# Security: Protect Your Service-Oriented Architecture Network

## Introduction

This paper covers the deployment, management, and governance architecture of entitlement management, including its influencing factors. Although service-oriented architecture (SOA) is the context of discussion, all of the discussion and insights apply equally well to application environments that are not yet service-oriented. This paper does not cover messaging and interface standards such as Security Assertion Markup Language (SAML), Extensible Access Control Markup Language (XACML), Web Services Security (WS-Security), Web Services Policy (WS-Policy), etc., or their usage in a SOA.

As the name suggests, a service-oriented architecture is one in which you package application functions as autonomous services that adhere to industry-standard interfaces (Web Services Description Language [WSDL] or Simple Object Access Protocol [SOAP], for example), and then deploy the services in an IT architecture that allows for their most effective use. You can rapidly reuse the component services and combine them to create new business offerings, and you can upgrade them individually for increased business agility. To achieve the promise of an SOA, however, you must provide critical non-business logic-related functions – particularly security – as a service. Thus you must externalize and manage security independently from the business logic-related services. This document addresses application security, and specifically fine-grained application access control or entitlements.

## The Need to Externalize Security

Reasons for externalizing and independently managing security abound; they include:

- Preserving service reuse: The security context within which a component service is executed is a function of the composite service within which it is invoked. It cannot be determined during the development of the component service. If authorization logic (who is allowed to use the service) is codified within the component service, then it will need to be modified for each use in different environments or within different composite services. This approach defeats one of the benefits of a SOA: components in an SOA should be completely reusable regardless of the context within which they are invoked.
- Avoiding overhead of inter-company coordination: The owners, administrators, and specialists for access policy are different from those who develop the business logic of the component service. They are often in different organizational domains. Requiring codification of the access-policy management at development time – and within the same package as the business logic – requires coordination that is unnecessary and inefficient.
- Improving visibility and auditability: Access policies need to be audited and checked outside of the service for compliance purposes. Access policy auditing is required for the component service as well as for the composite service or business process within which the component service is invoked. Auditing is important because of corporate governance and compliance needs. It is particularly important, however, in a SOA environment because in pre-SOA environments application functions could be used in only a few very limited,

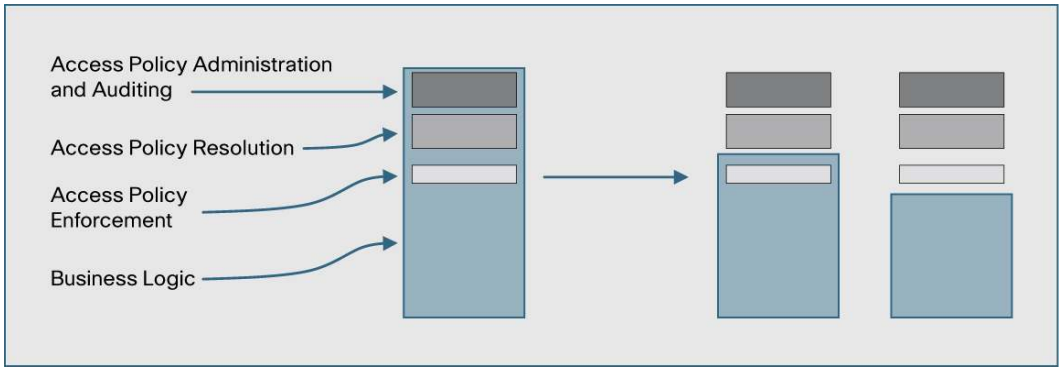
very controlled ways, but in a SOA world services may be invoked in very diverse and unanticipated ways. Auditing is a critical tool to anticipate problems before they occur and locate the root cause of problems when they do occur.

### Access Policy Management as a Service

In a well-designed SOA access policy, management itself is an important service, referred to as an infrastructure service. The component services and composite services, on the other hand, encode the business logic; these services are referred to as business services.

So what does externalizing access-policy management from the component service really mean? Figure 1 depicts the migration from tightly coupled monolithic applications to loosely coupled services. Pre-SOA application functions where security is managed within the business logic are depicted on the left. The architecture moves to a SOA-compatible service version of the same business logic with access-policy administration, resolution, and auditing externalized from the service and manifested as SOA-compatible infrastructure services themselves. In some cases the enforcement of the access policy is also externalized from the service. In general this scenario is not possible and the service can get the access-policy decision from the external security service and can enforce the decision itself.

**Figure 1.** Migration from a tightly coupled paradigm to a loosely coupled SOA




It is less critical to manifest access-policy enforcement as a separate infrastructure service because it is often tied very closely to the business logic and changes with the business logic. Thus access-policy enforcement is unlike access-policy administration, auditing, or resolution, which change at a different rate and at different times, and are owned by different people than the developers of the business logic.

For example, consider the following scenario:

- Component service: Order management
- Access policy:
  - Only people with the role of “broker” can enter an order
  - Only the owner or a manager of the owner of the order can update an order
  - Only the owner, a manager of the owner, or a subject of the owner with the role “reconcile” can read an order

In this case the developers of the order-management service can focus purely on implementing the most efficient order-management functions. The policy that specifies who can access the order management service and who can perform the functions it exposes needs to be managed externally and at a later time, potentially by multiple, independent people (more on this topic later). The access-policy resolution needs to access the appropriate contextual information, for example:

- Accessing a central Lightweight Directory Access Protocol (LDAP) directory in which the user roles may be stored
- Accessing another LDAP directory in which the user-manager relationship may be stored
- Accessing a separate local or remote database in which the order-owner relationship may be stored

The policy resolution determines if the given request should be permitted or denied, and the policy enforcement enforces that decision on the request.

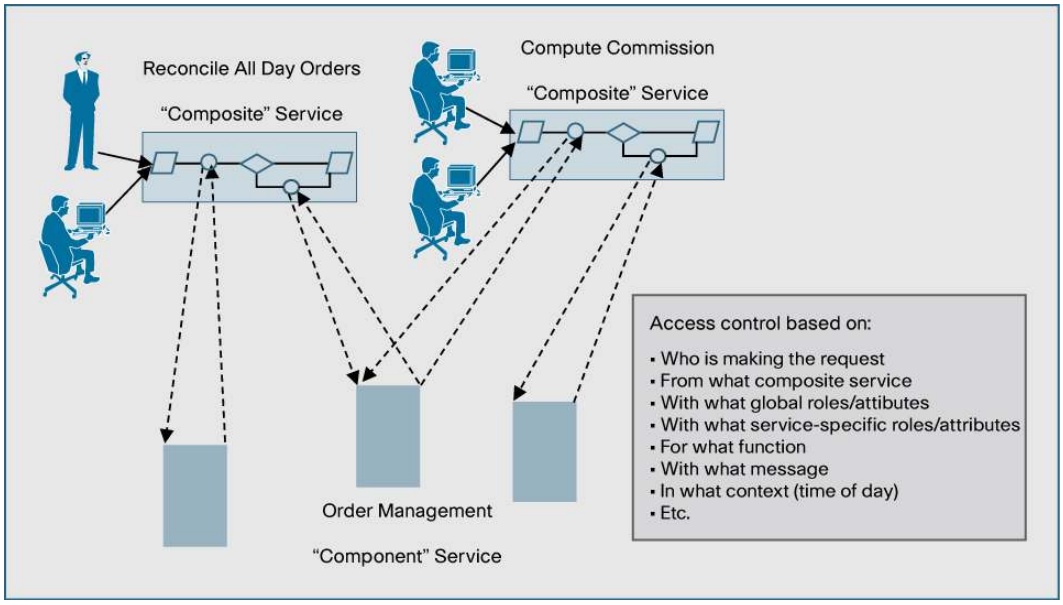
The benefit of separating the access-policy administration, resolution, and audit from the component service is that you can change the access policy to comply with changing security or compliance requirements without requiring any change or recoding of the component service. For example:

- A Patriot Act rule might require that an order from a user with a certain attribute (for example, users who belong to a set of identified organizations) and with an order value over a certain amount (e.g. Percentage of total assets held overseas exceeds 50%) should not be permitted
- A Sarbanes-Oxley rule might require segregation of duties between a broker and an administrator
- A business situation such as a merger and acquisition may require modification of the access policy to permit access to users who have the role of broker in one system and account manager in another system

You can effect these changes, which are independent of the business logic of the order-management service, by simply configuring a new or modified access policy; no changes in the order-management service itself are required.

Conversely, a new, more efficient order-management service does not require a recertification of the existing access policies. As the SOA deployment in an organization matures, the component services are invoked within composite services that determine part of the security context within which the invocation of the component service needs to be checked. In Figure 2 the order management component service may be invoked within a “reconcile-all-day-orders” composite service or within a new “compute-commission” composite service.

**Figure 2.** Order Management Component Service being invoked as part of two other business services




The access policy can account for the context of the composite service from which the component service is being invoked, who is initiating the composite service, etc. You can now use the order-management component service in ways unanticipated at the time the service was developed, and you can administer, resolve, and audit the appropriate access policy without loss of security and compliance and without requiring any rework in the component service.

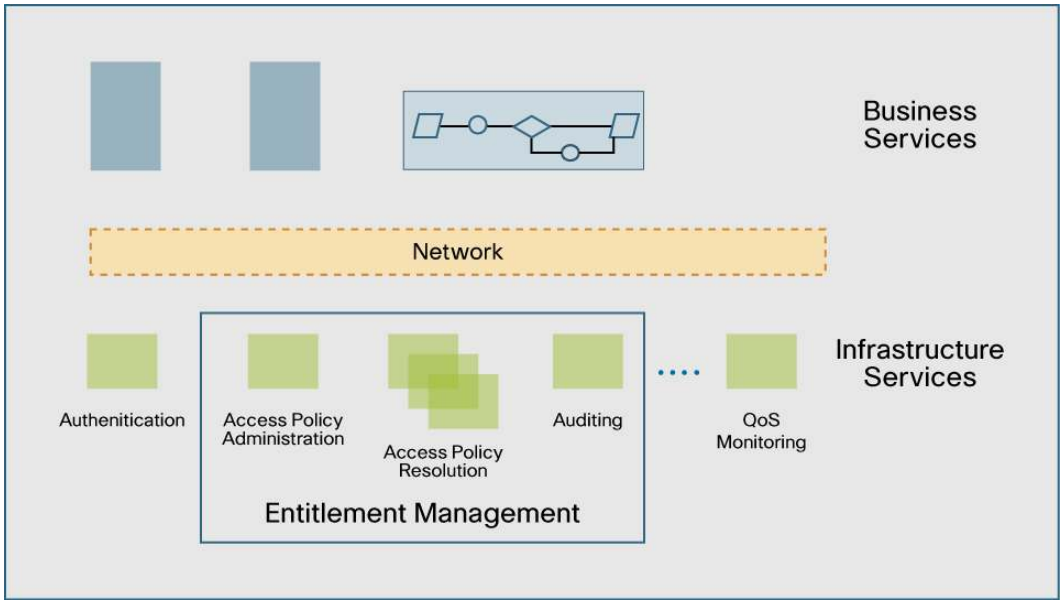
### So Where Do You Begin?

Begin by making sure that access-policy management – administration, resolution, and auditing – are not embedded inside a component or composite service. If the application logic is developed in a standard container model – for example, Java 2 Enterprise Edition (J2EE) or .Net – then try to ensure that the granularity of the discrete functions that need to be protected are exposed to the container interfaces. Then you can perform the access-policy enforcement by integrating a standards-based interceptor into the application infrastructure stack without changing the application code. This interceptor-based enforcer permits or denies access to a service resource by permitting or denying the corresponding container interface from invocation.

Similarly, if the application logic has a standard invocation model, such as SOAP, and the granularity of the resources being protected are at the granularity of the invocation interface, you can enforce the access policy by a standards-based interceptor within the SOAP stack. You can deploy the interceptors as code that is co-resident with the business service or as a separate infrastructure service that is invoked from within the application code. In general the application code invokes the policy-resolution service over standard interfaces (for example, XACML over SOAP) to get the access-policy decision and enforces the decision within the application code.

Figure 3 depicts a well-designed SOA that is loosely coupled with access-policy administration, resolution, and auditing as standards-compliant infrastructure services.

**Figure 3.** A well-designed SOA consuming external infrastructure services for policy administration, resolution and auditing




Unlike most other infrastructure services, the access-policy resolution for fine-grained accesses or entitlements has constraints that dictate the instance of the resolution service that you should use. Because the access policy is applied on every access and the policy resolution may require message context and other attribute information that is local to the business service, you will probably have to invoke a relatively local instance of the resolution service. It may be impractical from performance, scalability, and availability perspectives to use a centralized resolution service. Therefore it is important that a practical and effective security infrastructure for a SOA permit distributed access-policy resolution through multiple distributed instances of the resolution service.

### Progressing Toward a More Secure SOA

Now consider how to develop the end-state deployment architecture: What are some of the critical operations and policy administration concerns, and how do you deal with them?

Because there are many different owners of access policy for a given resource (for example, component service administrator, composite service administrator, enterprise security and Information Systems Security [InfoSec] teams, and enterprise and line-of-business compliance teams), it is imperative that the policy-administration service have a rich and effective delegation capability. It is critically important that the administration service not require all of these owners of access policies to coordinate their efforts or to administer a single unified policy at the same time.

Some of the conditions of the policy may need to be defined at different times. For example, the administrator of the composite service may want to suggest input regarding access policy of the component service at a much later time than the administrator of the component service wants to suggest input regarding the access policy of the component service. The compliance team also may want to change the compliance aspects of the access policy; for example, an order initiator cannot be the order approver, autonomously from the administration of the other aspects of the policy. In fact, compliance teams need to be able to change the policy to respond to a change in regulations without having to coordinate with the other administrators of access policy for a resource. In many instances it is important from a checks-and-balances perspective that the administrators be different and independent.

Depending on their role, when people log into the administration service, they should be able to administer only those aspects of access policy that they are permitted to administer. Thus the administration service itself needs to be entitled, and needs to have rich delegation capability.

If you have many autonomous administrators of policy and coordination among them is not required, obviously the policies that they define could conflict with each other. For example, the administrator of the reconcile-day-orders composite service may specify a policy allowing access to the order-management component service while at the same time the administrator of the order-management component service specifies a policy that denies that administrator access. Denial is perhaps because the user on whose behalf the composite service is being initiated is also the approver of an order, thus violating a segregation-of-duty policy for the order-management component service. Therefore, the administration infrastructure service should anticipate and handle access-policy conflicts. These conflicts should be resolved at the time of access using the most up-to-date, dynamic information, and the resulting policy decision should then be enforced on the resource access.

### The Need to Go Beyond User Roles in Setting Access Policies

A related and very important administration concern relates to user roles. Role Based Access Control (RBAC) or the use of roles in access policies is often considered a useful way to manage access to resources. The benefits of RBAC are well-documented. The main advantage of RBAC is ease of management – users typically outnumber roles significantly. Because a user can be a member of multiple projects, each project can have its own access requirements, and because user-to-project and project-to-access mappings can change, roles are a powerful abstraction to manage and enable this flexibility.

Important as RBAC may be, when deploying a security infrastructure service in a SOA (the security infrastructure service more accurately is a set of services – administration, resolution, and auditing – as mentioned earlier), role assignment can also cause impairment. Whether deploying a SOA or not, many organizations try first to reconcile all roles across the enterprise in a top-down fashion. This exercise is long, painful, and largely futile. Although there are a few enterprisewide roles, most roles are resource-specific. For example, a vice president in the corporate LDAP directory may be denied access to the development version of a business service.

Each resource has pertinent roles and appropriate levels of access for users. You can use these resource-specific roles in conjunction with global roles to form the basis of an effective RBAC solution. For example, an access policy may state that access to a business service is permitted to users who have a “controller” role in the enterprise LDAP directory or an “administrator” role for the service being protected. The access policies should allow specification of global and service-specific user roles. They should also allow for user- and service-specific attributes, for example, employment status, clearance level, geography, and organizational membership. Trying to incorporate these attributes into roles quickly leads to an explosion in the number of roles. Moreover, these attributes are often dynamically computed. Conversely, you can treat a role as simply another user attribute. It is, therefore, important for the security-administration service to allow use of generalized resource, user, environment (such as time of day), and invocation (such as the value of the transaction being requested) attributes in the specification and resolution of the access policies.

When you allow the use of resource-specific attributes, it is very important that you allow the resource owners to specify, assign, and manage the resource-specific attributes. Such distributed ownership and management of resource-specific attributes is consistent with an unstated principle that underlines SOA – namely, local control with global coordination. It is necessary for the smooth functioning of a practical SOA, and it expedites getting to a state of meaningful and effective RBAC. Now instead of trying to reconcile all roles across the enterprise in a top-down fashion and trying to keep them all consistent when user-to-role or role-to-access mappings change, most role assignments are delegated to the resource owners, who can define what they need for their resource and administer and manage appropriate changes at an appropriate pace.

## Conclusion

A service-oriented architecture is more than simply packaging application functions into business services that adhere to industry-standard interfaces. It requires the externalization of non-business logic-related functions from the application that need to be provided and used as a set of standards-compliant infrastructure services. Security is a critical infrastructure service that is essential to achieving the ready-to-use goals of SOA. If designed well it can facilitate the smooth operation and evolution of a SOA environment, and more importantly it can smooth the path to realize a SOA environment. If not, it can be the undoing of an otherwise sound SOA plan.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV
Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**



Printed in USA C11-453532-00 02/08