A Web Service for Hypermedia Role-Based Policies

Daniel Sanz, Ignacio Aedo, Paloma Díaz
Laboratorio DEI. Departamento de Informática
Universidad Carlos III de Madrid
{dsanz | aedo | pdp}@dei.inf.uc3m.es

Abstract

High level security is a key requirement in hypermedia/web applications. The systems opening to the Internet makes the research effort to move towards protecting the information transmission (i.e. SOAP messages, policy descriptions…), but little attention is paid to what the user can do with the system. Role-based access control (RBAC) allows to formulate the organization's resource access policy in a natural way, so a role-based access model for hypermedia will make it easier to integrate the security design with the rest of the system design. In service oriented architectures, an access policy service would allow to gather the management and deployment of the security policy in distributed and heterogeneous environments. This paper describes a role-based access control model for hypermedia called MARAH, and its implementation as a web service. An use case of the model in the design of the ARCE application is also discussed.

1. Introduction

The industry and research community trend towards the openness and interoperability among systems is very clear, what makes very important the adequate resource protection. The balance between service providing and asset protection becomes an strategic issue, so the security design has to be more than an implementation/deployment aspect: security is a first-class element whose requirements should be taken into account throughout all development process. This design integration would help to face the security analisys and design [4], but in order to achieve it with success, a formal basis gathering the relevant concepts in the protection domain is required, taking the form of formal models. In the highest abstraction level, access control is a security service that allows to regulate how users interact with the system.

Hypermedia/web applications are implemented in a variety of platforms, where the information is probably distributed and heterogeneous, and the access context can be very different each time. There is a clear need for a model that considers these issues and makes it possible to express the access policies in a flexible way. From an architectural point of view, it is recommended to keep some separation between the security controls and the controlled resources themselves, what makes the Web Services paradigm a perfect deployment platform to test our ideas.

MARAH is a role-based access model for hypermedia applications, that works with concepts of the hypermedia domain (i.e. contents or links), and is based on the RBAC model. RBAC articulates the access policy around the role concept: instead of granting permissions directly to system users, permissions are assigned to roles, and users are assigned to the roles they can play, acquiring its permissions. The role represents job functions, responsibilities or qualifications in the context of some organization, and groups both the permissions and the users allowed to use them.

The use of roles and proper hypermedia abstractions to express the access rights makes it possible to integrate MARAH in a development method, due to designers can decide about what the access policy is while designing other system aspects, such as navigation or content layout. This facilitates the policy definition, because now access control is not performed on the application code in terms of classes/objects, database tuples or other paradigm, but protected resources are the elements of a hypermedia design, that have a correspondence with one or more systems responsible of the information generation.

MARAH is available as Web Service, with the goal of providing the access policy information in a concrete domain. The implementation covers the core RBAC standard [2], and does not assume any specific hypermedia model, so it can be used from any hypermedia implementation. This work is organized as follows: section 2 presents a reduced version of the model, based on core RBAC specification. Section 3 describes the implementation and architecture of the MARAH web service. Section 4 introduces an use case in the ARCE application, and finally section 5 discusses some conclusions and future works.

2. The MARAH model

Initially MARAH was defined as an unique model, specifically designed for a hypermedia reference model called Labyrinth[6, 7]. After some implementation experiences [8, 11] and the creation of the RBAC ANSI standard [2], it is being restructured for a twofold purpose: to define the model features in an incremental way, as the RBAC family of models do, and to be independent of the underlying hypermedia model, and so applicable to any hypermedia architecture. We try to make more flexible the use of the model in a variety of situations, and to face its transformation to an independent service. We are working on the core of the model (Core MARAH in this paper), that, starting from the core of RBAC, includes the very basic elements for a hypermedia access control model.

Next subsection introduces the core RBAC model, as previous requirement to define the core MARAH model.

2.1. The core RBAC model

The core RBAC is one of the four model components defined in the RBAC reference model, and "defines a minimum collection of RBAC elements, element sets and relations in order to completely achieve a Role-Based Access Control system" [2]. The elements of the core RBAC model are:

- U, R, OPS, O
  Sets of Users, Roles, Operations and Objects respectively.
- $UA \subseteq R \times U$
  A many-to-many mapping user-to-role assignment relation.
- assigned_users : $R \rightarrow 2^U$
  The mapping of a role r onto a set of users allowed to hold it. Formally, assigned_users$(r) = \{u \in U \mid (u, r) \in UA\}$.
- $P \subseteq 2^{OPS \times O}$
  The set of permissions. A permission represents an approval to execute an operation on a protected object.
- $PA \subseteq P \times R$
  A many-to-many relationship mapping the permission-to-role assignment relation.
- assigned_permissions : $R \rightarrow 2^P$
  The mapping of a role r onto a set of permissions. Formally, assigned_permissions$(r) = \{p \in P \mid (p, r) \in PA\}$.
- Op : $P \rightarrow OPS$
  The permission to operation mapping, which gives the set of operations associated with permission p.
- Ob : $P \rightarrow O$
  The permission to object mapping, which gives the set of objects associated with permission p.
- S
  The set of sessions. A session represents an user interaction period, where the user activates some subset of the roles she is assigned to. Each user can establish one or more sessions.
- session_users : $S \rightarrow U$
  The mapping of session s onto the corresponding user.
- session_roles : $S \rightarrow 2^R$
  The mapping of session s onto a set of roles. Formally, session_roles$(s) \subseteq \{r \in R \mid (\text{session_users}(s), r) \in UA\}$
- avail_session_permissions : $S \rightarrow 2^P$
  The permissions available to the user in a session. Formally, avail_session_permissions$(s) = \bigcup_{r \in \text{session_roles(s)}} \text{assigned_permissions}(r)$

In addition to the sets, relations and functions, the RBAC standard also defines the system and administrative functional specification for the core model. Administrative functions allow to create and maintain the RBAC sets and relations, administrative review functions allow to perform administrative queries, and system functions allow to creat and manage user sessions and make access control decisions.

2.2. The core MARAH model

The core of MARAH is based on a hypermedia metamodel that assumes two fundamental elements of this discipline: the distinction between container and content, and the information linking. As in RBAC standard, to apply MARAH in a concrete system, a mapping from the access model components to the system components has to be provided. Due to MARAH is an access control model for hypermedia, this mapping should not be very complex, as MARAH provides the proper abstractions representing protected objects. Starting from the core RBAC specification, the Core MARAH model provides three contributions.

The first of these contributions is the definition of the protected objects. The hypermedia metamodel assumed by MARAH defines four object types[1] that can be identified in most hypermedia models:

Nodes abstract information containers, they have whole meaning in the application context. Represent presentation units.

[1]These object types reflect the application information structure, regardless how the information is modeled (i.e. OO model), stored or generated to give place to such structure.

Contents individual pieces of information that are included in the nodes. They can be simple (e.g. text) or complex (e.g. video).

Anchors represent areas or fragments in the node or content representation space (i.e. characters, pixels, rows, seconds…).

Links represent information connections, by means of relationships between a set of source and a set of target anchors.

Nodes and contents represent the system information, so they are the basis on which privileges are established. Anchors and links are used to represent relationships between information elements, so its definition is somehow tied to the elements in which are defined, and thus its access rights have to be derived from the permissions of these elements.

The second is the addition of relationships among objects. Core MARAH considers two relation types in the object set:

Location reflect the spatial/temporal placements among objects. For instance, the inclusion of contents into nodes or the definition of anchors.

Linking represent associative connections between nodes and/or contents.

As explained later, these relationships are used to propagate access rights in a variety of situations, so the administration cost is reduced due to there is no need to specify the permissions for each object.

The third contribution is the addition of one abstraction level on top of the system operations, reducing the number of permissions assignments by means of a classification of operations. The classification groups the operations and allows to easily set permissions involving several operations.

Core MARAH is an extension of Core RBAC standard that defines the object types[2], new elements and the relationships between the RBAC standard model, as shown in figure 1:

- $O = N \cup C \cup A \cup L$
  Sets of nodes (N), contents (C), anchors (A) and links (L) respectively.

- $OR = \{\succ_{loc}, \succ_{Ls}, \succ_{Lt}\}$
  Set of object relationships. Core MARAH defines three relationship types:

  Location. $\succ_{loc} \subseteq O \times O$. If a $\succ_{loc}$ b, there is a spatial/temporal relationship from object a to object b. For instance, if anchor a is defined to start in the third second of video b, then there is a location relation from the anchor to the video.

[2] Due to its generality, the RBAC standard leaves open this question.

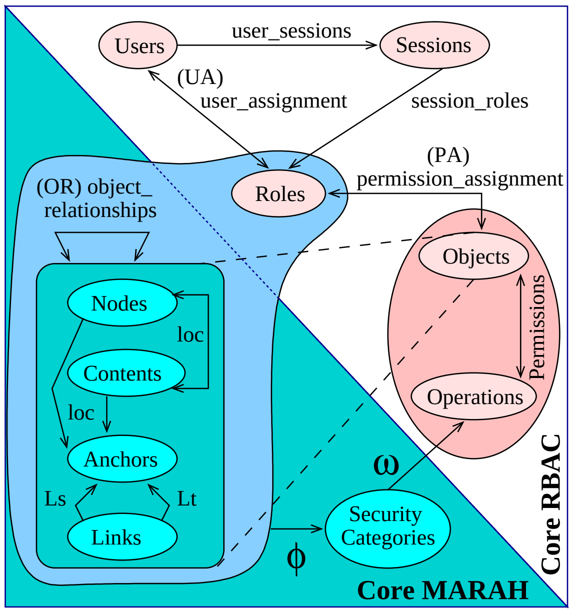


Figure 1. Core MARAH model and Core RBAC model

  Source linking. $\succ_{Ls} \subseteq A \times L$. If a $\succ_{Ls}$ b, the anchor a is an origin of the link b.

  Target linking. $\succ_{Lt} \subseteq A \times L$. If a $\succ_{Lt}$ b, the anchor a is a target of the link b.

- OPS
  Set of operations. Core MARAH only considers generic operations for system use, such as "seeNode" or "traverseLink". Like the object set, the operations are provided by the underlying hypermedia model.

- $SC = \{B, E\}$
  Set of security categories, with two elements. Each category represent abstract manipulation types that exist in any hypermedia system. The value B represents "browsing" and allows to see the element, and the value E means "editing" and allows to modify or create an element. A partial order is defined in SC such as $E > B$, since the edition manipulation includes the viewing manipulation.

- $\omega : OPS \to SC$
  Classification of operations function, that relates each operation with the manipulation type that its execution requires, for instance, $\omega(\text{getLinksToTarget()}) = B$.

- $\phi : R \times \{N \cup C\} \to SC$
  The clearance function returns the highest manipulation ability that a role can do on an object. This value is set by the security officer for contents and nodes,

and involves the creation of a set of permissions in P for the object and the operations classified with a lower or equal category than the one specified. Formally, if administrator sets the following clearance $\phi(r, o) = E$, then $\forall op \in OPS \mid \omega(op) \leq E \Rightarrow P = P \cup (op, o), PA = PA \cup \{(op, o), r\}$.

- $\prod : S \times O \rightarrow SC \cup 0$
  The authorization rule returns the highest manipulation ability available in a session for a given object. The value depends on the session active roles, the type and relationships of the involved object. The function is defined as follows:
  - For nodes and contents, returns the maximum of the session permissions:
    $$\prod(s, o) = \max(0 \cup \{\bigcup_{p \in \text{ASP}(s), Ob(p) = o} \omega(Op(p))\})$$
    $(\text{ASP}(s) = \text{avail\_session\_permissions}(s))$. For the maximum, a total order is defined such as $E > B > 0$. The value of 0 indicates the absence of permissions for the object in the session[3].
  - Anchors take directly the permission granted to the object on which are located.
    $\prod(s, a) = \prod(s, o) \mid a \underset{loc}{\succ} o$
  - For links, "B" access is allowed if there is access to at least one anchor in each link end. "E" access is allowed if all anchors have editing permissions:
    $$\prod(s, l) = \begin{cases} B & \text{if } \exists a_1, a_2 \mid a_1 \underset{Ls}{\succ} l, a_2 \underset{Lt}{\succ} l, \\ & \prod(if, a_1) = \prod(s, a_2) = B \\ E & \text{si } \forall a_i, a_j \mid a_1 \underset{Ls}{\succ} l, a_j \underset{Lt}{\succ} l, \\ & \prod(s, a_i) = \prod(s, a_j) = E \\ 0 & \text{otherwise} \end{cases}$$

- $\theta : OPS \times O \times S \rightarrow \{0, 1\}$
  The transition function checks if an operation can be performed on an object from a given session.
  $$\theta(op, o, s) = \begin{cases} 1 & \text{if } \omega(op) \leq \prod(s, o), \\ 0 & \text{otherwise} \end{cases}$$

The Core MARAH model defines also a set of administrative and system functions, in order to create, maintain and query the different sets and relations, as well as creating user sessions and taking access control decisions.

3. Core MARAH Web Service

Core MARAH is being implemented as web service. It acts as *policy decision point* (PDP), independent of the *policy enforcement point* (PEP) where the actions according to the PDP take place, typically in systems responsible of content generation, integration and/or delivery. Previous implementations of MARAH were integrated in the content server [8], making no distinctions between the PDP and PEP, and constraining its use to only one website. The MARAH service offers a RBAC standard implementation that is reusable by several applications within the organization. As figure 2 shows, the differences between MARAH

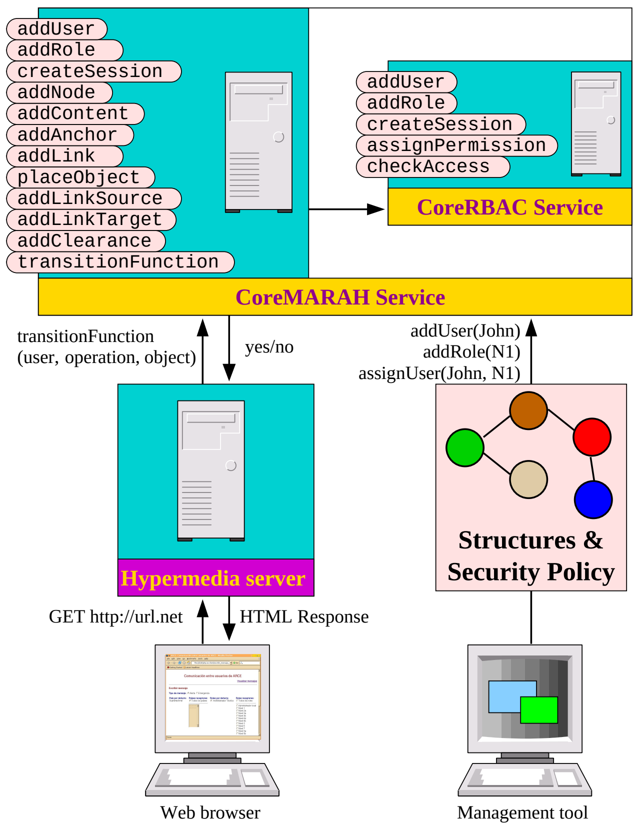


Figure 2. Core MARAH service architecture

and RBAC standard have led to a broad service split in two smaller cooperating subservices. This cooperation gathers the transformation rules from MARAH components to RBAC components described in previous section, in terms of service operation execution. From the point of view of the service clients, whether administrators or systems implementing PEPs, this separation is completely hidden. The service usage is very simple: first, access policy is notified to the service. Then, the service is requested for access control decisions involving operations, objects and roles previously defined. Here the term access policy means the definition of the set of objects and its relations, roles and clearances. It is important to note that the adoption of the web service paradigm and the creation of a policy service separated from the PEP introduces new challenges, due to there

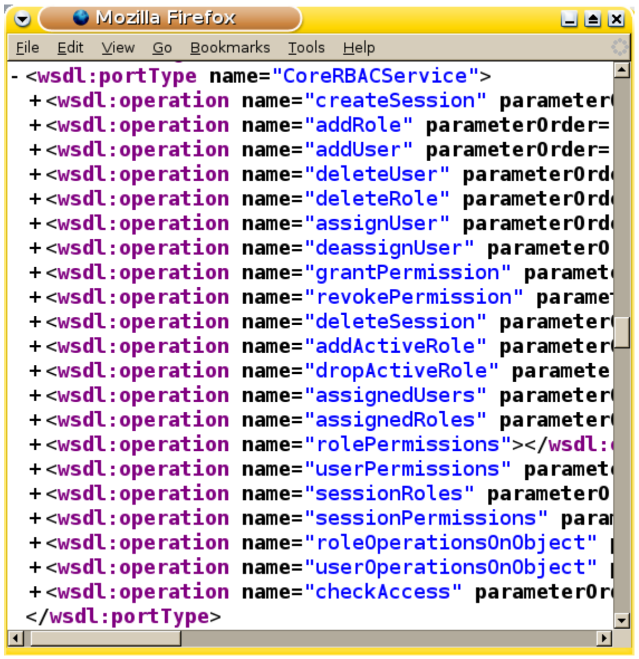
[3]The model does not provide an interpretation for this situation.

is a need to protect the information during transmission between PEP and PDP. Due to the focus of this work is the deployment of the abstract core access control model, and taking into account the advances to come in the web services security area [9], issues related to low level security in the service usage (i.e. identification, authorization, end-to-end encryption,. . .) have not been considered yet.

The Core RBAC web service implements the Core RBAC administrative and system functional specifications, providing operations for:

- Adding/removing users and roles,
- Assigning/deassigning users to roles, and permissions to roles,
- Creating/deleting sessions for an user and activating/deactivating some roles,
- Checking the access.

In order to provide a general service, all elements (roles, users, permissions and operations) are identified by `string` tokens, whilst the service returns information using primitive types such as string lists or boolean values. Figure 3 shows part of the web service definition (WSDL) for Core RBAC service.



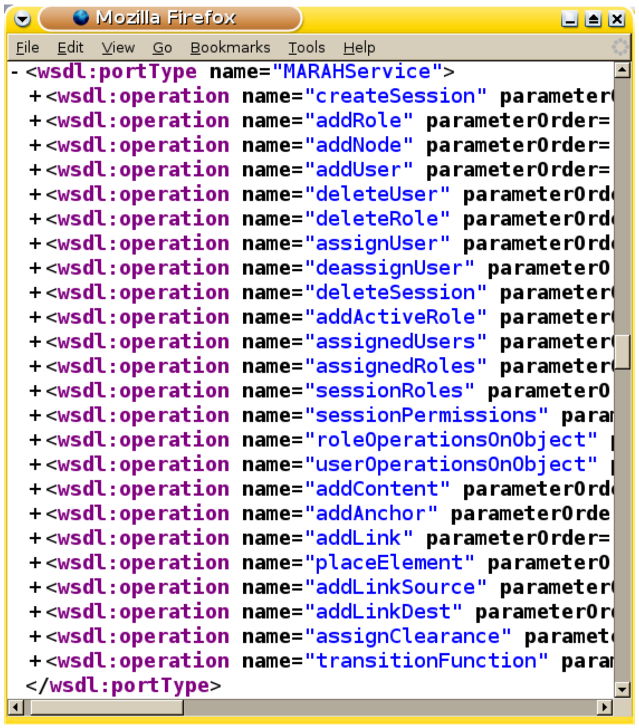
```
- <wsdl:portType name="CoreRBACService">
  +<wsdl:operation name="createSession" parameter
  +<wsdl:operation name="addRole" parameterOrder=
  +<wsdl:operation name="addUser" parameterOrder=
  +<wsdl:operation name="deleteUser" parameterOrd
  +<wsdl:operation name="deleteRole" parameterOrd
  +<wsdl:operation name="assignUser" parameterOrd
  +<wsdl:operation name="deassignUser" parameterO
  +<wsdl:operation name="grantPermission" paramet
  +<wsdl:operation name="revokePermission" parame
  +<wsdl:operation name="deleteSession" parameter
  +<wsdl:operation name="addActiveRole" parameter
  +<wsdl:operation name="dropActiveRole" paramete
  +<wsdl:operation name="assignedUsers" parameter
  +<wsdl:operation name="assignedRoles" parameter
  +<wsdl:operation name="rolePermissions"></wsdl:
  +<wsdl:operation name="userPermissions" paramet
  +<wsdl:operation name="sessionRoles" parameterO
  +<wsdl:operation name="sessionPermissions" para
  +<wsdl:operation name="roleOperationsOnObject"
  +<wsdl:operation name="userOperationsOnObject"
  +<wsdl:operation name="checkAccess" parameterOr
  </wsdl:portType>
```

Figure 3. WSDL fragment for the Core RBAC service

The Core MARAH service acts as client of the Core RBAC service, and implements administrative and system functions, that, in addition to the functionality provided by Core RBAC, allow to:

- Creating/deleting object instances, for each one of the four types,
- Creating/deleting object relationships (location, source linking and target linking),
- Granting/revoking clearances to roles,
- Checking the access, by the transition function (θ).

Figure 4 shows part of the WSDL definition for the Core MARAH service. The implementation of some of its operations make use of the Core RBAC service, some of them being simply wrappers for the RBAC service operations. Core MARAH service also uses string based primitive types.



```
- <wsdl:portType name="MARAHService">
  +<wsdl:operation name="createSession" parameter
  +<wsdl:operation name="addRole" parameterOrder=
  +<wsdl:operation name="addNode" parameterOrder=
  +<wsdl:operation name="addUser" parameterOrder=
  +<wsdl:operation name="deleteUser" parameterOrd
  +<wsdl:operation name="deleteRole" parameterOrd
  +<wsdl:operation name="assignUser" parameterOrd
  +<wsdl:operation name="deassignUser" parameterO
  +<wsdl:operation name="deleteSession" parameter
  +<wsdl:operation name="addActiveRole" parameter
  +<wsdl:operation name="assignedUsers" parameter
  +<wsdl:operation name="assignedRoles" parameter
  +<wsdl:operation name="sessionRoles" parameterO
  +<wsdl:operation name="sessionPermissions" para
  +<wsdl:operation name="roleOperationsOnObject"
  +<wsdl:operation name="userOperationsOnObject"
  +<wsdl:operation name="addContent" parameterOrd
  +<wsdl:operation name="addAnchor" parameterOrde
  +<wsdl:operation name="addLink" parameterOrder=
  +<wsdl:operation name="placeElement" parameterO
  +<wsdl:operation name="addLinkSource" parameter
  +<wsdl:operation name="addLinkDest" parameterOr
  +<wsdl:operation name="assignClearance" paramet
  +<wsdl:operation name="transitionFunction" para
  </wsdl:portType>
```

Figure 4. WSDL fragment for the Core MARAH service

Both Core RBAC and Core MARAH services are being developed using Java, and made available as web service using the Axis software, from the Apache project (http://ws.apache.org).

4. Use case: the ARCE system

The Core MARAH service is being probed in an use case of the ARCE system (networked application for emergency situations), which goal is to facilitate the efficient aid management in emergency situations among Latin-American countries [1]. One of the ARCE functions is to promote the communication between users, and so a messaging mechanism has been designed, similar to the email, but the receiver of the message is a role, and not a concrete user. There is an information flow policy such that one given sender can write messages only to a concrete set of target roles. There are roles representing different job functions and organizations in the Latin-american Association of Governmental Organisms of Civil Defence and Protection. Roles N1, N2, N3, N4, N5, N6, N7 and N8 are message senders. For the sake of clarity, we will describe a subset of the information flow policy.

- Role N2 sends messages to N2 receivers, with mandatory copy to N1. The rest of receivers can be selected by the sender when composing the message.

- Role N3 sends message to N3 users, with mandatory copy to N1. The rest of receivers can be selected by the sender, excepting N8, that can't receive messages from N3.

The messaging subsystem has three pages: the message composition page, the message list page (provides a table with message subjects) and the message view page (showing one message). The system has been modeled as follows:

- Nodes: the message composition, message list and message view are nodes. For each new composed message, a new node is generated representing it.

- Contents: Each form element in the message composition page is a content. In the message list, there is a table which rows representing links to messages. In the message view page, the message title and text are also contents. Each new message generates contents for the text and title.

- Anchors and links: from the message composition the user can navigate forwards and backwards to the message list. In this page, there is a link for each written message pointing to the corresponding message view node.

- Clearances: The message composition and list nodes have the "B" category for everybody. The message composition form has checkboxes in order to select the receiver roles. These checkboxes have different permissions according to the information flow policy:

  - If the sender *must* send the message to the receiver, the checkbox has a "B" clearance, stating that the checkbox value can't be modified

  - If the sender *can choose* if the receiver will read the message or not, then the checkbox has a "E" clearance

  - If the sender *can not* send the message to a receiver, there is no permission for the checkbox

According to this modeling, table 1 shows the clearances for the contents of message composition node, for roles N2 and N3 (the symbol $\emptyset$ indicates that no clearance is assigned). Each checkbox is named using the destination role. It is important to note that the system implementation (PEP) is responsible for taking the right action according to the response of the PEP, what includes choosing presentation issues. Figure 5 shows the aspect of the generated pages in conformance with the policyfor roles N2 and N3, where the non-editable checkboxes appear disabled. If some receiver must be notified, the corresponding checkbox appears marked and disabled.

	N2	**N3**
checkbox_N1	B	B
checkbox_N2	B	E
checkbox_N3	E	B
checkbox_N4	E	E
checkbox_N5	E	E
checkbox_N6	E	E
checkbox_N7	E	E
checkbox_N8	E	$\emptyset$

Table 1. Values of the clearance function for roles N2 and N3 in the ARCE messaging system

The messaging subsystem has been implemented in PHP due to this language offers SOAP functions that make it easy the coding of web service clients. The web server chosen to host the PHP pages is Apache. Nodes and contents are PHP objects that provide their information through the `show()` method, that includes a call to the Core MARAH service transition function. The PDP queries are performed by the class pdp.php, whose objects are built by providing the URL of the WSDL description of the Core MARAH service, as shown in figure 6.

Each node and content in Core MARAH corresponds to a PHP object, but nodes keep references to the objects representing their contents (and thus implementing the $\underset{loc}{\succ}$ relation), as shown in figure 7, where a generic node code is shown. Using this mechanism, a PHP node will perform several queries to the PDP during its dynamic generation. The process for the generation of the message composition node (see figure 5) is as follows: once policy is notified to PDP, an authenticated user tries to visualize the page, so

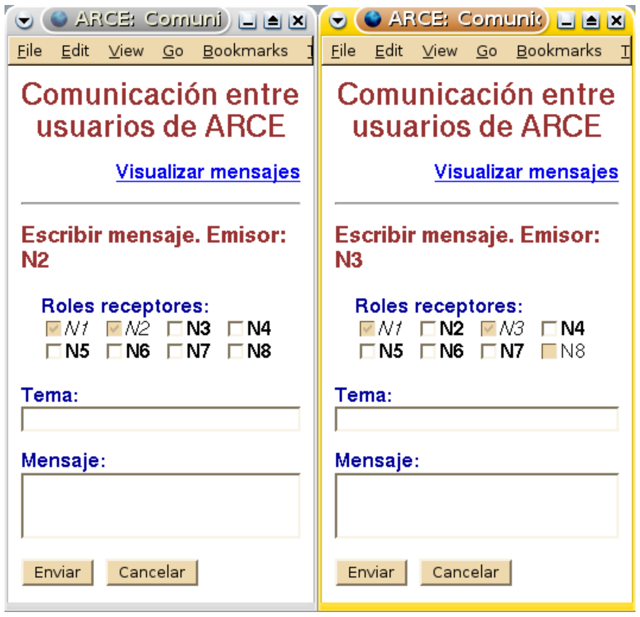


Figure 5. Views of message composition page for roles N2 and N3

```
<?php
class pdp {
 private $client;
 public function __construct($wsdl) {
   $this->client = new SoapClient($wsdl);
 }
 public function safe($user, $object, $operation) {
   return $client->transitionFunction($user,
                        $object, $operation);
 }
}
```

Figure 6. Core MARAH web service invocation from PHP

PHP calls `show()` on the PHP object implementing the page. First, the node checks access for itself, using context information that includes the user name and session identifier. After querying the PDP and checking the transition function value for the "B" category, the node starts to show its own contents, according to its own source code.

```
<?php
class node {
    private $pdp;
    private $id;
    private $contents;
    public function __construct($pdp, $id) {
        $this->pdp = $pdp;
        $this->id = $id;
    }
    public function placeContentNode($content) {
        $this->contents[] = $content;
    }
    public function show($user) {
        if ($this->pdp->safe($user, $this->id, "browse")) {
            $this->beginWrap();
            foreach ($this->contents as $content) {
                $content->show($user);
            }
            $this->endWrap();
        }
    }
```

Figure 7. Generic code for a node

Each content to show is a PHP object, that is notified to show itself by the `show()` method. In order to show its information, the transition function has to return 1 for the content itself, so the PDP is queried again. When the content includes anchors or other contents, the process is repeated again, so each HTML element sent to the client has been filtered by the access control mechanism, and so it is an allowed element. In this way, a given sender can not select roles for which sending a message is prohibited, because the contents that allow this action are skipped or otherwise disabled from the page.

Once the sender writes the message title and body, selects the receivers and sends the message, the system does the following:

- Generate a new node n representing it, including the message title and body in its contents. Define an anchor t and include it in the node.
- Add the message title content in the message list page, creating an anchor s for it
- Create a new link, whose source is the s anchor, and the target is t.
- Set the clearances corresponding to the new node for the selected roles, using the checkbox information. If a role N_i is marked, then $\phi(N_i, n) = B$.

Now we illustrate how links are skipped in the message list node. When processing this node, each table row represents a link to a message. In order to get access to a link, one of its sources and targets must be accessible. Since the target anchor takes the permission from the node in which is defined, the whole link has the permission of the destination. This implies that if the message is denied to some role, the link to it is too denied, so the PHP page will not add the corresponding row to the table. So, it can be said that there are one or more PEP for each object, but the PDP answers are provided by Core MARAH service.

5 Conclusions and future work

This work presents a role-based access control model for hypermedia systems, and its deployment as a web service, in response to the need of separating the access control decision of the policy enforcement. Moreover, a prototype that uses the web service has been built, in the ARCE application context. The model is deliberately simple, because its design criteria are based on the modularity, flexibility and minimalism, in similar philosophy as the used in the core RBAC design.

There are several future work directions. Firstly, the access model will be extended by adding several layers in order to add more advanced features. The first one will presumably be the Hierarchical MARAH, that will be based on Hierarchical RBAC and will extend it by providing new roles and objects relationships. This will allow to create richer structures where the permission inheritance will reduce even more the access rights management. Other foreseen extension are the inclusion of context parameters [5], specially useful in open environments because they provide more information to make access control decisions (i.e. location from where the access is requested). We also plan to add temporal constraints [10], that allows to enable or disable the use of roles or permissions assignments for a given time interval or period, as well as limiting the duration of the sessions or permissions.

Regarding the service, we plan to study how to apply the service in decentralized architectures where there are several PDP responsible of keeping parts of the enterprise-wide access policy. The PDPs can also be separated to some extent and be transformed into another service, able to generate an adequate version of the content according to the user, for instance using SMIL (http://www.w3.org/AudioVideo/).

Moreover, we will consider the integration with web services security standards, particularly WS-Policy [3], for environments where contents are provided by web services that describes and attach their access policy as part of the service description, maybe in different formats such as XACML or SAML. These policies would be used by one or more PDPs that gather and interpret them in order to provide an access control decision.

6 Acknowledgements

This work is part of the ARCE++ project (TSI2004-03394) funded by the Spanish Ministry of Science and Education (MEC) and a cooperation agreement between Universidad Carlos III de Madrid and Direccin General de Proteccin Civil y Emergencias (Ministry of the Interior).

References

[1] I. Aedo, P. Díaz, C. Fernández, and J. Castro. Supporting efficient multinational disaster response through a web-based system. *E-Gov Conference. Aix-en-Provence (France).*, pages 215–222, Sept. 2002.

[2] ANSI. Incits 359 2004. american national standard for information technology. role-based access control, 2004.

[3] S. Bajaj. Web services policy framework. http://www-128.ibm.com/developerworks/ library/specification/ws-polfram/index.html, 2004.

[4] G. Brose, M. Koch, and K. Lohr. Integrating security policy design into the software development process. Technical Report B-01-06, Freie Universitat, Berlin, Nov. 2001.

[5] J. Crampton. Specifying and enforcing constraints in role-based access control. In *Proceedings of SACMAT 2003, Como, Italy*, pages 43–50, 2003.

[6] P. Díaz, I. Aedo, and F. Panetsos. Labyrinth, an abstract model for hypermedia applications. description of its static components. *Information Systems*, 22(8):447–464, 1997.

[7] P. Díaz, I. Aedo, and F. Panetsos. Modelling the dynamic behavior of hypermedia applications. *IEEE Transactions on Software Engineering*, 27(6):550–572, June 2001.

[8] P. Díaz, D. Sanz, and I. Aedo. Marah: an rbac model and its integration in a web server. In K. Morgan and M. Spector, editors, *The Internet Society: Advances in Learning, Commerce and Security*, pages 243–252. WITPress, 2004.

[9] W. A. W. Group. Web services architecture. http://www.w3.org/tr/2004/note-ws-arch-20040211/, 2004.

[10] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. Generalized temporal role based access control model (gtrbac) (part i) - specification and modeling. https://www.cerias.purdue.edu/infosec/bibtex_archive/archive/2001-47.pdf, 2001.

[11] S. Montero, I. Aedo, and P. Díaz. Generation of personalized web courses using rbac: Courba, a practical experience. *AH 2002*, pages 419–423, May 2002.