Testing of Contextual Role-Based Access Control Model (C-RBAC)

Muhammad Nabeel Tahir m_nabeeltahir@hotmail.com
Multimedia University, Melaka
75450, Malaysia

Abstract

In order to evaluate the feasibility of the proposed C-RBAC model [1], the work in this paper presents the prototype implementation of C-RBAC model. We use eXtensible Access Control Markup Language (XACML) as a data repository and to represent the extended RBAC entities including purpose and spatial model.

Key words: C-RBAC Testing, XACML and C-RBAC, Policy Specification Languages

1 INTRODUCTION

1.1 EXtensible Access Control Markup Language (XACML)

The OASIS eXtensible Access Control Markup Language (XACML) is a powerful and flexible language for expressing access control policies used to describe both, policy and access control decision request / response [2]. XACML is a declarative access control policy language implemented in XML and a processing model, describing how to interpret the policies. It is a replacement for IBM's XML access control language (XACL) which is no longer in development. XACML is a language primarily aimed at expressing privacy policies in a form such that computer systems can enforce them. The XACML has been widely deployed and there are several implementations of XACML in various programming languages available [3]. The XACML is designed to support both centralized and decentralized policy management.

1.2 Comparison Between EPAL, XACML and P3P

Anderson [3] suggested that a standard structured language for supporting expression and enforcement of privacy rules must meet the following requirements:

Rq1. The language must support constraints on who is allowed to perform which action on which resource;

Rq2. The language must support constraints on the purposes for which data is collected or to be used;

Rq3. The language must be able to express directly-enforceable policies;

Rq4. The language must be platform-independent; and

Rq5. The language used for privacy policies must be the same as or integrated with the language used for access control policies.

Keeping in mind the above requirements, the comparison of P3P, EPAL, and XACML are summarized in Table 1 in which "√" means the language can satisfy the requirement, "×" means the language cannot satisfy the requirement and "?" means it is an unknown feature for the corresponding requirement and may depend on the language extension and implementation.

Table 1: Comparison of P3P, EPAL, and XACML (Anderson, 2005).

	P3P	EPAL	XACML
Rq1: Constraints on subject	×	√	√
Rq2: Constraints on the purposes	√	√	√
Rq3: Directly-enforceable policies	×	√	√
Rq4: Platform-independent	√	?	√
Rq5: Access control	×	×	√

Although P3P is a W3C recommended privacy policy language that supports purpose requirements and is platform-independent, P3P does not support directly-enforceable policies. P3P policies are not sufficiently fine-grained and expressive to handle the description of privacy policies at the implementation level. P3P mainly focuses on how and for what purpose information is being collected rather than on how and who can access the collected information. Thus, P3P is not a general-purpose access control language for providing technical mechanisms to check a given access request against the stated privacy policy especially in ubiquitous computing environment. EPAL supports directly-enforceable policies but it is a proprietary IBM specification without a standard status. According to a technical report comparing EPAL and XACML by Anderson [3], EPAL does not contain any privacy-specific features that are not readily supported in XACML. EPAL does not allow policies to be nested as each policy is separate with no language-defined mechanism for combining results from multiple policies that may apply to a given request whereas XACML allows policies to be nested. A policy in XACML, including all its sub-policies, is evaluated only if the policy's Target is satisfied. For example, policy "A" may contain two sub-policies "B1" and "B2". These sub-policies could either be physically included in policy "A" or one or both could be included by a reference to its policy-id, a unique identifier associated with each XACML policy. Thus making XACML more powerful in terms of policy integration and evaluation. EPAL [4] functionality to support hierarchically organized resources is extremely limited whereas XACML core syntax directly supports hierarchical resources [data-categories] that are XML documents. In an EPAL rule, obligations are stated by referencing an obligation that has been defined in the (vocabulary) element associated with the policy; in XACML, obligations are completely defined in the policy containing the rule itself. EPAL lacks significant features that are included in XACML and that are important in many enterprise privacy policy situations. In general, XACML is a functional superset of EPAL as XACML supports all the EPAL decision request functionality. XACML provide a more natural way of defining role hierarchies, permissions, permission-role assignment and it support the idea of complex permissions that are used in the systems implementing role-based access control models for distributed and ubiquitous environments. As a widely accepted standard, it is believed that

XACML is suitable for expressing privacy specific policies in a privacy-sensitive domain as healthcare.

2 CORE C-RBAC IMPLEMENTATION USING XACML

The implementation of core RBAC entities (USERS, ROLES, OBJECTS, OPS, PRMS) in XACML are presented in table 2.

Table 2: Core RBAC Entities in XACML.

Core RBAC Entities	XACML Implementation
USERS	<Subjects>
ROLES	<Subject Attributes>
OBJECTS	<Resources>
OPS	<Actions>
PRMS	<Policyset> <Policy>

The current XACML specification does not include the work for extended RBAC model but it has the core RBAC profile to implement the standard RBAC model. Therefore, XACML is further investigated and extended to support the proposed privacy access control model C-RBAC and privacy policies. Table 3 shows the proposed XACML extension for the privacy access control model.

Table 3: Extended Entities of C-RBAC Model.

C-RBAC ENTITIES	XACML/XML IMPLEMENTATION
PHYSICAL LOCATION	<PLOC>
LOGICAL LOCATION	<LLOC>
LOCATION HIERARCHY SCHEMA	<LHS>
LOCATION HIERARCHY INSTANCE	<LHI>
SPATIAL DOMAIN OVER LHS	<SSDOM>
SPATIAL DOMAIN OVER LHI	<ISDOM>
PURPOSE	<PURPOSE>
SPATIAL PURPOSE	<SP>
SPATIAL PURPOSE ROLES	<SPR>

2.1 Experimental Evaluation

We created different healthcare scenarios to analyze behavior of the proposed C-RBAC entities. By simulating different healthcare scenarios, we calculated response time including the access time (with and without authorization) and response time to derive spatial granularity, spatial purpose and spatial purpose role enabling and activation, have showed that the time required to collect contextual attributes, to generate a request and to authorize an access request have been in milliseconds and seconds that are considered to be tolerable in real time situations.

The use of XML as a tool for authorization raises questions as to expressiveness versus efficiency, particularly in a large enterprise. Ideally, authorization should account for a negligible amount of time per access but it is necessary that all access conditions be expressed and context be checked completely. In this implementation, all authorization policies are loaded into memory, independent of request comparison. Therefore, the time to read policies is not included into access time. Instead, authorization time consists of formal request generation, request parsing, contextual attribute gathering, request-policy comparison and context evaluation, response building, and response parsing. The experiments have been performed on a 2.66 GHz Intel machine with 1 GB of memory. The operating system on the machine is Microsoft Windows XP Professional Edition, and the implementation languages used is Microsoft C-Sharp (C#).

For the experimental evaluation, different healthcare scenarios that are mentioned throughout the thesis (the one presented in chapter 5 and section 7.3) have been executed to analyze the performance and expected output of C-RBAC model (Tahir, 2009a). According to those healthcare scenarios, contextual values including purpose setup, location modeling that include locations, location hierarchy schemas and instances, spatial purposes, spatial purpose roles and privacy policies have been defined in the system with their selectivity to 100 percent i.e. all policies, operations, purposes, locations and spatial purpose roles have been set to allow access for every access request. After creating the necessary objects and relations the response has been analyzed in order to verify that whether the proposed model correctly grant or deny access according to the privacy rules or not. Moreover, the response time has been also calculated at different levels to measure the computational cost for monitoring and evaluating the dynamic contextual values like purpose, location and time.

Figure 1 shows purpose inference algorithm based on the contextual values of the user. It includes time, location, motion direction, distance and user motion direction with measurement unit as meter, centimeters etc.

```
PurposeInference (s, $pos_1$, $pos_2$) {
// s $\in$ SESSIONS, $pos_1$ and $pos_2$ are user's current position and the position
to which user is heading to;

//Step 1: Getting the subject roles through the active session
SPR spr = SessionSPR(s);

//Step 2: Getting the current time
Time t = DateTime.Now;

//Step 3: Getting ploc in which user is located
PLOC $ploc_1$ = Ploc($pos_1$);
PLOC $ploc_2$ = Ploc($pos_2$);

//Step 4: Getting motion direction
DIRECTION dir = PlocDir($ploc_1$, $ploc_2$);

//Step 5: Getting distance measurement unit
DUnitPloc($ploc_2$) $\rightarrow$ dunit

//Step 6: Getting distance between the two physical locations
Distance dval = DisPloc($ploc_1$, $ploc_2$)

//Step 7: Retrieving the corresponding spatial purposes from the spatial
purpose global file (refer to figure 7.10)

Purpose p = Get_Purpose(spr, t, dir, $pos_1$, dval, DUnit)

Return p;
}
```

Figure 1: Purpose inference algorithm

Figure 2 shows the response time of purpose inference algorithm. As shown, the response time increases as the number of purpose inference requests increase. This is because of the constant movement of the user over the space defined within the system. For a single request, the system takes approximately 38 milliseconds to compute the purpose from the collected contextual attributes that are necessary input to the purpose inference algorithm.

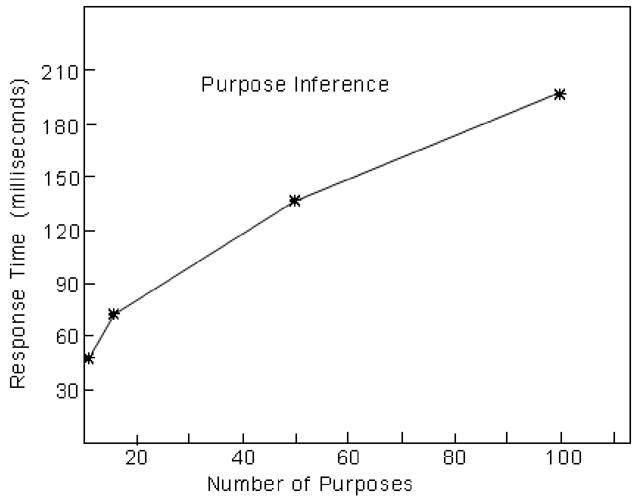

Figure 2: Purpose Inference Response Time.

Figure 3 shows the response time in general for purpose collection based on the user's current contextual attributes. Figure 4 shows the response time for purpose collection at location hierarchy schema and instance level. As shown, the response time increases as the number of logical or physical locations defined in schema or instances increases. It also shows that the response time at schema level is less than that of instance. This is because for each instance, the system collects the spatial purposes defined not only at an instance level but also from its corresponding schema from which it is instantiated (lhi is instance of lhs). Thus, the response time increases as the location granularity becomes finer.

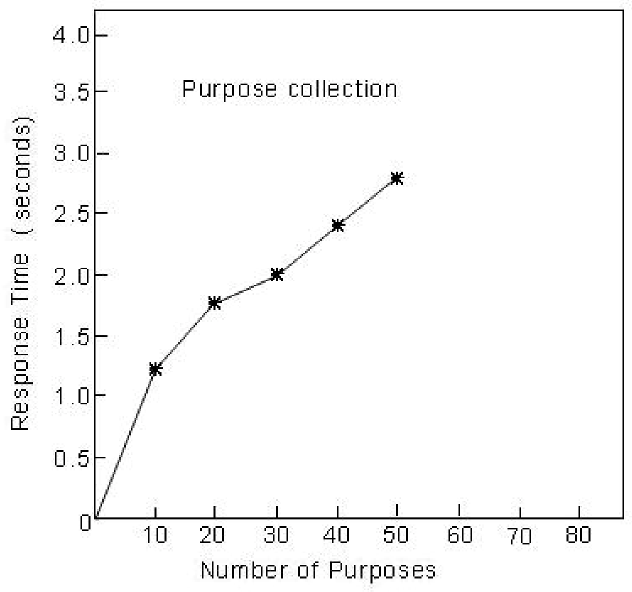

Figure 3: Purpose Collection Response Time in General.

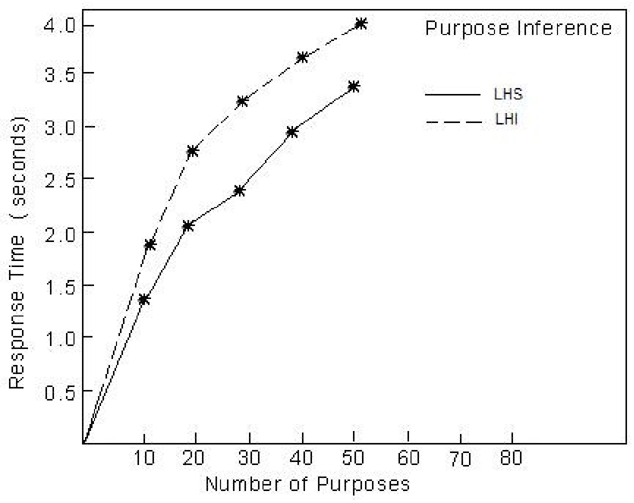

Figure 4: Purpose Collection Response Time at LHS and LHI Level.

Figure 5 shows spatial granularity mapping from LHS to logical locations lloc defined within the schema. It also shows the mapping response time to generate a set of physical locations ploc that are derived from lloc defined within the given LHS. Figure 6 shows the response time to derive physical locations from a given LHI.

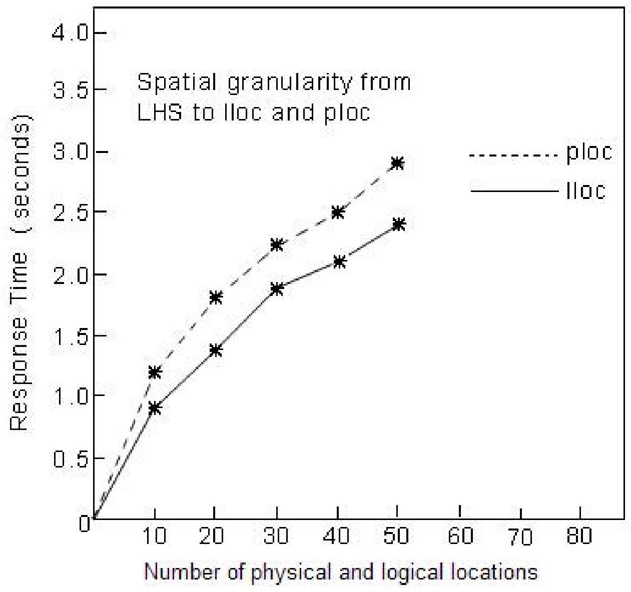

Figure 5: Response Time to Derive Physical and Logical Locations from a Given LHS.

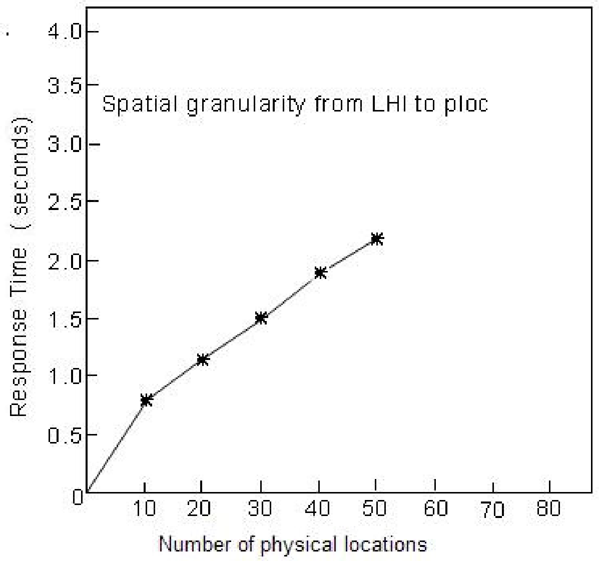

Figure 6: Response Time to Derive Physical Locations from a Given LHI.

Figure 7 shows the response time to activate a spatial purpose through C-RBAC constraints defined within the system. It has been observed that the activation of spatial purposes depends on the spatial granularity. For example the spatial purposes defined at location hierarchy schema level took more time to activate as compared to spatial purpose at physical location level. This is because at physical level, the system directly activate the spatial purpose for the given purpose and physical location whereas in case of location hierarchy schema, the system had to derive all logical locations and then to its corresponding physical locations first and then activate those corresponding physical locations with the given purpose.

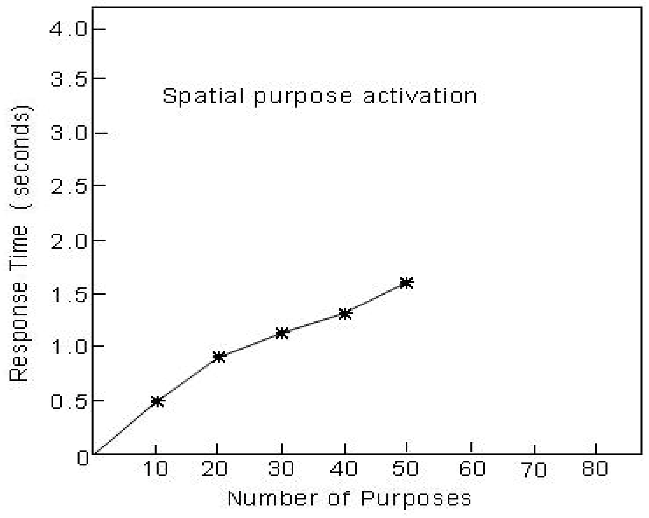

Figure 7: Response Time to Activate Spatial Purposes.

Figure 8 shows the response time to enable spatial purpose roles defined with different spatial granularities and purposes. The results have been analyzed by enabling a single spatial purpose role spr (without spatial purpose role hierarchy) and multiple spr in the presence of hierarchy. It is noticed that the enabling of roles defined without hierarchical relationships is less than to those defined with hierarchical relationships. This is because in case of hierarchical relationships, constraints are applied and evaluated based on the contextual values of the user before the system enable/disable spatial purpose roles defined within the C-RBAC implementation.

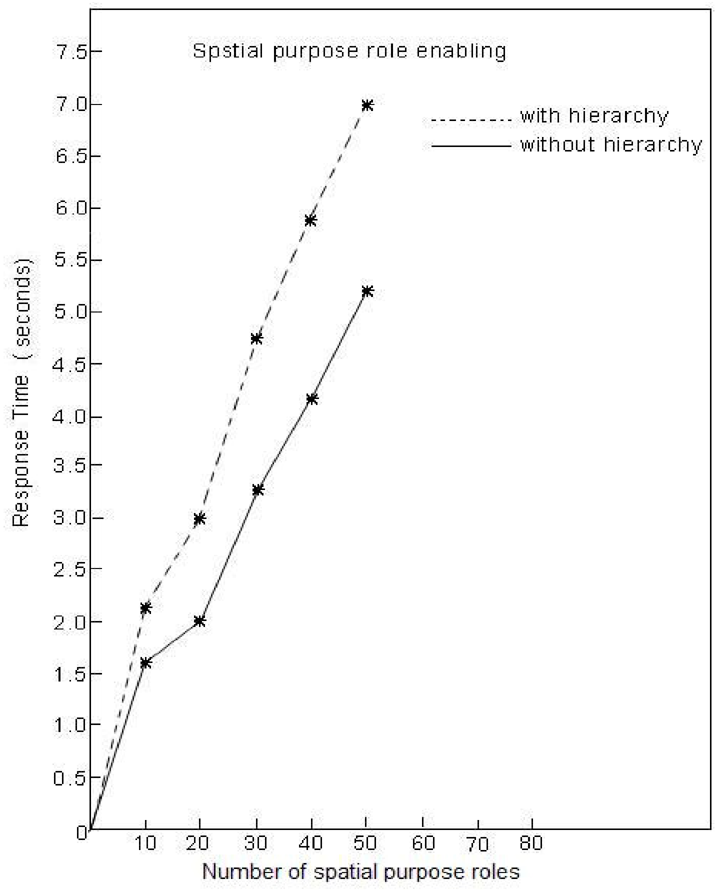

Figure 8: Response Time for Spatial Purpose Roles Enabling (with and without Hierarchical Relationships).

Figure 9 and 10 shows the response time for spatial purpose role activation and mapping of user session onto enabled and active spatial purpose roles respectively.

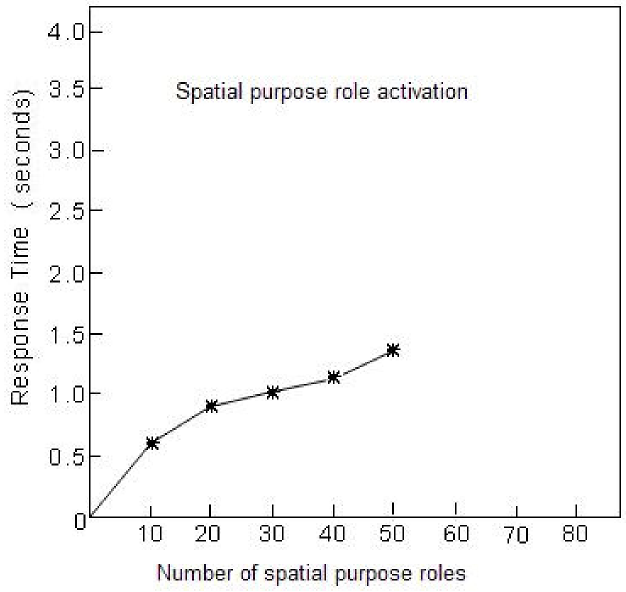

Figure 9: Response Time for Spatial Purpose Roles Activation.

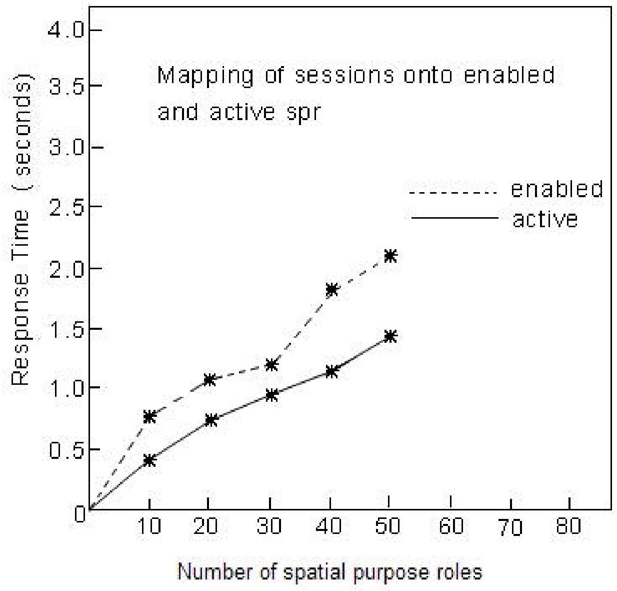

Figure 10: Response Time for Mapping a User Session Onto Enabled and Active Spatial Purpose Roles.

```
while(true){
          //Step 1: Gets the access requests from the subject
          If request(SUBJECTS s, OPS op, OBJECTS o, PURPOSES {$p_1$,$p_2$ ..., $p_n$ },
          RECIPIENTS {$rp_1$,$rp_2$ ..., $rp_n$}) {

               //Step 2: Processes the request
               //Step 2.1: Checks the object ownership
               OWNERS owr = object_owner(o)

               //Step 2.2: Checks the subject role
               ROLES r = subject_roles(s)

               //Step 2.3: Retrieves the corresponding privacy rules
               PRIVACY-RULES rule = GetPrivacyRules(r, op, o, {$p_1$,$p_2$ ..., $p_n$},{$rp_1$,$rp_2$ ...,
               $rp_n$})

               //Step 3: Makes a decision by
               DECISIONS d = deny or allow;

               //Step 3.1: Checks permission from the core C-RBAC model
               PRMS prms = assigned_permission($spr_{loc\_type, p}$)

               //Step 3.2: Checks legitimate purposes
               If(p' $\wedge$ rule.p = {$p_1$,$p_2$ ..., $p_n$}){
                       //Step 3.3: Checks legitimate recipients
                       If(rule.rp = {$rp_1$,$rp_2$ ..., $rp_n$}){

                       //Step 3.4: Checks the location granularity
               If (loc_type $\wedge$ rule.loc_type = {lloc, ploc, lhs, lhi, $sdom_{lhs}$, $sdom_{lhi}$}) {

                       //Step 3.5 Checks ssod and dsod constraints
               If ($r_{loc\_type, p}$) {
                       Apply_SSoDConstraints($r_{loc\_type, p}$);
                       Apply_DSoDConstraints($r_{loc\_type, p}$);

                       //Step 3.6 Final decision
                       d = rule.decisions
                       OBLIGATIONS {$obl_1$, $obl_2$ ..., $obl_n$} = rule.obligations
                       RETENTIONS rt = rule.retentions
               } } } }

               //Step 4: Returns a response and an acknowledgement
               If(d = allow){
                       //Step 4.1: Returns: allow, Obligations, Retention policy
                       Response(d, {$obl_1$, $obl_2$ ..., $obl_n$},rt)
                       } Else {
                       //Step 4.1: Returns deny, null, null
```

Figure 11: Access Control Decision Algorithm for the Proposed Privacy Based C-RBAC

It is observed that the response time to enable spatial purpose roles is more than that of activation and mapping time. This is because of object/classes based implementation in C# of the proposed C-RBAC model. During the execution of different healthcare scenarios, it is observed that at the time of login, the system has evaluated the contextual values of the user and enabled all the spatial purpose roles assigned by the administrator. From implementation point of view, role enabling means that the system loads all the assigned roles into the memory based on the contextual values and SSoD constraints. Then for each change in the user's context, the system decides whether to activate or deactivate the spatial purpose role by based on the DSoD constraints and new contextual values. Figure 11 shows the access control algorithm to evaluate the user's request and to grant/deny access based on the contextual values, enabled and activated roles.

For authorization, request generation time is approximately 2 seconds. The request parsing time is 1.28 seconds. The average time for the PDP to gather attributes and authorize a formal request is 3.5 seconds. All local transfer times are less than 1 seconds. Therefore, the total time to authorize an access is 6.78 seconds.

The average total time to determine which regular spatial purpose roles a user has assigned is 776 ms. Role assignment is trivially parallelizable because each role can be checked independently, so taking a distributed approach or using multi-threads could reduce this number to a fraction of this original value. If the time is reduced to a tenth of the original, it would take 77 ms to determine a user's roles.

Without authorization, the average time to perform an access is 703 ms. When authorization is added into this system, the total time for an authorized access is 7483 milliseconds (6.78 * 1000 + 703 = 7483 milliseconds = 7.5 seconds approximately). The 6.78 seconds access authorization time is 89% of the total system time. This additional time is easily tolerated in a system where tens of milliseconds are not critical. Role assignment can be determined per session or per access. The 77 milliseconds this process took is invisible during the login process. Per access, this 77 milliseconds added to the 6780 milliseconds (7.78 seconds) for authorization would account for 88% of the 7483 milliseconds (7.5 seconds) total access time. This result is still tolerable. Based on the results generated by measuring the response time for spatial granularity derivation, spatial purpose and spatial purpose role enabling and activation, request generation and evaluation and response time, it is concluded that the extensions introduced by C-RBAC are reliable and due to very less overheads, the model can be effectively used for dynamic context-aware access control applications.

3. CONCLUSION

In this paper, we simulated the different healthcare scenarios to analyze the behavior and to calculate the response time of the proposed C-RBAC model. Our findings include the access time (with and without authorization) and response time to derive spatial granularity, spatial purpose and spatial purpose role enabling and activation, have showed that the time required to collect contextual attributes, to generate a request and to authorize an access request have been in milliseconds and seconds that are considered to be tolerable in real time situations. The model implementation and its results also showed that the extensions introduced by C-RBAC have been reliable and due to very less overheads, the model can be effectively used for dynamic context-aware access control applications.

4. REFERENCES

[1] Tahir, M. N. (2007). Contextual Role-Based Access Control. Ubiquitous Computing and Communication Journal, 2(3), 42-50.

[2] OASIS (2003). A brief introduction to XACML. Retrieved November 14, 2008, from http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.htm.

[3] Anderson, A. (2005). A comparison of two privacy policy languages: EPAL and XACML. Sun Microsystems Labortory Technical Report #TR-2005-147, November 2005. Retrieved November 14, 2008, from http://research.sun.com/techrep/2005/abstract-147.html.

[4] IBM (2003). Enterprise privacy authorization language (EPAL). IBM Research Report June 2003. Retrieved November 14, 2008, from http://www.zurich.ibm.com/security/enterprise-privacy/epal.