miaa Guard
taking care of digital access

Externalised access model

miaa Guard's proposition puts access management outside of the applications, so-called "externalisation." This article zooms into the externalisation of three distinct aspects of access management:

- identity of a consumer or a professional in relation to the enterprise
- access to digital services according to the enterprise's business policy
- authorisation to perform an action onto a specific object

This article demonstrates how the combination of enterprise-specific identifiers, granular consents and ecosystem relations greatly simplifies application design, and yet, gives more control, security and flexibility with self-service workflows.

Note: one may observe that the name "miaa Guard" stands for guarding managed identity, access and authorisation.

Externalising identity management

To protect a user's privacy, miaa Access Management manages the identity of a user in the secure repository of an established Identity Management system.

> Note: It is good practice to externalise identity management and take it out of webservers and e-commerce platforms so that privacy-sensitive data and passwords are only managed within the Identity Management system through miaa Access Management.

In order to prepare access and authorisation decisions, miaa Access Management not only maintains general characteristics (core-identifiers) but also characteristics in relation to the enterprise (relationship-identifiers). While core-identifiers may include gender and birthdate, relationship-identifiers are unique for the enterprise and include claims such as "I've paid for this subscription," and "I'm a certified doctor."

In the words of Gartner, Homan Farahmand in [R12]: "Enterprises need to view core-identifiers and relationship-identifiers as two distinct constructs. Core-identifiers can be [federated], but each enterprise has to manage its own relationship-identifiers for each person to address other identity governance and access management requirements."

miaa Access Management verifies a user's claim against an authoritative source, such as a national register and an enterprise's CRM. After this so-called *identity proofing*, the status of the claim is updated from 'pending' to 'confirmed' or 'rejected.' Confirmed claims can then be used for access and authorisation, as described below.

> Example: Users can claim to be a doctor in a specific country. miaa Access Management can then verify this claim against Iqvia IMS Health.

OIDC 1.0 [S2] standardises the representation of identity in JWT tokens. In combination with ISO 29115:2013 [S4], it further standardises how the level of authentication is passed on by miaa Access Management in the form of "level of identity assurance."

> Note: It is good practice to take two-factor authentication out of the applications. Authenticating a user, however, should not be confused with 'granting access' (see below).

© 2019 miaa Guard NV – all rights reserved.

Conclusion

Externalising identity management yields significant advantages relative to privacy, identity assurance and identity proofing.

Externalising access management

As opposed to traditional monolithic applications, a microservice-based architecture comprises multiple loosely coupled modules ("microservices") that serve a specific business purpose. Such services can be developed, deployed and scaled independently, which greatly improves agility, robustness and consistency.

> Example: A clinic can decouple lab and health care management from actual patient management. When implemented as distinct microservices, a lab-oriented app and a consumer app can build on the same patient management microservice.

A microservice-based architecture poses some security challenges. As Gartner E. Wahlstrom said in [R9]: "when designing a microservice architecture, dealing with identity and security becomes a much more complicated task than in traditional monolithic applications: each individual component must know which user is interacting with it and which authorisations are granted to him. Only protecting services at the gateway would be an anti-pattern."

OAuth 2.0 [S1] standardises tokenised access control to enable loose coupling of microservices in a secure and efficient way. Good practice leverages the signed bearer access tokens that provide details about the access rights and are secured with the issuer's signature. It greatly simplifies *coarse-grained access control* in that an application only needs to verify the access token to decide whether a user has access or not.

To refine access control, Role-based Access Control ("RBAC") is often used because of its apparent simplicity. In RBAC, a role combines a number of permissions needed to perform a specific function of an application.

Traditional systems assign RBAC-roles to users, as standardised by NIST in [R1]. But since RBAC-roles essentially reflect a function of an application, this static assignment poses a number of challenges, as researched in [R2]:

- because roles are static and assigned at registration, the permissions are given to the user forever, until they are revoked explicitly.
- because roles are static, taking environmental aspects (such as country) into account is a challenge. In fact, roles are often duplicated to include each environmental aspect, which leads to so-called 'role explosion'.
- because roles are assigned at registration and represent permissions, they cannot be self-serviced and their assignment needs to be reviewed regularly and audited.

To overcome the issue of static roles, miaa Access Management's rule engine takes RBAC-roles into account but supplements this with contextual and user-related information. This so-called attributed-based access control ("ABAC") was first researched in [R3] and later standardised by NIST in [R4]. miaa Access Management puts its rule-based decisions in the JWT access token so that a microservice can immediately grant/deny access to a function just by interpreting the token.

> Example: Access tokens can be designed to include decisions such as 'has the right to use the update-patient function'. This RBAC-related decision, however, should not be confused with 'is authorised to update this particular patient record' which is an authorisation

© 2019 miaa Guard NV – all rights reserved.

decision (see below). Authorisation decisions are typically not put in an access token but are obtained through an API call.

Another limitation of traditional RBAC systems is its focus on the target application; RBAC-roles ignore any agreement or disagreement of the user relative to the policies adopted by the enterprise (so-called 'consent'). With UMA 2.0 [S5], the Kantara group proposes a standard for user-managed access as an extension of OAuth 2.0.

miaa Access Management fully adopts this emerging standard and makes any access decision first subject to the user's consent. The user thereby grants the enterprise authorisation to process his data in a very granular way by indicating the scope, the purpose and the period, as well as details about the corresponding policy in terms of language, territory and version.

Conclusion

Externalising access management yields significant advantages thanks to tokenised access control for microservices and rule-based access control to overcome the limitations of RBAC.

Externalising authorisation management

According to OAuth 2.0, access tokens are constructed before the user actually gains access to a system. As such, they are not designed to capture authorisation decisions relative to a particular object within that system after the user has gained access.

> Example: While access tokens may contain for example 'has the right to use the update-patient function,' they typically do not go as deep as 'is authorised to update this particular patient record'.

Authorisation, also known as *fine-grained access control*, focuses on the ultimate transaction. A transaction has a subject (the authenticated user) and an action (e.g. update-patient) but also an object (the patient record) and a context (e.g. country, level of assurance).

While a microservice-based architecture offers a separate microservice for different business purposes, the function of taking authorisation decisions can equally be offered as a microservice. Taking authorisations out of applications simplifies microservice development and makes them independent of policy evolutions.

> Example: The distinction between access by a care giver, a patient, a doctor, a nurse and a lab operator does not need to be made by a patient-management system. The patient-management system only needs to receive a decision: can this user, within this context, execute this action on this patient record. Evaluating the policy rules is then delegated to the authorisation microservice.

From a control perspective, externalised authorisation ensures consistency between applications and ensures security that takes a wider context into account beyond the scope of each microservice.

> Example: Apps that target different user segments (consumers versus care givers versus doctors) can access the patient-management system and be subject to always the same business policy. For example, before the nursing-app presents patient details on a user's device, it calls the API 'check authorisation for this patient-record.' From then on, it can allow the user to view, update or delete patient details.

© 2019 miaa Guard NV – all rights reserved.

XACML 3.0 [S3] standardises externalised authorisations. It defines a declarative fine-grained, attribute-based access control policy language, an architecture, and a processing model describing how to evaluate access requests according to policy-rules.

miaa Access Management offers the API adopting this standard for on-the-spot authorisations. Externalised authorisations using the standard also enable an API gateway to be adopted that controls access using externalised authorisation decisions.

> Example: An architecture based on Mulesoft can perform fine-grained access control based on authorisation decisions externalised to miaa Access Management. By caching the authorisation decisions locally, access control can happen within a few milliseconds.

Conclusion

Externalising authorisation management yields significant advantages because it greatly simplifies application design while it offers top-level security, very precise authorisation decisions and application-independent control.

Relation-driven authorisation management

While rule-based access control already resolves some of the static aspects of RBAC and many ABAC implementations, digital services that serve an ecosystem of users need to be much more dynamic and take the ever-changing interrelationships into account. Graph-based access control ("GBAC") introduced by Carminati et al [R7] builds on the generic lattice-based access control ("LBAC") model introduced by Koch et al [R6]. It represents a model to infer authorisation decisions using social relationships.

miaa Access Management builds on the model of Carminati et al based on a so-called 'directed graph' to establish relations between users and shared objects. Shared objects can be a patient's health record, a magazine subscription, an employer, etc. This directed graph of Carminati et al has a number of mathematical properties, e.g.

- 'inverse closure' that implies that every relation has an equivalent inverse relation

  Example: The relation 'has caring role for' has an inverse equivalent 'is patient of' so that queries can be done in both directions.

- 'forward chaining' that implies that groups of users can be formed to share the relations of that group with its members.

  Example: The model allows to extend doctor-patient relationships and to link relations to a group of users. This way, people belonging to a team (or family) share the patients with which the team is related. The resulting access rights, however, are still subject to rules that include attributes such as 'qualification'. This way a distinction can be made between a nurse and a doctor.

miaa Access Management also builds on the refinements of Fong et al in [R8] that assume that there exists a standard taxonomy of relationship types specific to the domain of application.

> Example: An enterprise can define its own relationship types, e.g. 'has caring role for,' 'is family member of,' 'is primary contact for,' 'is employee of,' etc.

miaa Access Management further extends the model to include *state*. The model of state transitions introduces intuitive workflows for self-service administration. For example, a user can

© 2019 miaa Guard NV – all rights reserved.

invite another user to establish a relation with a shared object, given the relation that he himself already has and given some other attributes of the user. The state of the relation can then evolve from pending to accepted or rejected.

> Example: An enterprise can define a workflow where a nurse has a relation 'has caring role for' which entitles her to invite the patient's mother to have a 'is family member of' relation with one of her patients.
>
> An enterprise can also define a workflow in which a patient is entitled to invite her doctor to establish the 'has caring role for' relation with her patient. Or a workflow in which a doctor invites his nurse to confirm the 'is employee of' relation with the clinic.

miaa Access Management also enables self-healing workflows such as an expiration workflow (whereby a relation has a limited lifetime) and an exclusion workflow (whereby a relation is deactivated for reasons of unpaid bills, misbehaviour or resignation).

miaa Access Management also enables self-enrolment workflows whereby a user declares to have a relation with a shared object, which must be confirmed/accepted by another user also having a relation with the shared object and some required attributes.

> Example: An enterprise can define a workflow whereby a nurse enrols herself as employee of a clinic, which is then confirmed by the responsible doctor. Or a workflow where a daughter can enrol as family member of the patient.

miaa Access Management also builds on the user-managed authorisation framework of UMA 2.0 [S5] and extends user consent to relationship types. This effectively provides for an intuitive model to manage user-to-user authorisations, whereby a user can accept or reject another user to be connected to the same shared asset, and whereby a user can refine the associated access rights.

> Example: A patient may still deny access to his records for certain people, even though they could have access thanks to their role and qualification.

Conclusion

Relation-driven authorisation management greatly simplifies application design because the design only needs to focus on its core function rather than on managing the ecosystem. In fact, managing the eco-system and deriving the authorisations from that eco-system has proven to be a science in its own right.

References

[S1] OAuth 2.0 - IETF **OAuth 2.0** Working Group RFC 6749 and RFC 6750, the industry-standard protocol for authorisation, October 2012.

[S2] OIDC 1.0 - **OpenID Connect 1.0** specifies a standardised identity layer on top of the OAuth 2.0 protocol, OpenID Foundation https://openid.net/connect/, November 2014.

[S3] XACML 3.0 - **eXtensible Access Control Markup Language 3.0**, OASIS. Defines a declarative fine-grained, attribute-based access control policy language, an architecture, and a processing model describing how to evaluate access requests according to the rules defined in policies, January 2013.

[S4] ISO/IEC 29115:2013 - **Entity authentication assurance framework**, ITU-T Recommendation X.1254, November 2013.

© 2019 miaa Guard NV – all rights reserved.

[S5] UMA 2.0 - **User-Managed Access 2.0** is a federated authorisation framework that defines an extension OAuth 2.0 grant type, Kantara.org, August 2016.

[R1] Sandhu, Ravi & Ferraiolo, David & Kuhn, D. (2000). **NIST model for role-based access control**: Towards a unified standard. Proceedings of the ACM Workshop on Role-Based Access Control. Pages 47-63.

[R2] Chen Liang & Crampton Jason, (2009), **Set covering problems in role-based access control**, Journal Proceedings of the 14th European Symposium on Research in Computer Security (ESORICS), Pages 689-704, Springer Berlin/Heidelberg.

[R3] Kuhn, D & J. Coyne, Edward & R. Weil, Timothy. (2010). **Adding Attributes to Role-Based Access Control**. Computer. Pages 43, 79-81. 10.1109/MC.2010.155.

[R4] Hu, Vincent & Ferraiolo, David & Kuhn, D & Schnitzer, A & Sandlin, K & Miller, R & Scarfone, Karen. (2014). **Guide to attribute based access control** (ABAC) definition and considerations. National Institute of Standards and Technology Special Publication. Pages 162-800.

[R5] Sebastian Ryszard Kruk, Slawomir Grzonkowski, Adam Gzella, Tomasz Woroniecki, and Hee-Chul Choi (2006). D-FOAF: **Distributed identity management with access rights delegation**. In Proceedings of the First Asian Semantic Web Conference (ASWC'06), volume 4185 of Lecture Notes in Computer Science, pages 140–154, Beijing, China, September 2006.

[R6] Koch, Manuel & Mancini, Luigi & Parisi Presicce, Francesco. (2005). **Graph-based specification of access control policies**. Journal of Computer and System Sciences. 71. 1-33. 10.1016/j.jcss.2004.11.002., November 2004.

[R7] Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu, and Bhavani Thurainsingham (2009). **A semantic web based framework for social network access control**. In Proceedings of the 14th ACM Symposium on Access Control Models and Technologies (SACMAT'09), pages 177–186, Stresa, Italy, June 2009.

[R8] Fong, Philip. (2011). **Relationship-based access control**: Protection model and policy language. CODASPY'11 - Proceedings of the 1st ACM Conference on Data and Application Security and Privacy. 191-202. 10.1145/1943513.1943539, January 2011

[R9] Wahlstrom E., (2018) Session **Integrate Identity With Microservice-Based Applications** at Gartner Identity & Access Management Summit in London on 5-6 March 2018.

[R10] Farahmand H., (2018) Does Decentralized Identity Need an Identity Neutrality Manifesto, Gartner Blog Network, July 12, 2018.

© 2019 miaa Guard NV – all rights reserved.