Inter-ReBAC: Inter-operation of Relationship-based Access Control Model Instances

Jason Crampton and James Sellwood

Royal Holloway University of London, Egham, United Kingdom.
jason.crampton@rhul.ac.uk,
james.sellwood.2010@live.rhul.ac.uk

Abstract. Relationship-based access control (ReBAC) models define authorization policies and make authorization decisions on the basis of relationships between the entities in a system. We present a framework through which multiple ReBAC model instances can interoperate so that requests initiated in one system may target resources in a second system. Further, our framework is able to support requests passing through a chain of inter-connected systems, thus enabling many systems to be connected together or a single large system to be decomposed into numerous component subsystems. Whilst the underlying principles of this framework can be applied to any ReBAC model, we introduce its formal application to our RPPM model [3], the first, and most actively developing, general computing ReBAC model.

Keywords: access control; path condition; relationship; principal matching; policy graph; principal activation; authorization; secure inter-operation

1 Introduction

Access control is a fundamental security service, employed within a system to manage the interaction between (user) processes and (system-protected) resources. Historically, access control in general-purpose computing systems has used discretionary and role-based access control (RBAC) models. Recent work on *relationship-based access control* (ReBAC), inspired originally by social networks [1, 2, 6], has shown that alternatives are viable for both specialist and general computing applications [4, 5, 8]. In ReBAC, the specification and evaluation of authorization policies is based on the relationships which exist between entities of a system. The use of paths of relationships has several advantages over roles, not least because they can naturally reflect the more complex, context-specific nature of human interaction which otherwise requires the use of many, highly parameterized, roles [3].

Whilst the consideration of paths of relationships is key to request evaluation in any ReBAC model, different models support varying capabilities and

constructs for representing and processing these relationships. However, the relationships between entities in the modelled system are, intuitively, always represented as an edge-labelled graph. Such system graphs indicate the scope of a model instance; the system's entities (vertices in the graph), relationships (labelled edges), and policies are bounded by the graph and the system's authorization requests are evaluated within it. Whilst a system is modelled by a single ReBAC model instance (and therefore by a single graph), there are several motivating situations in which multiple model instances may need to interact.

Firstly, systems are frequently connected together in order to support a wider range of services. In such cases, a subject in one system may request to perform an action on an object in a remote system. Currently such requests cannot be supported in ReBAC models unless a single "super-graph"[1] model captures every entity and relationship within the two, and all intermediary, systems. This requires global policies, outside of the "authority" of any one component system, putting at risk the autonomy of all. The inter-connection of discrete autonomous subgraphs is, therefore, desirable. Secondly, ReBAC models containing very large system graphs (stand-alone, or super-graph as just discussed) will be impacted by the fact that request evaluation in such models has a time complexity linked to the number of nodes in the graph, whether they are relevant to the request being evaluated or not. It is, therefore, desirable to decompose such very large system graphs into smaller discrete autonomous subgraphs. Whilst request evaluation complexity in each of these subgraphs will still be linked to the number of nodes, the practical complexity is expected to be greatly reduced as many requests will only involve a subset of subgraphs (assuming appropriate routing is available).

We, therefore, develop a framework by which two ReBAC system graphs may be connected and requests initiated in one may be authorized in the other. We introduce a formal application of this framework in RPPM, as we believe RPPM is naturally suited to this task. It supports the modelling of general computing environments, to which inter-operation is highly relevant. Further, its two step request evaluation process and use of security principals provide a natural "break point" at which a request may be transferred from one system to another (particularly when employing graph-based policies). RPPM provides a convenient basis for combining paths from multiple graphs without searching end-to-end across the, potentially numerous, inter-connected system graphs.

Due to space limitations we do not provide a review of the operation of stand-alone RPPM instances; this background can be found in [3, 4]. Section 2 introduces the Inter-RPPM framework as a specific example of our ReBAC inter-operation approach. Section 3 describes its request evaluation process. In Section 4 we discuss related work and in Section 5 we draw conclusions. The Appendix contains pseudocode for three algorithms required to support Section 3.

[1] We use the term super-graph to identify a large system graph which models multiple systems which might otherwise be modelled by distinct stand-alone system graphs.

2 Inter-RPPM

The goal of our inter-operation framework is to connect distinct and autonomous system graphs in such a way that we are not required to define and evaluate relationship paths traversing a single super-graph. We, therefore, introduce an inter-operation framework which maintains the autonomy of individual system graphs by preserving their individual request evaluation scopes, system models and policies. In order to provide connectivity between system graphs, a construct external to the system graphs is required. We, therefore, introduce the concept of a *bridged system group* and employ bridging relationships[2], called *bridges*, between *hub* entities within distinct component system graphs. A request to access a remote resource will need to be evaluated by a sequence of RPPM systems. Bridges provide the link through which we propagate information about the outcome of each system's "local" request evaluation to the next system in the sequence; where it subsequently informs another local evaluation.

We employ a hierarchical dot notation to label elements: that is $G.v$ and $G'.v$ represent different nodes (with the same label v) when G and G' are distinct system graphs.[3] When all of the elements of a tuple, such as (v, v', r), belong to the same system graph G, we will write $G.(v, v', r)$ to simplify notation.

Definition 1. *Let* $\mathcal{G} = \{G_1 = (V_1, E_1), \ldots, G_n = (V_n, E_n)\}$ *be a set of system graphs. A* bridge *is an edge of the form* $(G_i.v, G_j.v, \textsf{bridge-to})$ *such that* $G_i \neq G_j$. *A* bridged system group *is a pair* $(\mathcal{G}, \beta)$, *where* β *is a set of bridges.*

Informally, a bridged system group comprises two or more RPPM model instances whose system graphs are connected by one or more bridges. Each bridge connects two hub entities; one each from two distinct component system graphs. We illustrate this framework with a simple example which we construct from two, initially disconnected, model instances, identified by their system graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, shown schematically in Figure 1a.

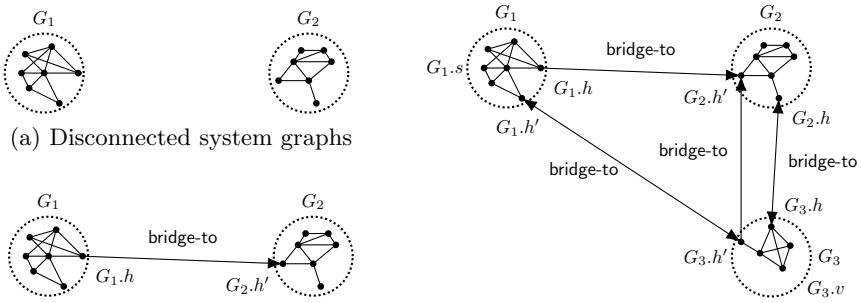
In Figure 1b we illustrate the inter-connection of G_1 and G_2 through the bridge $(G_1.h, G_2.h', \textsf{bridge-to})$ and develop the example further in Figure 1c to incorporate three system graphs inter-connected via six bridges.[4] Bridges are directed – $(G_1.h', G_3.h', \textsf{bridge-to})$ connects $G_1.h'$ to $G_3.h'$, but not vice versa – and are represented by an arrow. However, there may also exist a bridge $(G_3.h', G_1.h', \textsf{bridge-to})$ in which case we will use a double-headed arrow to represent the pair of bridges between $G_1.h'$ and $G_3.h'$.

Any bridged system group defines inter-system paths, obtained by traversing the bridges. A *system graph sequence* defines the sequence of system graphs along

[2] We require an extra-model administrator to be responsible for the management of these bridging relationships.

[3] Where there is no ambiguity through context or element naming we continue to leave out the prefix for convenience and clarity. So we do not prefix V_1 and E_1 in $G_1 = (V_1, E_1)$, for example.

[4] Note that the example of Figure 1c could equally represent three distinct system graphs which have been connected together, or a large stand-alone system graph which has been decomposed into three subsystems.



(a) Disconnected system graphs

(b) Bridged system group (1,2)

(c) Bridged system group (1,2,3)

Fig. 1: Bridged system group examples

such a path, where no system graph may be repeated and the directionality of the bridging relationships constrain the sequence. System graph sequences are a key component of the request evaluation process as they identify a path from the system originating a request to the system graph containing the target object. Multiple such paths may exist; we require that the access control policy decision point (PDP) within a system be able to determine the single, least cost path.

To achieve this we require that an extra-model administrator assign costs to bridges, and that hub entities maintain and exchange system path information using a modified path-vector routing protocol. Each hub communicates with adjacent hubs to which it directs a bridging relationship, retrieving the cost of their total (least) cost path to every other hub, and therefore every other system graph, in the bridged system group (where an infinite cost indicates an unreachable hub).[5] Upon receiving these path costs the hub adds to each the cost of the bridge connecting it to that neighbour; it will then update its local system path table to reflect the least cost path (if it didn't already know it) to every remote hub, along with which adjacent hub each least cost path passes via. A system's PDP can collate this information from each of its local hub entities in order to determine the single, least cost path to a target system graph, or to determine that no such path exists. Henceforth, when identifying a system graph sequence between two system graphs we assume the least cost such sequence.

3 Request Evaluation in Inter-RPPM

The basic RPPM model evaluates local requests within a stand-alone system graph using two steps: *compute principals* (where principal-matching rules are evaluated to determine whether security principals match to the request) and *compute authorizations* (where authorization rules are evaluated to determine if the matched security principals are authorized to perform the requested action).

[5] The directionality of bridges is enforced by hub entities not exchanging system path information against the direction of an incident bridge.

Within Inter-RPPM, remote requests are made by a subject $G.s$ to perform a remote action $G'.a$ (where $G \neq G'$) on a remote object $G'.v$. To support these remote actions we introduce an *originating remote request (ORR)* and an *incoming remote request (IRR)*.[6] The ORR represents the remote request as it is specified within the originating system graph. The IRR contains additional data which enable subsequent system graphs to contribute to the evaluation. When processing remote requests, every traversed system graphs' PDP employs the compute principals step (whether evaluating an ORR or IRR), but only the target system graph's PDP computes authorizations. Essentially, non-originating system graphs re-compute the set of matched principals based on the set computed by the preceding system graph and identified in the IRR.

A *policy graph* is used during the compute principals step to order the evaluation of principal-matching rules and to enable principals to be "activated" when specific other principals are matched to a request.

Definition 2. *Given a system graph $G = (V, E)$ and a set of principals P, a* policy graph G_ρ *is a directed acyclic graph with a unique root (of in-degree 0) such that each vertex is a principal-matching rule (ϕ, ψ, p). The set of principals includes a special principal called the* null *principal and the principal-matching rule for the root is defined to be* $(\mathsf{all}, \mathsf{none}, \mathsf{null})$.

The edges of the policy graph determine which rules trigger other rules. The concept of the principal-matching rule's target is extended to support rules in which the targets (positive, ϕ, and negative, ψ) may identify sets of principals which must exist (or not) in the current set of matched principals or, as previously, paths of relationships which must exist (or not) in the system graph. More formally, we evaluate a policy graph with respect to a request, in order to compute a set of matched principals, as per Algorithm 1 (see Appendix).[7]

A remote request is initiated in the originating system graph as an originating remote request (ORR), which is evaluated as per Algorithm 2 (see Appendix).[8]

Definition 3. *Given two distinct system graphs $G = (V, E)$ and $G' = (V', E')$ in a bridged system group $(\mathcal{G}, \beta)$, an* originating remote request *takes the form $G.q = (G.s, G'.v, G'.a)$,*[9] *where $G.s \in V$, $G'.v \in V'$ and $G'.a \in G'.A$.*

When processing the ORR, the originating system's PDP must take into account that the target object is located in a distinct system graph from the

[6] Recall that a system graph sequence will document the chain of system graphs between the originating system graph and the target system graph.

[7] Note that we do not add the null principal to the set of matched principals (line 4 of Algorithm 1) so that it does not interfere with authorization decisions, no matter the conflict resolution strategy employed; and that the algorithm is passed a set of matched principals to which any it matches are added.

[8] In Algorithm 2 and Algorithm 3 calls to EvaluatePolicyGraph have been simplified, removing the system graph and policy graph arguments so as to highlight the start entity, target entity, and set of matched principals arguments.

[9] Note that the originating remote request $G.q = (G.s, G'.v, G'.a)$ retains the same underlying *structure* as a local request made within the same system graph $G.q = G.(s, v, a)$.

subject. Therefore, the ORR is processed by performing compute principals between the subject and the hub entity which links the originating system graph to the next graph in the system graph sequence (lines 1, 3 and 4).[10] Note that line 4 passes an empty set to the EvaluatePolicyGraph algorithm (defined in Algorithm 1) as no principals have yet been matched to this remote request. The result of processing the policy graph is a set of matched principals for traversing the originating system graph. This is then passed, along with the details of the request, across the bridging relationship to the next system in the system graph sequence (lines 5 and 6).[11]

In non-originating system graphs, the remote request takes the form of an incoming remote request (IRR), evaluated as per Algorithm 3 (see Appendix); its processing differs in intermediate and target system graphs. Within an IRR, the subject component of the ORR is replaced with a tuple identifying the originating subject, the hub entity through which the request was received by the current system and the set of matching principals which resulted from the preceding system's processing of the request.

Definition 4. *Given a bridged system group $(\mathcal{G}, \beta)$ and a system graph sequence $(G_1, \ldots, G_{i-1}, G_i, \ldots, G_\ell)$, an* incoming remote request *processed by system graph $G_i = (V_i, E_i)$ takes the form $G_i.q = ((G_1.s, G_i.h, G_{i-1}.[\![\rho]\!]), G_\ell.v, G_\ell.a)$, where $G_i.h \in V_i$ is the hub through which the request entered G_i and $G_{i-1}.[\![\rho]\!]$ is the set of matched principals computed by the preceding system graph.*

When processing an IRR in an **intermediate system graph**, the system's PDP must take into account that neither the subject nor target object are located in the current system graph. Therefore, the IRR is processed by performing compute principals between two hub entities: the hub entity $G_i.h$ through which the request entered the system and the hub entity which links the intermediate system graph to the next graph in the system graph sequence (lines 4 and 5). The set of matched principals from the preceding system graphs is input into the graph policy evaluation (line 5) and may result in additional principals being activated (and added to it), as discussed in Section 2. As with processing an ORR, the cumulative set of matched principals that results is then passed, along with the details of the request, across the bridging relationship to the next system in the system graph sequence (lines 6 and 7). This process continues until it reaches the target system.

When processing an IRR in the **target system graph**, the policy graph evaluation is performed between the hub entity through which the request entered the system and the object of the request, note $G_i = G_\ell$ (line 10). Once again this makes use of the preceding system graph's set of matched principals. However, once a local set of matched principals is determined the compute authorizations step is performed. This allows a set of authorization decisions to

[10] The function DetermineLeastCostPathToSystemGraph requires the PDP to interrogate its local hubs and to collate their responses, as described in Section 2.

[11] An appropriate mechanism (e.g. digital signatures) must be in place to enable the receiving system to ensure the authenticity and freshness of such details.

be determined (line 11). A definitive decision is then determined, with RPPM's default decision process and conflict resolution process employed as with local requests (line 12).

4 Related Work

Whilst our work in respect of ReBAC inter-operation is novel, the inter-operation of other access control model instances has been considered in previous literature. In particular, research into the use of RBAC in multi-domain scenarios has led to several approaches to inter-operation based principally upon role mapping.

Shehab *et al.* define a distributed secure interoperability protocol in which users are assigned roles in remote domains based upon cross domain access agreements [10]. The order in which roles are acquired within each domain they access is referred to as the user's access path. These paths are checked to ensure the principles of autonomy and security [7] are both satisfied. In Inter-RPPM remote requests, by construction, are unable to target local resources and the outcome of local request evaluation is unchanged by the additions discussed in this paper. Therefore, both of these principles are satisfied trivially.

Shafiq *et al.* take this further by introducing a policy composition framework that integrates the RBAC policies of initially distinct domains [9]. They, additionally, implement a conflict resolution technique to deal with conflicts which may arise from the differences in how each domain models or references its access control policies. Within Inter-RPPM we have intentionally avoided integrating the policies of component model instances, and instead have provided a framework through which those policies may be applied to remote requests.[12] This places a greater requirement upon the manual definition of appropriate policies within each component model instance; however, it ensures a consistent policy language.

5 Conclusion

We have defined an inter-operation framework through which multiple RPPM system graphs may be connected and remote authorization requests may be evaluated. The connecting of systems is commonplace; however, the ability to decompose large systems into inter-connected smaller systems is a significant contribution of this paper and of particular importance in the application of ReBAC models. Decomposition enables a trade-off to be made between the computational complexity of a ReBAC model against the number of model instances which are employed to control authorizations within a large system.

The inter-connection of ReBAC (system) graphs through directed bridging relationships, and the use of system graph sequences and the path-vector routing protocol to identify paths between those graphs is applicable to remote request

[12] That being said we support the use of remote principals where desired to provide for more robust policies.

evaluation, no matter the model. This paper introduces its formal application in RPPM, although it is equally applicable to our more recent model, ARPPM [5] (something we leave for future work).[13] In cases where principal abstraction is not available, however, an alternative means of allowing each component graph's evaluation to contribute must be developed. We leave the development of such mechanisms to future work, only noting that an end-to-end path approach will produce a super-graph and thus, potentially, be limited by the increase in computational complexity associated with an increase in the number of nodes.

References

1. Carminati, B., Ferrari, E., Perego, A.: Enforcing access control in web-based social networks. ACM Trans. Inf. Syst. Secur. 13(1) (2009)
2. Cheng, Y., Park, J., Sandhu, R.S.: A user-to-user relationship-based access control model for online social networks. In: Cuppens-Boulahia, N., Cuppens, F., García-Alfaro, J. (eds.) DBSec. Lecture Notes in Computer Science, vol. 7371, pp. 8–24. Springer (2012)
3. Crampton, J., Sellwood, J.: Path conditions and principal matching: a new approach to access control. In: Osborn, S.L., Tripunitara, M.V., Molloy, I. (eds.) 19th ACM Symposium on Access Control Models and Technologies, SACMAT '14, London, ON, Canada - June 25 - 27, 2014. pp. 187–198. ACM (2014), http://doi.acm.org/10.1145/2613087.2613094
4. Crampton, J., Sellwood, J.: Relationships, paths and principal matching: a new approach to access control. CoRR abs/1505.07945 (2015)
5. Crampton, J., Sellwood, J.: ARPPM: administration in the RPPM model. In: Bertino, E., Sandhu, R., Pretschner, A. (eds.) Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy, CODASPY 2016, New Orleans, LA, USA, March 9-11, 2016. pp. 219–230. ACM (2016), http://doi.acm.org/10.1145/2857705.2857711
6. Fong, P.W.L.: Relationship-based access control: protection model and policy language. In: Sandhu, R.S., Bertino, E. (eds.) CODASPY. pp. 191–202. ACM (2011)
7. Gong, L., Qian, X.: Computational issues in secure interoperation. IEEE Trans. Software Eng. 22(1), 43–52 (1996), http://dx.doi.org/10.1109/32.481533
8. Rizvi, S.Z.R., Fong, P.W.L., Crampton, J., Sellwood, J.: Relationship-based access control for an open-source medical records system. In: Weippl, E.R., Kerschbaum, F., Lee, A.J. (eds.) Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, Vienna, Austria, June 1-3, 2015. pp. 113–124. ACM (2015), http://doi.acm.org/10.1145/2752952.2752962
9. Shafiq, B., Joshi, J., Bertino, E., Ghafoor, A.: Secure interoperation in a multidomain environment employing RBAC policies. IEEE Trans. Knowl. Data Eng. 17(11), 1557–1577 (2005), http://dx.doi.org/10.1109/TKDE.2005.185
10. Shehab, M., Bertino, E., Ghafoor, A.: SERAT: secure role mapping technique for decentralized secure interoperability. In: Ferrari, E., Ahn, G. (eds.) SACMAT 2005, 10th ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden, June 1-3, 2005, Proceedings. pp. 159–167. ACM (2005), http://doi.acm.org/10.1145/1063979.1064007

[13] We believe that administrative requests will always be local to a model instance (maintaining system autonomy) and so only operational requests (equivalent to the requests discussed in this paper) may be conducted remotely.

Appendix: Algorithms

The pseudocode for the algorithms used to support inter-connection of RPPM instances is shown below.

Algorithm 1 EvaluatePolicyGraph (ordered compute principals)

Require: System graph $G = (V, E)$, start node $u \in V$, target node $v \in V$, policy graph $G_\rho = (V_\rho, E_\rho)$, and set of matched principals $[\![\rho]\!]$
Ensure: Returns set of matched principals $[\![\rho]\!]$

while Perform breadth-first search of G_ρ starting at root vertex **do**
  Evaluate current vertex, $(\phi, \psi, p) \in V_\rho$
  if $G, u, v \models \phi$ **and** $G, u, v \not\models \psi$ **then**
    if $p \neq$ null **then**
      $[\![\rho]\!] \leftarrow [\![\rho]\!] \cup p$
    end if
  else
    Prune child vertices from evaluation
  end if
end while

Algorithm 2 ProcessORR

Require: System graph $G = (V, E)$, ORR $G.q = (G.s, G'.v, G'.a)$, policy graph $G.G_\rho = G.(V_\rho, E_\rho)$
Ensure: Malformed ORR rejected **or** set of matched principals $G.[\![\rho]\!]$ sent to next system graph in system graph sequence

$LCP_{G,G'} \leftarrow$ DetermineLeastCostPathToSystemGraph(G')
if G' reachable **then**
  $G.h \leftarrow$ IdentifyLocalHubForLCP$(LCP_{G,G'})$
  $G.[\![\rho]\!] \leftarrow$ EvaluatePolicyGraph$(G.s, G.h, \emptyset)$
  $G''.h \leftarrow$ IdentifyNeighbourHubForLCP$(LCP_{G,G'})$
  Securely send $G.q$ and $G.[\![\rho]\!]$ to $G''.h$ via $G.h$
else
  Reject malformed ORR
end if

Algorithm 3 ProcessIRR

Require: System graph $G_i = (V_i, E_i)$, IRR $G_i.q = ((G_1.s, G_i.h, G_{i-1}.[\![\rho]\!]), G_\ell.v, G_\ell.a)$, policy graph $G_i.G_\rho = G_i.(V_\rho, E_\rho)$
Ensure: Set of matched principals $G_i.[\![\rho]\!]$ sent to next graph in system graph sequence (intermediate system graph) **or** authorization decision made (target system graph)

if $G_i \neq G_\ell$ **then**
  // intermediate system graph
  $LCP_{G_i,G_\ell} \leftarrow$ DetermineLeastCostPathToSystemGraph(G_ℓ)
  $G_i.h' \leftarrow$ IdentifyLocalHubForLCP(LCP_{G_i,G_ℓ})
  $G_i.[\![\rho]\!] \leftarrow$ EvaluatePolicyGraph$(G_i.h, G_i.h', G_{i-1}.[\![\rho]\!])$
  $G_{i+1}.h \leftarrow$ IdentifyNeighbourHubForLCP(LCP_{G_i,G_ℓ})
  Securely send $G_i.q$ and $G_i.[\![\rho]\!]$ to $G_{i+1}.h$ via $G_i.h'$
else
  // target system graph
  $G_i.[\![\rho]\!] \leftarrow$ EvaluatePolicyGraph$(G_i.h, G_\ell.v, G_{i-1}.[\![\rho]\!])$
  $G_i.[\![\varrho]\!] \leftarrow$ ComputeAuthorizations$(G_i.[\![\rho]\!])$
  DecideAuthorizationResult$(G_i.[\![\rho]\!], G_i.[\![\varrho]\!])$
end if