Oracle Database 10g Release 2 Security

An Oracle White Paper
June 2005

ORACLE

Introduction ... 4
Transparent Data Encryption ... 4
  Application Transparent: No Views Required ... 5
  Transparent Key Management and Separation of Duty ... 6
  Performance ... 7
  Key Management Framework ... 7
  Encryption Algorithms ... 7
  Data Types Supported with TDE ... 7
  Database Encryption API ... 8
Batch jobs: Protection of username and password ... 8
Oracle Database 10g Release 2 Proxy Enhancements ... 8
  Proxy Technology Overview ... 9
    Protocol Support ... 9
    Credential Proxy ... 9
    Application User Proxy Authentication ... 10
Enterprise User Security ... 11
    Enterprise Privilege Administration ... 11
    Shared Schemas ... 12
    Password-Authenticated Enterprise Users ... 13
    Kerberos Based Authentication with Enterprise User Security ... 13
Protecting Sensitive Information At The Row Level ... 14
Oracle Database Row Level Security ... 14
Oracle Virtual Private Database ... 14
  Virtual Private Database Relevant Column Enforcement ... 15
  Virtual Private Database Relevant Column and Masking ... 16
    Partitioned Fine-grained Access Control ... 16
    Global Application Context ... 17
    Externalized Application Context ... 17
Data Classification with Oracle Label Security ... 18
  Identity Management Integration ... 19
  Oracle Policy Manager ... 20
Secure Application Role ... 20
Auditing For Compliance ... 21
    Robust, Comprehensive Auditing ... 21
    Efficient Auditing ... 22
    Customizable Auditing ... 22
    Fine-grained, Extensible Auditing ... 22

Enhanced Administrator Auditing 23
Auditing For Three-Tier Applications 24
Oracle Advanced Security 24
Industry Standard Encryption and Data Integrity 24
Easy Configuration, No Changes to your Applications 25
Strong Authentication Services for Oracle Database 10g 25
Closer Look At Kerberos Authentication for Directory Users 26
Closer Look at RADIUS (Remote Dial-in User Service) 26
PKCS #12 Support 27
PKCS#11 Support, Smart Cards/Hardware Security Modules 27
Industry Standards, Interoperable 27
A Closer Look At PKI 28
Wallets Stored in Oracle Internet Directory 29
Multiple Certificate Support 29
Strong Wallet Encryption 29
SSL 29
Java Security 30
JDBC Security 30
Secure Connections for Virtually Any Client 31
Use of the Secure JDBC Implementation 31
Summary 32

INTRODUCTION

The emergence of identity theft as an organized criminal activity has placed new focus on the need to protect sensitive information from both insiders and those who present fraudulent credentials to gain access to sensitive information

For over 25 years Oracle has delivered state-of-the-art security solutions to government and commercial customers worldwide. Oracle Database 10g Release 2 continues that tradition by introducing powerful new features to address security, privacy and regulatory compliance. The emergence of identity theft as an organized criminal activity has placed new focus on the need to protect sensitive information from both insiders and those who present fraudulent credentials to gain access to sensitive information. While no single security technology will provide absolute security, IT professionals are taking steps to lockdown access to data and closely monitor the enterprise for suspicious activity. Oracle Database 10g Release 2 introduces advanced encryption technology to protect sensitive information and important security features to help application developers build even more secure applications going forward.

TRANSPARENT DATA ENCRYPTION

Transparent data encryption sets the standard for database encryption by tightly coupling encryption with the database to provide a highly transparent encryption solution to protect sensitive data written to disk or backup media.

Perhaps the most important new feature in Oracle Database 10g Release 2 is the addition of transparent data encryption (TDE) to the Oracle Advanced Security option. Oracle can now transparently encrypt data on the network and inside the database. TDE sets the standard for database encryption by tightly coupling encryption with the database to provide a highly transparent encryption solution to protect sensitive data written to disk or backup media. Social security numbers, credit card numbers and other personally identifiable information (PII) can be easily encrypted without breaking the existing application. Most encryption solutions require specific calls to encryption functions within the application code plus the creation of addition views inside the database. This is expensive and time consuming because it requires extensive understanding of an application as well as the ability to write and maintain software. Most organizations don't have the time or expertise to modify existing applications to make calls to encryption routines. In addition, the task of retrofitting an existing application with encryption is manual and error prone. You have to hand code encryption calls everywhere for the application to work.

TDE solves this problem by deeply embedding encryption in the Oracle database. With a simple *alter table* statement an administrator can encrypt sensitive data within an application.

```
SQL>Alter table credit_rating modify(person_id encrypt no salt)
```

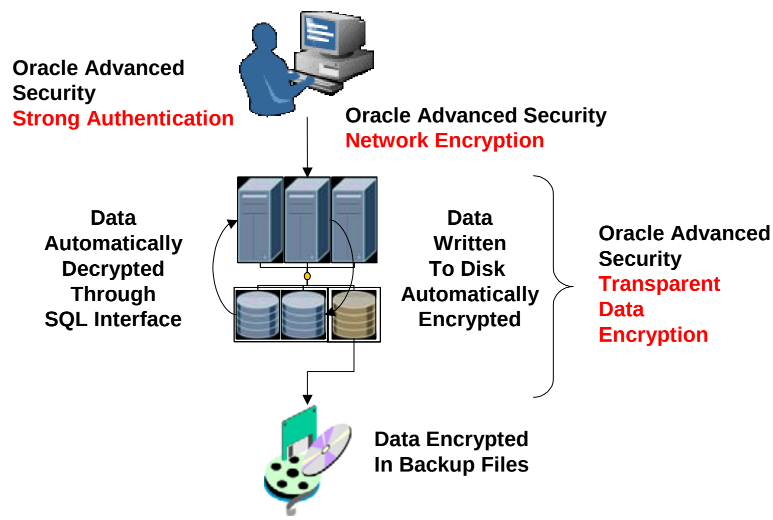
New application tables can simply specify the encrypt key word in the actual table definition.

```
SQL> Create table orders
     (order id number (12) not null,
      customer_id  number(12) not null,
      credit_card  varchar2(19) encrypt));
```

Application Transparent: No Views Required

Application logic performed through SQL will continue to work without modification. In other words, applications can use the same syntax to *insert* data into an application table and the Oracle database will automatically encrypt the data before writing the information to disk. Existing database backup routines will continue to work with the added assurance that sensitive data is encrypted on the backup tapes. Subsequent *select* operations will have the data transparently decrypted so the application will continue to work normally. This is important because existing applications expect to see application data unencrypted.

Credit card numbers, social security numbers and other sensitive information remains encrypted on backup media for regulatory compliance with protection against Identity theft.



TDE doesn't require the creation of database views, database triggers or modification to existing SQL. Tables containing sensitive data simply need to be altered with a single command and the data stored in those tables will be encrypted on disk.

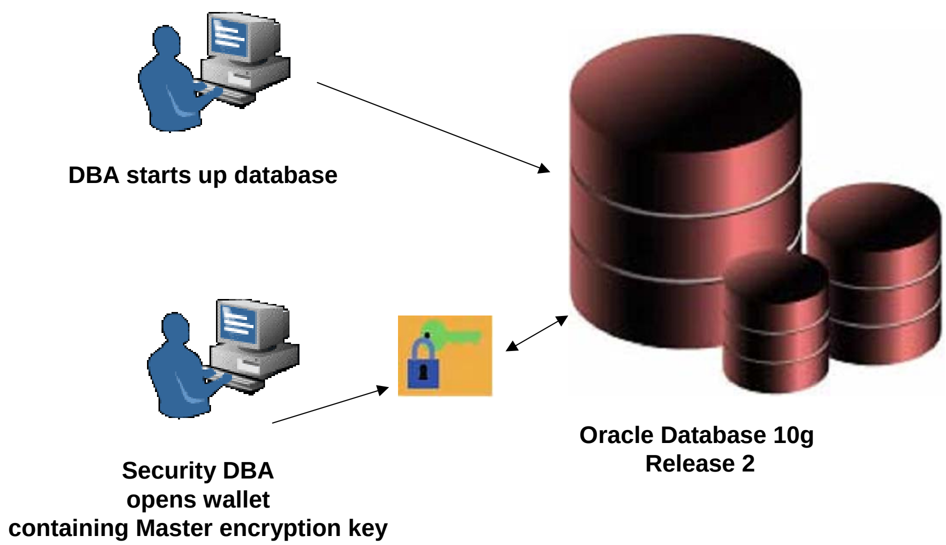
TDE doesn't require the creation of database views, database triggers or modification to existing SQL.

Transparent Key Management and Separation of Duty

Historically one of the most difficult tasks associated with encryption has been key management. Oracle has provided powerful encryption functionality inside the database since Oracle8i. However, that functionality required the application owner or user to manage the encryption key. TDE manages the encryption keys transparently. Encryption works by passing clear text data along with a secret, known as the key, into an encryption program. The encryption program encrypts the clear text data using the supplied key and returns the data encrypted. TDE solves the key management problem by generating and maintaining a master key for the entire database. The database master key is used to encrypt table specific encryption keys transparently created by Oracle. Without the master key, no data can be decrypted within the Oracle database. Upon starting up the Oracle database, an administrator opens an object known as an Oracle *Wallet* using a password. The wallet contains the master key and must be opened before encrypted data is accessed. The password for the wallet can be separate from the system or DBA password. The database master key must be initialized prior to encryption first being used in the database.

```
SQL> alter system set encryption key identified by "1wq!r23t";
```

A separate security DBA can be assigned responsibility for managing the master encryption key.



Performance

Encryption typically creates problems for existing application indexes because the index data isn't encrypted. TDE encrypts the index value associated with a given application table. This means that equality searches within an application will see little to no decrease in performance. For example, assume an application contains a table called CREDIT_RATING with a column named PERSON ID that needs to be encrypted.

```
SQL>Connect appowner

SQL>Alter table credit_rating modify(person_id encrypt no salt)

SQL>Create index person_id_idx on credit_rating (PERSON_ID)

SQL>Select score from credit_rating where PERSON_ID = '235901';
```

In this example, the PERSON_ID values would be encrypted in both the base table and the index. The Oracle database will use the new index even though the PERSON ID value is encrypted in the index.

Key Management Framework

TDE wallet integration has technology built-in which manages the master encryption key. This allows master encryption keys to be updated and encrypted data on backup tapes to be decrypted when loaded back into the Oracle database.

Encryption Algorithms

TDE supports the 3DES and the Advanced Encryption Standard (AES) encryption algorithms for key sizes up to 256 bits.

Data Types Supported with TDE

The first release of TDE will support the following data types. Additional data types will be supported in future releases.

Data Type	Transparent Data Encryption Supported
varchar2	Yes
nvarchar2	Yes
number	Yes
date	Yes
binary float	Yes
binary_double	Yes
timestamp	Yes
raw	Yes
char	Yes
nchar	Yes

Database Encryption API

For customers who wish to manually call encryption routines, Oracle provides the DBMS_CRYPTO toolkit. The DBMS_CRYPTO toolkit was introduced in Oracle Database 10g Release 1 and replaced the DBMS_OBFUSCATION_TOOLKIT introduced in Oracle8i. The DBMS_CRYPTO toolkit is easier to use than the DBMS_OBFUSCATION_TOOLKIT. The DBMS_CRYPTO toolkit supports the 3DES and AES encryption algorithms.

BATCH JOBS: PROTECTION OF USERNAME AND PASSWORD

Oracle Database 10g Release 2 introduces new functionality to dramatically improve the security of batch jobs which use a username and password to connect to the Oracle database. Historically, Oracle Wallets have been used to manage PKI credentials for authentication to the Oracle database. Starting with Oracle Database 10g Release 2, Oracle Wallets can be used to store a username and password. This enables a user to simply use a "/" to connect to the database, but still authenticate using a username and password.

```
sqlplus /@prodapp
```

This new capability will address the issue with exposing username and passwords on unix systems when executing the *ps -ef* command. This new feature is also useful for jobs which need to connect from a middle tier using a username and password and run a batch job within an application schema.

ORACLE DATABASE 10G RELEASE 2 PROXY ENHANCEMENTS

New in Oracle Database 10g Release 2 is the ability to specify a proxy account when connecting to the database using command lines tools such as SQL*Plus.

Oracle Database 10g Release 2 introduces exciting enhancements to the already robust proxy capabilities available with Oracle. The OCI proxy authentication feature was initially released in Oracle8*i*, and allowed a database client to set up, within a single database connection, a number of "lightweight" user sessions, each of which is associated with a different database user. New in Oracle Database 10g Release 2 is the ability to specify a proxy account when connecting to the database using command lines tools such as SQL*Plus. In other words, OCI is no longer a requirement for using the Oracle proxy technology.

```
sqlplus scott[app_schema]/mypassword@prodapp
```

This feature is also integrated with Oracle Enterprise User Security. Using the above example, the user *scott* could be an enterprise user managed in the Oracle Internet Directory. Using the new feature with the new Oracle Wallet capability to store username and password would change the above syntax to the following:

sqlplus [app_schema]/@prodapp

This feature is particularly useful for increasing the security of existing applications. Historically, applications have used the model of one big user to connect to an application. The proxy technology available since Oracle8i enabled applications to change that model. However, the capability wasn't available with command line tools. This new Oracle Database 10g Release 2 feature solves that problem and will enable developers to build even more secure applications.

Proxy Technology Overview

Proxy technology was implemented by Oracle so that a specific middle tier can be restricted to acting on behalf of a specified set of users. Once the middle tier has authenticated itself to the database, it can establish a lightweight session on behalf of those users without submitting user-specific authentication information such as passwords. Moreover, Oracle can be configured so that a specific middle tier can assume a specific set of database roles when acting at the database on behalf of a specific user. In other words, the database uses both middle tier identity and client user identity when determining what privileges to grant a middle tier acting for a user through a lightweight session.

Oracle's proxy authentication feature addresses a number of security problems associated with three-tier systems. Since each middle tier can be delegated ability to authenticate and act on behalf of a specific set of users, and with a specific set of roles, proxy authentication supports a limited trust model for the middle tier server, and avoids the problem of an all-privileged middle tier. It is also possible to give more privilege to a trusted middle tier (e.g., one that is within the corporate firewall) than to a less-trusted middle tier (e.g., one that is outside the firewall and thus more vulnerable to compromise). Moreover, because the identity of both middle tier and user are passed to the database through a lightweight user session, this feature makes it easier to audit the actions of users in a three-tier system, and thus improves accountability.

Protocol Support

Oracle8*i* supported the proxy authentication for communications to the database which used the Oracle Call Interface (OCI), Oracle9i proxy authentication supported "thick" Java Database Connectivity (JDBC) access to the database. Oracle Database 10g supports both "thick" and "thin" access to the database. A middle tier server can now access the Oracle Database 10g on behalf of a client user by establishing a lightweight session for that user through either OCI or JDBC. Oracle Database 10g Release 2 supports proxy connections for standard database connections using tools such as SQL*Plus.

Credential Proxy

Oracle8*i* supported proxy authentication for database users authenticated by password only; the password could be passed as an attribute to be verified by the database, or not, depending on an organization's security preferences.

Oracle Database 10g extends proxy authentication to include the Distinguished Name (DN).

Oracle Database 10g extends proxy authentication to include the Distinguished Name (DN). This provides strong, three-tier security by enabling an SSL credential to be passed to the database for purpose of identifying (but not authenticating) the user. (SSL cannot be used to authenticate a user through multiple tiers, since it is a point-to-point protocol rather than an end-to-end protocol.) For example, a user can authenticate to a middle tier using SSL, the middle tier can extract the DN from the certificate and pass it to the database. As an additional benefit, the DN or certificate is available in the lightweight session and the elements contained therein can be used with Virtual Private Database to limit access. For example, an organization could restrict data access based on the Organizational Unit (OU) element in a user certificate presented to the database.

The database can use the DN or certificate to look up a user in Oracle Internet Directory or other LDAP-based directory certified for Enterprise User Management (an Oracle Advanced Security feature). Integration of proxy authentication with Enterprise User Security enables the user identity to be maintained throughout all tiers of an application, yet the user need only be created once, in the directory. This also enables Enterprise User Security to be used in three-tier applications, instead of merely client-server, as was the case with Oracle8*i*.

Application User Proxy Authentication

Many applications use session pooling to set up a number of sessions which are reused by multiple users. In this context, "application users" are users who are authenticated to the middle tier of an application, but are not known to the database. Oracle introduces application user proxy authentication for these types of applications.

In this model, the middle tier passes a *client identifier* to the database upon session establishment. (The client identifier could be anything that represents the client connecting to the middle tier; a cookie, for example, or an IP address.) The client identifier, representing the application user, is available in user session information and can also be accessed within an application context (using the USERENV naming context), thus enabling applications to use Virtual Private Database to limit user access, even if the application users are not known to the database. Applications can set up and reuse sessions, while still being able to keep track of the "application user" in the session.

Applications can easily reset the client identifier and thus reuse the session for a different user, enabling high performance for web-based applications. For OCI-based connections, alteration of the client identifier is piggybacked on other OCI calls, to further enhance performance.

Application user proxy authentication is particularly valuable in e-business applications with thousands of users, as it supports per-user data access while meeting user scalability requirements.

Application user proxy authentication, available in thin JDBC, thick JDBC and OCI, provides the benefits of connection pooling without the overhead of setting up and managing separate user sessions (even “lightweight” ones), and enables even those applications whose users are unknown to the database to utilize Virtual Private Database. Application user proxy authentication is thus particularly valuable in e-business applications with thousands of users, as it supports per-user data access while meeting user scalability requirements.

ENTERPRISE USER SECURITY

Identity Management is one of the most critical operational components in any IT organization. Most organizations face daunting obstacles in user management. Users within an organization often have far too many user accounts, a problem exacerbated by the growth in web-based self-service applications —every other week, users have a new user account and password to remember. Organizations who want “per user” data access and accountability do not want the administrative challenges of managing users in each database a user accesses.

This problem is compounded for web-facing, e-business applications. An organization opening its mission-critical systems to partners and customers does not want to create an account for each partner in each database the partner accesses, yet “per partner” privilege and “per partner” accountability is highly desired. Oracle Enterprise user security addresses the requirement of per-user data access with centralized user management.

Enterprise Privilege Administration

An inherent challenge of any distributed system, including three-tier systems, is that common application information is often fragmented across the enterprise, leading to data that is redundant, inconsistent, and expensive to manage. Directories are being viewed by an increasing number of Oracle and third-party products as the best mechanism to make enterprise information available to multiple different systems within an enterprise. Directories also make it possible for organizations to access or share certain types of information over the Internet, for example, through a virtual private network. The trend towards directories has been accelerated by the recent growth of the Lightweight Directory Access Protocol (LDAP).

A specific type of enterprise information commonly proposed for storage in a directory is privilege and access control information. Both user privileges, represented as roles, and object constraints, represented as Access Control Lists (ACLs) listing those users who may access an object, may be stored in a directory.

Directory information which specifies users’ privileges or access attributes is sensitive, since unauthorized modification of this information can result in unauthorized granting or denial of privileges or access to users. A directory maintaining information on behalf of the enterprise must ensure that only authorized system security administrators can modify privilege or access information maintained in the directory. Oracle Internet Directory supports attribute-level access control and optional strong user authentication through SSL, and can be configured so that only specific users who are strongly authenticated are allowed to update directory information about user privileges or access.

Oracle8*i* introduced enterprise roles: centrally administered privilege sets, maintained in Oracle Internet Directory. Enterprise roles enable strong, centralized authorization of users. Also, an administrator can add capabilities to enterprise roles (granted to multiple users) without having to update the authorizations of each user independently.

Shared Schemas

The schema-independent user, introduced in Oracle8*i,* extends the benefits of directory integration by allowing the database to delegate administration of user identity, as well as privilege, to the directory. Schema-independent users—also known as users with *shared schemas*—are database users whose identity is maintained in a central LDAP repository; specifically, Oracle Internet Directory. When a schema-independent user connects to the database, the database queries the directory to determine if the user is registered there, and if so, to what database schema the user should be mapped, and what roles the user should obtain.

Schema-independent users reduce the administrative burden associated with managing users in the enterprise.

Suppose, for example, that there are 500 users of an application, who require access to data on several database servers in the enterprise. Instead of maintaining 500 different user accounts on each database, Oracle allows the system administrator to create a single shared schema (such as HRAPPUSER for the HR application), with appropriate privileges, on each database, and then create 500 enterprise users in an Oracle Internet Directory. When they connect to any specific database, these users are mapped to the appropriate schema on the database (e.g. HRAPPUSER), and inherit the privileges associated with the schema, as well as any additional privileges that are associated with the roles granted to them in the directory. Although these users share a common schema, individual users' identities are associated with their sessions by the database, and are used for access control or auditing purposes. Once created, these user accounts in LDAP can be used within multiple applications as well.

The shared schema feature has a number of benefits. It reduces the administrative burden associated with managing users in an enterprise, and allows effective management of much larger communities of users than was previously possible. Moreover, it can provide a mechanism for integrating user account and privilege management across tiers in a multi-tier system, as long as the middle tier also supports management of user identities and privileges in the directory. In such a system, new users and their privileges can be registered once in a directory, and this gives them appropriate access to the middle tier as well as any databases in the enterprise that they need to access. In the future, it should be possible to build three-tier systems (e.g., web storefronts) in which new users can register themselves with a web server, and the web server then creates an entry for these users in the directory, giving them access to information in appropriate databases which pertain to them.

Password-Authenticated Enterprise Users

In Oracle8*i,* Enterprise User Security relied on client-side wallets to authenticate enterprise users. This requires SSL to establish secure channels between (i) the client and the server, and (ii) the database server and an LDAP-compliant directory. The authentication mechanism uses SSL and X.509 v3 certificates, requiring installation of Oracle wallets on both the client and the server.

Although this is a highly effective mechanism to ensure the integrity of the user authentication process, it requires SSL configuration and client-side wallets. Because this requires an X.509 certificate issued by a trusted Certificate Authority for each enterprise user, overhead can be significant for large organizations. Both SSL and an Oracle wallet must be installed on both the client and the server. This is a backwards-compatibility issue for certain earlier releases, and adds complexity to the setup and configuration process.

In Oracle Database 10g, enterprise users can use password-based authentication, removing the requirement for client-side wallets and most Secure Socket Layer (SSL) processing.

In Oracle Database 10g, enterprise users can use password-based authentication, removing the requirement for client-side wallets and most Secure Socket Layer (SSL) processing. Furthermore, enterprise users can use a single enterprise username and password to connect to multiple databases, if desired. In addition, Oracle provides a User Migration Utility for use by an administrator to migrate users from multiple, independent databases to one central LDAP directory service for centralized user and privilege management. With its reduced processing overhead, improved ease-of-use, and simplified setup and administration, this release is particularly useful for large user communities accessing multiple applications.

Kerberos Based Authentication with Enterprise User Security

Oracle Database 10g Release 1 enterprise user security supports kerberos as an additional authentication mechanism. The addition of kerberos gives customers three options when using enterprise user security - SSL, password or kerberos based authentication.

PROTECTING SENSITIVE INFORMATION AT THE ROW LEVEL

Privacy and compliance requirements are increasing the need for row level security because more information is being stored in a single location.

Database objects include the tables and views which store and limit access to application data. A typical database application may contain dozens, hundreds or even thousands of database tables. Access to these tables is mediated using database object privileges such as SELECT, UPDATE, INSERT and DELETE. Object privileges can be granted directly to an application user or managed through enterprise roles. Roles contain the object privileges necessary to perform a specific job function. Oracle has robust support for roles, allowing application developers to break down access privileges into a least privilege model. Application users can have many roles active depending on their job responsibility. For example, the object privilege SELECT might be given to the HR_USER role. In most cases object privileges are sufficient to satisfy stated security policies. For example, a user can be denied access to purchase order information by simply ensuring the user does not the role containing access to the underlying purchase order application tables. However, in today's complex Internet connected world, object privileges sometimes aren't sufficient for controlling access. For example, data consolidation typically involves moving data from multiple databases into a single database. This may result in data from different organizations or companies being consolidated into the same database object. Object privileges stop at the object level and don't drill down to the row level or individual data element. Row level security is the ability to control access to individual rows within a database table after an application user has been given object privileges on the database table. This type of access control is difficult to implement programmatically and increases application complexity.

ORACLE DATABASE ROW LEVEL SECURITY

Oracle8*i* set a new standard in database security with the introduction of Oracle Label Security and Virtual Private Database (VPD). Oracle Database 10g introduces exciting new enhancements to both Oracle Label Security and Virtual Private Database. Oracle Database 10g allows Oracle Label Security policies to be managed in the Oracle Identity Management infrastructure. Oracle Database 10g Virtual Private Database introduces column relevant security policy enforcement and optional column masking. These features provide tremendous flexibility for meeting privacy mandates and other regulations.

ORACLE VIRTUAL PRIVATE DATABASE

Oracle Virtual Private Database has been evaluated twice (Versions 8i and 9.2) using the International Common Criteria at EAL4

Virtual Private Database was introduced in Oracle8i and includes programmable row level security and secure application context. Virtual Private Database enables a developer or DBA to attach a security policy to an application table, view or synonym. The security policy is invoked when SQL statements access the object associated with the policy. Oracle secure application context can be used in conjunction with the security policy to determine how to apply the policy. The security policy is written using PL/SQL.

Security is stronger, because it is enforced by the database, no matter how a user accesses data, this addresses the application security bypass problem.

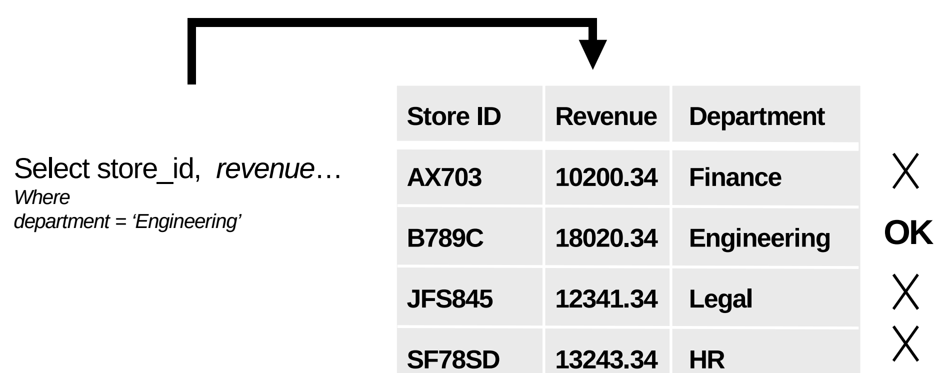
Within the enterprise, usage of Virtual Private Database can result in lower cost of ownership in deploying applications. Security can be built once, in the database, rather than in each application that accesses the data. Security is stronger, because it is enforced by the database, no matter how a user accesses data. Virtual Private Database is a key enabling technology for organizations building hosted, web-based applications, as well as for Oracle itself. It also address a common problem with application based logic and that is called the *application bypass* problem. The application bypass problem refers to situations where a user attempts to access data using an tool other than the application which is normally used to access data. In these situations data may be less secure because the security logic was in a single data access path and not bound to the data itself.

Direct or indirect access to a table with an attached security policy causes the database to consult a function implementing the policy. The policy function returns an access condition known as a predicate (a WHERE clause) which the database appends to the user's SQL statement, thus dynamically modifying the user's data access. A secure application context enables access conditions to be based on virtually any attributes an application deems significant, such as organization, cost center, account number, or position.

For example, an Web order entry system can enforce access based on customer number, and whether the user is a customer or a sales representative. In this way, customers can view their order status online (but only for their own orders), while sales representatives can view multiple orders, but only for the their own customers.

Virtual Private Database Relevant Column Enforcement

Oracle Database 10g allows Virtual Private Database policies to be associated with specific columns in application tables. Only when those columns are referenced is the policy invoked.



This option tells the database to return all rows regardless of the policy restriction, but mask, or null out, the values

Virtual Private Database Relevant Column and Masking

In addition to relevant column enforcement, Oracle Database 10g introduced a new enforcement option for policies applied to columns. This option tells the database to return all rows regardless of the policy restriction, but mask, or null out, the values for column in the rows which didn't meet the policies restrictions. For example, assume a virtual private database policy is written using PL/SQL and assigned to the *revenue* column. A SQL statement is submitted to the Oracle Database 10g and attempts to retrieve the *revenue* for all *departments*. The policy is enforced because the SQL statement references the revenue column. In Oracle8i and Oracle9i only rows matching the department *'Engineering'* would have been returned. In oracle Database 10g, the policy can be applied such that all rows are returned, but the column values in the *Revenue* column are masked for rows which don't match the department '*Engineering*'.

Select *revenue*….
Where
department = 'Engineering'

Store ID	Revenue	Department	
AX703		Finance	OK
B789C	18020.34	Engineering	OK
JFS845		Legal	OK
SF78SD		HR	OK

Partitioned Fine-grained Access Control

Oracle9i introduced the ability to partition security policy enforcement by application, thus facilitating Virtual Private Database deployment. For example, suppose both an Order Entry and Inventory application access the Orders table. The Order Entry application limits access based on customer number, while the Inventory application limits access based on part number. It is very useful to be able to "partition" fine-grained access control so that *different* security policies apply, depending upon which application is accessing the data. Otherwise, application developers of the respective Order Entry and Inventory applications have to agree upon a mutual policy, which may not be feasible or possible.

Oracle enables partitioning of Virtual Private Database through policy groups and a driving application context. A driving application context securely determines which application is accessing data, and policy groups facilitate managing the policies which apply by application. Oracle also supports default policy groups, which always apply to data access. For example, an application "striped" for application hosting using a subscriber ID could have a default policy, "Subscriber," that always enforces data separation by subscriber, and additional policy groups for

Inventory and Order Entry-based access, which apply depending on the particular application accessing data.

Partitioned application context facilitates the development of applications using Virtual Private Database, because applications can have different security policies based upon their individual application needs.

Global Application Context

In order to support hundreds of thousands of users, many web-based applications use connection pooling to achieve the required high scalability. These applications set up and reuse connections rather than create different sessions for each user. For example, two web users, Diane and Sanjay, connect to a middle tier application, which establishes a session in the database used by the application. The application is responsible for switching the username on the connection, so that, at any given time, it's either Diane or Sanjay using the session.

Global application contexts allow multiple sessions to share the same security attributes.

Oracle Virtual Private Database capabilities facilitate connection pooling by allowing multiple connections to access one or more *global* application contexts, instead of setting up an application context for each user session. Global application contexts provide additional flexibility for web-based applications to use Virtual Private Database, as well as enhanced performance through reuse of common application contexts among multiple sessions instead of setting up per-session application contexts.

Application user proxy authentication can be used with global application context for additional flexibility and high performance in building e-business applications. For example, suppose a web-based application that provides information to business partners has three types of users: Gold, Silver, and Bronze, representing different levels of information available. Instead of each user having his own session — with individual application contexts — set up, the application could set up global application contexts for Gold, Silver or Bronze and use the client identifier to point the session at the correct context, in order to retrieve the appropriate type of data. The application need only initialize the three global contexts once, and use the client identifier to access the correct application context to limit data access. This provides performance improvements through session reuse, and through accessing global application contexts set up once, instead of having to initialize application contexts for each session individually.

Externalized Application Context

Virtual Private Database does allow simple security attributes to be initialized from attributes stored in Oracle Internet Directory. The ability to identify attributes in Oracle Internet Directory that can be used for initialization of an application context further enhances the ability of organizations to leverage directory-based user management and derive a lower cost of ownership. For example, an Order Entry application context could be initialized externally by populating "cost center" attributes automatically, based on corresponding attributes defined for a user in Oracle Internet Directory. The ability to predefine "externally initialized" application contexts reduces the cost of development, since developers do not need to write LDAP calls to retrieve attributes from a directory into an application context. This also avoids duplication of data in both a database and a directory, by enabling Virtual Private Database to use attributes stored in Oracle Internet Directory for row level security.

In addition, Oracle Database 10g Release 1 introduced a new parameter with which helps increase the performance of VPD. This new option allows three parameters to be added to the "apply_policy" command: Static, context sensitive and dynamic

Static: When the predicate (WHERE-clause) does not change (for example "WHERE 9 a.m. < sysdate < 6 p.m."), the policy is only executed once (even if the result of the query varies depending on the time the statement is issued) and stored in memory, which greatly enhances performance.

Context Sensitive: This is used when the predicate changes infrequently, for example when another user logs on: "select * from orders WHERE customer_id in (select cust_id from customers where last_name = (sys_context('userenv','user_name'))); As long as one customer queries the orders table, the policy does not need to be executed each time, since it doesn't change. Only when another customer logs on and queries the same table, which of course results in different rows being returned, the policy is executed again. Depending on the frequency of changes, this is a significant performance enhancement.

Dynamic: For compatibility reasons, this is the default when not specified. The policy is executed each time a protected objects is accessed.

DATA CLASSIFICATION WITH ORACLE LABEL SECURITY

Oracle Label Security helps customers address regulatory compliance challenges by combining data classification technology with the concept of a user security clearance to enable strong access controls in the Oracle database.

Unlike Virtual Private Database where the DBA or developer writes the security policy using PL/SQL, Oracle Label Security provides a security policy and data dictionary for managing access to data using data classification and security clearances. Oracle Label Security helps customers address regulatory compliance challenges by combining data classification technology with the concept of a user security clearance to enable strong access controls in the Oracle database. Data classification labels, also referred to as sensitivity labels, can be assigned to individual rows in an application table, enabling row level security, or used within program units to assign classification labels to application modules. User security clearances and data classification labels can be used throughout the enterprise to make access control decisions.

Oracle Label Security has been evaluated twice (versions 8.1.7 & 9.2) using the International Common Criteria at EAL4.

Data classification labels are what determine an application user's ability to view and update application data. Data classification labels provide sophisticated controls which are not possible with traditional object level privileges. For example, suppose an order entry application security policy states that the application must be capable of limiting access to purchase orders labeled *Sensitive*? By default, giving an application user the SELECT privilege on the purchase orders table will allow the user to view all information. One approach to solving this requirement is to create two database views. The first view will exclude all the purchase orders deemed company sensitive and the second will include all the purchase orders. This approach is problematic because the security policy may change to include new levels of sensitivity. In addition, application users will need to be assigned the correct enterprise role depending on their authorization to view company sensitive information.

Data classification labels solve this security requirement and eliminate the need for additional views. Oracle Label Security data classification labels can contain three components: a single hierarchical level or classification, one or more horizontal compartments or categories and one or more groups.

Oracle Label Security Authorizations
Sensitive

Application Table

Case No.	Location	Department	Sensitivity Label	
AX703	Chicago	Finance	Unclassified	OK
B789C	Dallas	Engineering	Sensitive	OK
JFS845	Chicago	Legal	Highly Sensitive	X
SF78SD	Miami	Human Resource	Confidential: HR	X

Identity Management Integration

Managing Oracle Label Security in Oracle Identity Management provides an easy way to extend security clearances to the entire enterprise.

Oracle Database 10g allows Oracle Label Security policies to be centrally created in the Oracle Identity Management infrastructure. Leveraging the Oracle Internet Directory, the Oracle Label Security policies are created in a central location for the entire enterprise. This simplifies provisioning and administration of security across all databases in the enterprise or GRID. Organizational sensitivity labels and application user security clearances can be managed in one location.

If an application user needs additional access, their security clearance can be raised and the information will be automatically propagated to all registered databases in the enterprise. If an employee changes organizations, takes a leave of absence or is terminated, their access can be disabled from one location.

Oracle Policy Manager

Oracle Policy Manager is an administration tool for both Oracle Label Security and Oracle Virtual Private Database. Oracle Label Security policies can be managed using Oracle Policy Manager by connecting as the user LBACSYS or another user with appropriate privileges.

SECURE APPLICATION ROLE

Secure Application Roles can be used to restrict access to roles granted to a user based on a policy written in PL/SQL.

A long-standing security problem has been that of limiting how users access data, to prevent users from bypassing application logic to access data directly. For example, in web-based applications, even if users are known to the database, it may not be desirable to allow them to have direct access to data. To-date, this has been a very difficult security problem to solve, because there has been no secure way to validate which application is used to access data — e.g. a malicious user could write a program that *appears* to be a valid human resources application, but, in fact, is not.

Oracle addresses this challenge through a secure application role: a role invoked by a package. The package can perform any desired validation to ensure that the appropriate conditions are met before the user can exercise privileges granted to the role in the database. The database ensures that it is only the trusted package implementing the role that determines the correct access conditions. The set role operation is performed within the package.

In three-tier systems using proxy authentication, the package can validate that the user session was created by a middle tier, and thus that the user is accessing the database through the correct application. The secure application role can also ensure that a user connecting directly to the database is not able to access any data. A secure application role can enforce other security conditions, as well; for example, the user may not be allowed to access especially sensitive human resources data from the Internet.

A secure application role enhances the native strong authentication and row level security features of the database to prevent users from assuming any privileges unless the correct access conditions are met. Secure application role solves a very difficult security issue and help address web based access control issues.

```
sqlplus appsecofficer/mypassword

SQL> create role hr_actions identified using hr_sec_package;

SQL> grant select, insert, update on hr_workflow to hr_actions;

SQL> grant hr_actions to jsmith;

sqlplus jsmith/mypassword

SQL> set role hr_actions

ERROR at line 1:
ORA-28201: Not enough privileges to enable application role
'HR_ACTIONS'
SQL> execute hr_sec_package;
PL/SQL procedure successfully completed.
```

AUDITING FOR COMPLIANCE

Oracle standard and fine grained auditing can help address internal controls mandated by such regulations as Sarbanes-Oxley.

A critical aspect of any security policy is maintaining a record of system activity to ensure that users are held accountable for their actions. Auditing helps deter unauthorized user behavior which may not otherwise be prevented. It is particularly useful for ensuring that authorized system users do not abuse their privileges. Oracle builds upon the existing robust and comprehensive auditing capabilities of the database to include fine-grained auditing, that can serve as an “early warning system” of users misusing data access privileges, as well as an intrusion detection system for the database itself.

Oracle standard and fine grained auditing can help address internal controls mandated by such regulations as Sarbanes-Oxley.

Robust, Comprehensive Auditing

The Oracle audit facility allows businesses to audit database activity by statement, by use of system privilege, by object, or by user. For example, one can audit activity as general as all user connections to the database, and as specific as a particular user creating a table. One can also audit only successful operations, or unsuccessful operations. For example, auditing unsuccessful SELECT statements may catch users on “fishing expeditions” for data they are not privileged to see. Audit trail records can be stored in an Oracle table, making the information available for viewing through ad hoc queries or any appropriate application or tool, or combined with operating system audit trails on selected operating systems, for ease of management.

Efficient Auditing

Oracle implements auditing efficiently: statements are parsed once for both execution and auditing, not separately. Also, auditing is implemented within the server itself, not in a separate, add-on server which may be remotely situated from the statements which are being executed (thereby incurring network overhead). The granularity and scope of these audit options allow Oracle customers to record and monitor specific database activity without incurring the performance overhead that more general auditing entails. And, by setting just the options of interest, Oracle customers can avoid "catch-all, and throw away" audit methods which intercept and log all statements, and then filter them to retrieve the ones of interest.

Customizable Auditing

To record customized information that is not automatically included in audit records, Oracle can use triggers to further customize auditing conditions and audit record contents. Database triggers are user-defined sets of PL/SQL or Java statements, stored in compiled form. While users explicitly execute stored procedures, database triggers are automatically executed (or "fired") within the data server based on pre-specified events. A trigger is defined to execute either before or after an INSERT, UPDATE or DELETE, so that when that operation is performed on that table, the trigger automatically fires. For example, one could define a trigger on the EMP table to generate an audit record whenever an employee's salary is increased by more than 10 percent and include selected information, such as before and after values of SALARY.

Fine-grained, Extensible Auditing

Oracle introduces extensible, fine-grained auditing, that can alert administrators to misuse of legitimate data access rights as well as serving as an intrusion detection system for the database.

Oracle expands upon the existing robust, granular auditing capabilities of the database by introducing extensible, fine-grained auditing. Fine-grained auditing enables organizations to define specific audit policies that can alert administrators to misuse of legitimate data access rights.

Fine-grained auditing allows organizations to define audit policies, which specify the data access conditions that trigger the audit event, and use a flexible event handler to notify administrators that the triggering event has occurred. For example, an organization may allow HR clerks to access employee salary information, but audits access when salaries greater than $500K are accessed. The audit policy ("where SALARY > 500000") is applied to the EMPLOYEES table through an audit policy interface (a PL/SQL package). Oracle Database 10g extends support for Fine-gained auditing to INSERT, UPDATE and DELETE statements.

For additional flexibility in implementation, organizations can employ a user-defined function to determine the policy condition, and identify a relevant column for auditing ("audit column"). For example, the function could allow unaudited access to *any* salary as long as the user is accessing data within the intranet, but audit access to executive-level salaries when they are accessed from the Internet. An audit column helps reduce the instances of false or unnecessary audit records, because the audit need only be triggered when a particular column is referenced in the query. For example, an organization may only wish to audit executive salary access when an employee name is accessed, because accessing salary information alone is not meaningful unless an HR clerk also selects the corresponding employee name.

Oracle captures the exact SQL text of the statement the user executed in audit tables.

Oracle captures the exact SQL text of the statement the user executed in audit tables. In conjunction with other database features such as Flashback Query, fine-grained auditing can be used to recreate the exact records returned to a user. This may be especially important to organizations who have especially sensitive information they wish to share, for which they require strict accountability. For example, many law enforcement organizations at the international, federal, state and local level are increasingly becoming "e-businesses" by sharing information among themselves, yet it is more important than ever that they audit access to sensitive information, such as informant data, to know who accessed what *exact* data.

The event handler provides organizations with flexibility in determining how to handle a triggering audit event. A triggering audit event could be written into a special audit table for further analysis, or could activate a pager for the security administrator. The event handler allows organizations to fine-tune their audit response to appropriate levels of escalation.

Fine-grained auditing enables organizations to hone their auditing capabilities to capture and identify particular, specific data access of concern. In addition to providing more granular, targeted audit information, such as detecting misuse of legitimate access, fine-grained auditing can also serve as an intrusion detection facility for the Oracle Database 10g itself.

Enhanced Administrator Auditing

The Oracle database uses redo logs to record all changes made in the database. Redo logs provide recovery from an instance or media failure. Oracle applies the appropriate changes in the database's redo log to the data files, which update database data the instant that the failure occurred. The ability to capture all system activities in logs, in fact, has security benefits as well. The activities of all users—from the most privileged to the least—are captured in these logs.

Audit trails complement redo logs with their ability to hold users accountable for any and all actions taken against the database. Because audit logs are stored in the SYS schema, however, auditors who need to hold accountable users connected as SYSDBA or SYSOPER have a bootstrap issue. Oracle further strengthens auditing by specifically auditing users connected through SYSDBA and SYSOPER, and recording the audit trail on the operating system. As long as the auditor has root on the operating system, and the database administrator does not, customers can separate the function of the database administrator and the auditor.

This auditing feature benefits e-business customers in multiple ways. It facilitates the ability to track SYS operations and investigate suspicious activities, which is especially important because this user has numerous privileges. It enables e-businesses with strict auditing requirements, particularly banks and other financial services companies, to separate the function of the database administrator from the auditor.

Auditing For Three-Tier Applications

Many three-tier applications authenticate users to the middle tier, then the transaction processing monitor or application server connects as super-privileged user, and does all activity on behalf of all users. With Oracle, customers are not only able to preserve the identity of the real client over the middle tier and enforce “least privilege” through a middle tier, but can also audit actions taken on behalf of the user by the middle tier. Oracle’s audit records capture both the logged-in user (e.g., the middle tier) who initiated the connection, and the user on whose behalf an action is taken. Auditing user activity, whether users are connected through a middle tier or directly to the data server, enhances user accountability, and thus the overall security of multi-tier systems.

ORACLE ADVANCED SECURITY

Oracle Advanced Security protects privacy and confidentiality of data over the network by eliminating data sniffing, data loss, replay and person-in-the-middle attacks. All communication with an Oracle Database can be encrypted with Oracle Advanced Security. In addition, Oracle Advanced Security TDE encrypts data inside the database. Databases contain extremely sensitive information and restricting access by strong authentication is one of first lines of defense. Oracle Advanced Security provides strong authentication solutions leveraging a business’s existing security framework including Kerberos, Public Key Cryptography, RADIUS and DCE for Oracle Database 10g.

Industry Standard Encryption and Data Integrity

Oracle Advanced Security protects all communications to and from the Oracle Database. Businesses have a choice between using Oracle Advanced Security’s native encryption/data integrity algorithms and SSL to protect data over the network. Some of the typical scenarios requiring network level encryption include:

- Database Server is a behind a firewall and users access the server via client server applications
- Communication between the application server in a DMZ and the Database which is in behind a second firewall must be encrypted

Native Encryption and Data Integrity algorithms in Oracle Advanced Security require no PKI deployment. With each subsequent release of the database, newer encryption algorithms are included as they gain industry approval. The latest addition is the Advanced Encryption Standard (AES), an algorithm improved in security and performance over DES. The complete list of Encryption and Data integrity algorithms are

- AES (128, 192 and 256 Key)
- RC4 (40, 56, 128, 256 Key)
- 3DES (2 Key and 3 Key)
- MD5
- SHA1

SSL based encryption is available for businesses that have elected to provide Public Key Infrastructure to their IT deployments. New in the Oracle Advanced Security 10g release is the support for TLS 1.0 protocol. Oracle Advanced Security provides AES cipher suites with the TLS 1.0 protocol in Oracle Database 10g.

Easy Configuration, No Changes to your Applications

Configuring the network parameters for the server and/or client enables the network encryption/integrity function. Most businesses can therefore easily uptake this technology as there are no changes required in the application.

Strong Authentication Services for Oracle Database 10g

Unauthorized access to information is a very old problem. Business decisions today are driven by information gathered from mining terabytes of data. Protecting sensitive information is key to a business's ability to remain competitive. Access to key data repositories such as the Oracle Database 10g that house valuable information can be granted once users are identified and authenticated accurately. Verifying user identity involves collecting more information than the usual username and password. Oracle Advanced Security provides the ability for businesses to leverage their existing security infrastructures such as Kerberos, Public Key Infrastructure (PKI), RADIUS and Distributed Computing Environment (DCE) for strong authentication services to the Oracle Database 10g.

New in this release is the ability to check X509v3 certificate revocations using Certificate Revocation Lists stored in the file system, Oracle Internet Directory or using CRL Distribution Points.

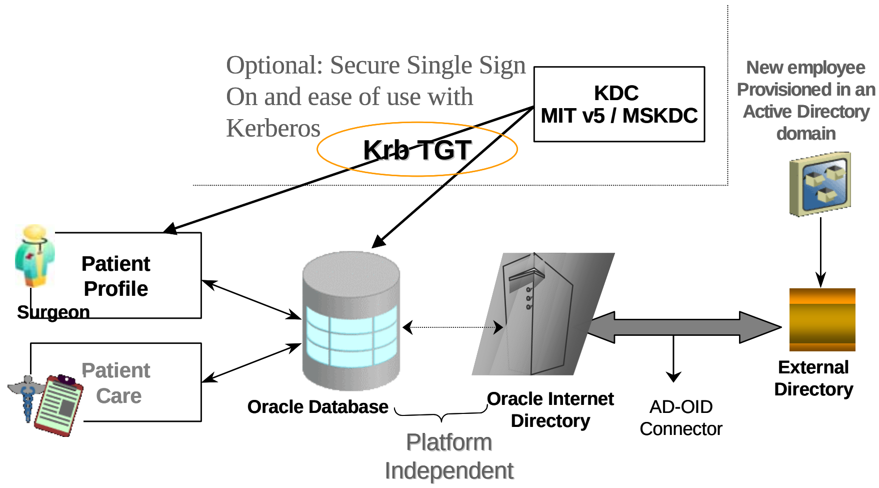
The ability for Oracle Database Servers or Database Clients /Users to use PKI Credentials stored in Smart Cards or other Hardware Storage Modules using industry's PKCS 11 standard. This is especially useful for users as it provides roaming access to the database via client server applications or web applications. Storing server credentials in a hardware module provides an additional level of security that some deployments require.

Kerberos integration for enterprise users in Oracle 10g eliminates the security versus Usability debate. Kerberos Authentication and SSL communication require the Oracle Database 10g Oracle Advanced Security option

Closer Look At Kerberos Authentication for Directory Users

This feature is new in Oracle Database 10g Advanced Security. For organizations that have shied away from Single Sign On with passwords, this feature provides security with usability as shown in figure below. Once an Oracle database is registered with a Kerberos Server and configured to support a Kerberos Service, enterprise users can authenticate to the database without any additional complications. Organizations that are already using a Kerberos Server and Oracle Advanced Security's Kerberos adapter can migrate their external database users to the directory to benefit from centralized user management. User Migration Utility assists in the migration task.

If the user is managed in a third party directory such as Active Directory, the Directory Integration and Provisioning Service must synchronize, in addition to other attributes, the user's Kerberos Principal into Oracle Internet Directory. Following is an illustration of this key benefit. In this scenario, a user is provisioned by an HR application into an Active Directory Domain for instance. The user is a member of a group in AD. The user along with his group membership is synchronized into Oracle Internet Directory by the AD-OID connector. When appropriate database roles are assigned to the OID group, members within that group have access to the database objects.



Closer Look at RADIUS (Remote Dial-in User Service)

RADIUS (RFC #2138) is a distributed system that secures remote access to network services and has long been established as an industry standard for remote and controlled access to networks. RADIUS user credentials and access information are defined in the RADIUS server to enable this external server to perform authentication, authorization and accounting services when requested.

ORACLE RADIUS support is an implementation of the RADIUS Client protocols that enables database to provide authentication, authorization and accounting for

RADIUS users. It sends authentication requests to RADIUS server and acts upon the server's responses. The authentication can occur either in synchronous or asynchronous authentication modes and is part of Oracle configuration for RADIUS support.

Oracle Advanced Security provides authentication, respects authorizations stored in RADIUS and basic accounting services to RADIUS users when accessing the Oracle database.

PKCS #12 Support

PKCS #12 support provides interoperability with third-party applications including browsers.

Oracle Advanced Security supports X.509 certificates stored in PKCS #12 containers, making the Oracle wallet interoperable with third party applications like Netscape Communicator 4.x and Microsoft Internet Explorer 5.x, and providing wallet portability across operating systems. Users who have existing PKI credentials may export them in PKCS#12 format and reuse them in Oracle Wallet Manager, and vice versa. PKCS#12 thus increases interoperability and reduces the cost of PKI deployment for organizations.

PKCS#11 Support, Smart Cards/Hardware Security Modules

Oracle Advanced Security supports the PKCS#11 interface.

An Oracle Wallet is a software container that holds the private key and other trust points of the certificate. Oracle Advanced Security supports the support PKCS#11 industry standard. This allows the private keys that were previously stored on the file system to be created and stored in secure devices such as Hardware Security Modules or Smart Cards that are available in the market. Oracle Advanced Security 10g Release 2 provides deeper integration with PKCS#11 enabling interoperability with pre-provisioned certificates on hardware devices.

Industry Standards, Interoperable

Oracle Advanced Security's SSL client can be used in any PKI that is industry standards compliant. For instance, certificates issued by Verisign, Thawte, RSA Keon and Oracle Certificate Authority can be used for authentication to Oracle Database 10g as they accept standard PKCS#7 certificate requests and issue X509v3 certificates. Oracle Advanced Security's provides an Entrust adapter that allows business applications to leverage Entrust's PKI with Oracle Database 10g.

Oracle Advanced Security includes a Kerberos client is compatible with a Kerberos v5 ticket that is issued by any MIT v5 compliant Kerberos server or Microsoft KDC. Businesses can continue to operate in a heterogeneous environment using Oracle Advanced Security's Kerberos solution.

Oracle Advanced Security provides a RADIUS client that allows Oracle Database 10g to respect the authentication and authorizations asserted by a RADIUS server. This feature is especially useful for businesses that are interested in two-factor authentication that establishes your identity based on what you know (password or PIN information) and what you have (the token card) provided by some token card manufacturers.

Oracle Advanced Security has supported SSL since Oracle Release 8*i*. New in Oracle Advanced Security 10g is TLS1.0 support, Smart Card support for Oracle Wallets and URL and LDAP support for certificate validation.

The new industry SSL protocol standard, TLS 1.0 is support with Oracle Advanced Security 10g. While TLS 1.0 is based on SSL 3.0, the more tangible benefits for Oracle users using TLS 1.0 are

- Improved efficiencies for CPU intensive cryptographic operations resulting in increased SSL based throughput
- Improved TLS Handshake Protocol that provides increased privacy and integrity for peer-to-peer communication

Oracle Wallet Manager continues to be the tool to use for certificate requests and other certificate management tasks for the end user. Additional command line utilities that assist in managing Certificate Revocation Lists (CRLs) and other Oracle Wallet operations are also available in this release.

Certification Revocation Lists published to an LDAP server, a file system or a URL are supported by Oracle's SSL infrastructure.

A Closer Look At PKI

Public Key Infrastructure (PKI) encompasses technologies, policies and procedures for authentication based on the principles of public key cryptography. Public Key Infrastructure (PKI) has emerged as the authentication technology which is most appropriate for securing Internet and e-commerce applications. There are a number of reasons for this. First, PKI is highly scaleable. Since users maintain their own certificates, and certificate authentication involves exchange of data between client and server only (i.e., no third party authentication server needs to be online), there is no limit to the number of users which can be supported using PKI. Moreover, PKI allows delegated trust. A user who has obtained a certificate from a recognized and trusted Certificate Authority (CA) can authenticate himself to a server the very first time he connects to that server, without that user having previously been registered with the system.

Oracle supports standard X.509v3 certificates and relevant Public Key Certificate Standards (PKCS) for certificate request and installation. This allows users to request certificates from any CA supporting these standards. It also allows users to install trusted root certificates from their choice of CA's, allowing the server to recognize and validate certificates issued by those CA's.

Oracle Database 10g Release 1 expanded PKI integration and interoperability through:

- PKCS#11 support
- Wallet storage in Oracle Internet Directory
- Multiple certificates per wallet
- Strong wallet encryption

Wallets Stored in Oracle Internet Directory

Oracle Enterprise Security Manager creates user wallets as part of the user enrollment process. The wallet is stored in Oracle Internet Directory, or other LDAP-compliant directory. Oracle Wallet Manager can upload wallets to—and retrieve them from—the LDAP directory.

Storing the wallet in a centralized LDAP-compliant directory lets users access them from multiple locations.

Storing the wallet in a centralized LDAP-compliant directory supports user roaming, allowing users to access their credentials from multiple locations or devices, ensuring consistent and reliable user authentication, while providing centralized wallet management throughout the wallet life cycle.

Multiple Certificate Support

Oracle Wallets support multiple certificates per wallet, including:

- S/MIME signing certificate
- S/MIME encryption certificate
- Code-signing certificate

Oracle Wallet Manager Version 3.0 supports multiple certificates for a single digital entity in a persona—with multiple private key pairs in a persona (each private key can match only one certificate). This enables consolidation of and more secure management of users' PKI credentials.

Strong Wallet Encryption

The private keys associated with X.509 certificates require strong encryption, over secure channels. Oracle replaces DES encryption with 3-key triple DES (3DES), which is a substantially stronger encryption algorithm and provides strong security for Oracle wallets.

SSL

Oracle implements the SSL protocol for encryption of data exchanged between database clients and the database. This includes data in Oracle Net Services (formerly known as Net8), LDAP, thick JDBC, and IIOP format. SSL encryption provides users with an alternative to the native Oracle Net Services encryption protocol which has been supported in Oracle Advanced Security (formerly known as Advanced Networking Option) since Oracle7. A benefit of SSL is that it is a de facto Internet standard, and can be used with clients using protocols other than Oracle Net Services.

In a three-tier system, SSL support in the database means that data exchanged between the middle tier and the database can be encrypted using SSL. The SSL protocol has gained confidence of users, and it is perhaps the most widely-deployed and well-understood encryption protocol in use today. Oracle's implementation of SSL supports the three standard modes of authentication, including anonymous (Diffie-Hellman), server-only authentication using X.509 certificates, and mutual (client-server) authentication with X.509.

Oracle Application Server also supports SSL encryption between thin clients and the Oracle Application Server, as well as between Oracle Application Server and Oracle Data Server. As in Oracle, anonymous, server-only, and client-server authentication via X.509 are supported.

SSL addresses the problem of protecting user data exchanged between tiers in a three-tier system. By providing strong, standards-based encryption, SSL provides system developers and users with confidence that data will not be compromised in the Internet. Note also that unlike password-based authentication, which authenticates client to server only, SSL can authenticate server to client as well as client to server. This is a useful feature when building a web-based three-tier system, since users often insist on authenticating the identity of a web server before they will provide the server with sensitive information, such as credit card numbers.

Java Security

Oracle8*i* was the first relational database to provide built-in support for Java, reinforcing its position as the database platform of choice for Internet developers. The security model in Oracle8*i* is that of JDK 1.1, which provided relatively coarse-grained access control. Oracle extends this security model to that of JDK 1.2, which includes a fine-grained, policy-based access control model. This model is more flexible and configurable than the previous Java security model, and is based on a permission class hierarchy.

JDBC Security

JDBC is an industry-standard Java interface that provides a Java standard for connecting to a relational database from a Java program. Sun Microsystems defined the JDBC standard, and Oracle Corporation, as an individual provider, implements and extends the standard with its own JDBC drivers. Oracle implements two types of JDBC drivers: Thick JDBC drivers built on top of the C-based Oracle Net Services client, and thin (pure Java) JDBC drivers to support downloadable applets.

Since thick JDBC uses the full Oracle Net Services communications stack on both client and server, it can take advantage of existing Oracle Advanced Security encryption and authentication mechanisms. Because the thin JDBC driver is designed for use with downloadable applets used over the Internet, Oracle includes a 100% Java implementation of Oracle Advanced Security encryption and integrity algorithms for use with thin clients. Oracle Advanced Security provides the following features for thin JDBC:

- Data encryption
- Data integrity checking
- Secure connections from thin JDBC clients to the Oracle Database 10g
- Ability for developers to build applets that transmit data over a secure communication channel
- Secure connections from Oracle Database 10gs to older versions of Oracle Advanced Security-enabled databases

Secure Connections for Virtually Any Client

Thick JDBC contains a complete implementation of a Oracle Net Services client in pure Java.

On the server, the negotiation of algorithms and the generation of keys function exactly the same as Oracle Advanced Security Net8 encryption, thus allowing backward and forward compatibility of clients and servers. On the clients, the algorithm negotiation and key generation occur in exactly the same manner as C-based Oracle Advanced Security encryption. The client and server negotiate encryption algorithms, generate random numbers, use Diffie-Hellman to exchange session keys, and use the Oracle Password Protocol, in the same manner as traditional Oracle Net Services clients. Thin JDBC contains a complete implementation of a Oracle Net Services client in pure Java. Consistent with other encryption implementations, the Java implementation of Oracle Advanced Security prevents access to the cryptographic algorithms, makes it impossible to double encrypt data, and encrypts data as it passes through the network. Users cannot alter the keyspace nor alter the encryption algorithms themselves.

Use of the Secure JDBC Implementation

The Oracle Advanced Security Java implementation gives developers the ability to build applets that transmit data over secure communication channels secured by Oracle Advanced Security. For example, it provides secure connections from any middle tier server with Java Server Pages (JSPs) to the Oracle Data Server and secure connections from Oracle Database 10gs to older versions of Oracle Advanced Security-enabled databases. This allows e-businesses deploying Oracle and other components to securely transmit a variety of information over a variety of channels.

SUMMARY

Just like previous versions of the Oracle Database, Oracle Database 10g Release 2 raises database security technology to a new level. Oracle's decade long commitment to independent security evaluations, coupled with Oracle's 25 plus years working with security conscious customers has enabled Oracle to establish itself as the database security leader. Robust support for data encryption, row level security, integrated identity management capabilities, fine-grained auditing, label security, proxy authentication, PKI support, and virtual private database are just a few of the technologies available with Oracle Database 10g Release 2. In addition, the capabilities in the Oracle Database 10g Release 2 are ideally suited for meeting the privacy and regulatory compliance challenges in today's global economy. Oracle Database 10g Release 2 identity management integration capabilities provide huge cost savings by dramatically reducing the complexity of managing application users. Oracle is an ideal platform on which to build and deploy secure applications for today's complex, Internet connected world.

Oracle Database 10g Release 2 Security
June 2005
Author: Paul Needham
Contributing Authors: [OPTIONAL]

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