Developers and Identity Services

- Modernizing Access Control with Authorization Service

An Oracle White Paper
November 2008

Developers and Identity Services

- Modernizing Access Control with Authorization Service

Developers and Identity Services

- *Modernizing Access Control with Authorization Service*

EXECUTIVE OVERVIEW

Service-Oriented Security (SOS) aligns with the overall Application-Centric approach of the entire Oracle Fusion Middleware platform - with the goal of providing a comprehensive, standards-based, developer-friendly platform. By leveraging and sharing many of the common Identity Services, SOS allows developers to spend the effort on where it counts the most – the application logic itself. However, to truly move away from the traditional silo-based approach in dealing with identity management, application developers building identity-enabled applications must look to the benefits of these services and understand when and how to use these services in their application design.

"However, to truly move away from the traditional silo-based approach in dealing with identity management, application developers building identity-enabled applications must look to the benefits of [Identity Services] and understand when and how to use these services in their application design."

This is the second in a series of "Developers and Identity Services" whitepapers, each intended to focus on a specific area of Identity Services from a developer standpoint. It builds on the concepts of Identity Services and Identity Externalization defined previously in - *Service-Oriented Security – An Application–Centric Look at Identity Management* - available at:

http://www.oracle.com/technology/products/id_mgmt/pdf/serv_oriented_sec.pdf

Our first paper, *Developing with Identity Services – Tacking Identity Data with Identity Hub* – available at:

http://www.oracle.com/products/middleware/identity-management/docs/tackling-identity-data-hub-whitepaper.pdf

examines the first key ingredient in any identity-enabled application – the identity data. As a developer begins to design the business logic of the application, the next security challenge is to ensure proper access to various aspects of the application – from data and transactions to application access. The topic of Authorization will be the focus of this paper.

INTRODUCTION

At the onset, application authorization appears to be an application-specific problem to be solved by the application developer. After all, it is the application and its resources that need to be protected. Traditional applications often define proprietary policies that can only be enforced and administered by proprietary mechanisms. A developer may want to restrict access to certain business logic. Business objects may also be subject to restrictions on the kinds of operations a user is authorized to perform - operations such as viewing or modifying. At the lower level, access to application or identity data from the repository must also be carefully considered.

Whether it is a packaged application from a vendor or a custom application within an enterprise, once an application is deployed in a production environment, any new requirements, customization or any modification to the authorization model would likely require a change in the code by the developers.

With applications scattered across an enterprise, administration of these proprietary authorization policies become a nightmare. In many situations, policies governing similar applications - for example, a set of financial applications - will likely to be similar. Administrators are forced to define and maintain different flavours of similar policies across these applications. And when it comes to compliance, the ability to audit and report on user entitlements across these disjoint systems becomes extremely difficult if not nearly impossible. These limitations highlight a general problem in that there is no enterprise level visibility and control over user entitlements – the set of privileges governing what an application user can do.

"These limitations highlight a general problem in that there is no enterprise level visibility and control over user entitlements – the set of privileges governing what an application user can do."

AUTHORIZATION CHALLENGE

Security requirements of an application are as important as the business requirement of any enterprise applications today. In fact, with the emergence of tighter corporate policies and government regulations, authorization has become an integral part of the business requirements. Developers are being asked to deliver applications that can enforce a high level of security right out of the box in both the enforcement and administration of user entitlements.

Protecting Business Logic

In a nutshell, securing the business logic of an application technically boils down to securing the application software components and the application business objects. The runtime logic needs to decide whether or not to allow access to a particular function or a particular piece of data. From a developer's perspective, if the application data contains all the data needed for making authorization decisions, he can implement the security around the application data itself. Traditionally, this is the approach and it gives the developer full control over the entire authorization model – from the types of information needed for the authorization decision to the decision logic itself. This creates a rather rigid model where authorization logic is completely hard-wired inside the application. While it gives the developer a great deal of control over the authorization model during development, it gives very little room for customization or expansion once the development has completed and the application has been deployed. Any changes needed in the authorization logic will likely require a code change. Also, any existing information from other applications or repositories must be pulled into the application metadata in order to be consumed, making data synchronization the responsibility of the developer as well. For example, a manager is allowed access over her direct reports' salary information. The manager-direct-report relationship, which may exist in a human resource management system (HRMS) or an LDAP directory, must be synchronized into the application metadata.

"The emergence of LDAP directory services as a centralized user and group repository allows application developers to leverage this information for both authentication and authorization purposes."

The emergence of LDAP directory services as a centralized user and group repository allows application developers to leverage this information for both authentication and authorization purposes. Some applications integrate directly with LDAP by directly connecting to the directory. In the J2EE world, Java Authentication and Authorization Service (JAAS) and Java Authorization Contract for Containers (JACC) provide a container layer of abstraction, freeing the developer from dealing with the underlying authorization provider and its implementation. They provide developers a framework to implement declarative security by defining application-level permissions and privileges, which are then mapped to a set of security roles. These roles are in turn assigned to either any individual user or a group from underlying provider. The permission and privilege definitions, along with the role mappings, are defined outside of the application in the various deployment descriptors, which can be modified without changing the application. Furthermore, changes to users, groups and group membership in the provider (for example, an LDAP provider) can alter the security of the application without involving the developers. However, certain changes – such as adding a new permission class or changing the descriptors would still require a change in the application or a redeployment of the application.

While it is not a complete de-coupling, this level of authorization externalization has given developers and application administrators some flexibility in customizing and expanding application security. But role-based security has its limitation. Suppose a financial application would like to enforce a policy that a customer service representative can only view customer data between the hours of 9am to 5pm. Here the current time is being used as part of the authorization criteria. In addition, enforce a policy where the customer service representative can only see your social security number on the screen but is not allowed to print it - while an account manager is allowed to do so when printing a report. In this example, the output type (screen vs printer) becomes part of the authorization criteria on the same subject (social security number). These kinds of authorization decision require a much richer context to provide all the necessary information for evaluation. Programmatic security is always an option by once again hard-wiring the decision logic into the code – which puts the developer back to the original dilemma. Once the logic is baked in, it is difficult for the deployment to customize them.

Enforcing Data security

Suppose in our financial application, a policy is defined to allow the regional account manager to only see the customer accounts in his region. From a developer standpoint, a way to enforce this is by programmatically generating the right filter during a database lookup – or, in other words, generating the right WHERE clause for a SELECT statement. While this is doable from within the application, the authorization decision should belong outside of the application. Access by any database client or any application should ideally be secured by the same logic. For example, a reporting utility should be governed by the same policy and should not be required to re-implement the predicate generation logic over and over again.

"What is more important is the ability to reflect any data security enforcement as a security policy rather than a baked-in logic in the code."

What is more important is the ability to reflect any data security enforcement as a security policy rather than a baked-in logic in the code. Using the same example as before – a policy stating that a manager is allowed access over her direct reports' salary information may need to be enforced at the data level.

Incorporating Risk

While user entitlements define who has access to what, enterprises are beginning to look for more context-aware authorization mechanisms to protect their enterprise environment. For example, in deciding whether a user should be allowed to transfer corporate funds to a holding account, a financial application may authorize the transfer if the risk is not too high where the risk is calculated based on certain criteria in the context of the transaction. In this example, a transaction might be considered as high risk if the user is connecting from outside the corporate network and is not using strong authentication. It might also be considered as high risk because the user has already made more than 4 transfers within the past 24 hours.

These types of risk-based policies in particular are very customer focused and tend to change frequently as well - making them difficult requirements for a developer to satisfy during the development cycle of the application.

Administration

"In most cases, it is business users rather than IT administrators who maintain and administer user entitlements. How user entitlements are represented is an important consideration."

When dealing with a single application, a proprietary handling of policy administration may not be a big deal. However, when you look at a typical enterprise with its large number of applications deployed, a consistent administration experience becomes crucial. Lack of any standards has allowed developers to take different approaches in how authorization policies are defined and administered. While the JACC/JAAS model provides some consistency in the way LDAP and J2EE security is used, developers are often asked to develop more business-user friendly interfaces within the application rather than forcing business users to deal with XML files and LDAP users and groups. In most cases, it is the business users rather than IT administrators who maintain and administer user entitlements. How user entitlements are represented is an important consideration.

Where user entitlements are stored is also an administration headache. For example, a user's LDAP group membership may determine her entitlements in a JAAS-enabled J2EE application while her responsibilities in some of the ERP systems are defined within the ERPs' own authorization models. With scattered authorization mechanisms across an enterprise, obtaining an enterprise view of user entitlements is a non-trivial task as it requires access to multiple systems.

Policy Sharing

" … Each application will carry a similar policy and will implement its own authorization decision logic – all doing virtually the same thing."

When looking across multiple applications, the lack of ability to share and reuse authorization policies is also apparent. Consider one of the previous policies mentioned – where the regional account manager is restricted to see and manage only the customers within the region. There could be multiple financial applications in place – each of which will likely require a similar policy to be created. The result – each application will carry a similar policy and will implement its own authorization decision logic – all doing virtually the same thing. This creates a huge headache for the administrators to, not only understand the different policy models, but also to maintain integrity across the enterprise.

Delegation

Delegation of access control is a common but difficult problem for any developer. In a financial application, an account owner might want to delegate certain capabilities, such as transfer privilege, to other account users. An account manager might delegate access to his clients to his manager while he goes on vacation. For a developer, the problem is not as simple as assigning the appropriate access to the account manager. Suppose while during the vacation, one of the clients has moved to another account manager under a different management chain. The initial delegation is now obsolete since the client no longer belongs to this management chain. These types of authorization integrity are difficult to capture within the application logic because the desired delegation model is often complex and customizable.

Audit

With government regulations such as Sarbanes-Oxley, the ability to report and review user entitlements across an enterprise is mandatory for enterprises. The pitfalls in policy administration, policy sharing and delegation are amplified in the ability to efficiently perform audit on user entitlements. Policies may be defined in ways that are not obvious to determine the exact user entitlements. Since each application has it own way of defining policies, any auditor or attester may have to deal with inconsistent formats spanned across multiple applications of various flavours in order to obtain the full picture of user entitlements. Similar to the administration pain, the lack of enterprise level visibility of user entitlements is a key contributor in the challenges and inefficiencies seen in the area of audit and compliance.

"Similar to the administration pain, the lack of enterprise level visibility of user entitlements is a key contributor in the challenges and inefficiencies seen in the area of audit and compliance."

AUTHORIZATION SERVICE – THE MODERN APPROACH

As we can see, there is a strong need to define a centralized Authorization Service that provides the necessary support for both the developers and application customers to satisfy their authorization requirements.

Externalization

The key concept of Authorization Service is to enable entitlement externalization. To a great extent, the JACC/JAAS model has taken a step in this direction. Entitlement externalization separates the authorization logic from the business logic within an application. The application takes care of enforcing the authorization check. For example, it triggers a check whenever a customer's social security number is to be fetched to see if the person accessing the application has access to this data. The decision, however, is made completely outside of the application. It is made by the authorization services based on policies defined within the authorization services. The application logic does not care whether the policy only allows the data to be fetched between business hours or disallows it if access to the application is through VPN. This distinction and separation allows authorization policies to change without impacting the application logic – or in other words, allowing a greater degree of customization without requiring a change in the application code.

"This distinction and separation allows authorization policies to change without impacting the application logic – or in other words, allowing a greater degree of customization without requiring a change in the application code."

PAP, PEP, PDP & PIP

To efficiently support entitlement externalization, the following services must be present as part of the authorization service to provide the needed functionalities:

- A **Policy Administration Point (PAP)** where administrators can create and modify application specific policies.

- A **Policy Enforcement Point (PEP)** that is responsible for triggering entitlement policy decisions within applications.
- A **Policy Decision Point (PDP)** that provides the actual entitlement decisions on behalf of the PEP.
- One or more **Policy Information Points (PIPs)** which supplies data and information to the PDP to help it make accurate policy decisions.

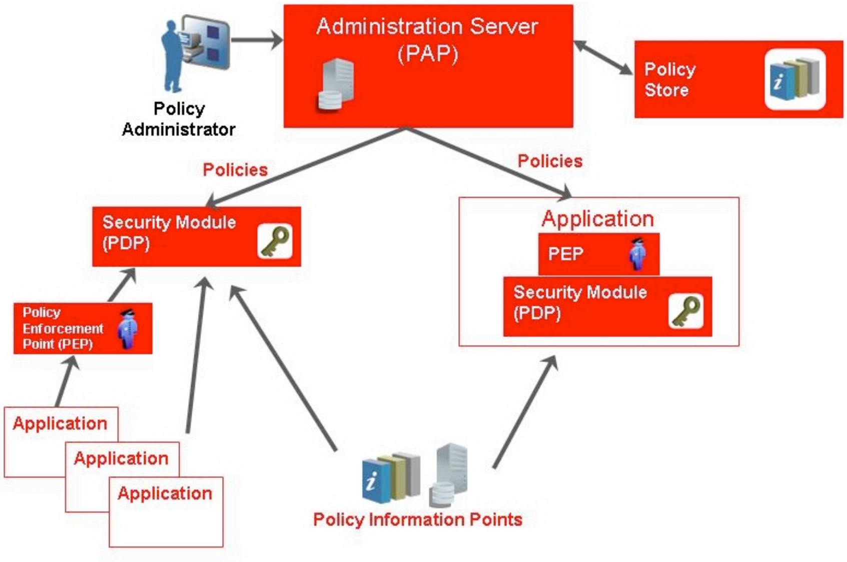

Figure 1. Authorization Service Architecture

The centerpiece of this architecture is the PDP. It must contain application specific policies and be able to take requests from the PEP such as, "*Can this user have this privilege on this resource?*" The PDP will consult its policies and either returns a *Grant* or *Deny* response to the application PEP along with any additional criteria or *Obligations* that the PEP must fulfill either before or after granting access to the user.

"By reaching out to the PIPs, PDP will consider Joe's current identity (including any groups, attributes or roles he may be aligned with), any attributes tat he may have (such as his State), and compare these against policies currently established for the resource 'SSN'."

For example, our financial application, the PEP, might ask the PDP, "Can user 'Joe' have the 'view' privilege for the resource 'SSN'?" By reaching out to the PIPs, PDP will consider Joe's current identity (including any groups, attributes or roles he may be aligned with), any attributes that he may have (such as his State), and compare these against policies currently established for the resource 'SSN'. Joe happens to be a customer service representative accessing a user record. However, Joe is accessing this data after business hours on a Friday evening. So the PDP returns a response of *Deny*. On Monday during business hours, Joe attempts to access the user record again. This time, the PDP returns a response of *Grant* with an *Obligation* requesting the financial application, the PEP, to log the SSN access by Joe.

Centralized Administration

"The PAP now offers an enterprise view and control of user entitlements across all the applications integrated with the authorization service.."

The centralized administration is provided through the Policy Administration Server (PAP). Authorization policies are defined and consolidated into a single policy store across applications. The PAP provides a consistent user experience for policy management. It also ensures a consistent format in how authorization policies are represented. From a compliance standpoint, operations such as audit and attestation will both benefit from the centralization of user entitlement definitions. The PAP now offers an enterprise view and control of user entitlements across all the applications integrated with the authorization service.

Sharable, Re-useable policies

Within a certain line of business there tends to be similarities across applications. There is a good likelihood that similar user entitlements are defined across these applications. By having these policies managed centrally (PAP), Authorization Service, acting as a shared service, can enforce consistent access and role mapping decisions across all these similar applications. Centralizing and externalizing the policy decision also eliminates the need for each application to make the same or similar set of authorization decisions.

"A centralized Authorization Service must provide the much needed externalization and consistency for both the developers as well as administrators."

A centralized Authorization Service must provide the much needed externalization and consistency for both the developers as well as administrators. It frees the developer from dealing with the actual authorization decision-making and allows him to concentrate on the application logic by decoupling the authorization logic from the application logic. By pushing out some of the complexity to the Authorization Service, the overall application is more capable and flexible in dealing with the evolution of authorization requirements.

EXTERNALIZE & CENTRALIZE WITH ORACLE ENTITLEMENTS SERVER

In traditional applications, security decisions are often hard-wired into the application logic. **Oracle Entitlements Server (OES)**, a new addition to the Oracle Access Management Suite, provides a centralized authorization service to support externalization of authorization policies and entitlements from within the application. OES provides a set of Security Modules, which act as Policy Decision Points (PDPs). These Security Modules evaluate entitlement policies on behalf of an application or a service. Applications and services can then act as Policy Enforcement Points (PEPs) to ask the Security Modules for decisions whether a user can perform an action on a particular resource. This means that application code is simplified and does not contain any security policies embedded as application logic.

OES also provides pre-built PEPs for certain structured runtimes such as Oracle Database, application servers and Microsoft SharePoint Server. Integration with Oracle Data Service Integrator and Oracle Virtual Private Database gives way in defining and enforcing data level security by providing an intermediate layer between the database client and the database that can enforce data security based on policies defined in OES.

“OES provides the ability to easily leverage existing enterprise data (RDBMS, LDAP, Web Services, customer sources, etc) in the policies for highly data-driven entitlement solutions.”

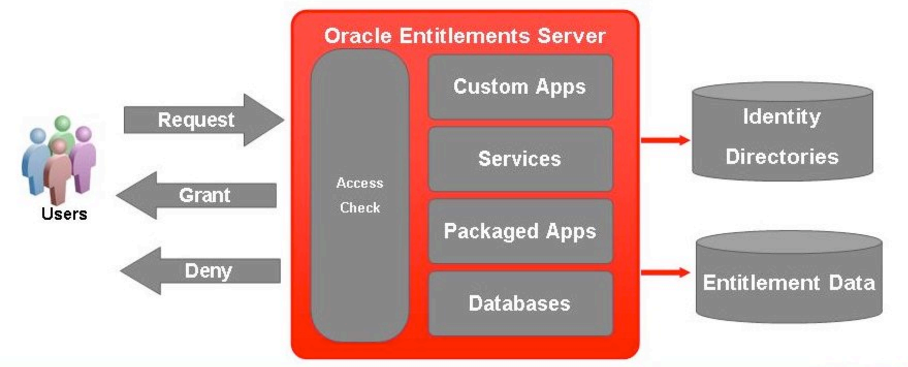

Figure 2. Oracle Entitlements Server

OES provides the ability to easily leverage existing enterprise data (RDBMS, LDAP, Web Services, customer sources, etc) in the policies for highly data-driven entitlement solutions. Entitlement policies can leverage attributes from these data sources to implement an Attribute Based Access Control (ABAC) system if so desired. In addition, extensions such as custom attribute retrievers and evaluation functions can be implemented to cater customer specific requirements.

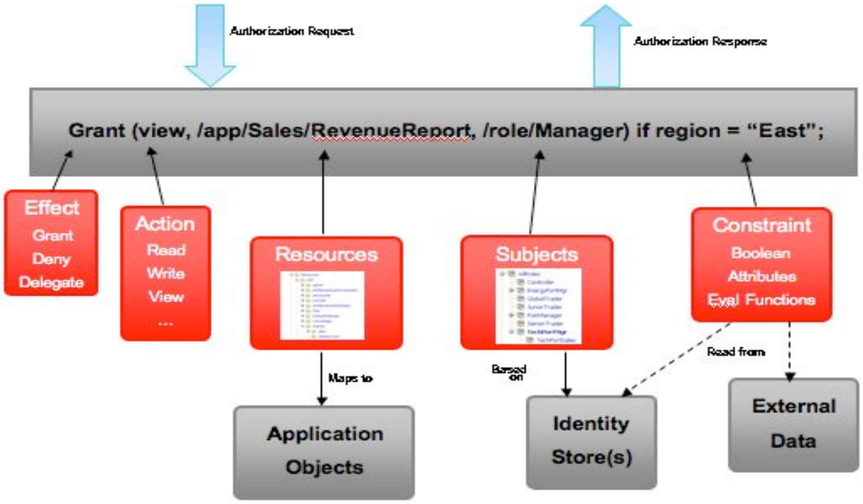

Figure 3. OES Policy

The management of all the authorization policies is centralized within OES. An administrator defines the fine-grained entitlement policies, which are then pushed to the Security Modules. This distribution can be done without disrupting the applications using the Security Modules. In this way, entitlement policies can evolve independently from the applications they are protecting. OES also provides management APIs for implementation of custom management application.

"Applications may use the XACML request/response protocol to obtain authorization decisions from the OES Security Modules. For interoperability, OES policies can be imported and exported in a standard XACML format.

In terms of standards, OES supports **eXtensible Access Control Markup Language (XACML)**. Applications may use the XACML request/response protocol to obtain authorization decisions from the OES Security Modules. For interoperability, OES policies can be imported and exported in a standard XACML format.

EXTEND AUTHORIZATION SERVICE WITH RISK-AWARE ORACLE ADAPTIVE ACCESS MANAGER

"In the context of authorization service, ARM provides another policy information point (PIP) that OES can leverage."

Oracle Adaptive Access Manager (OAAM), another member of the Oracle Access Management Suite, offers the capability to provide context-aware risk analytics in real-time to prevent fraudulent activities through its Adaptive Risk Manager (ARM) component. ARM evaluates risk by analyzing contextual data from a variety of sources, such as user profiles, device fingerprints, transactional data, IP Addresses, geographic location, and other 3rd party feeds. In the context of authorization service, ARM provides another policy information point (PIP) that OES can leverage. An OES policy may require the risk score to be below a certain threshold in order to grant access to a particular resource. In real-time, ARM is able to look at various risk factors and generate a risk score based on the context of the transaction.

CONCLUSION

Externalizing Authorization to an authorization service provides developers with a centralized and consistent way to implement authorization requirements while developing an application. For enterprises dealing with hundreds of different applications, this centralization of authorization policies is invaluable as it provides the much needed enterprise level visibility and control over user entitlements. This greatly enhances the lives of administrator – both IT users and business users. It also puts enterprises in a much better position to satisfy compliance and regulatory requirements in the area of reporting, performing audit and access certification.

With the modern approach of authorization service, developers, vendors and enterprises are now better equipped to face the constant evolution of authorization requirements.

For more information on Oracle's security technology,

Go to ***http://www.oracle.com/security***

Developers and Identity Services – Tacking Identity Data with Identity Hub
November 2008
Author: Stephen Lee
Contributing Author: William Dettelback, Nishant Kaushik

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Copyright © 2008, Oracle Corporation and/or its affiliates. All rights reserved.
This document is provided for information purposes only and the
contents hereof are subject to change without notice.
This document is not warranted to be error-free, nor subject to any
other warranties or conditions, whether expressed orally or implied
in law, including implied warranties and conditions of merchantability
or fitness for a particular purpose. We specifically disclaim any
liability with respect to this document and no contractual obligations
are formed either directly or indirectly by this document. This document
may not be reproduced or transmitted in any form or by any means,
electronic or mechanical, for any purpose, without our prior written permission.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.