authority, that will use the Distributed Claims mechanism to redirect the relying party to the identity agent; the relying party then connects to the UserInfo Endpoint of the identity agent and retrieves the claims.

8. Optionally, if appropriate, the relying party creates a local account for the user and stores the identifier and the values of the claims into it. If necessary, this step can be subject to further direct interaction between the user and the relying party, for example to ask the user to accept the relying party's terms and conditions.

After the conclusion of this procedure, the user has logged into the relying party and provided it with the appropriate personal information; if necessary, the relying party has created a new local account for the user.

7. Security Considerations

tbd

8. IANA Considerations

tbd

9. Normative References

[I-D.sanz-openid-dns-discovery]
Bertola, V. and M. Sanz, "OpenID Connect DNS-based Discovery", draft-sanz-openid-dns-discovery-00 (work in progress), October 2017.

[OpenID.Core]
Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.

[OpenID.Discovery]
Sakimura, N., Bradley, J., Jones, M., and E. Jay, "OpenID Connect Discovery 1.0", November 2014, <http://openid.net/specs/openid-connect-discovery-1_0.html>.

[OpenID.Registration]
Sakimura, N., Bradley, J., and M. Jones, "OpenID Connect Dynamic Client Registration 1.0", November 2014, <https://openid.net/specs/openid-connect-registration-1_0.html>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC6530] Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.17487/RFC6530, February 2012, <https://www.rfc-editor.org/info/rfc6530>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

Authors' Addresses

Vittorio Bertola
Open-Xchange
Via Treviso 12
Torino 10144
Italy

Email: vittorio.bertola@open-xchange.com
URI: https://www.open-xchange.com

Marcos Sanz
DENIC eG
Kaiserstrasse 75 - 77
Frankfurt am Main 60329
Germany

Email: sanz@denic.de
URI: https://www.denic.de