# A fully distributed OpenID Connect deployment based on domain names: id4me
## Challenges, lessons-learned and take-aways

Vittorio Bertola
*Open-Xchange*
*vittorio.bertola@open-xchange.com*

Marcos Sanz Grossón
*DENIC eG*
*sanz@denic.de*

**Abstract**

*id4me is an open project working on a public standard for identity management, creating an open, federated and interoperable identity framework on an Internet scale.*

*This paper presents deployment challenges and lessons learned, describes some problems that are not fully solved yet, and seeks for input and collaboration from the OAuth Security Workshop 2018[1] participants.*

### 1. Introduction

As online services become more and more targeted and customized, Internet users have to authenticate and provide personal information into each and every one of them.

Traditionally, authentication happens via a username and a password, but this method is becoming harder and harder to manage for users as the number of online services requiring authentication grows.

In the last few years, several over-the-top service providers (OTTs), such as Google and Facebook, have started to provide Internet-wide single sign-on (SSO) services based on their own credentials. This is very convenient, but has a series of drawbacks, prominently provider lock-in, privacy considerations, and the lack of interoperability, still requiring users to have multiple accounts - one per each OTT.

This is why the authors, as part of a broader group, started working on an identity management framework that could work just like those of the OTTs, but also be open, public and federated, allowing anyone to provide users with credentials valid for Internet-wide SSO. Such a system, by giving users a way to choose and change their Identity Provider (IdP), would empower the user rather than the provider, and protect the user's privacy and freedom.

The id4me architecture is based on OpenID Connect[2], the current *de facto* standard for online authentication systems, and on the Domain Name System (DNS), traditionally the cornerstone for Internet directories. A full description of the architecture can be found in the IETF draft repositories[3].

We will not enter here into a detailed description of the entire architecture, but we will rather discuss specific aspects throughout the paper.

### 2. Using DNS-based identifiers

As the objective is to provide a unifying and fully interoperable framework for use by the entire Internet, it is necessary to associate each possible online identity to an identifier that is guaranteed to be unique over the global network at any given moment.

The easiest way to do this is to rely on the Domain Name System, which offers a global, distributed way to create and manage unique identifiers. This can be accomplished by adopting fully qualified DNS hostnames as identifiers; email addresses enjoy the same qualities as well. Also, the latest extensions to the standards allow for fully internationalized DNS hostnames and email addresses[4], using any script and language. The DNSSEC extensions[5] provide for security properties like integrity and source authentication.

This choice would also allow for the integration into the global standard of the existing private namespaces; if the private SSO system uses email addresses as identifiers, they can use the email-to-hostname mapping system provided in the draft id4me specifications; if they use any other type of local string, the provider just needs to implement their own mapping mechanism to

[1] *"OAuth Security Workshop 2018 | ST - FBK | ST - Fondazione Bruno ...." https://st.fbk.eu/osw2018. Accessed 18 Jan. 2018.*

[2] (2014, November 8). Final: OpenID Connect Core 1.0 incorporating errata set 1. Retrieved January 18, 2018, from http://openid.net/specs/openid-connect-core-1_0.html

[3] (2017, October 27). An Architecture for a Public Identity Infrastructure Based on DNS and OpenID Connect. Retrieved January 18, 2018, from https://tools.ietf.org/html/draft-bertola-dns-openid-pidi-architecture-00

[4] (2010, August). "RFC 5890 - Internationalized Domain Names for Applications (IDNA ...." Accessed January 18, 2018. https://tools.ietf.org/html/rfc5890.

[5] (2005, March). "RFC 4035 - Protocol Modifications for the DNS Security Extensions." Accessed January 18, 2018. https://tools.ietf.org/html/rfc4035.

associate them to hostnames within a DNS zone that they own.

It might be that DNS-based OpenID identifiers, though standard-compliant, are not properly dealt with by some deployments. For instance, even the OpenID Foundation certification software was dealing wrong with this so-called "non-scoped" identifiers (those without an "@" character), a bug that was communicated to the foundation and quickly fixed[6]. The abundance (or lack thereof) of such kind of bugs still needs to be further assessed.

**3. Proving possession of control of the identifier**

In the OTT/social-media SSO landscape the user registers (or already has) an account at the OTT provider, which will assign an identifier to the user scoped within the boundaries of the provider. It's easy for the provider to check whether an identifier already exists, and avoid duplication. id4me lets the user bring their own identifier to the system (their domain name), thus the Identity Providers have to implement a mechanism for the user to prove possession of said identifier.

The id4me architecture does not address the issue of how to actually verify the true identity of the user in the real world and, accordingly, there is no requirement for an actual real-life identification of the users. Thus, possession/control of an identifier is expected to be proven in a technical way, that should be easy to automate, and does not require for example contractual information to be available or verified.

Different mechanisms exist nowadays to create a standard interaction between DNS providers and service providers (like web-hosting), which could be used to prove/exert control on a given domain's DNS. Domain Connect[7], initially created by GoDaddy, is a prominent example in this area. However, id4me has decided to use the IETF's soon-to-become-standard ACME[8] for this purpose.

ACME stands for *Automated Certificate Management Environment* and is a protocol that has been (and still is) under extensive security analysis by the IETF ACME working group. Originally conceived to be exclusively a protocol for the automation of certificate management, ACME requires the applicant of a domain-validated certificate to prove control of the domain name in question. This is accomplished by solving challenges, of which three different types exist: HTTP, TLS SNI (*Server Name Indication*) and DNS.

In order to avoid the overhead in setting up HTTP or TLS servers (s. also Discovery section) we have opted for the user (or their DNS provider in their behalf) to solve a DNS challenge, properly secured with DNSSEC[9] . By submitting an ACME pre-authorization request to the IdP (that plays the role of the ACME server) including the desired id4me in the ACME identifiers list, the IdP will try to validate control by the user of said id4me identifier in a challenge-response process.[10]

After successful validation, the IdP returns a one-time-usage magic link to the ACME client in the Location-Header of the last server response that will allow for the user to finish the id4me registration process and instantiate their id4me credentials. The following is an example of the last interaction between the ACME client and the IdP:

```
POST
https://acme.freedom-id.de/v1/authz/zpMDAxZ0NRhGS1jREXV
JW2wlkzrFrQAAAWD51xK_/0
Header:
{"nonce":"LmX8_6AmZg4HhmDeJnpyDQAAAWD51xK5","url":"http
s://acme.freedom-id.de/v1/authz/zpMDAxZ0NRhGS1jREXVJW2w
lkzrFrQAAAWD51xK_/0","kid":"https://acme.freedom-id.de/
v1/account/0","alg":"RS256"}

Payload:
{"type":"dns-01","keyAuthorization":"tjg83U7wYg2oNWaXgu
-Wsvr8nUf6ch8boXg5UQs8jb-k0f3ixtTLjERej5QW6sv5DLizDAAAA
WD51xK-.BzZPX5vawdddJUpTaDUGpUm0xMrJHsUUIwXcRzatorE"}


HTTP/1.1 200

[...]
Link:
<https://auth.freedom-id.de/init/magic/mJK6QGCffxssqSVp
qTKzOw0GY0f9gacxMi2Y0aL3wq>;rel="create-form"


{"type":"dns-01","status":"valid","token":"tjg83U7wYg2o
NWaXgu-Wsvr8nUf6ch8boXg5UQs8jb-k0f3ixtTLjERej5QW6sv5DLi
zDAAAAWD51xK-","url":"https://acme.freedom-id.de/v1/aut
hz/zpMDAxZ0NRhGS1jREXVJW2wlkzrFrQAAAWD51xK_/0"}
```

This way the initial id4me credentials should only be known to the IdP, not to the DNS provider.

**4. Discovery of the corresponding Identity Provider via DNS**

OpenID Connect Discovery 1.0[11] specifies the so-called "Issuer Discovery" process for any client to determine the location of an IdP. In this process the end-user supplies an identifier as input to the Relying Party (RP, synonym of the OAuth "client" in the OpenID Connect vocabulary). The RP should apply some normalization rules to the identifier to determine the "resource" and "host" and then make an HTTP GET request to the

[6] https://github.com/openid-certification/oidctest/issues/41
[7] (n.d.). The Spec – Domain Connect. Retrieved January 18, 2018, from http://domainconnect.org/specification/
[8] (2017, December 14). draft-ietf-acme-acme-09 - Automatic Certificate Management .... Retrieved January 18, 2018, from https://datatracker.ietf.org/doc/draft-ietf-acme-acme/

[9] Though this is only a recommendation in the ACME specification, not a requirement.
[10] A proper certificate request could also be used for this purpose, however, all actual certificate issuance/revocation mechanisms of the protocol are not really necessary in the context of id4me.
[11] (2014, November 8). Final: OpenID Connect Discovery 1.0 incorporating errata set 1. Retrieved January 18, 2018, from https://openid.net/specs/openid-connect-discovery-1_0.html

host's WebFinger[12] endpoint, by presuming all hosts run such one. OpenID Connect Discovery also leaves room for alternative “out-of-band” discovery mechanisms.

Though setting up a WebFinger service for each domain name hosting an id4me identifier is perfectly possible, it’s nonetheless an expensive overhead: it requires spawning and operating a webservice that it might not exist in the first place. In order to make the discovery process secure, it must properly run over HTTPS, which further complicates operation and introduces a security-dependence on the X.509 Public Key Infrastructure.

We explored appropriate lightweight technologies that might fit in the out-of-band discovery mechanism foreseen in the standard and decided that DNS itself was the most natural fit, because DNS hat to be set-up and run for a domain anyway and predates the instantiation of any WebFinger service.

Dedicated DNS Resource Record (RR) types for discovery purposes exist (NAPTR RR, SRV RR)[13][14][15][16] and there are a number of Internet standards that use them in different ways for discovery/bootstrapping. We opted for a human-readable TXT RR-based solution, fully described in a document in the IETF draft repositories[17], analog to what other security standards, like SPF[18], DKIM[19] or DMARC[20], do.

An id4me discovery DNS record looks like this:

```
_openid.yourname.example.de

IN                                                    TXT
"v=OID1;iss=auth.freedom-id.de;clp=identityagent.de"
```

Usage of TXT RR is simple and straightforward for the clients. Usage of DNSSEC on top provides the security properties needed.

**5. Distributing the location of claims about the subject**

OpenID Connect Core 1.0 foresees the possibility of an IdP that is not able to assert certain claims about the user, and provides for mechanisms to retrieve the answer from a third-party, the so-called Claims Provider. This can be done at the UserInfo Endpoint of the IdP, via Aggregated Claims (IdP itself gathers and returns the claim values asserted by the Claims Provider) or via Distributed Claims (IdP returns references to those claims located at the Claims Provider).

The id4me environment makes usage of Distributed Claims: domain name registries do not necessarily have user/registrant personal data and might delegate answers to their registrars. However we have found these mechanisms to be underspecified (like how to find out the proper location of the Claims Provider), which logically leads to insufficient/wrong support (bugs in the OpenID certification software, missing support in client libraries).

For example, at the time of this writing, the OpenID Foundation certification software tries to validate the JWT signatures on the claims asserted by the Claims Provider with the public key of the IdP, which consistently fails and thus hinders the id4me product’s certification[21].

For the other aforementioned problem we have decided to leverage on the previously described DNS discovery mechanism to allow for the IdPs to locate the domain’s Claims Provider (that’s the value of the “`clp`” property in the content of the TXT RR) and deliver this information seamlessly to the client, thus closing this underspecification of the standard until further normative advice exists on how to do this.

For the Claims Provider to be able to verify the validity of OAUTH bearer access tokens, which are opaque to the clients, they are implemented as JWT signed self-contained documents that include what claims have been consented (and which were not) for which client by the user.

The following is an example of a (decoded, signature-stripped) id4me access token:

```
{
  "sub": "yourname.example.de",
  "scp": [ "openid" ],
  "clm": [ "email" ],
  "dat": {
         "rejected_claims": [ "birthdate" ]
  },
  "iss": "https://auth.freedom-id.de",
  "exp": 1516200120,
```

[12] (2013, September). RFC 7033 - WebFinger - IETF Tools. Retrieved January 18, 2018, from https://tools.ietf.org/html/rfc7033
[13] (2000, February). RFC 2782 - A DNS RR for specifying the location of ... - IETF Tools. Retrieved January 18, 2018, from https://tools.ietf.org/html/rfc2782
[14] (2002, October). RFC 3403 - Dynamic Delegation Discovery System ... - IETF Tools. Retrieved January 18, 2018, from https://tools.ietf.org/html/rfc3403
[15] (2005, January). RFC 3958 - Domain-Based Application Service Location ... - IETF Tools. Retrieved January 18, 2018, from https://tools.ietf.org/html/rfc3958
[16] (2013, February). RFC 6763 - DNS-Based Service Discovery - IETF Tools. Retrieved January 18, 2018, from https://tools.ietf.org/html/rfc6763
[17] (2017, October 18). OpenID Connect DNS-based Discovery, Retrieved January 18, 2018, from https://tools.ietf.org/html/draft-sanz-openid-dns-discovery-00
[18] (2014, April). RFC 7208 - Sender Policy Framework (SPF) for ... - IETF Tools. Retrieved January 18, 2018, from https://tools.ietf.org/html/rfc7208
[19] (2011, September). RFC 6376 - DomainKeys Identified Mail (DKIM) Signatures - IETF Tools. Retrieved January 18, 2018, from https://tools.ietf.org/html/rfc6376
[20] (2015, March). RFC 7489 - Domain-based Message Authentication ... - IETF Tools. Retrieved January 18, 2018, from https://tools.ietf.org/html/rfc7489

[21] https://github.com/openid-certification/oidctest/issues/51

```
  "iat": 1516199520,
  "cid": "test-client"
}
```

### 6. Detaching from the existing X.509 PKI infrastructure via DANE

OpenID/OAuth communications make ubiquitous usage of TLS and HTTPS. TLS encryption is currently based on certificates issued by certificate authorities (CAs). Within the last few years, a number of CAs suffered serious security breaches, allowing the issuance of certificates for well-known domains to parties that did not own those domains. Trusting a large number of CAs might be a problem because any breached CA could issue a certificate for any domain name.

There are a number of approaches to close this loophole by using DNS. For instance the CAs themselves are promoting the deployment of the so-called Certification Authority Authorization[22] (CAA) RR. An alternative approach is the DNS-based Authentication of Named Entities[23] (DANE), that enables the administrator of a domain name to create and certify themselves the keys used in that domain's TLS servers by and storing pointers to them in the DNS[24].

id4me plans to make extensive usage of DANE to secure HTTPS communications because DANE embodies the security "principle of least privilege" that is lacking in the current public CA model. Current support in web libraries is limited, though, for DANE has only found wider usage in the email infrastructure so far.

### 7. Requiring dynamic discovery of supported claims

In the traditional scenario of an OpenID Connect SSO system controlled by a single entity, standardizing and discovering claim names is not so important; apart from the very few basic ones that are standardized in the core specification, the system owner can just make up the others as necessary.

In a public and federated scenario, however, two additional requirements arise:

1. All claim names, including the optional ones, have to be standardized, so that the different actors know how to refer to the same piece of information about the user;
2. RPs need a way to discover which claims are actually supported by the specific Claims Provider and IdP, so to know what they can ask for.

IdPs are recommended to announce metadata on the claims they support, however this information is static, and different Claims Providers might support different claim types.

Also, while id4me plans to standardize a broad number of claim names and types, this work has not started yet.

In the overall, this is still an area of work-in-progress.

### 8. Subject Identifier Types

A Subject Identifier is a locally unique and never reassigned identifier within the IdP for the end-user, which is intended to be consumed by the client. Two Subject Identifier types are defined by the OpenID Connect specification:

- “public”: This provides the same subject identifier value to all clients. It is the default behaviour.
- “pairwise”: This provides a different subject identifier value to each client, so as not to enable clients to correlate the end-user's activities without permission.

While it’s been shown that pairwise identifiers still allow linkability/correlation of identifiers under certain circumstances[25], they are a good idea in general and worth to support in id4me, just to “raise the bar”. However, it is counter-intuitive (to say the less) for the standard to allow for the client during the dynamic client registration process[26] to choose the Subject Identifier type they want to work with, specially if these are the very clients whose evil doings this feature wants to defend from.

One pragmatic solution for that could be to configure the id4me IdPs to only offer/support “pairwise” Subject Identifier types during client registration. That has proven a challenge, though, as one of the biggest identity server applications worldwide[27] (and the one being used in id4me at the moment) still does not have native support for this configuration option.

While still waiting for the “pairwise-only” configuration in the server to be possible, leakage of the public Subject Identifier value to clients has been found in our tests, even if the clients registered as “pairwise”. The public value was unwillingly disclosed by the server to the clients within the self-contained bearer access token (but

[22] (2013, January). RFC 6844 - DNS Certification Authority Authorization ... - IETF Tools. Retrieved January 18, 2018, from https://tools.ietf.org/search/rfc6844

[23] (2012, August). RFC 6698 - The DNS-Based Authentication of Named Entities (DANE .... Retrieved January 18, 2018, from https://tools.ietf.org/html/rfc6698

[24] DANE needs the DNS records to be signed with DNSSEC for its security model to work.

[25] (2009, September 24). Pairwise identifiers and linkability online (1/3) – Random Oracle. Retrieved January 18, 2018, from https://randomoracle.wordpress.com/2009/09/24/pairwise-identifiers-and-linkability-online-13/

[26] (2014, November 8). Final: OpenID Connect Dynamic Client Registration 1.0 incorporating .... Retrieved January 18, 2018, from https://openid.net/specs/openid-connect-registration-1_0.html

[27] 90 mio end-users as of July 2017, data provided by the vendor

only under certain configuration options). This bug has been communicated to the vendor and it will be fixed in the upcoming version of the software, expected end of January at the moment of this writing.

**9. Discussion / Future Work**

Formal methods in security engineering have led to discoveries of flaws and vulnerabilities in SSO deployments[28] and even in the protocols themselves[29]. id4me would benefit from a formal analysis as well and readers are welcome to get in touch with the authors if interested/capable of carrying out such analysis.

Plain bearer access tokens have some security drawbacks[30] and thus we would like to explore some proof-of-possession approach in the next implementation iteration, probably OAuth Token Binding [31].

At this point in time we are also studying the compatibility of id4me with similar initiatives also based on domain names, like MojeID[32], that could deliver a unified user experience.

[28] (2008, October 27). Formal analysis of SAML 2.0 web browser single sign-on: breaking the .... Retrieved January 18, 2018, from http://dl.acm.org/citation.cfm?id=1456397
[29] (2016, October 24). A Comprehensive Formal Security Analysis of OAuth 2.0. Retrieved January 18, 2018, from http://dl.acm.org/citation.cfm?id=2978385
[30] (2017, November 13). draft-ietf-oauth-security-topics-04 - IETF Tools. Retrieved January 18, 2018, from https://tools.ietf.org/html/draft-ietf-oauth-security-topics-04
[31] (2017, October 26). draft-ietf-oauth-token-binding-05 - OAuth 2.0 Token Binding - IETF Tools. Retrieved January 18, 2018, from https://tools.ietf.org/html/draft-ietf-oauth-token-binding-05
[32] (n.d.). MojeID. Retrieved January 18, 2018, from https://www.mojeid.cz/