Integrating Enterprise Security Infrastructure with Cloud Computing

Oliver Pfaff
Corporate Technology
Siemens AG
81739 Munich, Germany

Sebastian Ries
Corporate Technology
Siemens AG
81739 Munich, Germany

Abstract

Traditional enterprises have their roots in on-premises computing. They operate security infrastructure on-premise. This includes corporate user repositories, authentication and authorization systems. This infrastructure backs and enables workforce productivity. It has to be integrated when utilizing Cloud computing.

Hence traditional enterprises encounter specific integration challenges with respect to their security infrastructure when considering Cloud computing.

This text analyses these challenges, identifies best practice approaches and red-flags common pitfalls.

1. Introduction

Traditional enterprises that consider Cloud offerings[1] need to integrate their existing security infrastructure especially corporate user repositories, authentication as well as authorization systems. This infrastructure resides on-premises.

Its integration is crucial: the mentioned on-premises security infrastructure backs and enables the productivity of the workforce of an enterprise.

Moreover this integration is encountered with several types of applications in the Cloud including:

- Own applications deployed to IaaS or PaaS offerings
- 3rd party applications especially SaaS offerings

The initial situation is shown in Figure 1.

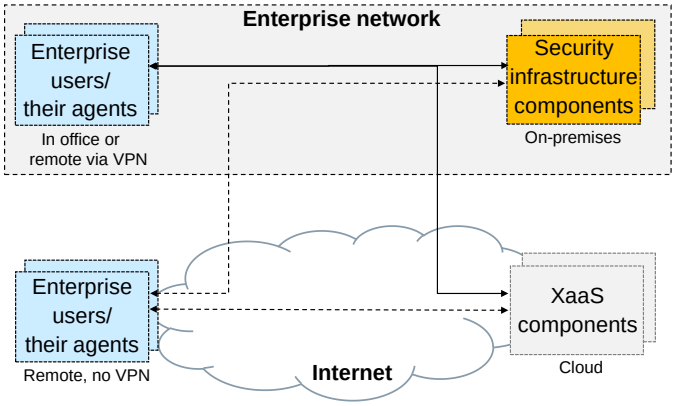

Figure 1. Initial situation

This integration needs to be implemented with care: several pitfalls do exist when it comes to integrating enterprise security infrastructure with Cloud computing. These pitfalls happen to present a major source of complication in Cloud adaptation projects for enterprise IT.

This text identifies important security integration pitfalls and does elaborate on them - especially their avoidance/ mitigation. This results in a checklist and suggestions for:

- Application owners and architects: *how to design applications whose deployment model includes Cloud and which shall integrate with existing security infrastructure?*
- Service owners, people in charge of security infrastructure: *how to expose security infrastructure when Cloud-based applications are being added?*
- Providers of Cloud-based services: *how to address the integration with security infrastructure that Cloud providers sustain on-premises?*

[1] This text puts a focus on deployments where XaaS components reside outside the internal network(s).

2. Anti-patterns

An anti-pattern is a "*common response to a recurring problem that is usually ineffective and risks being highly counterproductive*" (cf. http://en.wikipedia.org/wiki/Anti-pattern). This text uses this terminology to spot common pitfalls in security infrastructure integration.

2.1. XaaS requires on-boarding of enterprise users

XaaS offerings may require enterprise users to on-board in order to use services in the Cloud[2]. This on-boarding of enterprise users encompasses various implementation options such as user self-registration, classical or federated provisioning as well as user bulk loading.

Issues: The on-boarding of enterprise users may lead to the duplication of user identity information with resulting synchronization needs. Moreover, authentication account zombies might be created. This refers to user accounts in the Cloud which are equipped with initial authentication credentials (e.g. passwords) and that are affiliated to the enterprise[3]. Such objects become critical in case of leavers: usually enterprises are prepared to reconcile their on-premises security infrastructure within a limited timeframe of say 24 hours in case of events such as quitting. But for various reasons they are often not well-prepared to reconcile dependent objects in external systems. This presents a major security risk.

Mitigation: Inject all information about logged-in users by means of information-rich security token objects that are consumed by XaaS components (cf. anti-pattern 2.4). Avoid XaaS components that depend on data for not logged-in users resp. mitigate this concern in an adequate way (cf. anti-pattern 2.7).

Caveat: XaaS components which depend on user information that is not available in the enterprise security infrastructure. In particular this may be encountered with 3rd party SaaS offerings.

Recommendation: Limit XaaS user repositories to information not available in enterprise infrastructure. XaaS systems should not create own user repositories that duplicate information held in corporate user repositories. Avoid storing initial authentication credentials in XaaS user repositories. As a Cloud provider, expose an IdM service allowing subscribers to manage such information.

Aspects of user on-boarding aspects are illustrated in Figure 2. This figure shows critical (duplicated information, credentials) as well as uncritical (information not available in native repositories, identifier for correlation purposes) items.

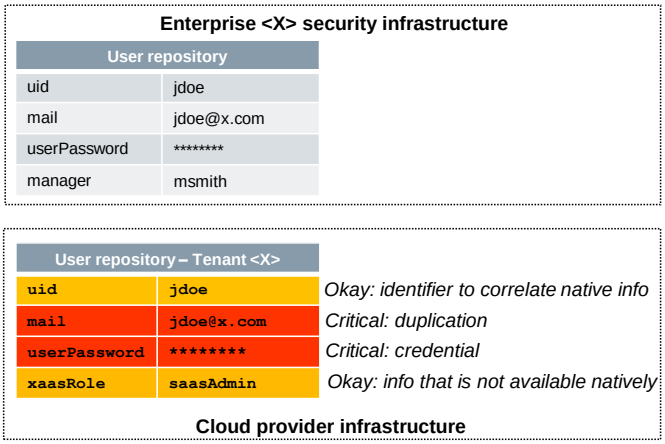

Figure 2. Critical aspects of user on-boarding

2.2. XaaS asks for initial authentication credentials

Before users are granted access to protected XaaS resources they need to be authenticated. XaaS components may implement this by challenging for initial authentication credentials by themselves. Examples for such credentials are: static passwords, one-time-passwords, security questions and answers. Their validation may happen against reference information that is stored in native enterprise infrastructure or XaaS offerings.

Issues: Lack of SSO user experience across on-premises and XaaS applications. Risk of identity theft in the case of long-lived credentials[4].

Mitigation: Retain the verifier role for initial authentication on-premises. Do not expose credential validation functionality[5]. Do expose authentication functionality[6]. Only reveal information about authentication events to the Cloud (in form of security tokens i.e. protected objects).

Caveat: A number of proposals exist to externalize initial authentication including federated IdM protocols such as SAML, WS-Federation and OAuth/OpenID Connect [1], [4], [5], [10]. This requires smart choices where relying parties (XaaS components) and asserting parties (in the enterprise infrastructure) have to be in synchronization.

Recommendation: XaaS components should not ask for initial authentication credentials by themselves. If user authentication is needed, XaaS components should redirect or direct[7] to asserting parties in the enterprise infrastructure.

[2] This refers to the overall enterprise user population or large subpopulations, not the administrators of a Cloud subscription only.

[3] User account in the Cloud for enterprise <X> user John Doe (*John Doe@Tenant<EnterpriseX>*)

[4] Some credentials are stored in hashed form esp. passwords in LDAP. In this case the original credentials (passwords) do not get exposed in the backend but might have frontend exposure

[5] Input: identifier, initial authentication credential

Output: Boolean, (opt.) state information

[6] Input: authentication request (without initial authentication credentials)

Output: security token reporting on initial authentication event

[7] Details depend on the application protocol and the type of user agent. For example: in case of HTTP, XaaS components should redirect (HTTP 30x responses) in case of Web browsers as user agents (so-called passive

Their duty is to perform user authentication and report on user authentication events.

Inappropriate ways of responding to not or insufficiently authenticated requests are sketched for HTTP in Figure 3.

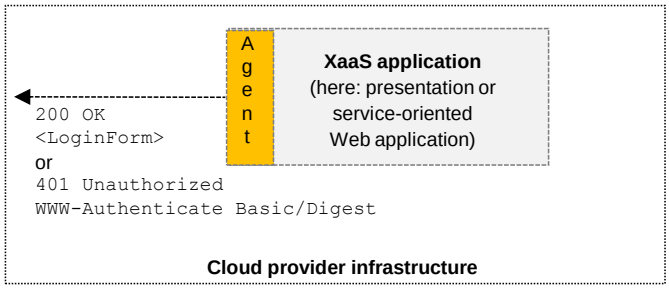

Figure 3. Inappropriate ways of responding to not or insufficiently authenticated HTTP requests

2.3. XaaS depends on backchannel interactions for SSO

To provide SSO user experience, XaaS components may depend on direct exchanges with services provided by the enterprise network. Such exchanges do not traverse the user agent. Examples are: artifact profile exchanges in case of SAML and UserInfo service exchanges in case of OpenID Connect. The utilization of backchannel interactions for the purpose of SSO can be mobile-friendly because it avoids supplying potentially complex security tokens across constrained user agents.

Issues: Backchannels place a burden on the enterprise. They mandate to make certain endpoints in the enterprise security infrastructure public-facing.

Mitigation: It is not needed for an asserting party for Web-based federated SSO to expose public-facing endpoints.

Caveat: Other use cases may require backchannels and/or public-facing endpoints for user authentication. Examples are SLO and the authentication of employee users accessing the enterprise infrastructure remotely via public network infrastructure only.

Recommendation: Provide SSO user experience through frontend exchanges to reduce the exposure of on-premises security infrastructure[8]. XaaS offerings should be prepared for handling front- and backchannel interactions.

Backchannel interactions for SSO use cases are illustrated in Figure 4.

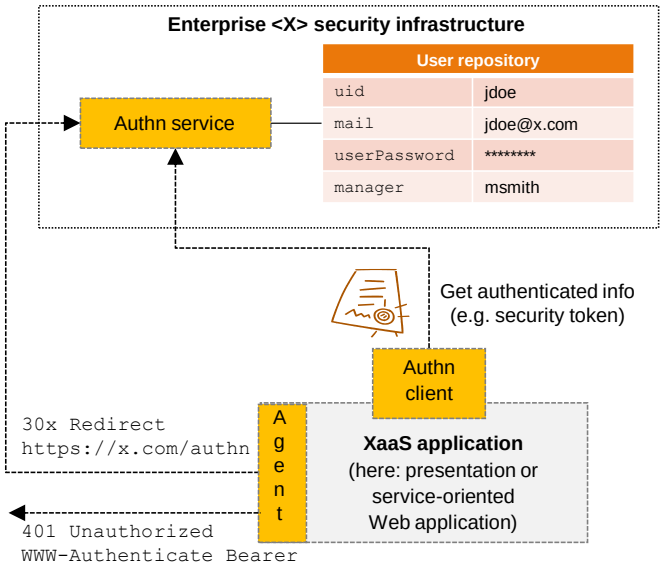

Figure 4. Backchannel interactions for SSO

2.4. XaaS pulls properties of logged-in users

After a XaaS application has authenticated a user[9], it may want to look up further properties by user identifier e.g. user attributes or affiliations (role assignments or group memberships). This implies querying an on-premises corporate user repository or user service.

Issues: In some cases, issues arise from security tokens that contain only partial information about logged-in users. In other cases, versatile security tokens are provided but downstream consumers honour their identifier portions only meanwhile they require supplementary user properties.

Mitigation: Change from pull to push models with respect to the properties of logged-in users.

Caveat: This requires information-rich, versatile security token objects sent from asserting parties in the enterprise security infrastructure to relying parties in XaaS. It also requires the coordination between endpoints terminating the federated authentication protocol on relying party side and the actual consumer of this information e.g. Web application in XaaS.

Recommendation: Inject information on logged-in users as part of the authentication and SSO process. Do not expect corporate repositories to be accessible for XaaS components.

Pulling properties of logged in users is shown in Figure 5. This figure shows the approach of accessing corporate user repositories by native means (which is to be avoided).

clients) and direct (HTTP 401 responses) in case of browser-based or mobile apps (so-called active clients).

[8] Unless public-facing endpoints are required for other reasons such as the support of constrained devices or clients.

[9] By consuming security tokens issued by asserting parties that are provided as a party of the enterprise security infrastructure.

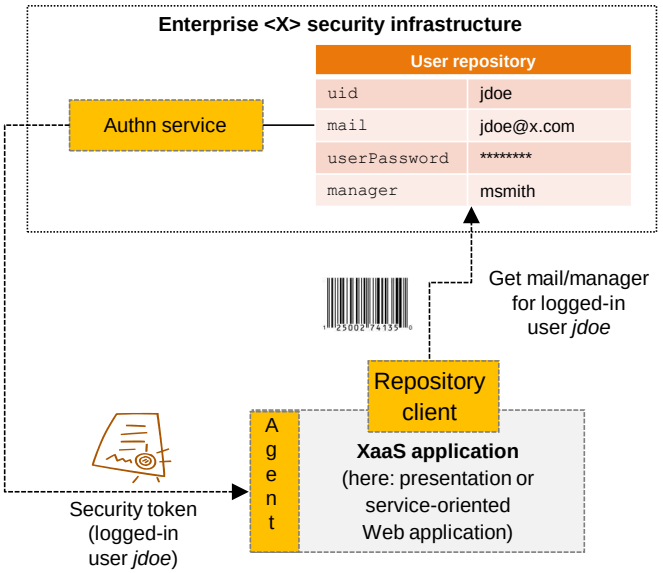

Figure 5. Pulling properties of logged-in users

2.5. XaaS cares for enterprise-initiated login only

The federated login might be implemented in a way that mandates users to first authenticate against the security infrastructure of the enterprise/enterprise before accessing any XaaS offering[10].

Issues: Users bookmark XaaS resources and browse from bookmarks. This is not addressed by enterprise-initiated login only and results in unsatisfactory user experience.

Mitigation: Even when the so-called 'IdP-initiated login' presents the designated user experience, XaaS as well as enterprise security infrastructure components must be prepared to handle XaaS-initiated logins.

Caveat: The handover of unauthenticated requests from XaaS to enterprise security infrastructure needs to be performed in a way that is transparent to the users.

Recommendation: Support XaaS-initiated as well as enterprise-initiated user login to XaaS components.

2.6. XaaS depends on backchannel interactions for authorization decision making

XaaS components may include PEP components that call remote PDPs for authorization decision making. The remote PDP is assumed to be provided as part of the security infrastructure of an enterprise. This can be done by means of e.g. XACML [8] context requests/responses.

Issues: Performance and availability become concerns if XaaS components interact with remote security infrastructure for the vast majority of requests. Note that amount as well as time-criticality of traffic between PEPs and PDPs (authorization decision requests and responses) usually is magnitudes larger than between PDPs and PAPs (authorization policies and other instructions).

Mitigation: Allocate PDP components at XaaS. Supply policy/rule objects (e.g. XACML policy objects) resp. configuration information (e.g. resource listings or classifications) to them.

Caveat: Requires mutually agreed standards for managing the lifecycle of authorization policy objects.

Recommendation: Keep authorization decision making local (here: in the Cloud). If needed: plan for a Cloud-hosted PDP component which is supplied with the company's policy/rule objects resp. configuration information.

2.7. XaaS asks for data of not logged-in users

While processing requests of logged-in users, XaaS applications may need to access data of other users (who may but do not have to be logged-in simultaneously) e.g. to send out email notifications. For further details about such use cases see [7].

Issues: Enterprises are usually unwilling to provide access to their corporate user repositories to clients from the Internet via native repository protocol means (e.g. LDAP and SQL) as this would put core assets at risk.

Mitigation: Implement and expose a dedicated service (e.g. RESTful HTTP service according SCIM [3]) that is security-enabled and that provides a façade for querying the native user repository in a specific and constrained fashion[11]. As XaaS component: support externalization of repository clients e.g. use plug-in interfaces to deploy service clients.

Caveat: XaaS offerings that internalize or hardcode the user lookup implementation by e.g. implementing native repository protocols such as LDAP or SQL especially when this implementation does not provide sufficient means of configuring the repository endpoints as well as contents structure.

Recommendation: Create a dedicated service (e.g. RESTful HTTP service) to query information about native users. Closely monitor this service and make sure that its clients must register and are authenticated and authorized on a sound level. Do not expose user repositories to external repository clients in their native form, i.e. via LDAP or SQL.

Pulling properties of not logged in users is shown in Figure 6. This figure shows the approach of accessing corporate user repositories by native means (which is to be avoided).

[10] Aka: IdP-initiated exchanges or unsolicited responses (SAML 2.0)

[11] Query response contents filtered according the approved needs of the client representing the XaaS application

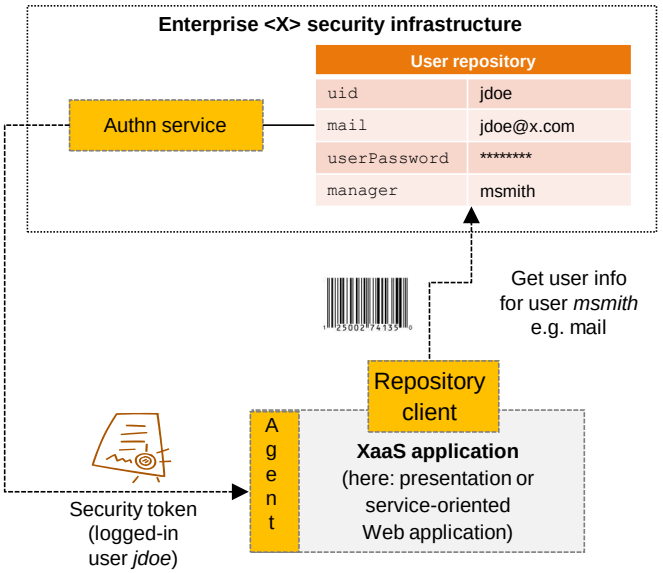

Figure 6. Pulling properties of not logged-in users

2.8. XaaS depends on backchannel interactions for other use cases e.g. auditing

XaaS components may send requests to on-premises responders in order to fulfil other security-related use cases such as the dropping of auditing data.

Issues: Inbound requests to on-premises responders are typically blocked by a company's firewall and would require changes to the configuration of the network boundary.

Mitigation: Use on-premises clients to establish a (bidirectional) connection to XaaS applications (e.g. WebSocket, AMQP). After this has been established, the client may receive information from XaaS applications.

Caveat: This requires additional implementation efforts on side of the XaaS components (e.g. message brokers) and their subscribers.

Recommendation: Rely on the "*we call you – don't call us*" pattern of interaction: use on-premises clients which establish (bidirectional) connections to XaaS applications.

2.9. XaaS uses proprietary Web SSO protocols

Web applications that are protected with Web SSO agents which are provided by the company's chosen Web SSO product shall be moved from internal networks to IaaS or PaaS offerings in the Cloud.

Issues: Native SSO schemes in many traditional Web SSO products suffer from limited authentication request expressiveness as well as non-versatile security tokens. This often results in difficulties when trying to implement use cases around re-authentication as well as step-up authentication. In contrast to applications that reside on-premises such schemes become a need in order to protect public-facing XaaS endpoints.

Mitigation: Abandon legacy Web SSO mechanisms. Switch to standards-based Web SSO protocols.

Caveat: There is a plethora of standards-based Web SSO protocols to choose from. Candidates include SAML, WS-Federation and OAuth/OpenID Connect.

Recommendation: Use standards-based Web SSO protocols emphasizing on authentication request expressiveness and security token versatility.

2.10. XaaS does not provide public-facing endpoints

Access to XaaS components might be limited to computing devices from the enterprise network. Devices that reside outside the enterprise premises must use VPN technology[12].

Issues: Usage of VPNs presents issues for users of constrained computing devices e.g. tablets or smart-phones.

Mitigation: As XaaS component: support browser-based clients by exposing public-facing endpoints for HTTP and WebSocket by means of the URL access schemes `https` resp. `wss` i.e. HTTP resp. WebSocket-over-SSL/TLS[13].

Caveat: Public-facing XaaS endpoints require new means of protection e.g. throttling or context-aware adaptive user authentication including mechanisms such as re-authentication and step-up authentication. Corresponding mechanisms are uncommon in '*behind the firewall*'-enterprise IT.

Recommendation: Use/create XaaS offerings that expose public-facing endpoints especially HTTP and WebSocket-over-SSL/TLS. Ensure/implement means of protection that extend beyond the enterprise IT best current practices.

3. Conclusions

This text identified anti-patterns of integrating existing enterprise security infrastructure with Cloud computing[14]. Such anti-patterns are often encountered. They present a major source of complication in Cloud adaptation projects.

The identified anti-patterns are related to:

- *Communication/network infrastructure*: anti-patterns 2.3, 2.4, 2.8 and 2.10
- *Authentication repositories/systems/protocols*: anti-patterns 2.1, 2.2, 2.3, 2.5 and 2.9
- *Attribute/affiliation repositories*: anti-patterns 2.1, 2.4 and 2.7
- *Authorization systems*: anti-pattern 2.6

No matter whether one is creating or sustaining a Cloud computing offering or planning to use it as a subscriber, the identified security infrastructure integration anti-patterns are to be avoided or at least mitigated.

This text provided suggestions on their avoidance and mitigation.

[12] In contrast to the anti-patterns above, in which the integration of enterprise security infrastructure imposes challenges on the Cloud, this anti-pattern arises from over-utilizing the enterprise's security infrastructure.

[13] Enterprise firewalls are usually configured to accommodate such outbound traffic; the default port 443 is usually open for outbound traffic.

[14] See [9] for an overview of identity-related Cloud use cases.

This text did not cover Cloud security concerns beyond the integration with existing enterprise security infrastructure such as the operational security of Cloud offerings. Additional criteria apply to assess the overall security of a Cloud provider (cf. [2], [6]).

4. Abbreviations

AMQP	Advanced Message Queuing Protocol
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
IAM	Identity and Access Management
IdM	Identity Management
IdP	Identity Provider
LDAP	Lightweight Directory Access Protocol
OAuth	Open Authorization
PaaS	Platform as a Service
PAP	Policy Administration Point
PEP	Policy Enforcement Point
PDP	Policy Decision Point
REST	REpresentational State Transfer
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SCIM	System for Cross Domain Identity Management
SLO	Single Log-Out
SQL	Standard Query Language
SSL	Secure Sockets Layer
SSO	Single-Sign-On
TLS	Transport Layer Security
URL	Uniform Resource Locator
VPN	Virtual Private Network
WS	Web Services
XaaS	Any <X> as a Service including IaaS, PaaS, SaaS
XACML	eXtensible Access Control Markup

5. References

[1] S. Cantor, J. Kemp, R. Philpott and E. Maler (eds.): Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 2005. (http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)

[2] Cloud Security Alliance: Security Guidance for Critical Areas of Focus in Cloud Computing v3.0, 2011. (https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf)

[3] T. Drake (ed.): System for Cross-Domain Identity Management: Protocol. IETF Draft (work-in-progess), 2014. (https://tools.ietf.org/html/draft-ietf-scim-api-13)

[4] M. Goodner and A. Nadalin (eds.): Web Services Federation Language (WS-Federation) Version 1.2, OASIS Standard, 2009. (http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.html)

[5] D. Hardt (ed.): The OAuth 2.0 Authorization Framework, IETF RFC 6749, 2012. (http://tools.ietf.org/html/rfc6749)

[6] W. Jansen and T. Grance: Guidelines on Security and Privacy in Public Cloud Computing. NIST Special Publication 800-144, 2011. (http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf)

[7] M. Oosdijk, B. Hulsebosch, N. van Dijk, R. van Rijswijk and H. Zandbelt: Provisioning scenarios in identity federations. Surfnet Research Paper, 2010. (http://www.surf.nl/binaries/content/assets/surf/en/knowledgebase/2010/EDS-4+Provisioning+Scenarios+in+Federations+Final.pdf)

[8] E. Rissanen (ed.): eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS Standard, 2013. (http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.doc)

[9] M. Rutkowski (ed.): Identity in the Cloud Use Cases v1.0. OASIS Committee Note 2012 (http://docs.oasis-open.org/id-cloud/IDCloud-usecases/v1.0/cn01/IDCloud-usecases-v1.0-cn01.html)

[10] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros and C. Mortimore: OpenID Connect Core 1.0. OpenID Foundation Standard, 2014. (http://openid.net/specs/openid-connect-core-1_0.html)