ORACLE 9I APPLICATION SERVER VERSION 2 SECURITY

John Heimann, Oracle

INTRODUCTION

This paper describes security features of Oracle9*i* Application Server (Oracle9*i*AS) version 2. It presents an overview of web security requirements, and then discusses the security architecture implemented in Oracle9iASv2 to meet these requirements. The paper provides specific focus on new security features introduced in Oracle9iASv2.

SECURITY FOR WEB APPLICATIONS

The web has replaced client-server as the preferred way in which organizations provide their users with access to business applications and data. There are many well-understood reasons for this, including greater scalability, reduced cost, and opportunities to reach new users and markets. The risks of deploying business applications on the internet have also been widely recognized. Risks include:

- limited knowledge of user identities,
- little or no control over the behavior of users (it's hard to punish misbehavior of a user if you don't know who they are, or if they are not a member of your organization),
- greatly increased exposure of systems and data to malicious users,
- disclosure or corruption of information in insecure networks,
- attacks which exploit specific open features of the internet such as worms, cross-site scripting, etc.

Web applications developers have had to address these risks for several years now. Some recent trends which have added to the complexity of web application security include the deployment of multiple applications through a single information portal, increasing use of Java for web application development, and the increasing complexity and scale (in terms of number of users served) of deployed applications.

ORACLE9IAS SECURITY OVERVIEW

Oracle9iAS version 1 introduced Oracle9iAS Single Sign-On (SSO), which supports a broad range of applications, and included security services in many components, in particular Oracle HTTP Server and Oracle9iAS Portal. In Oracle9iAS version 2, Oracle introduces a comprehensive security framework supporting all Oracle9iAS components, as well as third-party and custom applications deployed on Oracle9iAS. The framework is based on Oracle9iAS SSO for authentication, Oracle Internet Directory for authorization and user provisioning, and the Oracle Java Authentication and Authorization Service (JAAS) provider for security in Java2 Enterprise Edition (J2EE) applications.

SINGLE SIGN-ON IN ORACLE 9IAS

An important security feature of Oracle9*i*AS is support for single sign-on (SSO) to web-based applications. There are a number of reasons why businesses are considering SSO. These include the increasing use of web-based eBusiness applications which companies are deploying for use by employees, customers and partners. Without SSO, each user must maintain a separate identity and password for each application she accesses. Maintaining multiple accounts and passwords for each user is insecure and expensive.

MULTIPLE ACCOUNTS AND PASSWORDS ARE INSECURE

Most users cannot remember more than a few passwords. Users who maintain more than one login account often choose easy-to-remember passwords, choose identical passwords for different accounts, reuse passwords when asked to change them, or write passwords down. All these practices reduce password security. Writing passwords down or choosing easily remembered (and thus easily guessed) passwords increases the risk of passwords being compromised. Both reusing passwords when asked to change them, or using the same password on multiple systems, increase the potential damage if a password is compromised. Although many systems implement password management mechanisms which force users to choose complex passwords, or prevent them from reusing a password, these mechanisms often backfire: users figure out ways to defeat them, and thus reduce security even more. For example, forcing users to use random passwords almost guarantees that the passwords will be written down.

If a user joins or leaves an organization, or changes functional roles within the organization, the privileges which that user has when accessing applications supported by the organization change. Multiple, independent accounts per user often means that the associated user privileges do not keep up with organizational changes. For example, user accounts and access privileges may remain in the system long after the user has left the organization or changed roles. This opens the system up to potential attack by disgruntled former employees.

MULTIPLE PASSWORDS ARE EXPENSIVE

Managing multiple accounts and passwords per user is expensive. In many enterprise deployments, a substantial fraction of the system administrators' time is spent on account- and password-related problems, including initial creation of users' accounts when they join the organization, deletion of accounts when they leave or change roles, and resetting passwords that have been forgotten. Having several accounts per user multiplies the associated demands on the system administrator.

Among the problems which system administrators must deal with are having to access multiple systems, through multiple, possible different administrative interfaces, to add or remove user accounts on each system.

THE ORACLE9IAS SSO SOLUTION

Oracle web SSO technology provides single sign-on for web users. It is designed to work in environment such as that provided by Oracle9*i*AS, where multiple web-based applications are accessible through the Application Server. The Oracle strategy for SSO encompasses a variety of technologies. For the growing field of web-based applications, Oracle has developed an SSO framework and the Oracle9*i*AS SSO Server, which is specifically designed to provide web SSO.

The Oracle SSO approach has a number of benefits. It provides a framework for secure SSO from browser clients to web-based applications, including Oracle Applications and Tools, through standard protocols. It supports both partner applications, which take full advantage of the SSO framework, as well external applications for support of legacy and third party products. Partner applications work within the SSO framework and rely on the SSO service for authentication of users; external applications continue to use their own usernames and passwords. The Oracle9iAS SSO approach is based on cookies, which are created both by partner applications and by a centralized server called the Oracle9*i*AS SSO Server.

Unocal is a leading Oracle customer who is using Oracle9iAS. Unocal has selected Oracle9iAS SSO to provide single sign-on into their corporate information portal, myUnocal.com, and is working on global implementation of this standard. myUnocal provides business application services to Unocal employees around the globe, and supports Unocal's goal of consolidating business data and services. Oracle9iAS SSO will allow myUnocal employees to authenticate once and access those applications

to which they are entitled, without having to remember separate usernames and passwords for each application.

Components

Oracle9*i*AS *SSO Server*

The core of Oracle SSO technology is the Oracle9*i*AS SSO Server. This technology was originally introduced in Oracle9iASv1 as a component of Oracle9iAS Portal, but in Oracle9iASv2 the SSO Server is an infrastructure component, and does not require Portal to be installed.

The Oracle9*i*AS SSO Server authenticates users, and passes their identity securely to partner applications. It prompts users for a username and password when they access the system for the first time in a given time period (usually a day), and verifies the password presented by the user.

Oracle9*i*AS SSO Server uses cookies, which are formatted pieces of information stored on a browser client by a web server. Cookies allow web servers to store and retrieve information about the client user, effectively maintaining client state information in the otherwise stateless web environment. Cookies are supported by all current browsers, although they can be disabled by the user (in which case Oracle9*i*AS SSO Server will not provide SSO). Cookies can be persistent, which means that they are saved on the client hard disk, even when the browser is shut down; or they can be non-persistent, in which case they are deleted when the browser is shut down. When a user authenticates to the Oracle9*i*AS SSO Server, the server sets an SSO cookie in the user's browser. When a browser then accesses the Oracle9*i*AS SSO Server, the presence of a valid SSO cookie confirms that the user has been authenticated.

Partner Applications

Partner applications are those which work within the SSO framework. Specifically, they are designed (or have been modified) to delegate responsibility for user authentication to the Oracle9*i*AS SSO Server. They accept the user identity presented to them by the Oracle9*i*AS SSO Server.

Since partner applications take advantage of the authentication services of the Oracle9*i*AS SSO Server, they do not need to implement their own authentication modules. User administration is simplified for partner applications, since there is no need to manage passwords for these applications. Deploying an application as a partner application thus can reduce both development and ongoing administrative expenses.

Mod_osso

`Mod_osso` is a new feature introduced in Oracle9iASv2. It is an extension to the Oracle HTTP Server which enables the HTTP Server to be an SSO partner application. Applications running underneath the HTTP Server, such as servlets, can then obtain a user's authenticated identity from `mod_osso` in the form of an Apache header. `Mod_osso` therefore allows applications to participate in the Oracle9iAS SSO framework without requiring that the applications embed specific partner application logic. It is the recommended way for applications running on Oracle9iAS to participate in Oracle9iAS SSO.

External Applications

External applications are those which retain their own usernames and passwords, and do not delegate responsibility for authenticating users to the Oracle9*i*AS SSO Server. These applications have not been developed or modified to work within the SSO framework. A typical external application might be one developed or deployed by a third party, such as a portal website which requires username and password for access to custom services like email.

Since external applications do not take advantage of the Oracle9*i*AS SSO Server authentication mechanism, they must implement their own authentication modules and manage their user

passwords. Because different external applications may have different web forms for entry of usernames and passwords, supporting external applications may require application-specific customization of the Oracle9*i*AS SSO Server. Users or system administrators may be required to take some specific action when installing or changing usernames and passwords, such as entering this information via a special registration page. Setting up such registration pages may require development of application-specific functionality within the Oracle9*i*AS SSO Server.

Although it is preferable to deploy applications using `mod_osso` or as partner applications, it is not always practical to retrofit an existing application. It is for this reason that external applications are supported by Oracle9*i*AS SSO Server. Supporting both partner and external applications in Oracle9*i*AS SSO Server provides maximum flexibility to the system integrator, since new applications as well as legacy web applications can be supported in a single SSO framework.

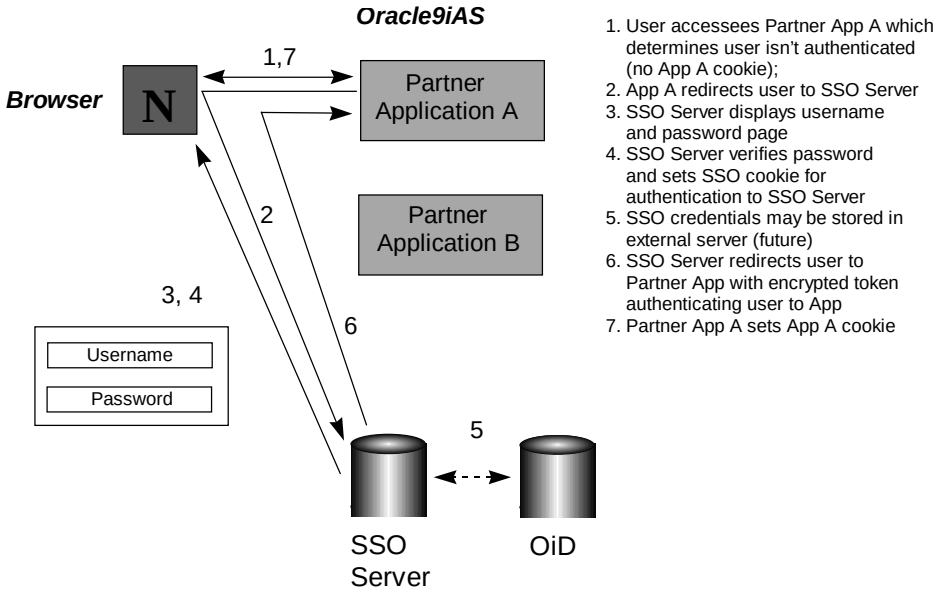
Functional Overview

Initial Authentication

When a user attempts to access a partner application for the first time, he is redirected by the partner application to the Oracle9*i*AS SSO Server. The Oracle9*i*AS SSO Server checks to determine whether the user has a valid SSO cookie set; if not, it requests that the user authenticate by submitting a username and password. Once the user has done so, the Oracle9*i*AS SSO Server verifies the password and then sets an SSO cookie in the user's browser. This cookie is used to authenticate the client to the Oracle9*i*AS SSO Server in subsequent HTTP interactions with the Oracle9*i*AS SSO Server.

The SSO cookie is encrypted by the Oracle9*i*AS SSO Server, so that it cannot be set or read by a third party. Cookies expire after a certain period of time, as set by the administrator (typically 8 hours); or when users shut down their browser. As distinguished from partner cookies, SSO cookies are not persistent, and are deleted when a browser is shut down.

Figure 1: Authentication to Oracle9*i*AS SSO Server and Partner Application



Note that interactions between the Oracle9*i*AS SSO Server, browser clients, and applications are all via standard HTTP. No special requirements are placed on clients, other than that they support cookies. It is recommended that SSL be enabled between Oracle9*i*AS SSO Server and the client to prevent usernames, passwords, and SSO cookies from being intercepted by a third party, who could possibly use them to spoof the Oracle9*i*AS SSO Server.

Authentication to Partner Applications

Once a user has been authenticated and an SSO cookie has been set, the Oracle9*i*AS SSO Server directs the user back to the partner application, and includes an encrypted token, which contains the user's identity, in the partner application URL. The token is encrypted in a key which is shared only by the Oracle9*i*AS SSO Server and the partner application. This assures the partner application that the token is authentic, and was created by the Oracle9*i*AS SSO Server.

When the partner application receives and decrypts the URL token, it can determine whether to grant the authenticated user access to the application. To grant access, it sets a partner application cookie in the user's browser. The partner application cookie allows the application to identify and grant access to the client user, without having to redirect the user to the Oracle9*i*AS SSO Server for authentication. Partner application cookies, like SSO cookies, expire after a certain, application-specific period of time. Unlike SSO cookies, partner cookies may be persistent or non-persistent (that is, they may or may not survive the shutdown of a browser). The expiration period for a partner application cookie is determined by the application, and may be different than the SSO cookie expiration time.

As with SSO cookies, it is recommended that SSL encryption be used to protect the cookie exchange between browser and partner application.

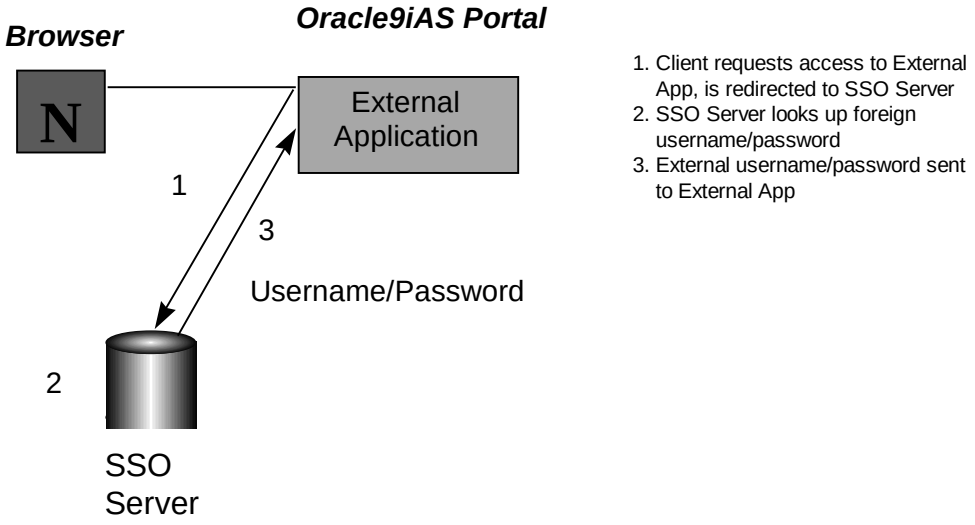


Figure 1: Authentication to External Application

Authentication to External Applications

External applications cannot accept an authenticated identity directly from the Oracle9*i*AS SSO Server. The Oracle9*i*AS SSO Server provides SSO to external applications which support authentication via web forms. The Oracle9*i*AS SSO Server provides SSO to external applications by means of a secure password store mechanism. The password store maintains application-specific usernames and passwords in a table within the Oracle9*i*AS SSO Server. Access to this table is restricted by the Oracle9*i*AS SSO Server, and passwords are further protected via encryption. When a user who has been authenticated by the Oracle9*i*AS SSO Server needs to access an external application, the Oracle9*i*AS SSO Server retrieves the user's username and password for that specific application from the password store, formats them in the appropriate web form, and submits them to the application. This is done transparently to the user.

Note that SSL encryption between Oracle9*i*AS SSO Server and external applications can be used to prevent exposure of application passwords in the network.

LDAP Integration

Directories supporting the Lightweight Directory Access Protocol (LDAP) are increasingly used as a single source of enterprise-wide information about users. These directories provide a convenient mechanism for provisioning (creating and configuring) and managing users who use multiple applications or servers in an enterprise. This is because LDAP is a widely supported, Internet-standard protocol, and because an LDAP directory can be used as a convenient, single source of

information about users, accessible throughout the enterprise. Oracle Internet Directory (OID) is particularly well suited for this type of application, since it provides a secure, scalable, performant, and highly available directory service (for more on OID, refer to the section entitled "Directory-Enabled Security via OID").

Oracle Oracle9*i*AS SSO Server verifies SSO usernames and passwords using OID. When a user submits an SSO username and password as part of the initial authentication, the Oracle9*i*AS SSO Server compares the username and password with that maintained in OID. If the comparison succeeds, the SSO username and password are considered to be verified. Note that in Oracle9iASv1, maintaining user identities in OID was optional, whereas in Oracle9iASv2 this is now the default. Managing usernames and passwords in OID is consistent with the overall Oracle9i platform (database and application server) strategy of managing user information for components in a single, standard, centralized LDAP repository.

In Oracle9iASv2, OID provides extensible authentication through a Password Verifier API. The Password Verifier API allows OID to accept user authentication data (in the form of <username>/<password verifier>) and check its validity using custom built or third party authentication mechanisms. Since Oracle9iAS SSO depends on OID to validate authentication data, the OID Password Verifier provides extensible authentication to Oracle9iAS SSO, and allows it to support a wide range authentication technology.

THIRD-PARTY INTEGRATION

Oracle9iAS SSO provides an API for integration of third-party authentication and single sign-on; this feature was introduced in Oracle9iASv1.0.2.2. The API allows SSO to be configured to obtain user identity from a trusted external authentication mechanism, and allows integration of Oracle9iAS into an SSO framework provided by a third party product, such as Siteminder® from Netegrity, Inc.

PKI SUPPORT

PKI authentication is beginning to replace passwords in many applications. In web-based applications, PKI authentication is typically performed through an exchange of X.509 certificates, as part of a Secure Sockets Layer (SSL) session establishment. PKI by itself can be used to provide SSO, since a user with a certificate can authenticate to multiple applications without entering a password.

In Oracle9iASv2, users can authenticate to the Oracle9*i*AS SSO Server via PKI. This will provide SSO both to web-based applications supported by Oracle9*i*AS SSO Server, and to other PKI-enabled applications. Instead of providing an SSO username and password, users will authenticate to the Oracle9*i*AS HTTP Server via SSL with client and server X.509 certificate exchange. The Oracle9iAS SSO Server will obtain the user's SSL-validated certificate from the HTTP Server, and look up this certificate in Oracle Internet Directory (OID). If the user is found, OID will return the user's SSO identity to the Oracle9iAS SSO Server. Authentication to partner and external applications will then be performed by the Oracle9*i*AS SSO Server, using the cookie-based approach described previously.

Benefits of this approach are that applications which work within the Oracle9*i*AS SSO Server framework will automatically be PKI-enabled when the Oracle9*i*AS SSO Server is PKI enabled. The Oracle9*i*AS SSO Server and OID assume responsibility for name mapping. Moreover, since getting and checking a cookie is much less processing-intensive than performing an SSL exchange, using PKI for initial authentication to the SSO framework and cookies for authentication to partner applications should have better performance than a PKI-only authentication approach. For web applications which are characterized by many short-lived sessions, this can lead to significant improvement in server performance and throughput.

Finally, PKI-enabling the Oracle9*i*AS SSO Server is the Oracle strategy for allowing users to authenticate to Oracle Applications using PKI. Since Oracle Applications participate in the SSO framework through the Application Portlet in Oracle9iAS Portal, support for PKI in Oracle9*i*AS SSO will allow PKI authentication to Oracle Applications.

OTHER SECURITY ENHANCEMENTS

In Oracle9iASv2, SSO includes a number of enhancements which improve the flexibility and security of the SSO solution. Among these enhancements are Single Sign-Off, Paranoid Application support, and Global Inactivity Detection.

SINGLE SIGN-OFF

Single Sign-Off allows users to terminate not only an SSO session, but also to terminate sessions with partner applications. This feature is important since partner application timeout periods can be longer than the SSO session, so it is possible for a user not to have a valid SSO session (because the SSO session cookie is timed out) but to continue to have a valid partner application session if the partner application cookie has a longer validity period than the SSO cookie. Single Sign-Off allows the user to log out of the SSO session and all partner applications through a single action, e.g., when going home for the day.

PARANOID APPLICATION SUPPORT

Paranoid application support allows partner applications to force the SSO server to re-authenticate a user whether or not the SSO cookie is still valid. Before this feature was introduced, if a partner application session timed out but the SSO session was still valid, the partner application would redirect the user to the SSO server, but the SSO server would simply return the user's identity without re-authenticating the user. The paranoid application feature allows particularly sensitive ("paranoid") applications to require more frequent re-authentication than the SSO server requires. It also permits event-driven authentication, so that a user can be forced to re-authenticate when performing some particularly sensitive action within an application.

GLOBAL INACTIVITY DETECTION

Partner applications can timeout a session based on inactivity using the paranoid application feature just described. In 9iASv2 it is also possible to configure global inactivity detection, so that failure to use any partner application for a specified period of time will cause SSO session timeout. This feature can be used to implement security policies requiring user re-authentication if the user is inactive for given period.

THREE-TIER INTEGRATION

The Oracle9*i*AS SSO Server provides SSO for web client access to web servers. Web servers are increasingly being deployed as the middle tier in a three-tier architecture, where they provide access to a back-end tier database. Users of web applications that require access to the database should not have to supply a database username and password for access to data stored there. Although the Oracle9*i*AS SSO Server does not support non-web based applications, the Oracle9*i* database includes features specifically designed to support secure access to databases through three-tier architectures. For more information, please refer to the Oracle white papers "Securing Three-Tier Systems with Oracle8*i*" and "Database Security in Oracle8*i*".

SSO SUMMARY

Oracle9iAS SSO provides an authentication framework which can be highly useful when deploying multiple web applications. SSO improves the user experience, reduces application development costs since applications do not need their own authentication mechanisms, reduces system

administration costs, and improves system security since problems associated with multiple passwords are eliminated.

DIRECTORY-ENABLED SECURITY VIA OID

Oracle has standardized on LDAP as the common mechanism for Oracle products to manage enterprise information about users and services in the enterprise. To support this, Oracle has developed a highly scalable, reliable, and secure LDAPv3-compliant directory, Oracle Internet Directory (OID) based on Oracle9i's proven database technology. OID provides LDAP directory services to Oracle products, and Oracle products in turn certify their LDAP implementation against OID.

DIRECTORY AUTHORIZATION

Oracle products use OID to manage enterprise security information, in particular that which is shared among Oracle enterprise components. This information includes user identities, authentication data such as SSO passwords, and authorization data such as roles or group membership. In Oracle9iASv2, OID is a core component of the Oracle9iAS infrastructure, and is the common repository for user, authentication, and authorization information. It replaces component-specific repositories which existed in Oracle9iASv1. OID provides a common, LDAPv3 standard framework for representing user privileges. Use of OID not only provides a common privilege management mechanism for Oracle9iAS components, but also provides a vehicle for integrated management of privileges between Oracle9iAS and other LDAP-compliant enterprise components.

To manage privileges in OID, Oracle9iASv2 also introduces Delegated Administrative Services (DAS), an application which allows system administrators and, where appropriate, users themselves, to manage user information in OID. DAS includes both a web based GUI application and a set of APIs for administration of OID data.

EXTENSIBLE AUTHENTICATION

OID provides a centralized, secure repository for user password information. In Oracle9iASv2, OID introduces support for extensible authentication through a Password Verifier API. This API provides support for a variety of custom and third party authentication mechanisms, and was discussed in the section on Oracle9iAS SSO.

THIRD-PARTY DIRECTORY SUPPORT

To provide a single directory-based security framework across multiple applications, customers may need to integrate OID with other, third party directory products. Oracle9iASv2 therefore introduces the Directory Integration Platform (DIP). The DIP provides a framework for building connectors between OID and third party directories, and supports referral and synchronization between OID and other directories.

Oracle HTTP Server Security

Oracle HTTP server is the Oracle web server component of Oracle9*i*AS. It is based on the open source Apache web server. The Apache server is among the most widely-adopted web server products; it supports a rich set of existing applications, and provides a flexible and well-understood security model. Apache is a very well-tested platform on which to deploy secure applications. Customers familiar with Apache should find it easy to build and deploy secure web applications using Oracle HTTP Server.

HTTP SERVER SECURITY SERVICES OVERVIEW

Oracle HTTP Server extends Apache with a variety of standard and Oracle-unique enhancements (or "mods," as they are referred to in the Apache community). It allows users equipped with web browsers to access Oracle9*i*AS using standard web protocols. It provides a basic HTTP listener capability (HTTP and secure HTTP, or HTTPS), and provides access to both static web pages and dynamic content through a variety of interfaces.

Oracle HTTP Server security services include the ability to restrict or allow access to files and services based on the identity of users established via basic challenge/response operations, via client-supplied X.509 certificates and via IP or hostname addresses.

Another important feature of Oracle HTTP Server security is protection of data exchanged between clients and the server. This is provided via the SSL protocol, which also provides data integrity and strong authentication of both users and HTTP servers.

In addition, Oracle HTTP Server provides logging and other facilities needed to detect and resolve intrusion attempts. It provides integration with the other Oracle9*i*AS components and products such as the Oracle8i database. In this way, the Oracle HTTP Server provides a comprehensive set of security services for building web applications.

ACCESS CONTROL

When URL requests arrive at the Oracle HTTP server they are processed in a number of steps which are implemented via the mod/plug-in architecture common to all the popular webserver/listeners. Access control is applied early in the request processing cycle.

Oracle HTTP Server access control is based on Apache access control mechanisms which allow the server administrator to restrict access to particular files, directories, or URLs on the server. For each restricted object on the server, the administrator can specify, by means of a *directive*, that access to the object is denied or allowed based on the value of one or more *attributes* associated with the requester. The administrator can configure directives such as `deny`, `allow` and `order` to inhibit further processing, based on user attributes such as hostname, IP address, or browser type. Restrictions can be applied to particular files, directories or URL formats using the `<files>`, `<directory>` and `<location>` configuration directives, respectively.

In the following example, requests originating from any IP address in the 192.168.1.* range or with the hostname `us.oracle.com` are allowed access to files in the directory `/internalonly/.`

```
<Directory /internalonly/>
  order deny,allow
  deny from all
  allow from 192.168.1.* us.oracle.com
</Directory>
```

Note that allowing or restricting access based on hostname *for Internet access* is not considered a very good method of providing security, since hostname is easy to spoof. While the same is true of IP addresses, sabotage is slightly more difficult. Thus, access control via IP/hostname for *intranet* use is reasonable in many situations where the same cannot be said for Internet IP and hostname restrictions.

Although the Oracle HTTP Server is based on the open source Apache Server, it contains some access control enhancements which improve security. For example, the Apache Server provides for access restrictions per directory/folder via files with the suffix `.htaccess`. The processing of these

files is disabled by default in Oracle HTTP Server, since `.htaccess` processing involves both security and performance-degradation problems.

USER AUTHENTICATION AND AUTHORIZATION

In many applications it is desirable to control access to resources on the web server based on user identity. Oracle HTTP Server provides several mechanisms for user authentication, including client authentication over the Single Sockets Layer (SSL) using X.509 certificates, username/password (as in `basic` authentication), and other forms. A server administrator can specify, via Apache directive, that access to specific URLs is restricted to specific users, who must be authenticated via a specific mechanism.

Once user authentication is established, additional rules can be applied restricting or allowing further processing of URL requests. Access control directives based on user identity can be combined with directives based on IP address or hostname (as described above), so that user requests must satisfy both directives in order to allow further processing. For example, you can constrain access to a particular URL so that only certain named users can access it, and then only from within the corporate intranet. You can do this by configuring the directive for the URL so that it requires user authentication, denies access except for specific named users, and requires that the IP address be within a certain range (to ensure that the user is within the intranet).

The Apache directive access control mechanisms implemented in Oracle HTTP Server thus provide a great deal of flexibility in managing user access to objects.

MOD_OSSO

`Mod_osso` is a new feature of the Oracle HTTP Server in Oracle9iASv2, and allows the HTTP Server to become an SSO-enabled Partner Application. `Mod_osso` is more fully described in the section of this paper on Oracle9iAS SSO.

SECURE SOCKETS LAYER (SSL)

The Secure Sockets Layer provides point-to-point security between Oracle9*i*AS HTTP Server and client browsers. Security-related services provided by SSL include authentication, authorization, confidentiality and data integrity. These are discussed below.

SSL CONFIDENTIALITY

The primary service provided by SSL is confidentiality: messages are encrypted so they cannot be read and understood by third parties. SSL uses a standard set of cryptographic mechanisms to encrypt data and distribute keys between communicating devices. The specific set of encryption, integrity protection, and key distribution algorithms chosen, together with encryption key length used, define a *ciphersuite*. The Oracle9*i*AS SSL implementation supports a wide range of standard ciphersuites. In particular, Oracle9*i*AS supports those ciphersuites which use X.509 certificates for authentication and key distribution (also referred to as PKI authentication).

Oracle HTTP Server allows SSL sessions to be cached, so that multiple message interchanges between two IP addresses can be exchanged under one session. Session caching is very important for performance reasons. Establishment of SSL sessions is very CPU-intensive, and has been known to take up to 90% of available CPU resources. SSL session caching is specified via the `SSLSessionCache` directive, whose parameter specifies the file or shared memory segment where SSL session information is maintained.

SSL Client Authentication

SSL can also be used to provide client authentication using X.509 certificates, as part of a public key infrastructure (PKI) deployment. Oracle HTTP Server can be configured to restrict access to files and services based on information in the client's X.509 certificate. Information that can be used in making an access decision includes the distinguished name (DN) in a client certificate, profile information contained within the DN, and the certificate trust point (that is, the Certificate Authority which issued the user's certificate). SSL can be configured to accept trust points it recognizes, or those signed by trust points it recognizes, and so forth. Authentication can be based on lists of partial or full distinguished names, or "wild-carded" versions of those names.

Once SSL authentication occurs, the information available in the certificate can be used in directives such as `<directory>`, `<files>`, and `<location>`, as described above. SSL authentication can be combined with Basic authentication and/or host-based access control. In this way you can allow or restrict different combinations of SSL- and basic-authenticated users' access to files and services, and combine such restrictions with host-based access control.

A new feature in Oracle9iASv2 is support for SSL client authentication to Oracle9iAS SSO. Refer to the section on PKI support in Oracle9iAS SSO for more information.

SSL Environment Variables

Oracle9*i*AS allows for security-related information about the SSL session, referred to as *environment variables*, to be passed to web server applications such as CGI scripts, servlets, and Perl scripts. Applications can use these environment variables to perform additional access control or authorization on user requests, based on information about the user, or the type of SSL session established on behalf of the user.

Environment variables include such information as:

- The URL which arrived in the HTTPS request
- The size of cipher key used in SSL session
- The ciphersuite used in the SSL session
- The distinguished name from the client certificate

SSL Logging

The Oracle HTTP server also provides for logging of SSL-related information. This can be used to determine if intrusions were attempted, and if they succeeded. It can also be used in determining the source of intrusion attacks, or for other purposes.

Secure Access To an Oracle Database

Oracle9*i*AS makes it easy to build three-tier systems using an Oracle database back-end repository. Oracle9*i*AS provides a number of mechanisms for accessing the database and invoking applications on the database. The most commonly used of these are fat client JDBC and `mod_plsql`, a plug-in to the Oracle HTTP Server that allows Oracle9*i*AS to invoke database applications written in the Oracle database programming language, PL/SQL. Since JDBC (when running on a fat client) and `mod_plsql` access the Oracle database using the Oracle client-server networking protocol, developers using fat client JDBC or `mod_plsql` can take advantage of Oracle Advanced Security, an Oracle9i option, to protect data exchanged between Oracle9*i*AS and an Oracle9i database. Oracle Advanced Security provides encryption, integrity protection, and advanced authentication

services to Oracle database clients and servers. It supports industry standard encryption protocols such as SSL, and standard encryption algorithms including RSA's RC4, DES and 3DES.

Note that Oracle has worked with many firewall vendors to ensure that data encrypted by Oracle Advanced Security is supported by all leading commercial firewall products. Oracle Advanced Security ensures that data exchanged between Oracle9iAS and Oracle9i is secure even against an inside attacker with access to a company's internal information.

Oracle9i also provides a feature called proxy authentication. This feature is designed to address a performance problem associated with three-tier application design. Specifically, it allows Oracle9iAS to access an Oracle9i database and get specific Oracle9i-user privileges without having to logout and login again each time Oracle9iAS switches user contexts. Moreover, it addresses the security problem of granted limited (but not complete) trust to the middle tier application server when accessing a database on behalf of authenticated users. In the past, application designers either had to grant the middle tier superuser privilege (e.g., SYS or root) so that it could access the database on behalf of any user, or else store database user passwords in the middle tier. Both of these solutions are insecure.

Proxy authentication allows Oracle9iAS to establish a single authenticated session (e.g., using fat client JDBC or `mod_plsql`) with an Oracle9i server, and act on behalf multiple Oracle9i users, without having to submit separate authentication credentials for each user within the session. The application server must specify which user it is acting on behalf of, and must have been granted privilege to act on that user's behalf by Oracle9i. Moreover, Oracle9i can use both the authenticated identity of the Oracle9iAS, and the identity of the user on behalf of whom the Oracle9iAS performs proxy authentication, when making access decisions or writing audit records of events. Proxy authentication allows Oracle9i to delegate limited trust to a middle tier Oracle9iAS, without having to grant it superuser privilege on the database, or store multiple database user passwords in Oracle9iAS.

JAVA SECURITY IN ORACLE9IAS

Java, and in particular Java2 Enterprise Edition, has emerged as the development environment of choice for many new web applications. Java2 Enterprise Edition defines a Java2 Security Model and a security framework referred to as Java Authentication and Authorization Service (JAAS). Oracle9iAS implements this framework through a fully-J2EE compliant JAAS provider. The JAAS provider makes user authentication, authorization, and delegation services accessible to application developers, and allows them to integrate these services into J2EE application environments.

THE JAVA2 SECURITY MODEL

The Java2 Security Model is defined by the Java division of Sun Microsystems, Inc.. It is capability based, and allows developers to specify protection domains, and security policies associated with those domains. Security policies specify the set of permissions associated with Java classes which execute within those domains. Permissions define specific types of access granted to specific objects; e.g., read access on directory /salaries/.

THE JAAS PROVIDER

The Oracle9iAS JAAS provider implements the Java2 Security Model, allowing application developers to obtain authenticated user (principal) identity from a set of standard authentication services provided by JAAS, and to manage the privileges which principals have for accessing objects. It also supports privilege delegation, for managing privileges of methods invoked by principals.

JAAS AUTHENTICATION

The Oracle9iAS JAAS provider supports a flexible authentication framework. It provides specific mechanisms for authentication, based on SSL and SSO, but also allows developers to integrate custom authentication modules through the standard JAAS Login Module API.

SSL AUTHENTICATION

SSL authentication allows users who have client X.509v3 certificates to authenticate to JAAS, and thus to J2EE applications, using these certificates. SSL authentication uses `mod_ossl` in the Oracle HTTP Server to obtain an authenticated user identity from the client X.509 certificate, as validated through an SSL exchange. This identity can then be provided to Java applications through JAAS.

SSO AUTHENTICATION

SSO authentication allows Java applications to use Oracle9iAS SSO for user authentication. In this case, authenticated user identity is obtained from `mod_osso`, and made available to Java applications through JAAS.

CUSTOM AUTHENTICATION

The Oracle9iAS JAAS provider supports the standard JAAS Login Module API, which allows developers to integrate custom authentication methods into JAAS.

JAAS AUTHORIZATION

In addition to providing a complete role-based access control model for authorization, the Oracle9iAS JAAS implementation provides developers with architectural flexibility when managing authorizations. Choices include managing authorization centrally using LDAP, and through the file system via an XML-based API

LDAP-BASED AUTHORIZATION

Oracle9iAS JAAS user information and authorizations can be stored in OID, Oracle9iAS' scalable, secure LDAP directory. Managing authorizations in LDAP is particularly useful when user communities are large, and scalability and centralized management become critical.

XML-BASED AUTHORIZATION

The Oracle9iAS JAAS provider also supports a fast, lightweight implementation of the authorization API using XML as an encoding mechanism. This API allows Java developers to retrieve user and role information securely from operating system files rather than from OID. This choice is useful for lightweight deployments of Oracle9iAS where scaling to large user communities is not critical.

Note that unlike some JAAS implementations, in which user passwords are stored in unencrypted form, passwords are encrypted when using Oracle9iAS JAAS XML authorization. Moreover, the Oracle implementation provides full support for role-based access control and the Java2 permission model.

JAAS DELEGATION

The Oracle9iAS JAAS provider has support for privilege delegation, allowing a Java application to run with the privileges of a specified user. Both RunAsClient and RunAsID are support. RunAsClient means that a Java application (e.g., enterprise bean, servlet, JSP) can be configured to run with the permissions associated with the current client user. RunAsID means that a bean, servlet or JSP can be configured to run with the permissions associated with a specified user (e.g., run as "DBAdmin"). This allows developers to enforce the principle of least privilege in their

applications, allowing users only those privileges needed to perform a function, since users can only exercise privileges in the context of a well-formed business rule (e.g., an enterprise bean).

SECURITY FOR JAVA APPLICATION ACCESS TO DATABASE

In addition to protecting information exchanged between web clients and Java applications via HTTPS (as described in the section on Oracle9iAS HTTP Server Security) Oracle9iAS can also protect information exchanged between Java applications and back-end databases using the Oracle Advanced Security protocol, and Oracle9i proxy authentication. These features have already been discussed in the section on Oracle HTTP Server security.

ORACLE9*i*AS PORTAL SECURITY

Oracle9*i*AS Portal is a key component of the Oracle product offering in the "enterprise portal" category. Web products in this emerging class provide a gateway to business-related information on corporate intranets. Although it is initially targeted at the enterprise portal market, Oracle9*i*AS Portal can be scaled to provide access to much larger, Internet-scale communities.

Oracle9*i*AS Portal allows Oracle9*i*AS customers to organize their web content and applications, and provide this to users in a logical, consistent web portal format. It also provides a set of tools for creating and managing users and their access to Oracle9*i*AS Portal content.

Enterprise portals, as both a consolidation and extension of existing market spaces, can utilize three powerful components:

- The strong information management technology inherent in the Oracle database
- A wide range of applications which manage critical business data, including enterprise resource planning (ERP), customer relationship management (CRM), and business intelligence (BI) offerings
- A framework (Oracle9*i*AS Portal) which leverages this technology to bring the applications together with other datastores on the intranet.

The functionality built into Oracle9*i*AS Portal provides a common framework across multiple Oracle products and applications. A customer who has purchased portal-enabled Oracle products can easily extend portal integration to other uses in an incremental fashion, as dictated by business needs and priorities.

This section provides an overview of the security features and architecture of the Oracle9*i*AS Portal. It addresses the concept and representation of users and groups, and the relationship of users to database schemas. Then it addresses the issues of authentication, session management and authorization, and the various components of the architecture that implement these features.

ORACLE9*i*AS PORTAL SECURITY OVERVIEW

Oracle9*i*AS Portal provides a secure platform for integrating various applications into a single consolidated portal environment, and provides the management tools and interfaces to administer this environment in an efficient manner.

Oracle9*i*AS Portal provides a comprehensive and extensible authorization model, based on user privileges and groups, for user access to content and applications on the portal. It also provides a flexible integration model for tying applications into the portal authentication framework, allowing them to be deployed as portal, partner, or external applications. Oracle9*i*AS Portal also supports auditing of security-related events through its event logging service.

PORTAL USERS

In the Internet computing model, where millions of users may potentially be accessing a portal, it is important to keep the representation of users as lightweight as possible. To manage large numbers of users, and large amounts of data, in a secure, scaleable, and fault-tolerant way, Oracle9*i*AS Portal leverages the security and data management technology of the Oracle database.

Oracle9*i*AS Portal defines its own user accounts, which are referred to as "lightweight" because they *do not* each have a unique database schema associated with them in the Oracle database. By contrast, each Oracle9*i*AS Portal user account *does* correspond uniquely to an Oracle9*i*AS SSO user account. The Oracle9*i*AS Portal provides the mechanism by which users of Oracle9*i*AS SSO-enabled applications are managed.

INSTALLED USERS

When Oracle9*i*AS Portal is installed, a default set of user accounts is created. These include accounts for the portal administrator and a `public` account. The `public` account allows certain portal content to be publicly accessible; that is, it allows the content to be accessed by users who do not have their own Oracle9iAS Portal user accounts, or by users who do have accounts but who have not yet logged in. By default, two portal administrative accounts are created. One, `portal`, is created for an Oracle9iAS Portal administrator who is also a database administrator (DBA) on the associated Oracle database. The other, `portal_admin`, is for Oracle9*i*AS Portal administrators who do not need DBA privileges.

CREATING USERS

In Oracle9*i*ASv2, Portal has migrated to use of OID for managing users. User management is performed using the new OID DAS framework.

PORTAL GROUPS

Oracle9*i*AS Portal supports groups, which are used for two main purposes. Groups provide a convenient means of granting privileges to a collection of users in one action. In addition, certain attributes in the Portal system can be associated with a group, and if a user has a default group specified in his preferences, then those attributes can be applied to his session. Examples of these attributes include a default home page for a group, or a default style.

Note that groups can be comprised of users, as well as other groups. This allows for construction of hierarchical groups. A user belonging to a subgroup of another group is a member of the parent group, by virtue of group membership. Privileges granted to the parent group are thus assumed by the subgroup user.

CREATING GROUPS

To create a group, a user must have been granted group creation privilege. By default, this privilege is granted to any authenticated user, although it can be restricted to specific users. When a user creates a group, he defines its name, a short description of the group, whether the scope of the group is restricted to a particular content area, and whether or not to hide the existence of the group from other users. He then specifies the group's membership, and may grant it certain privileges, if authorized to do so by the Oracle9*i*AS Portal administrator.

A group can be specified to be private. This means that only users designated as owners of the group will be able to see it in any lists of groups. No other users, including members, will be able to see the group. However, this *private* property does not in any way affect the behavior of the group, such as providing a user privileges through group membership.

PORTAL AUTHENTICATION

The Oracle9*i*AS Portal is implemented as an Oracle9*i*AS SSO partner application for the purpose of user authentication. Note that users who are not authenticated may be allowed access to certain content on Oracle9*i*AS Portal via the `public` user account.

PORTAL AUTHORIZATION

Authorization is the process of controlling access to various areas of the Oracle9*i*AS Portal, based on the identity of the user. Once the user is identified, after going through the process of authentication, the Oracle9*i*AS Portal determines the appropriate authorization of the user based on his identity. Oracle9*i*AS Portal provides an extensible set of privileges which are used to define the access control lists for each object in the portal. The following sections describe this model.

PORTAL PRIVILEGES

Oracle9*i*AS Portal includes a wide range of privileges which can be assigned to users for specific types of access to specific objects (such as web pages or folders) on the portal. Privileges are ranked within a given object type, and higher privileges subsume all privileges below them. Thus, if a user has been directly granted a certain privilege with regard to particular pages, then she also has all the underlying privileges for those pages. This hierarchical or cumulative behavior allows the application to check whether or not a given user has *at least* the permission to perform some action.

There are two types of privileges, *Global Privileges* and *Instance-Specific Privileges*, which provide a convenient mechanism for managing user access to objects in the portal environment.

GLOBAL PRIVILEGES

Global privileges are those which apply across all objects of the specified type. For example, if you are granted the `ANY_PAGE/EDIT` privilege, then you can edit any page in the system, regardless of whether or not you have an explicit, instance-specific grant on that page.

INSTANCE-SPECIFIC PRIVILEGES

Instance-specific privileges are granted to a user or group to define privileges on a specific instance of an object, such as a specific page, or a specific folder or item.

APPLICATION INTEGRATION

One of the key features of Oracle9*i*AS Portal is the ability to integrate various applications into its framework to provide a seamless experience for those using it to access their business applications. Oracle9*i*AS Portal depends on Oracle9*i*AS SSO for this purpose, and supports varying degrees of integration between portal security and application security.

The tightest integration occurs when an application is implemented on the Oracle9*i*AS Portal itself and obtains user identity directly from Oracle9*i*AS Portal. Such applications are called *portal applications*. The Oracle9*i*AS Portal is itself a partner application for Oracle9*i*AS SSO, allowing it to obtain a user's identity from the SSO server. Portal applications obtain a user's identity directly from the Oracle9*i*AS Portal once the user has authenticated via Oracle9*i*AS SSO.

Other applications accessible through Oracle9*i*AS Portal may be Oracle9*i*AS SSO partner applications or external applications, as described in the section on Oracle9*i*AS SSO. Partner applications need not be implemented on the portal itself, but do participate in the Oracle9*i*AS SSO authentication framework which Oracle9*i*AS Portal also uses. External applications maintain their own authentication mechanisms, and do not share directly in the authentication framework provided by Oracle9*i*AS SSO.

SUPPORT FOR HTTPS

For increased security, some installations may require the use of SSL. Both Oracle9*i*AS SSO Server and Oracle9*i*AS Portal can be run in HTTPS mode. Alternatively, for performance reasons it may sometimes be more desirable to have a configuration whereby Oracle9*i*AS SSO Server is run in HTTPS mode, while Oracle9*i*AS Portal is run in HTTP mode.

AUDITING

Besides monitoring activity which could indicate unauthorized system use, auditing of security related events is often the most effective means to ensure that authorized users adhere to system usage policies, and thus "keep honest users honest."

The Oracle9*i*AS Portal has a logging service which logs certain security events and which can be invoked to log arbitrary events defined by portal applications. The reporting features of the Oracle9*i*AS Portal application building capabilities can then be leveraged to view the logged data. Event logging can be used to audit security-relevant events, and to detect possible attempts to undermine system security, or to use the system in a way that is contrary to security policy.

CONCLUSION

Security is a critical concern when deploying web applications. Oracle9*i*ASv2 provides a solid framework for building web applications using the Apache-based Oracle HTTP Server, Oracle's J2EE framework, and Oracle9*i*AS Portal. Oracle9*i*AS security starts from the basic, well-tested and highly configurable web security services provided by Apache, adds a comprehensive set of web single sign-on, directory-based authorization and user management, and Java2 security services, and extends them further with portal security and application integration mechanisms. Oracle9*i*AS also supports secure access to Oracle database systems using Oracle Advanced Security. These features ensure that Oracle9*i*AS is the application server of choice when building and deploying secure web applications.