Trust-Based Secure Workflow Path Construction

M. Altunay, D. Brown, G. Byrd, and R. Dean

North Carolina State University, Raleigh, 27695, USA
{maltuna, debrown, gbyrd, ralph_dean}@ncsu.edu

Abstract. Security and trust relationships between services significantly govern their willingness to collaborate and participate in a workflow. Existing workflow tools do not consider such relationships as an integral part of their planning logic: rather, they approach security as a run-time issue. We present a workflow management framework that fully integrates trust and security into the workflow planning logic. It considers not only trust relationships between the workflow requestor and individual services, but also trust relationships among the services themselves. It allows each service owner to define an upper layer of collaboration policies (rules that specify the terms under which participation in a workflow is allowed) and integrates them into the planning logic. Services that are unfit for collaboration due to security violations are replaced at the planning stage. This approach increases the services owners' control over the workflow path, their willingness for collaboration, and avoids run-time security failures.

1 Introduction

Workflow management tools, making use of available services created by the SOC-based communities, play a focal role in dissecting complicated user applications into smaller tasks, assigning each task to a suitable service, and orchestrating the workload among these services. Workflow management, requiring disparate services to collaborate and interact on demand, raises important security and trust issues. At the planning stage, a workflow engine must evaluate the trust relationships among the services as well as the trust relationships between the end user and individual services. The trust relationships among the services may be driven by factors such as industry-specific regulations, existing business partnership agreements, and competition among the services. Moreover, the identities of the workflow requestor and collaborating services, along with the prospective benefits from participation in a workflow, should have an impact on the willingness of a service to join a workflow. Orchestrating services without modeling such complicated trust relationships may result in security violations and reluctance of services for participation.

OGSA [1] and its implementation Globus Toolkit [2] are one of the most significant service-based efforts that provide the necessary middleware to support autonomous inter-organizational sharing of resources and services. Currently, the grid's primary administrative entity, the Virtual Organization (VO) [3], defines a community of users and imposes an "all-or-nothing" style of authentication and authorization. While resource administration is performed locally, in each owner's

B. Benatallah, F. Casati, and P. Traverso (Eds.): ICSOC 2005, LNCS 3826, pp. 382 – 395, 2005.

domain, the identities and roles of VO members are determined by VO-wide credentials and/or policies, thus making a resource's access control policies conform to the trust model established by the VO. Such uniformity of access eases the problems of resource discovery and selection, and allows the user to form workflows or pipelines of resources to solve complex problems.

The establishment of this community, however, requires infrastructure that can be cumbersome in a more dynamic environment characterized by short-lived collaborations and relationships. The grid-oriented VO requires pre-established trust relationships among the member organization. Service-oriented applications, on the other hand, rely on short-term *ad hoc* collaborations. These collaborating services may not share common goals – rather, they spontaneously collaborate on behalf of a third party, the workflow requestor. Such loose collections of web services may include resources owned by rival companies, be separated by corporate firewalls, otherwise be inhibited from working collaboratively, even on behalf of a third party (the end user) that is authorized for each resource individually. The data, the computations involved, and even the databases and queries used, can all be sensitive and proprietary, due to competitive and regulatory constraints. This style of "co-opetition" is not likely to happen under the VO model – a more dynamic, decentralized collaborative trust model is required.

Based on the characteristics of these new heterogeneous and dynamic environments, we have identified the following list of security requirements for workflow management. Throughout this paper, a *workflow participating entity (WPE)* is any resource (e.g., web service, computational resource, or storage site) that may be chosen to participate in a workflow. An execution path is the specific set of interactions among workflow entities that satisfies the requirements of the workflow. A workflow requestor is the end user on behalf of whom the workflow is executed.

- *Recognition of collaboration requirements among workflow participating entities.* Complex workflows may require a resource owner to interact and collaborate with many other WPEs on behalf of the workflow requestor. Trust models of these WPEs may prevent them from joining a given workflow due to interactions with other participants or the workflow requestor. Ignoring trust relationships between parties during the planning stage may result in security failures (e.g. access denials or firewall failures).
- *A decentralized workflow authorization model.* It is highly likely that WPEs constituting a workflow will have domain-specific security policies and requirements that are confidential [16]. Thus, workflow engines should have decentralized access control models that leave the final access decision to each WPE. Moreover, the workflow engine should not assume any knowledge about the internal security policies of each WPE.
- *Context-based, collaboration-aware access control mechanisms.* Classical identity-based models or the families of role-based (RBAC) [4] and task-based (TBAC) [5] access control models assume that a resource owner has prior knowledge of the user. This assumption is not adequate for today's highly dynamic, market oriented web services paradigm, wherein the services are offered to anyone with the necessary credentials. Proposed access control models based on trust management [6] address this problem. However, trust management-based access control models

still need to be incorporated with a high level abstraction that encompasses trust and collaboration policies, as described in Section 2. These policies will allow a resource owner to evaluate incoming access requests based on the context of a workflow and the established trust relationships among the WPEs.

Below, Figure 1 illustrates the workflow path construction throughout the planning stage. The planning engine starts with finding sets of candidate services that are functionally capable of performing the workflow tasks (candidate services are shown in curly brackets in Figure 1). Then each task is mapped to a service, and the workflow path is sent to the execution stage. We propose that the workflow planning engine needs to recognize, identify, and evaluate the complex trust relationships between WPEs during the planning stage. Modeling and evaluating trust relationships during planning stage is requisite if the above requirements are to be met. Thus, the workflow planning engine should (1) identify prospective collaborating parties from a given path, (2) orchestrate the trust evaluations among these parties, and (3) have a robust selection algorithm for building a secure and reliable workflow path from given candidate resources.

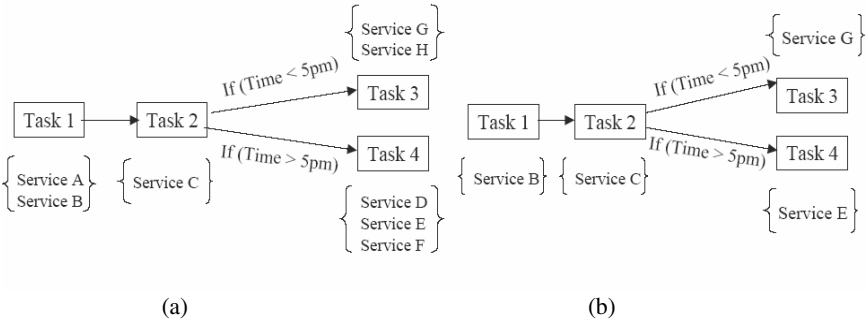

Fig. 1. (a) The tentative workflow path at the beginning of planning stage. (b) The workflow path at the end of the planning stage.

To support such environments, we present a workflow management framework (Section 3) that incorporates collaborative trust and security models at the workflow planning stage. Our framework provides an organizational model rather than an architectural specification. The primary component of this framework is a workflow engine that can identify and evaluate complex trust relationships among workflow entities. It provides a decentralized authentication and authorization model, in which each workflow entity is responsible for creating and enforcing its own access control policies. Our framework increases the availability of complex and trusted distributed environments for application communities that are unable or reluctant to create formal virtual organizations. It also provides the tools that support the complex trust relationships that already exist in these domains, allowing new opportunities for collaboration and scientific discovery.

2 New Workflow Paradigm: Collaboration-Based Secure Workflow Path Formation

Conceptually, during the execution of a typical workflow, the data transfer and control flow defined by the workflow engine specifies the neighboring relationships among arbitrary resources. The interactions occurring between WPEs in a workflow path can be examined in two categories: bilateral and indirect.

Any interaction between two WPEs that are immediate neighbors of each other is a *bilateral relationship*, even when the flow of interaction seems to be one-sided. To illustrate this, the simple collaboration scenario shown in Figure 2 seemingly involves a one-sided relationship: Service A presents an input file to Service B and Service B determines if it trusts Service A. However, there are actually two relationships: (1) Service A determines that it trusts Service B and agrees to share a copy of its result file, and (2) Service B determines that it trusts Service A to access with the specified input file. Both of these actions involve risk: from Service A's perspective, Service B could be a rival company that is not willing to share its results (i.e. application logic); from Service B's perspective, Service A could be a malicious user who sends a Trojan horse.

Indirect interactions occur between WPEs that are not immediate neighbors of each other, such as A and C in Figure 2. The interaction between such WPEs occurs through intermediate services. Based on the level of interaction, a service owner may want to put restrictions on the identities of its non-immediate neighbors. Moreover, the identities of such neighbors may significantly affect the willingness of a WPE to participate in a workflow. There are several reasons why such indirect trust relationships must be carefully evaluated. (1) Confidential documents or the results of a sensitive algorithm are typically passed among several WPEs throughout an execution path; thus even a non-immediate neighbor might have access to confidential data. (2) Industry-specific and government-based regulations [21] place important restrictions on the identity of collaborating partners, even when such collaborations are indirect. In industries such as health care and bioinformatics, every individual organization is held accountable for their direct and indirect collaborators with whom they exchange data. (3) Existing partnership agreements and competition among businesses prevent them from doing business with some certain organizations. Even when such interactions are safe from a security standpoint, the higher-level business logic forbids them.

Current workflow management systems ignore bilateral and indirect interactions, and only apply unilateral security checks, merely checking credentials of the workflow requestor against each WPEs. Two scenarios arise where our proposed framework provides a distinctive advantage over existing workflow planning systems. Presume that the user has authorized access to all WPEs, and has delegated his access rights to them.

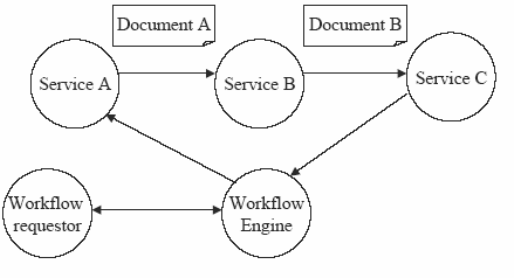

Fig. 2. Collaboration scenario

In the first scenario, a desired interaction cannot

occur and this is not determined until run-time. For example, Service B needs to access a file from Service A but security policies, perhaps a firewall rule set, deny the interaction. This fault will require a costly run-time re-evaluation of the planning algorithm.

In the second scenario, a desired interaction occurs that, due to higher-level business policies, ought not be allowed. Given the current difficulty in setting up fine-grained, dynamic VO's and the costly nature of security related run-time failures, many installations are overly permissive. For example, Service B's business logic prevents it from interacting with Service A due to its partnership agreement with another business, but since the WPEs are using delegated rights, Service B is unable to recognize its interaction with Service A.

Our framework realizes the evaluation of bilateral and indirect trust relationships, and requires every WPE to express its security requirements through an upper layer of collaboration policies. These policies allow a service owner to explicitly communicate its trust requirements for bilateral and indirect interactions occurring in a workflow. By harnessing such requirements, collaboration policies express the conditions under which a WPE is willing to participate in a workflow and the allowed actions during the workflow execution.

2.1 Collaboration Policies

Collaboration policies define a set of requirements and rules that must be met for participating in a workflow. These policies define trust requirements of neighboring services for collaboration, the propagation of access rights throughout the execution path (delegation policies), and the trust requirements from a workflow requestor. Each collaboration policy is comprised of four attributes: (1) authorization rules for neighboring WPEs, (2) the radius of the partial workflow path that a WPE needs to evaluate in terms of security, (3) the delegation of credentials throughout a workflow, and (4) the trust relationship with the workflow requestor. Separate collaboration policies for upstream and downstream neighbors may be defined.

Authorization rules for neighbor WPEs: The level of interaction between WPEs is determined by the position of services in the execution path. The immediately neighboring services are expected to exchange documents, executables and other arguments directly, whereas non-immediate neighbors interact with each other through intermediate services. A service owner may have different security requirements based on the level of interaction required with each WPE. A collaboration policy must express how all these different requirements combined together and an overall authorization decision is made for a given path.

For example, an immediate neighbor (service B in Figure 2) may be required to have the proper authorization rights to invoke service C, whereas, a non-immediate neighbor (service A in Figure 2) may be applied to a weaker set of authorization rules that covers only a partial set of required authorization attributes such as country location, company information or a reputation value.

At the policy writing time, each service owner must decide (1) the size of the partial workflow path that needs to be examined (i.e. the distance between the service owner and the WPEs that security checks must be applied), and (2) the set of access rules that should be applied to the entities within the partial path. Note that the size of the partial workflow path that should be examined is independent of a specific

workflow instance — rather it is dependent on the sensitivity of the resource being offered and the local regulations governing this resource. For example, a service, which is only interested in checking its immediate upstream and downstream neighbors, has a partial workflow path of two nodes. No matter which specific workflow instance this service participates in, it will only examine two WPEs that happen to be at the specified distances from the service owner.

In order to ease writing such policies, we define two security functions, S and A, where S denotes strong authorization requirements and A denotes lighter attribute-based requirements. Both S and A define the access control rules based on the interaction level between entities. We use the distance between WPEs as an indicator of interaction level, which is defined as follows:

Distance (WPE1, WPE2): x, where there are x number of hops between WPE1 and WPE2 for each linear path between them.

S(WPE, obj), where obj denotes the object access is being requested, and WPE denotes the workflow entity requesting access. One way to express a neighbor WPE is by using **direction:distance** pairs. The direction:distance pairs identify the WPE for which this policy rule is applied. Direction could be either upstream (up) or downstream (down). S shows that the requesting WPE must be applied to the same security policies that are applied when the access is requested by this WPE individually (without being part of a workflow).

A(WPE, attr, obj), where obj and WPE are used in the same manner as used in the S function. Attr denotes a list of attribute-based requirements. This list of required attributes for authorization does not constitute the full set of attributes that are required when the access is requested individually. Rather, these attributes are geared towards internal business rules and industrial restrictions such as checking the geographical origin of an organization or the rivalry information.

For example, the below sample policy shows that as long as one of the two upstream neighbors are strongly authorized and they are not a certain rival company, the access for this upstream path is allowed, whereas for the downstream path, only the country information of the immediate neighbor is needed.

CP: {((S(up:1, obj1) ∧ A(up:2, Organization Name, obj1)) ∨ (S(up:2, obj1) ∧
A(up:1, Organization Name, obj1))) ∧
A(down:1, country information, obj2)}

The delegation of credentials throughout the workflow path: During the execution of a workflow, the delegation of credentials from WPEs or workflow requestor may be necessary. Each WPE, before joining a workflow, must express its delegation rules in its collaboration policy. Such rules define (1) if delegated credentials are accepted for authorization, (2) whether the downstream delegation of WPEs' credentials are allowed in case workflow execution requires such an action, and if so the trust requirements from the delegated parties, (3) whether the WPE accepts delegated credentials (delegated from an upstream neighbor) in order to invoke another service or to propagate them to another WPE.

The above rules are necessary for a number of reasons. (1) A WPE or a workflow requestor may be willing to delegate their credentials to a second party in order to complete a job without carefully contemplating the security consequences. The WPE

that receives a request with delegated credentials must be able to distinguish such credentials and apply appropriate security policies. Current grid execution environments (Globus Toolkit) do not allow discrimination of delegated credentials from the original ones, due to the usage of proxy credentials that provide single sign-on. Our framework, by allowing a WPE to evaluate its direct and indirect neighbors' credentials, enables a WPE to determine if delegated credentials are used in an execution chain. Armed with such information, a WPE may refuse to allow access to a delegation chain. (2) During workflow execution, WPEs shares their resources on behalf of a workflow requestor. These shared resources may involve a WPE's credentials. Before joining a workflow, each WPE should define their rules for allowing downstream delegation. Once such information is made available to the workflow engine at the planning stage, a more efficient but non-secure path may later be discovered to be suitable, thus increasing the performance of the workflow. (3) A WPE may need to accept delegated credentials from an upstream neighbor either to propagate them along the execution path or to invoke another downstream service. Even though, acceptance of delegated credentials seems relatively safer than delegating one's own credentials, a WPE must carefully analyze the consequences of accepting such credentials.

In order to ease writing collaboration policies, we define three delegation functions:

Du(distance, obj, conditions), where distance shows how many downstream hops the delegated credentials has traveled, obj denotes the object that is being requested, and conditions defines the rules that must be satisfied for accepting access with delegated credentials. Conditions define the authorization requirements in terms of combination of S and A functions. Du communicates under which conditions access to an object with delegated credentials is accepted.

Dd(credential, distance, conditions), where distance shows how many hops of re-delegation is allowed, conditions hold the same meaning as in Du. Credential denotes the credential being delegated downstream. Dd communicates whether this WPE is willing to allow downstream delegation of its credentials.

Dt(credential, distance, transient/final, conditions), where all attributes hold the same meaning as above except transient/final. Transient/final denotes whether the credentials should be passed onto another WPE for the final service invocation or the receiving WPE will perform the downstream service invocation with the delegated credentials. Dt communicates the conditions under which a delegated credential is accepted so that these credentials either be propagated to another WPE or used for a downstream service invocation.

The radius of partial workflow graph for security examination: Based on its collaboration policy, a WPE may require examining a partial graph of the workflow. The radius for such a sub-graph is calculated from the collaboration policy and communicated to the workflow engine. Depending on the policy, the number of neighbors that should be evaluated in the downstream and upstream directions may be different, thus requiring two radii, one calculated for each direction.

The trust relationship with the workflow requestor: In addition to defining the required access rules for its neighboring WPEs', a WPE may also need to define access rules for the workflow requestor. The functions defined previously, S and A,

can be used to state such rules with the special keyword **requestor** instead of direction:distance pairs. Note that the distance between a WPE and the workflow requestor is dependent on the specific workflow path instance; however, the collaboration policies must be expressed independent of any specific workflow instances. Therefore, at the policy writing time, the location of the workflow requestor is unknown, and must be signaled by the keyword **requestor** in S and A.

2.2 Collaboration Policy Semantics

A flexible yet expressive collaboration language is needed to express the collaboration policies. XACML allows a resource owner to express access control policies for a resource. Due to its richness and flexibility, we decided to express our collaboration policies in XACML. Collaboration policies, combining existing lower level access control policies with additional constraints, require an extensible language that allows defining new attributes and subject groups easily.

We have also examined several other languages. BPEL4WS is widely accepted as the de-facto flow language for workflow management systems. However, security is not a major concern in the BPEL4WS – rather, BPEL4WS provides a facility to exchange messages between collaborating organizations, and assumes that underlying standards such as WS-SecureConversation and WS-Security would provide the necessary security extension. WS-SecureConversation and WS-Security standards provide "message-level" security on top of the SSL layer by specifying needed security tokens in a SOAP message (Kerberos tickets, X509 credentials, or SAML assertions). However, neither expresses the "business-level" interactions and security/trust requirements among the collaborating partners.

WS-Trust and its allied standards support security token interoperability and traditional bi-partite trust relationship. It does not explicitly model business level trust relationships between entities when they are acting on behalf of a third party.

3 Secure Workflow Management Framework

Typically, workflow planning engines retrieve a list of suitable resources from the discovery service (MDS) [7] [8], and map each task to a resource. During the mapping process, the planning engines evaluate several constraints such as resource availability, current load, wait time, data locations and the cost associated with each resource [18, 19, 20]. For example, The Pegasus [18] planning engine selects resources that are closer to the required data locations. If no resource is particularly close, then a random decision is made. The Nimrod/G [19] planning engine evaluates the computational costs and selects resources within workflow requestor's price range. The GridFlow [20] planning engine focuses on time management. The estimated execution times for each candidate resource is evaluated, and an optimal workflow path is built that can satisfy the time restrictions. Security and trust is of little or no consideration during the mapping process. Rather, classical systems defer authorization and trust evaluation to the execution stage.

The workflow planning logic must evaluate the existing trust relationships, and dynamically build alternative execution paths when a functionally desirable execution

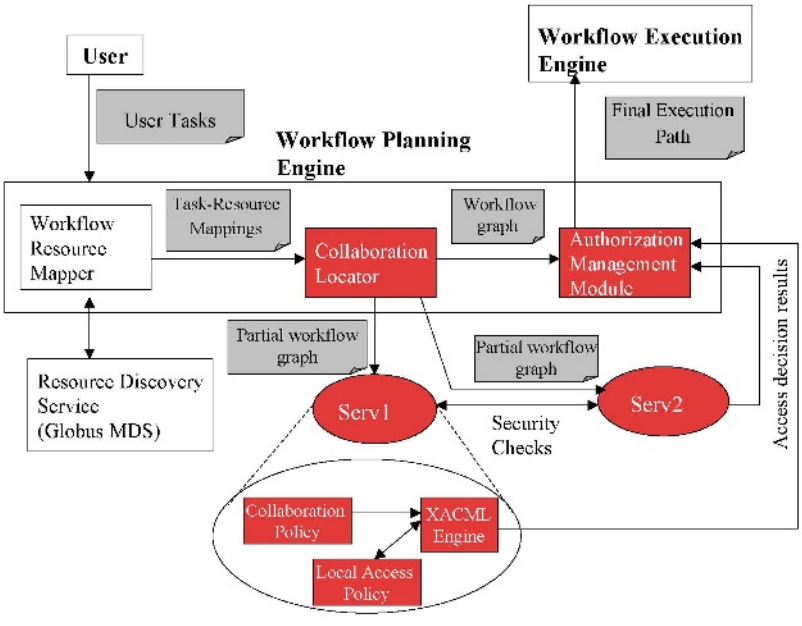

Fig. 3. Secure Workflow Framework, operating with the Globus Toolkit. The dark components indicate the proposed additions to the frameworks. The light gray boxes are the documents exchanged among the components.

path discovered to be infeasible from the security standpoint. To accomplish this, we extend the workflow planning stage with a Collaboration Locator Module (CLM) and the Authorization Management Module (AMM) (see Figure 3).

3.1 Collaboration Locator Module

The Collaboration Locator Module (CLM) is responsible for locating the WPEs that collaborate with each other. Each WPE communicates the required radiuses of partial workflow graphs to CLM. Based on the radius information, the CLM determines the partial graphs that need to be sent to each WPE, and also determines the security checks among the WPEs. Based on these checks, each WPE initiates the security evaluations. (Note that WPE collaboration policies need not be communicated to the CLM.)

The sample scenario shown in Figure 2 is accompanied with the following collaboration policies for each of the WPEs.

CPd(A):{S(down:1, document A); Dd(X.509, 2, Condition1: {S(down:1, credential) for one hop delegation}, Condition2: {(S(down:1,credential) ∧ S(down:2, credential)) for two hops delegation}; radius:2; S(requestor, service A)},

CPu(B):{S(up:1, service B); Dt(X.509, up:1, final, none); radius:1; S(requestor, service B)}

CPd(B):{S(down:1, document B); none; radius: 1}

CPu(C): {((S(up:1, service C)) ∨ (S(up:2, service C) ∧ A(up:1, Organization Name (X.509 DN), service C) ∧ Du(2, service C, none))) ; none; radius: 2; S(requestor, service C)}

A's collaboration policy indicates that the strong authorization of an immediate downstream neighbor (expressed by S(down:1, document A)) and the workflow requestor are required (expressed by S(requestor, service A)). Downstream delegation of service A's credentials are allowed under two conditions (expressed by Dd(X.509, 2, Condition1, Condition2)): (1) either the immediate downstream neighbor is strongly authorized and the credentials are delegated for one hop distance (expressed by Condition1: {S(down:1, credential) for one hop delegation}), or (2) two subsequent downstream neighbors are strongly authorized, and the delegation of credentials for distance of two is allowed (Condition2: {(S(down:1,credential) ∧ S(down:2, credential)) for two hops delegation}). Finally, the radius of upstream workflow graph that needs to be evaluated is two (expressed by radius:2). Note that the radius is two due to the conditions for the delegation rule.

B's collaboration policy consists of two parts: upstream and downstream policies. Both policies can be expressed together, however separation eases the job of policy writer. The upstream policy requires the immediate upstream neighbor and the workflow requestor to be strongly authorized ((S(up:1, service B) and S(requestor, service B)). In case of an upstream delegation, B accepts the delegated credentials if it would be the final entity for downstream service invocation (Dt(X.509, up:1, final, none)). B's downstream policy does not allow delegation of B's credentials and indicates that it should strongly authorize its immediate downstream neighbor (S(down:1, document B)).

C's collaboration policy indicates that an upstream path is only authorized if (1) the immediate neighbor is strongly authorized or (2) the non-immediate neighbor at distance 2 must be strongly authorized and should allow downstream delegation of its credentials, and the immediate neighbor must not be a rival company ((S(up:2, service C) ∧ A(up:1, Organization Name (X.509 DN), service C) ∧ Du(2, service C, none))). C requires strong authorization of the workflow requestor (S(requestor, service C)), and does not allow downstream delegation.

Having only received the radius information from the above collaboration policies, CLM decides that following security checks among the WPEs are necessary.

A → C for service C; B → C for service C; C → B for document B; B → A for document A; A → B for service B, and C → A for service A. CLM also prepares partial graphs of the workflow and sends those to each WPE. Finally, CLM passes the list of required security checks to AMM, which orchestrates the communication among WPE and replaces insufficient pairs with matching ones.

3.2 Authorization Management Module

The AMM performs two functionalities: it orchestrates the trust evaluations between WPEs, and based on the access decisions, it finds alternative execution paths.

A key point of AMM is the decentralized authorization framework. Each WPE evaluates the access requests from its candidate neighbors. Based on the results of security checks, each WPE determines whether or not to participate in the workflow and the conditions under which the participation may take place. WPEs send their decisions and

conditions on the partial workflow path. AMM examines the WPEs' decisions and checks if the conditions are met. Note that AMM only serves to orchestrate authorization, rather than making the final access control decisions on behalf of resources.

In order to ease the communication between AMM and WPEs, we provide a simple message format.

(WPE, path): {decision, conditions: (required upstream delegation distance, (allowed upstream delegation distance, transient/final), allowed downstream delegation distance)}. Decision denotes if a given partial workflow map is authorized by this WPE. The list of conditions shows if the approval of this partial path is dependent on delegation of credentials. The list, respectively, shows how many hops of upstream delegation is required for authorization (corresponds to Du), how many hops of upstream delegation is accepted along with a transient or final flag (corresponds to Dt), and the number of hops allowed for downstream delegation of this WPE's credentials (corresponds to Dd).

Re-visiting the sample scenario from Figure 2, CLM triggers services A, B and C to start trust evaluations among each other. Assume A is strongly authorized for service C, whereas B is not. However, B's company information indicates that it is not part of a rival organization. Also, B and C are both strongly authorized to A, and finally, A and C are strongly authorized to B. The following messages would be created:

(C, 1): {approved, conditions: (1, 0, 0)}. This message shows that C is willing to authorize this path, but it requires upstream delegation of distance one: neighbor (B) must use the delegated credentials from its immediate upstream neighbor (A) (indicated by 1 in the message). C neither authorizes any downstream delegation nor accepts transient delegation of upstream credentials.

(A, 1): {approved, conditions: (0, 0, 2)}. This message shows that A is only willing to allow downstream delegation of its credentials up to distance of 2.

Note that the authorization of C to A seems like an unnecessary operation for this specific partial graph. However, consider the following scenario: there is a subsequent downstream service, say D, that accepts upstream delegated credentials from distance of two. In the case that C does not have the required credentials for access, the workflow engine would immediately know that this path is still feasible by looking at the message from A indicating that A may delegate its credentials to C (of course given that A is authorized to D). Also note that requirement for authorization of A to D is indicated in D's upstream collaboration policy which would state that credentials delegated up to distance of two are accepted.

(B, 1): {approved, conditions: (0, (1,final), 0)}, which shows that B accepts the upstream delegation of credentials from A to invoke service C (indicated by (1, final)); however it neither allows downstream delegation, nor requires upstream delegation.

Having looked at all the messages, AMM starts checking the conditions on the path. C's conditional approval (shown by (1,0,0)) depends on A's willingness to delegate (which holds true as shown in (0,0,2)) and B's willingness to accept the delegated credentials (which holds true as shown by (0, (1, final), 0)). Therefore, AMM concludes that the required conditions for this path have been achieved and a feasible path is found.

The second important function of AMM is to suggest alternative execution paths in case a chosen path turns out to be infeasible from security standpoint. AMM must select which WPE to replace in a given infeasible path. There could be several WPEs that do not mutually trust each other on a complicated workflow path. Therefore, the algorithms for selecting which WPE to replace become crucial. There are several important issues to consider while replacing an unfit WPE and locating a new one for that task. (1) The number of neighbors of the WPE. A high number of neighbors indicates that replacement is going to cause many re-evaluations of trust relationships. Therefore, it is safer to replace a WPE that has few neighbors. (2) The collaboration policy of a WPE. Replacing a WPE that has a relatively light collaboration policy (i.e. accepts and allows delegation, small radius, does not require strong authorization) with a WPE that has more restrictive requirements may cause more trust re-evaluations. (3) The number of neighbors that do not authorize this specific WPE. In some cases, a WPE may be found to be unfit by only one other WPE, while the rest of the WPEs authorize it. In most cases, it is safer to replace a WPE that is found unfit by the majority. However, a robust selection algorithm must always re-visit issues (1) and (2).

4 Related Work

There are several workflow security frameworks that target the needs of large organizations [9][10][11]. They focus on synchronizing the access to required privileges to execute a task with the progression of a workflow. These approaches require a central workflow authority to have the access rights associated with each workflow subject and transfer these rights to the subjects based on the workflow progress. In a heterogeneous environment, objects and subjects may be web services belonging to different organizations. Therefore, the ownership of such access rights by the central authority becomes impossible, thus making them insufficient for heterogeneous dynamic environments.

Other approaches by Bertino [12], Tan [13], and Hung [14] use authorization constraints to extend RBAC models. A security policy designed by a workflow requestor may express constraints such that the resulting execution path must conform. These efforts approach the authorization problem from the workflow requestor's perspective, and allow her/him to express trust requirements sought from a candidate resource to perform a job. However, this approach does not touch upon the other side of the authorization problem: the authorization of workflow requestor to the candidate resources and the trust relationships among the resources.

Kang [15], Koshutanski [16], and WAS framework [17] recognize the inter-organizational, heterogeneous nature of new-generation workflows. Kang requires each participating organization to map its entire role structure to the role domain of the workflow. The WAS framework focuses on deciding the required rights in order to run a task through different domains, and on providing restricted delegation by using source-code analysis of tasks. Both approaches require a central authority, with access to the internal policies of each WPE, that can assign each task with a pool of privileges and roles. The main drawback of this model is that it does not allow building dynamic workflows, where the workflow engine assumes no knowledge about the internal security policies of a participating organization.

Koshutanski's framework focuses on providing authorization mechanisms between a workflow requestor and WPEs. Instead of revealing access policies, each organization sends a mobile process to the end user. Upon executing the mobile processes on the user side, an access control decision is made. The reliance on mobile processes introduces other security issues, such as how the end user can verify the code, and how the code should identify the credentials required to make the authorization decision. Moreover, this framework does not consider the trust relationships among WPEs.

5 Conclusion

Construction of heterogeneous and dynamic workflows requires collaboration and interaction among disparate services on demand; necessitating expression and evaluation of trust relationships at the planning stage. Unfortunately, no current workflow tool considers such relationships during the planning stage: rather, they assume homogeneous execution communities where each party shares common long-term goals and implicit trust. Our framework differentiates itself by defining a security architecture for heterogeneous and short-lived collaboration environments. It introduces collaboration policies that express the conditions to enter a workflow and allowed actions during execution, by harnessing existing lower level access control policies and additional workflow-oriented constraints. As a result, workflow entities are assured to abide by their internal security policies, and infeasible execution paths are replaced with suitable service pairs that are willing to collaborate at the planning stage. Our framework, by providing the tools to support complex trust relationships, increases the willingness of services for collaboration where creating formal homogeneous communities are expensive.

References

1. Foster I., Kesselman C., Nick J., Tuecke S.: Open Grid Service Infrastructure WG, Global Grid Forum (2002)
2. Foster I., Kesselman C.: Globus: A Metacomputing Infrastructure Toolkit. Intl J. Supercomputer Applications (1997) 11(2):115-128
3. Foster I., Kesselman C., Tuecke S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Intl. J. Supercomputer Applications (2001) 15(3)
4. Sandhu R.: Role-Based Access Control Models. IEEE Computer (1996) 29(2):34-47
5. Thomas R.K., Sandhu R..: Towards a Task-based Paradigm for Flexible and Adaptable Access Control in Distributed Applications. ACM SIGSAC New Security Paradigms Workshop (1992-93) 138-142
6. Blaze M., Feigenbaum J., Ioannadis J., Keromytis A. D.: The role of trust management in distributed systems security. In Secure Internet Programming: the Security Issues for Mobile and Distributed Objects. Springer-Verlag (1999) 185-210
7. Raman R., Livny M., Solomon M.: Matchmaking: Distributed Resource Management for High Throughput Computing. Seventh IEEE Intl. Symp. on High-Performance Distributed. Computing (HPDC) (1998)

8. Czajkowski K., et al.: Grid Information Services for Distributed Resource Sharing. 10th IEEE Intl. Symp. on High-Performance Distributed Computing (HPDC-10) (2001)
9. Atluri V., Huang W-K.: An Authorization Model for Workflows. Fifth European Symp. on Research in Computer Security (1996) 44-64.
10. Knorr K.: Dynamic access control through Petri net workflows. 16th Conf. on Computer Security Applications (ACSAC'00) (2000) 159-167
11. Huang W-K., Atluri V.: SecureFlow: A Secure Web-enabled Workflow Management System. 4th ACM Workshop on Role-based Access Control (1999)
12. Bertino E., Ferrari E., Atluri V.: The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. ACM Trans. on Information and System Security (1999) 2(1):65-104
13. Tan K., Crampton J., Gunter C. A.: The Consistency of Task-Based Authorization Constraints in Workflow Systems. 17th IEEE Computer Security Foundations Workshop (CSFW'04) (2004) 155-169
14. Hung P.C.K., Karlapalem K.: A secure Workflow Model. Australasian Information Security Workshop Conference (2003) 33-41
15. Kang M.H., Park J. S., Froscher J. N.: Access-Control Mechanisms for Inter Organizational Workflow. Sixth ACM Symp. on Access Control Models and Technologies (2001) 66-74
16. Koshutanski H., Massacci V.: An Access Control Framework for Business Processes for Web Services. ACM Workshop on XML Security (2003) 15-24
17. Kim S-H., Kim J., Hong S-J., Kim S.: Workflow-based Authorization Service in Grid. Fourth Intl. Workshop on Grid Computing (GRID'03) (2003) 94-100
18. Deelman E., Blythe J., Gil Y., Kesselman C., Mehta G., Patil S., Su M-H., Vahi K., Livny M.: Pegasus: Mapping Scientific Workflow onto the Grid. Across Grids Conference (2004) 11-20
19. Buyya R., Abramson D., Giddy J.: Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. Fourth Intl. Conference On High Performance Computing in Asia-Pacific Region (HPC ASIA'00) (2000) (1): 283-289
20. Cao J., Jarvis S. A., Saini S., Nudd G. R.: GridFlow: Workflow Management for Grid Computing. Third IEEE/ACM Intl. Symposium on Cluster Computing and the Grid (CCGRID'03) (2003) 198-205
21. Standards for Privacy of Individually Identifiable Health Information (HPR). 45 CFR 164.C. Federal Register (2003) 68(34):8334 – 8381