Federated Privileged Identity Management for Break-the-Glass: A Case Study with OpenAM

Davy Preuveneers and Wouter Joosen

iMinds-DistriNet, Department of Computer Science, KU Leuven, Leuven, Belgium
`first.last@cs.kuleuven.be`

Abstract. As next generation health monitoring and Ambient Assisted Living applications are opening up towards a variety of stakeholders and platforms, enforcing secure and reliable access to patient data by authorized users − anytime and anywhere − is paramount. However, static access control policies do not offer the flexibility to deal with unanticipated emergency situations where access to patient information is essential. In this paper, we discuss privileged identity management and share our experience obtained in a case study using OpenAM − a state-of-the-art open source federated identity and access management solution − for the implementation of a reusable policy for break-the-glass procedures in federated healthcare services.

Keywords: identity, access, break-the-glass, policy, e-health

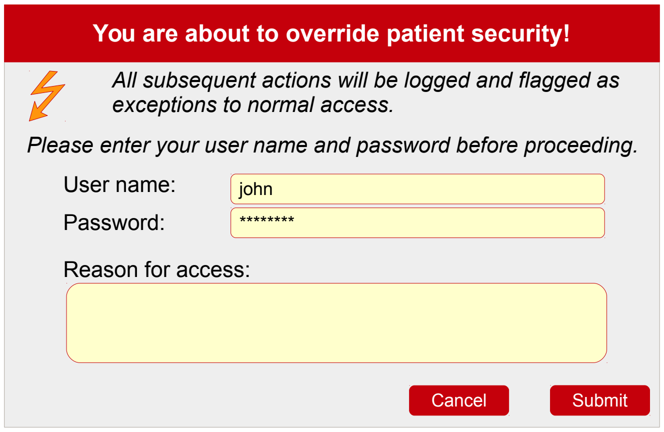
1 Introduction

There is a growing interest from policy makers, academics, providers and consumers in e-health [1] to improve the efficiency of healthcare service delivery. Innovations in the technological space − ranging from mobile applications and websites dedicated to patient self-healthcare, towards electronic exchange of patient records between care givers − have emerged to achieve this objective.

With cloud computing stepping up as the next paradigm shift in distributed systems and services, future e-health applications will be outsourced to the internet. Therefore, the number of platforms and stakeholders will continue to increase. As a consequence of this technological trend, the distributed nature of information and health data repositories calls for stringent access constraints to safeguard the patient's privacy, but without jeopardizing his health or safety due to rigid enforcement of such security policies.

The main purpose of identity and access management (IAM) platforms is to address authentication, authorization, access and auditing as a common concern

In: E.A.A. Jaatun, E. Brooks, K.E. Berntsen, H. Gilstad, M. G. Jaatun (eds.): Proceedings of the 2nd European Workshop on Practical Aspects of Health Informatics (PAHI 2014), Trondheim, Norway, 19-MAY-2014, published at http://ceur-ws.org


You are about to override patient security!

All subsequent actions will be logged and flagged as exceptions to normal access.

Please enter your user name and password before proceeding.

User name: john

Password: ********

Reason for access:

Fig. 1: An example of a *Break-the-glass* security warning

for online service providers. The added value that many of these IAM solutions have to offer is their capability of federated single sign on (SSO), simplifying access to partnering healthcare services with a single login. However, to minimize the risk of unintentionally revealing sensitive data, fine-grained access control for a restricted set of authorized users is a key information security requirement.

Contrary to many other vertical domains, consumers of healthcare IT applications can be faced with emergency or disaster situations where the ability to override access control on demand is a necessity, as rigid enforcement of security constraints should never risk patient health or safety. Such access is enabled by a *Break-the-glass* [2] procedure. This is the practice of enabling a healthcare practitioner to view a patient's medical record, or a portion thereof, under emergency circumstances, when that practitioner does not have the necessary system access privileges. This access is attributable to an authenticated user, temporary and auditable. The way this *break-the-glass* functionality is usually offered in software applications boils down to prompting the user with a warning that he does not have the rights required to access the necessary information, as shown in Fig. 1. An explanation of the required emergency access must be provided, and the action will be audited.

Electronic patient record (EPR) systems offered as Software-as-a-Service (SaaS) applications in the cloud have the benefit of scaling up on demand and reducing the total cost of ownership by sharing the same service instance across multiple tenants (i.e. the healthcare practices or clients of the multi-tenant SaaS application). Even if different service variants exist to address the unique needs of various types of healthcare professionals (e.g. general practitioners, dentists, ophthalmologists, out-of-hours service centers), they can share common software assets to authenticate and authorize healthcare professionals. However, these service variants may be subject to different legislative rules and contextual constraints for break-the-glass procedures to gain access to medical records. For example, a GP in out-of-hours service center cannot access a patient's record if he is not on duty. Multi-tenant EPR SaaS applications therefore need customization capabilities to adapt medical record access control (1) per tenant or healthcare practice, and (2) per service variant. These concerns are the main motivation why access control and the enforcement of break-the-glass procedures should be decoupled from the underlying service logic.

This paper focuses on privileged identity management to control the access of authorized users and other identities to elevated privileges across multiple systems deployed at a healthcare service provider. In general, privileged identity management focuses on managing and securing privileged accounts and controlling access to sensitive data. Privileged identities are accounts that hold elevated permission to access data, run programs, or change configuration settings. Traditionally, identity and access management (IAM) solutions do not manage such accounts, which makes it almost impossible to:

1. identify the list of privileged identities and login credentials that exist in the system or on the network.
2. enforce compliance with strong password management and authentication policies for privileged identities.
3. limit the individuals who know about these privileged login credentials to mitigate risks associated with insiders misusing these identities.
4. delegate access to privileged accounts for appropriate personnel to reduce the number of privileged accounts required by an organization.
5. increase accountability with fine-grained activity logging to audit who gained access to system resources or data using these privileged identities.

The objective is to enable separation of privileges, manage self-service requests to privileged accounts, and provide auditing capabilities. In most organizations, privileged accounts are not clearly defined and when they are not tightly managed, they present a high security risk for the organization. Detecting inappropriate access to privileged accounts and determining which healthcare providers were involved in unauthorized activities (through a break-the-glass procedure) is challenging. We share our experience with using ForgeRock's OpenAM [1] – a state-of-the-art open source federated identity and access management solution – for the implementation of a reusable policy for a variety of break-the-glass procedures.

After reviewing related work in section 2, we present an electronic healthcare record related use case with break-the glass procedures in section 3 as a motivating example. Section 4 discusses the access control primitives used in OpenAM, and the actual implementation is presented in section 5. In section 6 we evaluate the strengths and weaknesses of our implementation. We conclude in section 7 summarizing the main insights and identifying possible topics for future work.

[1] http://forgerock.com/products/open-identity-stack/openam/

2 Related work

Povey [3] was one of the first to argue for a break-the-glass concept as a way to deal with the inability to encode each and every access control policy of an organization because not all kinds of requests and conditions can be anticipated. He argued that regardless of how flexible or expressive access control mechanisms become, there will always be a gap between what organizations need, and what security mechanisms can actually implement. He proposed an access control paradigm involving partially-formed transactions whose effects can be rolled back, allowing users access to restricted resources if they are willing to acknowledge that they do not have the right permission.

The eXtensible Access Control Markup Language (XACML) [4] specification is the most well known XML-based language for expressing security policies. Security policies are ways to describe who has access to what resources under what conditions. The main purpose of XACML is to allow for security policies to be defined in a technology neutral way such that the policies can be reused. Version 3 of the XACML specification was approved as an OASIS standard in January 2013. While the first draft of the XACML v3.0 specification emerged in 2009, there is not yet widespread adoption of this standard in contemporary identity and access management systems.

Decat et al. [5] propose a middleware for efficient and confidentiality-aware federation of access control policies in the context of cloud computing and Software-as-a-Service (SaaS) in which a tenant rents access to a shared web application hosted by a provider. They propose a mechanism for policy federation, the ability to decompose tenant policies and evaluate the resulting parts near the data they require as much as possible while keeping sensitive tenant data local to the tenant environment. With a case study on home patient monitoring, the authors show that policy federation effectively succeeds in keeping the sensitive tenant data confidential and at the same time improves policy evaluation time in most cases.

Alqatawna et al. [6] introduced a discretionary overriding mechanism in XACML. This is one way for handling hard to define and unanticipated situations where availability of information is critical. The override mechanism gives the subject of the access control policy the possibility to override a denied decision, and if the subject should confirm the override, the access will be logged for special auditing. The authors achieved this capability by means of obligations and a general obligation combining mechanism.

Similar work was proposed by Brucker et al. [7]. They extended SecureUML [8] − an UML-based security language for access control − with support for break-the-glass strategies. They presented a security architecture with corresponding support as well transformation capabilities from break-the-glass SecureUML policies into XACML.

Rumpole [9] is a flexible break-the-glass access control proposed Marinovic et al. This model aims to address concerns that current access control models make decisions without considering and investigating the reasons why access is denied, and that these models do not explicitly represent and reason over conflicting and missing information about subjects and the context. The access control model that they propose uses Belnap's four-valued relevance logic [10] to represent conflicting and missing (unknown) information, allowing the policy to make a more informed decision when faced with missing or inconsistent knowledge. In traditional logical calculus, there are only two possible values (*true* and *false*) for any proposition. Belnap's four-valued logic is an example of a many-valued logic in which there are more than two truth values. It deals with multiple possibly conflicting information sources. The possible values are (1) *true* if only true is found, (2) *false* if only false is found, (3) *true and false* if both conflicting values are found, and (4) *unknown* if no information is provided by any source.

With federated privileged identity management, our objective is not to complicate access control policies even further with break-the-glass procedures, but rather to keep the specification and managerial complexity of such policies in a federated health service ecosystem as low as possible.

3 Use cases of break-the-glass and emergency scenarios

As the objective is to offer simplified management of emergency access and break-the-glass procedures through reusable access policies, we will document three real-world use cases (applicable in a Belgian context) involving elevated privileges to access electronic medical records with different types of healthcare service providers and caregivers. The aim is to determine whether or not a healthcare provider or caregiver has permission to perform an action on a resource (e.g. consulting a medical file) for a given patient as a function of a given context. Fig. 2 illustrates a simplified workflow for access control to a medical file with contextual constraints and patient consent directives. Note that this workflow also allows patients to be given a stark choice between participating or opting out, with a warning that opting out means that they are also refusing access to their medical records in emergency or break-the-glass circumstances.

3.1 General practitioner in a medical center during office days

Dr. Smith, a general practitioner, works in a medical center where he and his colleagues administer first line primary care to patients during office hours. The medical center hosts a central server containing a list of medical documents per patient, where the GPs keep the medical history of their patients on file.

For a doctor to gain access to the medical records of a patient, a therapeutic relationship between both the patient and doctor needs to be established and patient consent directives apply. For example, a patient can prohibit access to a medical file in general or only for particular healthcare professionals.

3.2 General practitioner on call in an out-of-hours service center

When their own GP is not available (e.g. at night or during the weekends or holidays), patients with pressing medical problems can call a general practitioner's

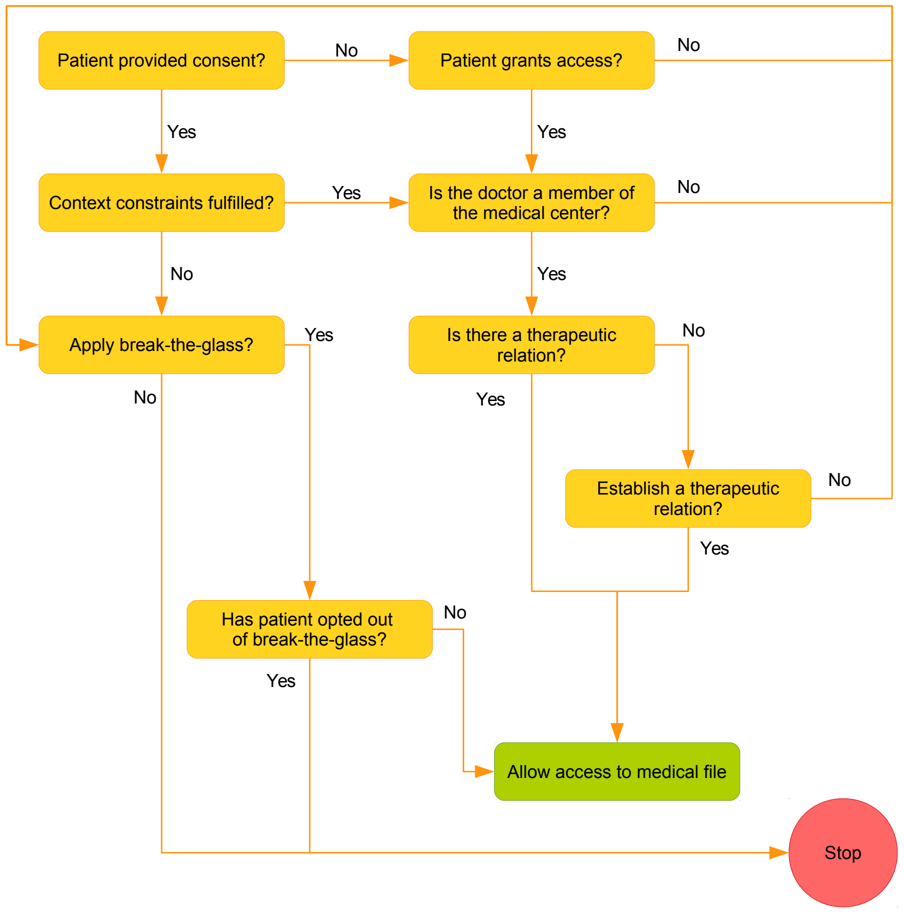

Patient provided consent?
No
Patient grants access?
No
Yes
Yes
Context constraints fulfilled?
Yes
Is the doctor a member of the medical center?
No
No
Yes
Apply break-the-glass?
Yes
Is there a therapeutic relation?
No
No
Yes
Establish a therapeutic relation?
No
Yes
Has patient opted out of break-the-glass?
No
Yes
Allow access to medical file
Stop

Fig. 2: Simplified access control workflow with patient consent directives for a healthcare professional requesting access to a medical file

out-of-hours service center. During regular office hours, these service centers are closed and patients will have to visit their own general practitioner. Each GP is on duty in such a service center in his area for one day every 8 weeks. Contrary to the EPR software that they would use in their home practice, the software in these service centers does not keep track of the medical history of the patient, even if the patient visits the service center multiple times. Instead, the patient's own GP will receive a report mentioning which examination was done and which medication was prescribed by the GP at the service center. Furthermore, legislative rules on privacy require that medical records at these service centers should be deleted or anonymized after 30 days.

Beyond the existence of a therapeutic relationship, the GP cannot access his own records if he is no longer on call in the service center. For example, a break-the-glass procedure is necessary, if the GP on call later on needs to access his own medical report to discuss it with the patient's own GP.

3.3 Emergency access for caregivers

An imminent threat to the health or safety of a patient may grant special user permissions and authorized access to caregivers to gain access to protected health information when particular emergency conditions are met. These context constraints [11] are declared upfront and form the basis for conditional permissions (e.g. time and location dependencies, separation of duties), and specific patient consent directives regarding preferences in an emergency situation apply.

The permissions that are granted to certain caregivers in advance can be enabled through self-declaration of an emergency situation. Similar to the break-the-glass procedure, emergency access is subject to alerts and auditing after the fact.

3.4 Requirements

The federated privileged identity management solution we envision will offer authentication, authorization and access capabilities for a variety of Software-as-a-Service applications in the cloud. These applications are hosted by service providers, and tenants rent access to a shared instance of this application. From the scenarios above, and through discussions with relevant stakeholders, we elicited the following high-level requirements:

1. The system must support smartcard-based authentication with electronic identity cards, with username/password credentials as a fallback login.
2. The system must support single-sign on so that users can access federated healthcare services without the need to authenticate multiple times.
3. Each tenant of a healthcare service should be able to specify his own authentication and access policies, including break-the-glass procedures.
4. The system must be able to customize and enforce the contextual constraints in the access policies per service and tenant.
5. The system must support delegation to privileged accounts with elevated access rights and increased auditing of their use.

In the following sections, we will discuss how these requirements were fulfilled on top of ForgeRock's OpenAM software system.

4 Identity and access management with OpenAM

From the aforementioned scenarios, we can distinguish three important aspects for access control to a patient's medical files, and that is (1) the applicability of patient consent directives, (2) the existence of a therapeutic relationship, and (3) context constraints that impose conditional permissions for access in regular and emergency circumstances. In the following subjects, we will discuss how these aspects can be taken into consideration.

4.1 Identity management

Our solution must provide support for a variety of stakeholders in the healthcare services that will link to the OpenAM-based platforms:

- Patients
- Family, friends and other caregivers
- Doctors, physicians, nurses or other healthcare professionals
- Administrators

Each of these users needs to be enrolled with an identity provider such that they can login to require access to protected resources. The objective is not to describe their roles and responsibilities in detail, but to illustrate the differences and subtleties in access control. For example, a patient has access to his own records, and can grant or restrict access to his records for certain healthcare professionals. He can also delegate access permissions and administrative privileges to a family member (e.g. a parent delegating to an adult child). The patient can also grant other caregivers conditional access to medical records under predefined emergency circumstances. Doctors and other healthcare professionals are capable of temporary delegation to other colleagues in the same practice (e.g. when going on holiday), and they can apply a break-the-glass procedure to gain access to medical records when no therapeutic relationship with the patient exists. However, they cannot modify the access policies of the patient. Such capabilities are only offered to administrators of the system.

In a multi-tenant Software-as-a-Service (SaaS) deployment, a single web application instance serves multiple organizations (i.e. tenants) and users within this organization. As such, the federated identity and access management solution should not only be able to distinguish between the different healthcare SaaS instances, but also between the different tenants (and their users) of a single healthcare web application instance. Nonetheless, each tenant should have an administrator which will manage the users within the organization, and administer the access policies for the different services.

4.2 Attribute-based access control and XACML policies

OpenAM is an identity and access management solution with support for XACML policies. In such policies, the following terminology is used:

- A *resource* defines the data, system component or service to be accessed.
- The *subject* is the actor who makes a request to access a certain resource.
- The *action* declares the operation (e.g. read, write, update, delete) on the resource for which permission is requested.
- The *environment* is a set of attributes (independent of a particular subject, resource or action) that are relevant to an authorization decision.
- A *target* defines the conditions that determine whether a policy applies to a particular request.
- An *obligation* is a directive on what must be carried out when an access is approved (e.g. notify the administrator).

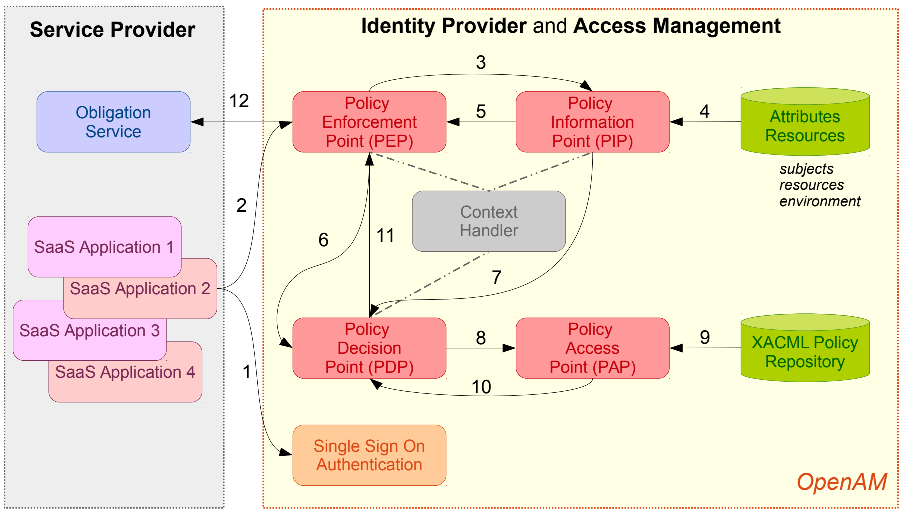


Fig. 3: XACML components in the identity and access management system

Attribute-based access control is a paradigm whereby access rights are granted to users through the use of policies which combine various types of attributes. XACML is such a policy-based and attribute-based access control standard.

4.3 Basic building blocks for XACML-based access control

Key components for decision making in XACML-based architectures, such as OpenAM (as depicted in Fig. 3), are the *Policy Enforcement Point* (PEP) and the *Policy Decision Point* (PDP). The PEP protects the resource. It receives incoming access requests, enforces access control decisions and executes obligations. It sends the XACML access request to the PDP which evaluates applicable policies and returns an authorization decision. If the policy request from the PEP does not contain all the required attributes about the subject, the resource being requested, or the environment, the *Policy Information Point* (PIP) will collect them for the PDP to be able to evaluate the relevant policies. The *Policy Administration Point* (PAP) creates, stores, manages and federates XACML security policies across the organization. The *Context Handler* works as an intermediate between the PDP, PEP and PIP to convert the requests and decisions from a native format to the XACML canonical form back and forth.

5 Implementation

In this section, we outline how the requirements identified in section 3.4 have been addressed on top of OpenAM. This implementation features two healthcare services (corresponding to the two scenarios in sections 3.1 and 3.2), and the federated privileged identity management framework.

5.1 Federated authentication with single-sign on

OpenAM separates identity providers from service providers, but they interact with one another in a circle of trust using the Security Assertion Markup Language (SAML) 2.0 [12] standard. SAML provides a secure, XML-based solution for exchanging user security information between an identity provider and a service provider. The identity provider stores and serves identity profiles, and handles authentication. The service provider offers services that access protected resources. A circle of trust groups at least one identity provider and at least one service provider who agree to share authentication information.

Username/password and smartcard authentication are supported out-of-the-box in OpenAM, though we used the eID Identity Provider software[2] to allow a web application to be made accessible with an eID.

5.2 Healthcare services offered as SaaS applications

For our experimental setup, we implemented a single service with different access policies for each of the use cases in section 3. The SaaS service host medical records as JSON documents on top of CouchDB[3] NoSQL database backend. What sets the first two use case scenarios apart is that the software for general practitioners in a medical center (cfr. scenario 1) keeps the medical history of their patients on file, whereas the software for the general practitioner on call in an out-of-hours service center (cfr. scenario 2) does not. The third scenario is used by patients directly for healthcare self-management purposes.

Although the Belgian government introduced the Kmehr standard[4] for medical information, we describe – for the sake of simplicity – a medical record with the following meta-properties:

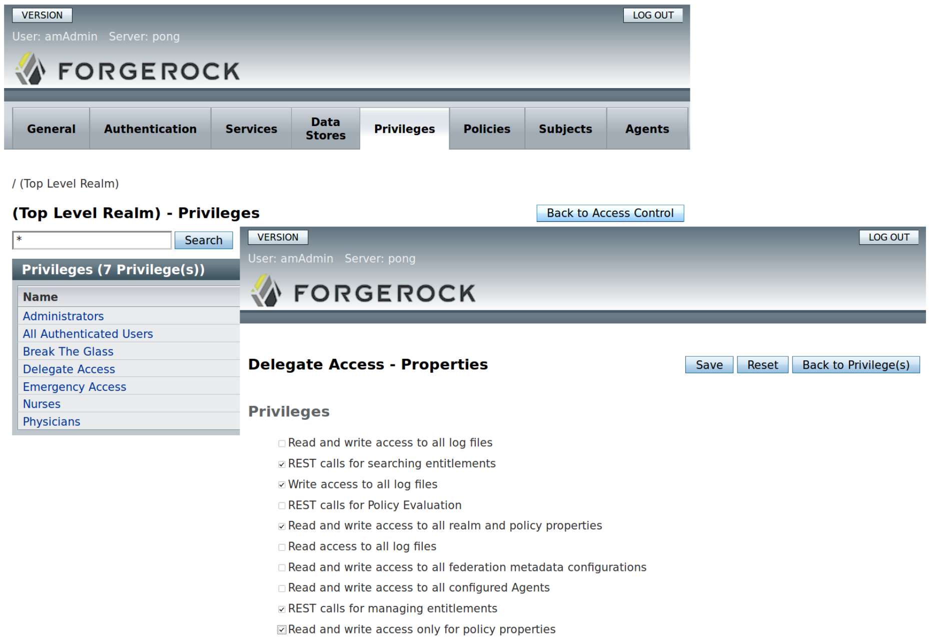
- **ResourceID**: an unambiguous identifier referring to the resource
- **Description**: a short free-text account of the resource
- **Owner**: the creator responsible for granting access to the resource
- **Subject**: the person or patient to which this resource refers
- **Type**: the nature or format of the resource using a controlled vocabulary
- **Date**: the date and time of creation of the resource

Each entry is exposed as a RESTful URI based on the universally unique identifier of the resource, and additional content is stored on a file system in a hierarchical structure using the same identifiers. As explained earlier, access to these services is governed by different rules.

5.3 Privileges delegation per service, tenant and user

With OpenAM, we can easily map tenants on the notion of *realms*. A realm is a mechanism to group configuration and identities together. For example, OpenAM can have different realms configured, one for physicians, a second for nurses

[2] http://eid.belgium.be/en/
[3] http://couchdb.apache.org/
[4] https://www.ehealth.fgov.be/standards/kmehr/



(Top Level Realm) - Privileges

Privileges (7 Privilege(s))

Name
Administrators
All Authenticated Users
Break The Glass
Delegate Access
Emergency Access
Nurses
Physicians

Delegate Access - Properties

Privileges

- ☐ Read and write access to all log files
- ☑ REST calls for searching entitlements
- ☑ Write access to all log files
- ☐ REST calls for Policy Evaluation
- ☑ Read and write access to all realm and policy properties
- ☐ Read access to all log files
- ☐ Read and write access to all federation metadata configurations
- ☐ Read and write access to all configured Agents
- ☑ REST calls for managing entitlements
- ☑ Read and write access only for policy properties

Fig. 4: Privilege delegation per realm in ForgeRock's OpenAM

and a third for administrators. Subdivisions in these groups with additional privileges can be defined as a child of this realm (e.g. a sub-realm for supervisory duties). We also added other groups dedicated for break-the-glass procedures and emergency situations, as illustrated in Fig. 4.

OpenAM has a capability of delegating administrator privileges to certain group of users that need to have permissions to modify OpenAM configurations. For example, the *Read and write access only for policy properties* privilege gives user permissions to create/modify/delete policies in a realm.

Our proof-of-concept implementation on top of OpenAM allows for privileged accounts to be created per tenant to be used for break-the-glass procedures or in emergency scenarios. These are not tied to particular people, but can be enabled through self-service requests. The password of the privileged account is never revealed, and automatically changed after each request. This way, the user cannot share access to the privileged account with unauthorized people. If allowed, individual users can also delegate their entitlements to another user for a period of time (e.g. during absence of work) using the same principle of delegating access to a privileged account they can administer themselves.

Before a user can request usage of a privileged account, the server checks whether access is granted either for the user himself, or for a group to which the user belongs. Inappropriate use of privileged accounts is detected through increased auditing. While not implemented, the session created with this privileged account can be time constrained to reduce the risk of abuse.

Listing 1.1: RESTful creation of an account in a sub-realm 'administrators'

```
$ curl --request "POST" --header "iplanetDirectoryPro: AQ...*"
  --header "Content-Type: application/json"
  --data '{ "username": "john", "userpassword": "secret", "mail": "john@host.com" }'
  https://idp.host.com/openam/json/service001/administrators/users/?_action=create
```

The above functionality is provided as a custom implementation on top of OpenAM's RESTful APIs[5] for managing the life cycle of user accounts per realm and sub-realms. The creation of an account with these APIs is illustrated with the *curl* Linux command line utility in Listing 1.1. The *iplanetDirectoryPro* header represents the token that the user received after authentication with which he can be identified in subsequent interactions.

5.4 Declaration of default access policies per services

Each the realms and sub-realms can define its own access policies. The hierarchical nature of sub-realms means that privileges defined in a certain realm are inherited into all sub-realms. Listing 1.2 provides a basic XACML policy for reading medical records, with a simplified representation for break-the-glass procedures.

Listing 1.2: Example break-the-glass-policy

```
<PolicySet xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" PolicySetId="readpolicyset"
        PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm
        :first-applicable">
  <Target>
    <Actions>
      <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
          read
          </AttributeValue>
          <ActionAttributeDesignator AttributeId="action:id"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ActionMatch>
      </Action>
    </Actions>

    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            medicalrecord
            </AttributeValue>
            <ResourceAttributeDesignator AttributeId="resource:type"
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>

```

[5] http://openam.forgerock.org/openam-documentation/openam-doc-source/doc/dev-guide/index/chap-rest.html

```
30    <Policy PolicyId="readpolicy:1" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0
31         :rule−combining−algorithm:first−applicable">
32      <Rule RuleId="readrule:1" Effect="Permit">
33        <Condition>
34          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string−is−in">
35            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string−one−and−only">
36              <ResourceAttributeDesignator AttributeId="resource:owner:id"
37                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
38            </Apply>
39            <SubjectAttributeDesignator AttributeId="subject:treated"
40                DataType="http://www.w3.org/2001/XMLSchema#string"/>
41          </Apply>
42        </Condition>
43      </Rule>
44    </Policy>
45
46    <Policy PolicyId="readpolicy:2" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0
47        :rule−combining−algorithm:first−applicable">
48      <Rule RuleId="readrule:2" Effect="Permit">
49        <Condition>
50          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean−one−and−only">
51            <SubjectAttributeDesignator AttributeId="subject:break_glass"
52                DataType="http://www.w3.org/2001/XMLSchema#boolean"/>
53          </Apply>
54        </Condition>
55      </Rule>
56    </Policy>
57
58    <Policy PolicyId="readpolicy:3" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0
59        :rule−combining−algorithm:first−applicable">
60      <Rule RuleId="readrule:3" Effect="Deny"/>
61    </Policy>
62  </PolicySet>
```

Lines 5-15 state that this policy set is only applicable for access requests involving *read* operations. Furthermore, these read operations should be executed on a resource of type *medicalrecord* as stated in lines 17-27. The individual policies and corresponding rules in the lines below define constraints for granting read access, breaking the glass, and denying read access. For a complete account of all relevant policies, the XML representation would become very long and verbose. Nonetheless, each of these realms can customize and specify additional policies that differ in various contextual constraints:

– Time and/or location dependencies on the role holding the permission for persons requesting access beyond their regular shift or primary workplace.
– Dynamic separation of duty constraints, where two mutual exclusive roles are never activated simultaneously for the same user.
– Assignment of additional temporary duties, such as administrative or supervisory roles.
– Cardinality constraints that limit the number of people who may hold a permission at any one time.

Obviously these constraints can be combined. An example that combines time, location and cardinality constraints is a policy that states that *only one nurse per 12-hour shift on a hospital floor can obtain head nurse privileges*. Similar to the APIs present earlier for identity management, OpenAM also offers RESTful interfaces to administer the life cycle of access policies.

6 Qualitative evaluation

Our federated privileged identity management proof-of-concept was developed on top of the OpenAM identity and access management platform, based on use cases and scenarios from stakeholders in the business of offering healthcare services to medical professionals. The usability of the prototype has not yet been quantitatively evaluated, which is why we focus on some qualitative aspects of our solution which go beyond the requirements of section 3.4.

6.1 Strengths

- Our prototype allows for the complex access control and privileged identity management to be isolated from the healthcare service as a separate component, reducing the management overhead of the healthcare service itself.
- From a managerial point of view, our proof-of-concept offers a unified and consistent interface towards the management of security policies for federated healthcare services. A single interface suffices to manage the data access control for all federated healthcare services.
- To avoid becoming a single point of failure for authentication and authorization, our federated privileged identity management solution leverages OpenAM's built-in replication capabilities to offer failover support.

6.2 Weaknesses

- The XACML standard we used to define access policies is not for the faint of heart. The level of abstraction should be raised to reduce the complexity and to simplify their use for non-experts.
- Obligations are common for break-the-glass procedures. XACML offers combination algorithms to resolve conflicts between contradicting policies or rules. However, these are not yet available for obligations.
- The performance overhead of XACML-based policy evaluation is often a reason to fall back on simpler RBAC policies. However, with OpenAM's built-in clustering support for enhanced scalability, the higher expressiveness of ABAC-based XACML policies is a clear benefit.

7 Conclusion

In this paper, we discussed our privileged identity management that we built on top of OpenAM, a state-of-the-art open source federated identity and access management solution. The main objective as the implementation of a reusable policy for break-the-glass procedures and emergency scenarios in federated healthcare services. We elicited a set of requirements from 3 distinct use cases. Some of the key features that our proof-of-concept provides include:

- Support for single sign on authentication using different means to log in

- Customization of access policies per service and per tenant
- Delegation to privileged accounts with elevated access rights

We explained how our solution offers identity and access management capabilities for different healthcare services, but also how we support different tenants (and users within these tenants) per service using realms and sub-realms to group configuration, identities and access policies together. While our proof-of-concept has many benefits, the usability can be improved further by raising the level of abstraction of XACML policies. For example, a policy language could make the definition of security policies less complex, and hence less error prone. Additional tool support could then translate these new policies in a XACML compliant format. Also, while it is technically feasible to reuse existing security policies, we have not yet set up an experiment with real stakeholders in which the reuse and reduction in policy management overhead is measured.

Beyond the above aspects, as future work we will investigate the often heard performance concerns with XACML-based policy evaluation with systematic benchmarks, and explore to what extent our prototype on top of OpenAM can benefit from the latter's replication mechanisms to ensure scalability under increasing load.

Acknowledgments

This research is partially funded by the Research Fund KU Leuven.

References

1. Pagliari, C., Sloan, D., Gregor, P., Sullivan, F., Detmer, D., Kahan, J., Oortwijn, W., MacGillivray, S.: What is eHealth (4): a scoping exercise to map the field. J Med Internet Res **7**(1) (2005) e9
2. SPC: Break-Glass - An Approach to Granting Emergency Access to Healthcare Systems. Technical report, Joint NEMA/COCIR/JIRA Security and Privacy Committee (SPC) (December 2004)
3. Povey, D.: Optimistic Security: A New Access Control Paradigm. In: Proceedings of the 1999 Workshop on New Security Paradigms. NSPW '99, New York, NY, USA, ACM (2000) 40–45
4. XACML-V3.0: eXtensible Access Control Markup Language (XACML) Version 3.0. Candidate OASIS Standard 01. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cos01-en.html (September 2012)
5. Decat, M., Lagaisse, B., Joosen, W.: Middleware for efficient and confidentiality-aware federation of access control policies. Journal of Internet Services and Applications **5**(1) (2014) 1–15
6. Alqatawna, J., Rissanen, E., Sadighi, B.: Overriding of Access Control in XACML. In: Proceedings of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks. POLICY '07, Washington, DC, USA, IEEE Computer Society (2007) 87–95
7. Brucker, A.D., Petritsch, H.: Extending Access Control Models with Break-glass. In: Proceedings of the 14th ACM Symposium on Access Control Models and Technologies. SACMAT '09, New York, NY, USA, ACM (2009) 197–206

8. Lodderstedt, T., Basin, D.A., Doser, J.: SecureUML: A UML-Based Modeling Language for Model-Driven Security. In: Proceedings of the 5th International Conference on The Unified Modeling Language. UML '02, London, UK, UK, Springer-Verlag (2002) 426–441
9. Marinovic, S., Craven, R., Ma, J., Dulay, N.: Rumpole: A Flexible Break-glass Access Control Model. In: Proceedings of the 16th ACM Symposium on Access Control Models and Technologies. SACMAT '11, New York, NY, USA, ACM (2011) 73–82
10. Belnap, NuelD., J.: A Useful Four-Valued Logic. In Dunn, J., Epstein, G., eds.: Modern Uses of Multiple-Valued Logic. Volume 2 of Episteme. Springer Netherlands (1977) 5–37
11. Strembeck, M., Neumann, G.: An integrated approach to engineer and enforce context constraints in RBAC environments. ACM Trans. Inf. Syst. Secur. **7**(3) (2004) 392–427
12. Lewis, K.D., Lewis, J.E.: Web Single Sign-On Authentication using SAML. CoRR **abs/0909.2368** (2009)