HAL open science

Scalable and Efficient Reasoning for Enforcing Role-Based Access Control

Tyrone Cadenhead, Murat Kantarcioglu, Bhavani Thuraisingham

To cite this version:

Tyrone Cadenhead, Murat Kantarcioglu, Bhavani Thuraisingham. Scalable and Efficient Reasoning for Enforcing Role-Based Access Control. 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy (DBSEC), Jun 2010, Rome, Italy. pp.209-224, 10.1007/978-3-642-13739-6_14 . hal-01056687

HAL Id: hal-01056687

https://inria.hal.science/hal-01056687v1

Submitted on 20 Aug 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Scalable and Efficient Reasoning for Enforcing Role-Based Access Control

Tyrone Cadenhead *, Murat Kantarcioglu, and Bhavani Thuraisingham

Department of Computer Science,
The University of Texas at Dallas,Richardson, TX 75083
{thc071000, muratk, bxt043000}@utdallas.edu
http://www.utdallas.com

Abstract. Today, many organizations generate large amount of data and have many users that need only partial access to resources at any time to collaborate in making critical decisions. Thus, there is a need for a scalable access control model that simplifies the management of security policies and handles the heterogeneity inherent in the information system. This paper proposes an ontology-based distributed solution to this problem, with the benefits of being scalable and producing acceptable response times.

1 Introduction

Organizations have large or varied access policies that suggest a need for more scalable and interoperable systems. There is still work to be done in addressing these two issues, in particular to build integrated access control systems that are simple to manage and require little changes as growth occurs within the organization. The semantic web offers benefits to organizations that wish to migrate to take advantage of the interoperability, reuse and the semantics it offers. However, access control is still a concern with this migration. This web environment must respect the privacy of sensitive information that it stores, and also support the implementation of models that grant privileges to the data. The Health Insurance Portability and Accountability Act, HIPAA [11], require that we prevent unintended disclosure of any part of the patient's record, thus requiring careful coordination and rules that associate users to the various segments of patients' records. To model such constraints, we utilize an existing security model, role-based access control (RBAC) [5, 7], which already simplifies the security management of resources. In this paper, we leverage this feature of RBAC with existing semantic web technologies to allow for flexible integration and easily extensible semantic rules to automatically enforce access policies and ensure consistency of policies. In RBAC, the administrator statically defines privileges for users based on the user's role by associating permissions with the roles. This,

* This work is partially supported by Air Force Office of Scientific Research MURI Grant FA9550-08-1-0265, National Science Foundation Grant Career-0845803 and National Institutes of Health Grant 1R01LM009989

however, does not consider the case when a patient visits the emergency room for immediate treatment and the user is not preassigned to a role with access to the record. Although in this paper we use the healthcare domain as an illustration, our approach is applicable to other domains as well.

To make RBAC more dynamic, we extend our solution to the temporal RBAC (TRBAC) [2, 13] model to allow roles to be enabled for a duration of time, which will allow temporary privileges to users for various reasons. Organizations, however, have different structures and disparate platforms supporting their security guidelines, and so the extended model must be able to support existing systems that may already have their own access control mechanisms. In addition, such a model must address the scalability and manageability issues that may arise as more systems are integrated. Therefore, we need an access control model that is not only temporally flexible, but also provides interoperability between platforms without sacrificing the simplicity of the model. In general, ontologies enable reconciliation and translation between different standards and so our extended model could incorporate both the properties of ontologies and RBAC. For example, hospital operational environments require collaboration between different specialists and the exchange of their expertise and knowledge [26, 14], together with flexible access to records and resources. Ontologies also adhere to a description logic (DL) modeling formalism, thus adding the benefits of a DL knowledge base and an inference service, which we can use to check the consistency of policies. Also, we can extend the expressiveness of an ontology knowledge base with semantic rules. Moreover, using ontologies would provide the added benefits to build a scalable and efficient semantic web implementation of TRBAC that could provide accurate reasoning about what privileges the user should have on a resource.

The main contributions of this paper are the following: (i) to implement TRBAC using existing semantic technologies; (ii) to reason about particular users over a large number of instances in a DL knowledge base (KB); and (iii) to offer the ability to efficiently and accurately reason about access rights under any situation.

To achieve the first objective, we transform the access control policies into the semantic web rule language (SWRL) [12], which is more consistent with the web ontology language (OWL) specification (the W3C Recommendation for ontologies [18]). Second, various organizations, like hospitals, can integrate our semantic TRBAC model with their existing policies. Finally, using our solution allows the storing of data in a standard data interchange format, the querying and the reasoning to be done by a semantic query language and a description logic reasoner.

We realize the second objective by partitioning our knowledge base (which has a terminology box (TBox) and a assertion box (ABox)) into a set of smaller knowledge bases, which have the same TBox but a subset of the original ABox. A TBox consists of a set of classes and properties that model a domain, and the ABox contains the instances created based on the TBox. In addition, the underlying knowledge base data is expressed in the Resource Description Framework

(RDF) [22], the standard language for storing metadata about web resources. A global knowledge base normally resides in memory and offers real-time reasoning and querying, but is not scalable as we stream more data to it. The main reason is that the instances in the ABox grow as we scale our implementation, while the size of the TBox remains relatively the same. We propose a solution that only perform reasoning with a subset of instances (from the smaller ABoxes) in memory for a given RBAC session.

Finally, our proposed solution attempts to address two key objectives. First, efficiency determines the response time to make a decision; and second, correct reasoning ensures that all the data assertions (facts) are available when applying the security policies. To illustrate, we consider the healthcare domain, where making decisions in emergencies for a patient are critical and must be made in short periods of time.

This paper is organized as follows. Section 2 reviews previous work. Section 3 provides a theoretical background for the reader. Section 4 describes our architecture. Section 5 details our approach and techniques used to realize the scalability and efficiency of our implementation. Section 6 presents our experimental results. Section 7 provides a discussion of future research directions.

2 Related Work

Research has been active in the area of access control dealing with reasoning and scalability [4, 27, 6, 15, 17]. Previous approaches to modeling RBAC with description logic include the work by Cirio et. al. [4]. Their approach extends RBAC with contextual attributes and use a DL reasoner to classify users and resources and also verify the consistency of the access control policies. Moreover, they allow roles to be determined based on the users' attributes, which is unlike our approach where roles are temporally determined. Another approach is that done by Zhao et. al.[27]. They describe a formalism of RBAC using the DL language $\mathcal{ALCQ}$. In their work, they show how to use DL to model policy constraints (separation of duty, security role hierarchies). Furthermore, Finin et. al. [6] show different approaches to support RBAC in OWL. They investigate the use of OWL to unify parallel approaches to policy needs in real world domains. They discuss the use of rules and attributes in enforcing policies in SWRL and N3 logic with respect to examples with separation of duty. However, they did not show any experimental results for their domain. Our approach adds one more facet to the RBAC model, by addressing the scalability and efficiency aspects of RBAC reasoning with a temporal extension. There is also work on access control in the policy language XACML (the AOSIS standard for access control), where [15] addresses reasoning aspects and [17] focuses on the scalability aspects. Some other approaches address the scalability or the efficiency issues [20, 23, 9]. For example, Levandoski and Mokbel [16] store RDF data into relational databases, and use clustering and partitioning to reduce the number of joins and improve querying time. Database approaches to reasoning, however, require all instances of the data in the entire knowledge base (KB). To the best of our knowledge, this

is the first work that implements TRBAC in a semantic web environment, with an emphasis on the healthcare domain, using a distributed modular approach.

3 Theoretical Background

We first present a description of the standard RBAC and the extended features in TRBAC . Next, we provide a brief background of description logics and rules.

3.1 RBAC

This model [5, 7] generally comprises of loosely coupled components: (i) a *user* is usually a human or an autonomous agent; (ii) a *role* is a collection of permissions needed to perform a certain job function; (iii) a *permission* is an access mode that can be exercised on an *object*; and (iv) a *session* relates a user to roles.

- $PA : Roles \rightarrow Permissions$ the permission assignment function, that assigns to roles the permissions needed to complete their jobs;
- $UA : Users \rightarrow Roles$ the user assignment function, that assigns users to roles;
- $user : Sessions \rightarrow Users$, that assigns each session to a single user;
- $role : Sessions \rightarrow 2^{Roles}$, that assigns each session to a set of roles; and
- $RH \subseteq Roles \times Roles$, a partially ordered role hierarchy (written $\geq$).

To avoid confusion of the term role, we will use the term "role" when we are referring to the roles in the RBAC model, and instead use the term "property" when we are referring to the binary relations in DL.

3.2 TRBAC

The Temporal-RBAC (TRBAC) model as described in [2] is an extension of RBAC models that supports temporal constraints on the enabling/disabling of roles. TRBAC also supports periodic role enabling and disabling, and temporal dependencies among such actions. Such dependencies are expressed by means of role triggers that can also be used to constrain the set of roles that a particular user can activate at a given time instant. The firing of a trigger may cause a role to be enabled/disabled either immediately, or after an explicitly specified amount of time. The enabling/disabling actions may be given a priority that may help in solving conflicts, such as the simultaneous enabling and disabling of a role. We now formally introduce TRBAC extensions as presented in [2].

A periodic time is represented as the pair $\langle [begin, end], P \rangle$, where P is a periodic expression denoting an infinite set of periodic time instants, and $[begin, end]$ is a time interval denoting the lower and upper bounds that are imposed on instants in P. Two event expressions $enableR_1$ and $disableR_2$ are conflicting if $R_1 = R_2$. $(Prios, \preceq)$ is a totally ordered set of priorities, such that for all $x \in Prios$, $begin \preceq x \preceq end$ and we write $x \prec y$ if $x \preceq y$ and $x \neq y$. Periodic events are of the form $(I, P, p : E)$, where I is a time interval; P is a periodic

expression; and $p : E$ is a prioritized event expression with $p \prec end$. Also, the role triggers are of the form: $E_1, \ldots, E_n, C_1, \ldots, C_k \rightarrow p : E$ after Δt, where the $E_i s$ are simple event expressions, the $C_i s$ are role status expressions, $p : E$ is a prioritized event expression with $p \prec end$, and Δt is a duration expression.

3.3 Description Logics

Just for this section, we will adopt the normal meaning of roles to formally describe the language $\mathcal{ALCQ}$. This language like other DL languages are decidable fragments of first order logic (FOL).

$\mathcal{ALCQ}$ consists of a countable set of individuals Ind, a countable set of atomic concepts CS, a countable set of roles RS and the concepts built on CS and RS as follows:
$C, D := A|\neg A|C \sqcap D|C \sqcup D|\exists R.C|\forall R.C|(\leq nR.C)|(\geq nR.C)$,
where $A \in CS, R \in RS$, C and D are concepts and n is a natural number. Also, individuals are denoted by $a, b, c, \ldots$.

This language includes only concepts in negation normal form. The complement of a concept $\neg(C)$ is inductively defined, as usual, by using the law of double negation, de Morgan laws and the dualities for quantifiers. Moreover, the constants $\top$ and $\bot$ abbreviate $A \sqcup \neg A$ and $A \sqcap \neg A$, respectively, for some $A \in CS$. An interpretation $\mathcal{I}$ consists of a non-empty domain, $\Delta^{\mathcal{I}}$, and a mapping, $.^{\mathcal{I}}$, that assigns

- to each individual $a \in Ind$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- to each atomic concept $A \in CS$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
- to each role $R \in RS$ a relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

The interpretation I extends then on concepts as follows:
$\neg A^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$
$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \exists y((x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$
$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \forall y((x, y) \in R^{\mathcal{I}} \Longrightarrow y \in C^{\mathcal{I}})\}$
$(\leq nR.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \#\{y|((x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\} \leq n\}$
$(\geq nR.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \#\{y|((x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\} \geq n\}$

We can now define the notion of a knowledge base and its models. An $\mathcal{ALCQ}$ knowledge base KB is the union of

1. a finite terminological set (TBox) of inclusion axioms that have the form $\top \sqsubseteq C$, where C is called inclusion concept, and
2. a finite assertional set (ABox) of assertions of the form $a : C$ (concept assertion) or $(a, b) : R$ (role assertion) where R is called assertional role and C is called assertional concept.

We denote the set of individuals that appear in KB by $Ind(KB)$. An interpretation I is a model of

- an inclusion axiom $\top \sqsubseteq C$ ($I \models \top \sqsubseteq C$) if $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$

- a concept assertion $a : C$ ($I \models a : C$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- a role assertion $a, b : R$ ($I \models (a, b) : R$) if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Let KB be the $\mathcal{ALCQ}$-knowledge base of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. An interpretation I is a model of KB if $I \models \phi$, for every $\phi \in \mathcal{T} \cup \mathcal{A}$. A knowledge base KB is consistent if it has a model. Moreover, for φ an inclusion axiom or an assertion, we say that $KB \models \varphi$ (in words, K entails φ) if for every model I of K, $I \models \varphi$ also holds.

The consistency problem for $\mathcal{ALCQ}$ is ExpTime-complete. The entailment problem is reducible to the consistency problem as follows: Let KB be an $\mathcal{ALCQ}$ knowledge base and d be an individual not belonging to $Ind(KB)$. Then,

- $KB \models \top \sqsubseteq C$ iff $KB \cup \{d : \neg C\}$ is inconsistent and
- $KB \models a : C$ iff $KB \cup \{a : \neg C\}$ is inconsistent.

This shows that an entailment can be decided in ExpTime. Moreover, the inconsistency problem is reducible to the entailment problem and so, deciding an entailment is an ExpTime-complete problem too.

3.4 SWRL Rules

A SWRL rule has a function-free Horn-like syntax that share the common unary and binary predicates with its OWL counterpart. The antecedent of a SWRL rule is referred to as the rule body and the consequent as the head. The head and the body are composed of a conjunction of one or more atoms. SWRL rules reason about OWL individuals. Atoms can be of the form $C(x)$, $P(x, y)$, $sameAs(x, y)$ or $differentFrom(x, y)$, where C is an OWL DL description, P is an OWL property, and x, y are either variables, OWL individuals or OWL data values. The model-theoretic semantics for SWRL is a straightforward extension of the semantics for OWL. Extensions of OWL interpretations are bindings that map variables to elements of the domain [12]. Horn rules and DL languages such as $\mathcal{ALCQ}$ and OWL-DL ($\mathcal{SHOIN}(\mathcal{D})$) are decidable, but when the latter is combined with SWRL, the language may no longer be decidable. However, [19] proposes a combination of both languages that is decidable, where each variable of a rule is required to occur in the body (i.e. the DL-safe rules).

4 Overview of Our Approach

Our architecture is composed of three stages:

1. We partition the global KB, KB_{global}, into n smaller KBs, as shown in Figure 1(a). Then, we store each partition to disk by indexing each assertion (or asserted statement) subparts, namely the subject, predicate and object (s p o), referred to as a RDF triple [22, 3]. This facilitates faster retrieval at run-time once we identify the target KB for the assertion. In addition, we build this step offline and we restrict each partition size to ensure that it fits into the memory on the machine.

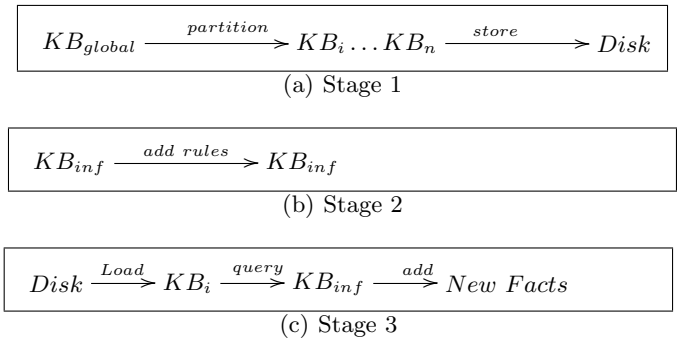


Fig. 1. Architecture

2. Then, we load the SWRL rules into a new KB, KB_{inf} (see Figure 1(b)). These rules are a finite set of authorizations defined by the organization's security policies, and therefore they are used to determine which assertions are relevant to determine any policy objective. Adding SWRL rules to KB_{inf} does not have a huge impact on the reasoning time as indicated by our experimental results. This is due to the fact that we are only retrieving a small subset of triples which reduces the number of symbols in the ABox when the rules are applied.
3. Finally, starting with one patient, we retrieve the relevant assertions from disk, one step at a time, and update our inference KB (see Figure 1(c)). Once this is done, the rules in KB_{inf} could also cause new facts to be added to KB_{inf}. Some of these facts could be special assertions to indicate what rules were fired. These special assertions could then provide a feedback as to whether the inference was consistent with the organization policies.

When there is an access request for a specific patient, we start executing stages two and three. Moreover, these two are our inferencing stages where we enforce the security policies. These can also be executed concurrently for many patients.

Moreover, this architecture improves performance. Firstly, the partitions, $KB_1, \ldots, KB_n$, function as materialized owl knowledge bases that perform inferencing when the data is loaded into them. In particular, performing consistency tests in the preprocessing stage avoids the delays in querying at run-time, since these tests are usually expensive [23] and OWL reasoning is exponential in time and memory in the worst case [1]. Secondly, KB_{inf} stores SWRL rules to do RBAC policy inferencing at run-time on a very small subset of assertions.

4.1 Defining KB partitions

We believe that performing reasoning in KB_{inf} instead of KB_{global} will scale our approach. We never directly apply the SWRL rules to KB_{global}, but instead we apply them to KB_{inf}, which has a smaller subset of instances. We believe that if we do not restrict the number of individuals in a KB, then all the satisfied SWRL

rules will fire and add many irrelevant assertions (which could be recursive). Each partition has a local reasoner, so by restricting the number of symbols in a KB, we can perform distributed local reasoning instead of full-scale reasoning. We think this is more efficient in a RBAC setting.

Definition 1. *(Domain Modules). The set $\mathfrak{D}$ consists of disjoint modules, where*

- *RBAC defines $Users, Roles, Sessions, Permissions,$ and $Objects$;*
- *the hospital extends $Users$ to employees; $Roles$ to the organizational structure; $Objects$ to Records (plus other resources like equipments, etc.); and*
- *the hospital defines $Patients$ (plus other stakeholders like suppliers, etc.).*

In order to ensure that the TBox, $\mathcal{T}$, is the same in all our KBs but the ABox, $\mathcal{A}$, is different, we define $KB_i = \{\mathcal{T}, \mathcal{A}_i\}$. Also, we allow arbitrary partitioning. For example, $KB_{patient}$ is further partitioned into $KB_{patient_1}, KB_{patient_2} \ldots$. Furthermore, $KB_{global} = (\mathcal{T}, \mathcal{A}_{global})$ and we distribute the ABox, $\mathcal{A}_{global}$, over each KB_i, such that $\mathcal{A}_{global} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \ldots \cup \mathcal{A}_n$.

4.2 Mapping RBAC to Description Logic

We use $\mathcal{ALCQ}$ Knowledge Bases to represent the concepts in the TRBAC model. For example, the underlying core RBAC concepts are represented in the TBox terminology. In addition, Role hierarchy is supported by concept hierarchy in the TBox. For example, since *Roles* is a finite set of job descriptions, they are organized so that family physicians and emergency physicians are both physicians. In this paper, we focus on scalability, but a more detail work of mapping RBAC to $\mathcal{ALCQ}$ can be found in [27].

5 Our Approach

Performing reasoning with large ABoxes must be efficient and scalable. Already, there are reasoners that implement an optimized version of the tableaux algorithm. For example, Pellet, a complete DL reasoner [24], performs well with moderate TBox and ABox sizes in memory. There are also some approaches to perform scalable [9] and optimized reasoning in a modular fashion. Other approaches intend to use secondary storage by partitioning the ontology [24].

Table 1. Memory Exception after adding 1000 individuals + 16(Rules)

Inference Statistics					
Individuals	112	336	560	784	1008
Time(ms)	152	211	276	448	552

Table 1 shows the performance of scaling the ABox, when all assertions and rules are in one knowledge base (which causes memory exception). This prompts

us to divide the global KB into autonomous KBs so we could take advantage of the loosely coupled RBAC components. We use SPARQL [25] [21] , a query language for RDF data, to retrieve both new and existing knowledge from our knowledge bases. When we need to make a policy decision, we issue SPARQL queries over the relevant partitions and then combine each partition result with the ones already in KB_{inf}, so that we can perform further reasoning tasks. Such technique is also called conjunctive query [24, 8], and is suitable for reasoning and retrieving instances over large ABoxes.

5.1 Partitioning

We first define terminologies for special properties (or mappings) that restrict how the domain modules communicate with each other, then we define how an assertion is placed into a partition, and finally, we give special names for a mapping that physically connect two KBs.

Definition 2. *(Mapping Function). The set $\mathfrak{M}$ consists of unique atomic properties (binary relations) connecting two domain modules in $\mathfrak{D}$ so that we have:*

- *RBAC assignments: the mappings user-role, role-user, role-permission, permission-role, user-session, role-role and role-session;*
- *Hospital extensions: the mappings patient-user, user-patient and patient-session; and*
- *Patient-Record constraint: the one-to-one mappings patient-record and record-patient,*
  where $user \in Users$, $role \in Roles$, $permission \in Permissions$, $session \in Sessions$, $patient \in Patients$ and $record \in Records$.

Definition 3. *(Home Partition). We define a home partition KB_i for all the triples, where $i = 1, 2, \ldots n$, such that*

- *the TBox, $\mathcal{T}$, is in KB_i; and*
- *for all assertions of form $C(x)$ and $R(x, Y)$, both have the same home, KB_i, and C is a concept, R is an atomic property, x is restricted to individuals in $\mathfrak{D}$ and Y is either an individual or a literal (for object or dataType property respectively). In particular, the home is determined by x, the domain of R.*

Definition 4. *(P-link). A P-link is a directed arc that allows navigation from one materialized KB to the next. An atomic property $\rho \in RS$, the set of properties in $\mathcal{ALCQ}$, is a P-link if $\rho \in \mathfrak{M}$. Also, a P-link has a home partition.*

The representation of the RDF triples on disk must be able to facilitate scalable reasoning, as well as answer TRBAC queries. The basic intuition is that we do not need global knowledge to answer TRBAC queries. To facilitate this, we connect triples from one partition to the next by using partition links (*P-links*). This allows the combining of local knowledge bases without physically joining the partitions. Also, the number of *P-link*s is finite: (i) the basic RBAC modules are fixed, (ii) we only have a finite set of *P-links* definitions in the

TBox (since there is a finite set of property definitions); and (iii) our ABox is distributed over a finite set of partitions of finite sizes. Another benefit of these links is that they allow arbitrary partitioning.

5.2 A Query-retrieval Process

Definition 5. *(Policy Query). Given a set of partitions KS, a policy (or access) query q against KS is a tuple (s, α, K, Ψ, o), where s is of the form $[t_1, t_2]$, α is an individual, K is an ordered set of partitions, Ψ is a set of access policy rules and o is the output of a positive query. Furthermore, K represents a flow and is of the form $\langle KB_1 \prec \ldots \prec KB_m \rangle$ such that $KB_1 \prec KB_2$ means KB_1 precedes KB_2, and the query process starts from KB_1 and ends in KB_m. Also, KB_k and KB_{k+1} are connected by a P-link, where $k < m$. In addition, Ψ are SWRL rules of the form : $H_1 \wedge \ldots \wedge H_{m'} \longleftarrow B_1 \wedge \ldots \wedge B_{n'}$ where $B_i, H_j, 1 \leq i \leq n', 1 \leq j \leq m'$ are atoms of the following form $C(i)$ or $P(i, j)$.*

A policy query for a patient *Bob* in session $[t_1, t_2]$ would be
$([t_1, t_2], Bob, \langle KB_{patient} \prec KB_{user} \prec KB_{role} \rangle, \Psi, o)$.

A rule in Ψ would be:$Patient(?x1) \bigwedge patUser(?x1, ?x2) \bigwedge patSess(?x1, ?x4) \bigwedge patRec(?x1, ?x3) \bigwedge userRole(?x2, ?x5) \bigwedge userSess(?x2, ?x4) \bigwedge roleSess(?x5, ?x4) \bigwedge rolePerm(?x5, ?x6) \longrightarrow canAccess(?x2, ?x3) \bigwedge grantPerm(?x2, ?x6)$,
which means that a *user* on duty, who plays the appropriate *role* (e.g. patient's physician) will be granted access to the patient's *record* within the specified *session* (e.g. the patient's session).

Figure 2 outlines a trace of a policy query for a patient *Bob* entering the hospital at interval $[t_1, t_2]$. At each stage of the query, we are retrieving a set of results (on the right) for the individual (possibly many) and session on the left. In the diagram, we assume *Sam* is *Bob*'s physician.

To determine the first output, o_1, we issue the first query against $KB_{patient}$. We then execute the other queries against KB_{doctor} and KB_{role}, so that at each step the results are added to KB_{inf}, until we have $o = o_1 \cup \ldots \cup o_3$.

Queries are also used to retrieve facts from a KB, including those inferred as a result of the SWRL rules in KB_{inf}. However, under the open world assumption of OWL and SWRL, a negative response to a query is not treated as failure to prove the query (a KB is assumed incomplete). Furthermore, OWL assumes monotonic reasoning. This is unlike the database approach, which uses default reasoning, where the assumption is that what is not known to be true is believed to be false. Nevertheless, in the partitioned approach we presented for RBAC, a policy query is positive if $o_i \neq \varnothing \; \forall i$. This avoids the problem of a user having the ability to perform unnecessary and potentially harmful actions merely as a side effect of granting access using incomplete knowledge from a set of KBs.

5.3 Optimization and Heuristics

We use two types of indexing to improve the efficiency of our approach by (i) indexing the assertions; and (ii) creating a high level index. In the first case, we

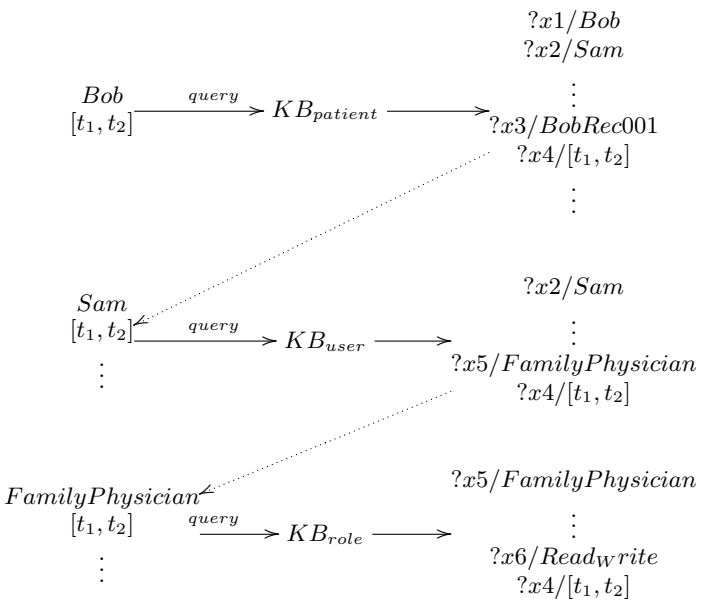


Fig. 2. A trace for a patient *Bob*.

index the subparts (s p o) of a triple to facilitate faster retrieval at query time. We do this with LARQ [1], which is a text indexing API for SPARQL. This allows us to find a triple by a subject (s), a predicate (p) or an object (o), without the cost of a linear search over all the triples in a partition. In the second case, we keep another level of index structure that points to the location of the partitions on disk (by using a Lucene index [10]). At retrieval time, we use the index to locate a partition for an assertion. Thus, retrieving the relevant assertions is at most linear with respect to the number of partitions.

We limit n for concepts $C \sqsubseteq \geq nR.C$ and $C \sqsubseteq \leq nR.C$. For instance, with n_u users and n_r roles, the combination of users and roles is at most $n_u \times n_r$. Also, a given user, a, could theoretically have n_r assertions of form $R(a, r_i), i = 1 \ldots n_r$, and this could cause the home partition of a to not fit into memory for local reasoning. In our domain, however, this is less likely to be the case; for example, there are different competences among users and roles are based on the user's ability. Moreover, an added benefit of TRBAC is the temporal constraints on roles; during an interval, the number of enabled roles may not be that large.

5.4 Correctness

As stated in Section 3.3, complex concepts are built from atomic concepts in CS, and properties in RS of the language $\mathcal{ALCQ}$. Table 2 displays the predicate

[1] `http://jena.sourceforge.net/ARQ/lucene-arq.html`

logic translation of these concepts [1, sec. 2.2].

Table 2. $\mathcal{ALCQ}$ rule engine

Group 1	
$D(x) \longleftarrow C(x)$	$C \sqsubseteq D$
$C(x) \longleftarrow R(x,y), D(y)$	$\exists R.D \sqsubseteq C$
$D(y) \longleftarrow R(x,y), C(x)$	$C \sqsubseteq \forall R.D$
$\neg A(x)$	$\neg A$
Group 2	
$C(x) \longleftarrow \exists y_1 \ldots y_n.R(x,y_1) \wedge \ldots \wedge R(x,y_n) \wedge \bigwedge_{i<j} y_i \neq y_j$	$C \sqsubseteq \ \geq nR.D$
$\bigvee_{i<j} y_i = y_j \longleftarrow \forall y_1 \ldots y_{n+1}.R(x,y_1) \wedge \ldots \wedge R(x,y_{n+1}),\ C(x)$	$C \sqsubseteq \ \leq nR.D$
Group 3	
$C(x) \longleftarrow D_1(x) \wedge \ldots \wedge D_n(x)$	$D_1 \sqcap \ldots \sqcap D_n \sqsubseteq C$
$C(x) \longleftarrow D_1(x) \vee \ldots \vee D_n(x)$	$D_1 \sqcup \ldots \sqcup D_n \sqsubseteq C$

Let F infer assertions from KB_i, such that $F(\mathcal{T}, \mathcal{A}_i) = Inf_i$, Q be a query over a set of triples and S be a subset of KBs. For *RBAC system* discussed in this paper, our partitioning based reasoning scheme correctly infers all the necessary triples needed for enforcing security policy.

Theorem 1. $Q(F(KB_{global})) \equiv F(\bigcup_{i \in S} Q_{S_i}(F(KB_i)))$

Proof Sketch Our goal is to prove that our partitioning scheme correctly infers all the relevant triples associated with a given session, user, role and permission. Basically, we argue that in order to correctly infer all the triples associated with the KB_{global}, we can just do reasoning using each partition and combine the selected results of the local reasoners.[2]

To prove this claim, we will use the rule engine given in Table 2. First, we argue that the information needed to use the rules given in Table 2 is already captured by the TBox and the local ABox instances. To prove this we will examine all the rules given above and argue that correct application of those rules could be done without combining instances in different partitions.

Please note that the first rule in group 1 could be correctly applied by just using the TBox. The second and third rules in group 1 could be correctly applied by using the triple $(x\ R\ y)$ given in some local partition and TBox since the definition of R in TBox precisely specifies the domain and the range of the relation. The fourth rule in group 1 is just the negation of an atomic concept.

For rules in group 2, for correct reasoning, we need to find out triples of the form $(x\ R\ y_i), \forall i$. Since our partitioning puts all the triples with subject x to

[2] Please note that, all the local reasoners share the same TBox

the same partition, all needed triples for correct inference will be in the same partition.

For the rules in group 3, we need to have all concepts D_i to be present at the time of reasoning. Clearly some of the concepts could be in different partitions. The way our system works is we query all the materialized results for each partition (i.e., $F(KB_i)$) related to RBAC query and get the D_i concepts needed. Using these D_i concepts and a TBox, we infer all $C(x)$ and associated triples in memory (i.e., in KB_{inf}). □

In our partition approach, we evaluate each policy query (s, α, K, Ψ, o), against the ordered set K of materialized KBs.

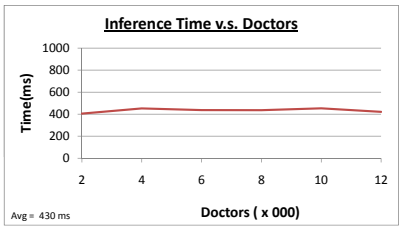
Therefore, $Q(K) = Q(\langle KB_1 \prec \ldots \prec KB_m \rangle) \equiv Q(\langle KB_{global} \rangle)$.

6 Experiments and Results

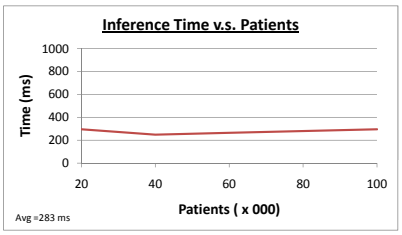
We perform experiments with synthetic data supporting large number of users, patients, rules, and other RBAC assertions (like sessions and role instances) on a Dell Inspiron 2.4GHz with 8GB RAM. Each user and patient has on average 30 object and data type properties. We then collect the time for inferencing the action to be taken for each patient. In particular, we scale one variable in the set {(D)octors, (N)urses, (P)atients (R)ules, TBox} at a time, while keeping the other variables constant. We use Protégé to build our TBox, and Jena[3][3] to build our ABoxes and programmatically extend the TBox. We use Lucene [10] for indexing the partitions and LARQ with SPARQL for the sequential and incremental extraction of triples from each materialized partition. We use Java as a programming language and we plug in pellet[24] as a reasoner for Jena. Also, these technologies are all open source. In addition, we follow the $\mathcal{ALCQ}$ specifications, where we only support atomic properties in our TBox, i.e. no transitive, inverse properties, etc. The time to build a set of materialized KBs range from one to three hours, while the time to retrieve the assertions and check the policies in KB_{inf} are presented in Figure 3 and Figure 4. We use various sources, such as WebMD[4], pubmed [5], and related literature to investigate the healthcare domain, in order to implement healthcare-specific data. For the TRBAC domain, we use the reference literature to design the data and rules for the security aspects. Also, we create individuals and their properties randomly in Jena, using the appropriate set of resources and data types.

Performance of our inference KB, KB_{inf}, displays fluctuations as we simulate various activities in our experiment. This is due to the fact that the index of any of the most accessed KBs will cache previous results, and this has unpredictable behavior. Also, for our base-line, we compare our performances to those in [15], which evaluates reasoning for a similar policy language, XACML. In addition, we use a naive approach to generate assertions for an in memory KB (see Table 1).

[3] `http://jena.sourceforge.net/`
[4] `http://www.webmd.com/`
[5] pubmed.gov



(a) Scaling Doctors using constants: 4200(N), 2100(P), 16(R)

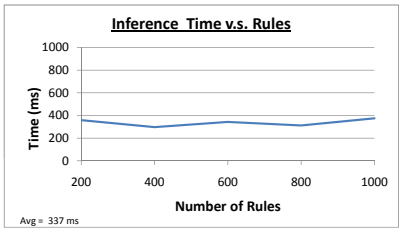


(b) Scaling Patients using constants: 3862(D), 3940(N), 16(R)

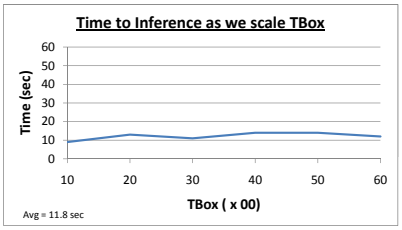
Fig. 3. Scaling Individuals

We perform various runs, each time with different combination of individuals from our hospital domain.

The results in Figure 3(a) and 3(b) show that we achieve almost constant time for determining a policy decision as we scale the instances in our application. These results are consistent with (i) initiating the flow with one patient and (ii) using an index for locating both a partition and an assertion.



(a) Scaling Rules using constants: 320(D), 640(N), 1280(P)



(b) Scaling TBox using constants: 320(D), 640(N), 1280(P), 16(R)

Fig. 4. Scaling TBox and Rules

The results in Figure 4(a) also show that even though we scale the number of rules, our run-time is fairly constant. This is because the rules only fire for one patient. The results in Figure 4(b), on the other hand, show that scaling our TBox terms does have some performance limitations. This could be that the DL reasoner must perform more expensive tests each time the TBox size increases. Our implementation is therefore quite scalable with respect to ABox and SWRL rules, but not so with TBox. Nevertheless, we expect the TBox size to be fairly constant for our chosen domain. Furthermore, when we compare our inference times to the verification times in [15] for policies using pellet reasoner, our approach gives similar performances for Figure 3(a), Figure 3(b) and Figure 4(a), where we use a small TBox.

7 Conclusions

We presented an implementation of a scalable TRBAC application, with particular reference to the healthcare domain. We addressed the problem of having a successful and scalable application in the semantic web, as well as highlighted the limitations of performing reasoning with very large ABoxes. We provided our solution to this problem by partitioning the ABoxes. We advocated that by retrieving only the relevant assertions, our reasoning task can be reduced from being global to being local and still correctly and efficiently determine the policy-decisions with respect to a patient. In the future, we would like to use data from other domains besides healthcare. In addition, we plan to use TBox-partitioning techniques to allow the scaling of concepts.

References

1. Baader, F., McGuinness, D.L., Nardi, D., and Patel-Schneider, P. F. : The Description Logic Handbook: Theory, Implementation and Applications: Cambridge University Press (2007)
2. Bertino, E., Bottati, P.A, Ferrari, E.: TRBAC: A Temporal Role-Based Access Control Model: ACM Transactions on Information and System Security August 2001, no. 3, vol. 3, pp. 191-223
3. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D. Seaborne, A., Wilkinson, K.: Jena: Implementing the Semantic Web Recommendations: HP Laboratories Bristol HPL-2003-146, 2003
4. Cirio, L., Cruz, I.F., Tamassia, R.: A Role and Attribute Based Access Control System Using Semantic Web Technologies: On the Move to Meaningful Internet Systems '07: OTM '07 Workshops, 2007
5. Ferraiolo, D.F., Sandhu, R., Gavrila, S.: Proposed NIST Standard for Role-Based Access Control: ACM Transactions on Information and System Security, pp. 224-274, August, 2001
6. Finin, T.W., Joshi, A., Kagal, L., Niu, J., Sandhu, R.S., Winsborough, W.H., Thuraisingham, B.M.: ROWLBAC-Representing Role Based Access Control in OWL: In. Proceedings of the 13th ACM symposium on Access control models and technologies, pp. 73-82, 2008
7. Ferraiolo, D.F., Kuhn, D.R., Chandramouli, R.: Role-Based Access Control: Artech House, Inc. (2003)
8. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive Query Answering for the Description Logic SHIQ: Journal of Artificial Intelligence Research 31 (2008) 157-204
9. Guo, Y., Heflin, J.: A Scalable Approach for Partitioning OWL Knowledge Bases: International Workshop on Scalable Semantic Web Knowledge Bases, 2006
10. Hatcher, E., Gospodnetic, O., McCandless, M.: Lucene in Action, Second Edition: Manning (2004)
11. HIPAA: U.S. Department of Health and Human Services, `http://www.hhs.gov/ocr/privacy/`
12. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML: W3C Member Submission 21 May 2004. Latest version is available at `http://www.w3.org/Submission/SWRL/`

13. Joshi, J. B.D., Bertino, E., Ghafoor, A.: A Generalized Temporal Role-Based Access Control Model: IEEE Transactions on Knowledge and Data Engineering, vol. 17, January, 2005
14. Kataria, P., Juric, R., Paurobally, S., Madani, K.: Implementation of Ontology for Intelligent Hospital Wards: Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)
15. Kolovski, V., Hendler, J., Parsia, B.: Analyzing web access control policies: Proceedings of the 16th international conference on World Wide Web, May 08-12, 2007, Banff, Alberta, Canada
16. Levandoski, J.J., Mokbel, M.F.: RDF Data-Centric Storage: ICWS, 2009
17. Liu, A.X., Chen, F., Hwang, J., Xie, T.: Xengine: a fast and scalable XACML policy evaluation engine: ACM Sigmetrics Performance Evaluation Review (SIGMETRICS '08), 2008
18. McGuinness, D.L., Harmelen, F.: OWL Web Ontology Language Overview: World Wide Web Consortium W3C Recommendation 10 February 2004. At `http://www.w3.org/TR/owl-features/`
19. Motik, B., Sattler, U., Studer, R.: Query Answering for OWL-DL with Rules. : Web Semantics: Science, Services and Agents on the World Wide Web, Volume 3, Issue 1, pp. 41-60, July 2005
20. Owens, A., Seaborne, A. Gibbins, N.: Clustered TDB: A Clustered Triple Store for Jena: WWW '09, 2009
21. Perez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL: ACM Transactions on Database Systems (TODS), 2009
22. Resource Description Framework (RDF): Concepts and Abstract Syntax: W3C Recommendation, 2004, `http://www.w3.org/RDF/`
23. Sirin, E., Parsia, E.: Optimizations for Answering Conjunctive ABox Queries: First Results: In. Proceedings of the '06 International Workshop on Description Logics, 2006
24. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A Practical OWL-DL Reasoner: Journal of Web Semantics, 2007
25. SPARQL Query Language for RDF, W3C Recommendation 15 January 2008, `http://www.w3.org/TR/rdf-sparql-query/`
26. Winter, A., Brigl, B., Wendt, T.: A UML-based Ontology for Describing Hospital Information System Architectures: Studies in health technology and informatics '01; pp. 778-82, 2005
27. Zhao, C., Heilili, N., Liu, S., Lin, Z.: Representation and Reasoning on RBAC: A Description Logic Approach: Theoretical Aspects of Computing - ICTAC '05, Second International Colloquium, pp. 381-393, 2005