SECURITY SOLUTIONS IN APPLICATION HOSTING

Paul D. Needham, Oracle Corporation
Mary Ann Davidson, Oracle Corporation

INTRODUCTION

The era of having individual companies install and maintain complex application software is over. Application designs are changing as a result of the technology breakthroughs which have been realized in the past decade. Software applications are no longer sold directly to customers. Application service providers, or hosting companies, now install and maintain centralized applications which can be used by multiple organizations simultaneously.

This change will result in multibillion dollar cost savings for organizations around the world. Information technology (IT) departments are shrinking and the financial resources associated with those departments are refocused into areas such as research and development and achieving the primary business objectives of the organization.

In order for this model to reach its full potential, it must be cost effective for the application service provider (ASP) and trustworthy. Central to both of these issues is security and how it can be achieved in a single application installation environment. While the Internet provides the information super highway to pass information around the world between the application user and the hosting facility, the database provides the infrastructure to manage the terabytes of information which will exist in an a hosted application. Multiple companies, using the same application and database, require a security infrastructure to protect their individual data.

This paper examines three of the application hosting security challenges and presents the Oracle security technology which can help address each of these challenges. The security technology discussed includes the Oracle Virtual Private Database and Oracle Label Security.

APPLICATION HOSTING SECURITY CHALLENGES

Applications built since the beginning of the computer age have not been designed for an ASP environment. The assumption was that individual companies would purchase the latest application software, take it back to their individual IT department, buy new computer hardware if necessary, upgrade power supplies, train database administrators, purchase backup recovery systems, and install the application client software on each desktop PC. This was an extremely expensive, painful and accepted fact of life until four years ago when the Internet changed everything. The remaining challenges to widespread adoption of the ASP model include the following:

1. Maintaining a one-to-one relationship between the application, database and application subscriber is financially prohibitive and defeats the purpose of the ASP model.
2. Retrofitting existing applications and building security directly into new applications is expensive.
3. Building highly granular access controls, necessary for a hosted application, is prohibitively expensive.

Oracle9*i* provides the technology to address these three challenges. Applications can be retrofitted and designed for the ASP model using Oracle9*i*. A single database and application installation can host multiple subscribers. Granular access controls can be easily designed and implemented using Oracle9*i* technology.

THE NEED FOR SINGLE DATABASE INSTANCE APPLICATION HOSTING

Creating and building separate databases for each application subscriber is not a cost-efficient model for an ASP. While technically possible, the separate database model would quickly become unmanageable and a dramatic increase in cost would be realized with each new subscriber. To be successful, the ASP model requires that a single application installation host multiple subscribers. For example, a human resources application must be capable of securely holding information from multiple companies. Installing the human resources application each time a new subscriber signs up with an ASP is not a practical solution.

ENABLING THE APPLICATION FOR HOSTING

A recurring challenge organizations face is the "application security problem." When access control is embedded in an application (instead of being enforced directly on the data), users who have access to ad-hoc query or reporting tools bypass the security mechanisms of the application. Strong security policies, centrally managed and applied directly to data, enables security to be enforced no matter how a user gets to the data, whether through an application, by a query, or using a report-writing tool. A centrally-managed and applied security policy also offers a lower cost of ownership: organizations can build security once, in the data server, instead of building security into every application which accesses the data. Applications are not going to be redesigned and rewritten overnight to perform optimally in an ASP environment. However, granular access controls can be built into the database and enforced by the database to provide data separation between hosted companies, or hosting subscribers.

HIGHLY GRANULAR ACCESS CONTROLS

Many existing applications are also faced with problems of enforcing complex access control policies, required to safeguard sensitive information (the disclosure of which might have severe social and legal ramifications). Human resources and medical information systems, for example, have strict requirements for security and privacy. These applications typically enforce multiple security rules, depending upon who is accessing the data, and what his function is.

Consider the following:

A bank wishes to allow its customers to do on-line banking. The bank wants to ensure that customers can only review transactions and account balances for their own accounts, and not anyone else's account.

A very large manufacturing company has a centralized human resource database which incorporates data from multiple subsidiaries, divisions and departments. There are multiple users of human resource information, and different security policies apply to each type of access:

- Employees can view their own HR records, and modify information such as marital status, number of dependents, address, and phone number, but they cannot modify their own salary.
- Managers can view all information for employees who work for them, directly or indirectly.
- HR specialists can review (and update) employee records within their area only. For example, an HR specialist might only be able to update records for employees in the Aircraft division, whose last names begin with the letters A through F.

A large telecommunications company provides long distance service for several local calling areas, and maintains call information (for multiple local companies) in a central database. The local calling companies must be able to review long distance calling information for their local customers, but they must not ever be able to see customer information for their local calling competitors.

A manufacturing company has a large direct sales force. With their sales force automation system, sales representatives can write or update orders on-the-spot while visiting customers. The sales force automation software must ensure that sales representatives can only enter or update orders for their own customer accounts, and no one else's account. Area managers, however, can enter or update orders for any customer in their territory.

A Web hosting company wishes to run the HR and Payroll business of other companies. Different companies want different personalizations. Some want access to raw data to run business analysis reports that best suit their corporate standards. The hosting company wants to use a best-of-breed HR application, but creating a completely new system for the company is not an economically-viable model.

A Department of Defense organization wishes to limit data access based on a security label (Unclassified, Secret, or Top Secret). Users (with appropriate privileges) can only query records at their security classification and below. For example, a user cleared to Secret can retrieve records labeled Secret and Unclassified, while a user cleared to Unclassified can only review Unclassified records.

Each of the above scenarios has the same basic problem: the need to mediate data access at a very fine level of granularity.

THE VIRTUAL PRIVATE DATABASE

The following sections describe the functionality of the Virtual Private Database— fine-grained access control and a related feature, secure application context— provided in Oracle8*i* and enhanced for Oracle9*i*.

The Virtual Private Database offers the following benefits:

- You can lower your cost of ownership. Organizations can reap huge cost savings by building security once, in the data server, instead of implementing the same security in each application that accesses data.
- You can eliminate the "application security problem." Users no longer bypass security policies embedded in applications because the security policy is attached to the data. The same security policy is automatically enforced by the data server, no matter how a user accesses data, whether through a report-writing tool, a query, or through an application.
- You can do things you never could do before. In the past, organizations couldn't give customers and partners direct access to their production systems, because there was no way to secure the data. Hosting companies couldn't have data for multiple companies reside in the same data server, because they could not separate each company's data. Now, all these scenarios are possible, because fine-grained access control gives you server-enforced data security with the assurance of physical data separation

DYNAMICALLY MODIFIED QUERIES

Fine-grained access control relies upon "dynamic query modification" to enforce security policies on the objects with which the policies are associated. Here, "query" refers to any selection from a table or view, including data access through a query-for-update, insert or delete statements, or a subquery, not just statements which begin with SELECT.

A user directly or indirectly accessing a table or view having a security policy associated with it causes the server to dynamically modify the statement based on a "WHERE" condition (known as a predicate) returned by a function which implements the security policy. The user's SQL statement is modified dynamically, transparently to the user, using any condition which can be expressed in, or returned by a function.

Consider an HR clerk who is only allowed to see employee records in the Aircraft Division. When the user initiates the query "SELECT * FROM emp," the function implementing the security policy returns the predicate "division = 'AIRCRAFT'", and the database transparently rewrites the query, so that the query actually executed becomes "SELECT * FROM emp WHERE division = 'AIRCRAFT'".

SECURE APPLICATION CONTEXTS

Many organizations want to make access control decisions based on something about the user (such as the user's role in the organization, his organizational unit, whether he is a customer or partner), and something about the application itself (whether the user is accessing a general ledger application as the GL manager, or the human resources application as an employee).

Secure application contexts enhance the ability of developers to implement the Virtual Private Database within Oracle8*i*. While this feature is not required, it makes implementation of fine-grained access control far easier and more robust. Application contexts offer the following benefits:

1. *Extensibility* - Application contexts are completely user-definable, as are their attributes. As a result, each application can have its own application-specific context, with different attributes.
2. *Ease of use* - Applications can set, verify and retrieve a particular context attribute conveniently and unambiguously. Oracle8*i* offers a system function (SYS_CONTEXT) which allows you to specify the exact context and attribute you want to set.

ORACLE LABEL SECURITY

Oracle Label Security -- The Virtual Private Database technology in Oracle8i is the foundation for a new product from Oracle called Oracle Label Security. Oracle Label Security is based on labeling concepts used by government and defense organizations to protect sensitive information and provide data separation. In these environments labels typically are composed of a hierarchical level and one or more categories or compartments. Oracle Label Security can be used to retrofit existing applications for the ASP model. It can also be used to provide security in Healthcare and CRM applications.

Oracle Label Security offers the following benefits:

- Oracle Policy Manager, a GUI for administering labels and authorizations. Oracle Policy Manager will also serve as the administration tool for user defined Virtual Private Database policies. Oracle Policy Manager will be integrated into the Oracle Enterprise Manager.
- Oracle Label Security *automatically* provides out-of-the-box VPD enforcement. Customers need no longer code a VPD application; Oracle Label Security is a VPD product.
- Provides data labeling required by government and defense organizations on standard operating system platforms.
- Label-based access control for application service providers. Data labels provide a degree of granularity which cannot be easily achieved with raw application data.
- Available with the Oracle8*i* Enterprise Edition on standard operating systems.

ORACLE LABEL SECURITY POLICIES

Oracle Label Security is built around the concept of a label based access control (LBAC) *policy*. Using Oracle Label Security, users define and associate labels and user authorizations with each policy. Labels are defined by a security administrator using a GUI. The policy is then applied to an entire application or specific application tables and labels are authorized for users. For example, a *Defense* policy and *Human Resources* policy could be defined within an Oracle8*i* database. The *Defense* policy could define labels *Secret*, *Top Secret* and *Confidential*. The *Human Resources* policy could define labels *Senior VP*, *Manager* and *HR Only*.

ORACLE LABEL SECURITY LABELS

An Oracle Label Security *Label* can be thought of as an additional table attribute. The label is added to the base table definition of application tables and used by Oracle Label Security to enforce access mediation at the row level. The Oracle Label Security label design is one of the key architectural features which allows customers to implement sophisticated access controls. The label contains three components: a single hierarchical level or classification, one or more horizontal compartments or categories and one or more groups.

LABEL COMPONENTS

Level -- The level is a hierarchical component which denotes the sensitivity of the data. A typical government organization might define levels confidential, sensitive and highly sensitive. However, there is no requirement to define more than one level. For example, a commercial organization might define a single level for company confidential data or application hosting requirements.

Compartment - The compartment component is sometimes referred to as a category and is non hierarchical. Typically one or more compartments are defined to segregate data. For example, a compartment might be defined for an ongoing strategic initiative or map to a hosted application subscriber. Data related to the initiative could be labeled with the newly-defined compartment. Oracle Label Security supports up to 9999 unique compartments.

Group - The group component is used to record ownership and can be used hierarchically. For example, two groups called *Senior VP* and *Manager* could be created and subsequently assigned as children of the *CEO* group, creating an ownership tree. Labels can be composed of a standalone level component or a level component can be combined with compartments, groups or both.

LABEL TAGS

A *label tag* is stored with the actual data for optimization. The example application table below shows a table with four attributes: Employee Number, Hire Date, Grade, and Department. The fifth and final component is the label attribute added by Oracle Label Security. The values listed under the Subscriber Label are the external representations of the labels.

EXTERNAL REPRESENTATION

The external representation of a label is composed of the three label components, separated by a semicolon. The label *Business-On-Line: ACME : Senior VP* is composed of the following label components:

- Level = Business-On-Line
- Compartment = ACME
- Group = Senior VP

EXAMPLE APPLICATION TABLE:

Employee	*Hire Date*	*Grade*	*Department*	*Subscriber Label*
12345	Sep 12, 1999	3	Engineering	Business-on-line: ACME: Manager
45673	Apr 01, 1984	5	Human Resources	Business-on-line: ACME: Manager
32100	July 03, 1991	2	Finance	Business-on-line: ACME : Senior VP

The labels in the example table all have the same level, Business-On-Line, and the same compartment, ACME. However, the group component shows two potential values, Manager and Senior VP. For example purposes, assume the Label Security administrator defined the Senior VP group as a parent group of the Manager group.

ORACLE LABEL SECURITY MEDIATION

Oracle Label Security mediates access to rows in database tables based on a label contained in the row, a label associated with each database session, and Oracle Label Security privileges assigned to the session. Oracle Label Security provides access mediation on application data after a user has been granted the standard Oracle8*i* SYSTEM and OBJECT privileges. For example, assume a user has SELECT privilege on an application table. If the user executes a SELECT statement on the table, Oracle Label Security will determine access to individual rows based on the privileges and access labels assigned to the user by the security administrator. Oracle Label Security can also perform security checks on UPDATE, DELETE and INSERT statements.

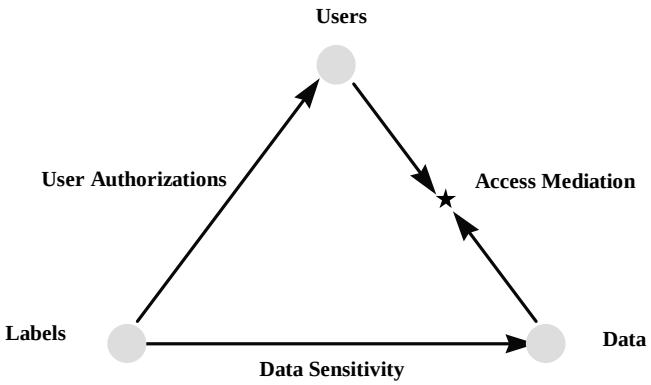

Figure 1 - Labels, User Authorizations and Data Sensitivity

Figure 1 shows how labels, users and data are related. In the lower left corner, labels are defined. Labels are then assigned to users, as *authorizations*, and to data, for data sensitivity. Access mediation takes place by comparing the user authorizations and the data sensitivities. In other words, Oracle Label Security compares the *label tag* assigned to a row in an application table with the label authorizations assigned to the application user. For example, if a user is authorized to view data with a maximum sensitivity of *Secret*, the user would not be allowed to view data which has a sensitivity of *Top Secret*.

USER LABEL AUTHORIZATIONS

Oracle Label Security user authorizations must be established by a security administrator before an application user can access an application table protected by Oracle Label Security.

Oracle Label Security user label authorizations are defined as follows:

Maximum Level — The maximum sensitivity level a user is authorized to access. In a hosting environment only single level may exist. In government and defense environments four or five levels might be defined.

Minimum Level — The minimum sensitivity level a user is authorized to write data. For example, an administrator can prevent users from labeling data as *Public* or *Internet* by assigning a minimum level of *Company Confidential*.

Default Level — The level used by default when a user connects to the database. For example, a user can set his or her default level to Secret. When he or she connects to the system, the default level will be initialized to Secret.

Row Level — The level used to label data inserted into the database by the user through the application or directly through a tool such as SQL*Plus.

Read Compartments — The set of compartments assigned to the user and used during READ access mediation. For example, if a user has compartments *A,B and C*, he could view data which has compartments *A and B* but not data which has compartments *A,B,C and D*. The read algorithm will be defined in the next section.

Write Compartments — The set of compartments assigned to the user and used during WRITE access mediation. For example, a user could be given READ and WRITE access to compartments *A and B* but READ-ONLY access to compartment *C*. If an application record was labeled with compartments *A,B and C*, the user would not be allowed to update the record because he or she does not have WRITE access on compartment *C*.

Read Groups — The set of groups assigned to the user and used during READ access mediation. For example, if a user had the group *Manager*, he could view data which has the *Manager* group but not data which had the *Senior VP* group. The read algorithm will be defined in the next section.

Write Groups — The set of groups assigned to the user and used during WRITE access mediation. For example, a user could be given READ and WRITE access to group *Senior VP* but READ-ONLY access to group *Manager*. If an application record was labeled with a single group, *Manager*, the user would not be allowed to update the record because he or she does not have WRITE access on the *Manager* group.

USER PRIVILEGE AUTHORIZATIONS

Oracle Label Security user privilege authorizations are defined as follows:

READ Privilege — The READ privilege allows a user to access all data protected by Oracle Label Security, however, access mediation is still enforced on UPDATE, INSERT and DELETE operations. Oracle Label Security makes no mediation check on SELECT operations.

FULL Privilege — The FULL privilege turns off all Oracle Label Security access mediation. A user with the FULL privilege can perform SELECT, UPATE, INSERT and DELETE operations with no label authorizations. Note that Oracle SYSTEM and OBJECT privileges are still enforced. For example, a user must still have SELECT on the application table. The FULL privilege turns off the access mediation check at the individual row level.

WRITEDOWN — The WRITEDOWN privilege allows a user to modify the level component of a label and lower the sensitivity of the label. For example, application data which is labeled *Top Secret: Alpha, Beta* could be changed to *Secret: Alpha, Beta*.

WRITEUP — The WRITEUP privilege allows a user to modify the level component of a label and raise the sensitivity of the label. For example, application data which is labeled *Secret: Alpha, Beta* could be changed to *Top Secret: Alpha, Beta*. Note that the *Maximum Level* label authorization assigned to the user would limit modification.

WRITEACROSS — The WRITEACROSS privilege allows a user to modify the compartments and groups in a label to any valid compartment and group defined in Oracle Label Security for the policy. For example, if application data which is labeled *Secret: Alpha, Beta* could be modified to *Secret: Alpha, Beta, Delta* even though the user was not authorized for the *Delta* compartment.

PROFILE ACCESS — The PROFILE ACCESS privilege allows a user to assume the Oracle Label Security authorizations of another user. For example, user *Scott* who has access to compartments *A,B, and C* could assume the profile of user *Joe* who has access to compartments *A,B, C and D*. This functionality might be useful in an environment where an application uses a single application account for all application users. The application account could use the PROFILE ACCESS privilege to immediately assume a designated label security profile when an application users connects to the system.

POLICY ENFORCEMENT OPTIONS

Oracle Label Security enforcement can be customized on a per policy basis. For example, a *Human Resources* policy and a *Defense* policy could exist in the same Oracle8*i* database and provide different degrees of protection. The *Human Resources* application could enforce READ protection only and the *Defense* policy could enforce READ and WRITE protection. The Oracle Label Security enforcement options are as follows:

READ CONTROL — Applies policy enforcement to all queries; only authorized rows are accessible for SELECT, UPDATE, and DELETE operations.

INSERT CONTROL — Applies policy enforcement to INSERT operations, according to the Oracle Label Security algorithm for write access.

UPDATE CONTROL — Applies policy enforcement to UPDATE operations on the data columns within a row, according to the Oracle Label Security algorithm for write access.

DELETE CONTROL — Applies policy enforcement to DELETE operations, according to the Oracle Label Security algorithm for write access.

WRITE CONTROL — Determines the ability to INSERT, UPDATE, and DELETE data in a row. If this option is set, it enforces INSERT_CONTROL, UPDATE_CONTROL, and DELETE_CONTROL.

LABEL DEFAULT — If the user does not explicitly specify a label on INSERT, the users default *row label* value is used. By default, the *row label* value is computed internally by Oracle Label Security using the label authorization values specified for the user. A user can set the row label independently, but only to:

- A level which is less than or equal to the level of the session label, and greater than or equal to the user's minimum level.
- Include a subset of the compartments and groups from the session label, for which the user is authorized to have write access.

LABEL UPDATE — Applies policy enforcement to UPDATE operations that set or change the value of a label attached to a row. The WRITEUP, WRITEDOWN, and WRITEACROSS privileges are only enforced if the LABEL_UPDATE option is set.

LABEL CHECK — Applies READ_CONTROL policy enforcement to INSERT and UPDATE statements to assure that the new row label is read-accessible by the user after and INSERT or UPDATE statement.

NO CONTROL — Applies no enforcement options. A labeling function or a SQL predicate can nonetheless be applied.

ORACLE LABEL SECURITY ALGORITHMS

The following algorithms assume that READ CONTROL and WRITE CONTROL are enforced. The algorithms show how Oracle Label Security mediates access between a user and labeled data.

THE READ ALGORITHM

1. Is the user's level equal to, or greater than, the level of the data?
2. If so, does the user have access to at least one of the groups present in the data label?
3. If so, does the user have access to all the compartments present in the data label?

If the answer is no at any stage in this evaluation process, then Oracle Label Security denies access to the row, and moves on to evaluate the next row of data.

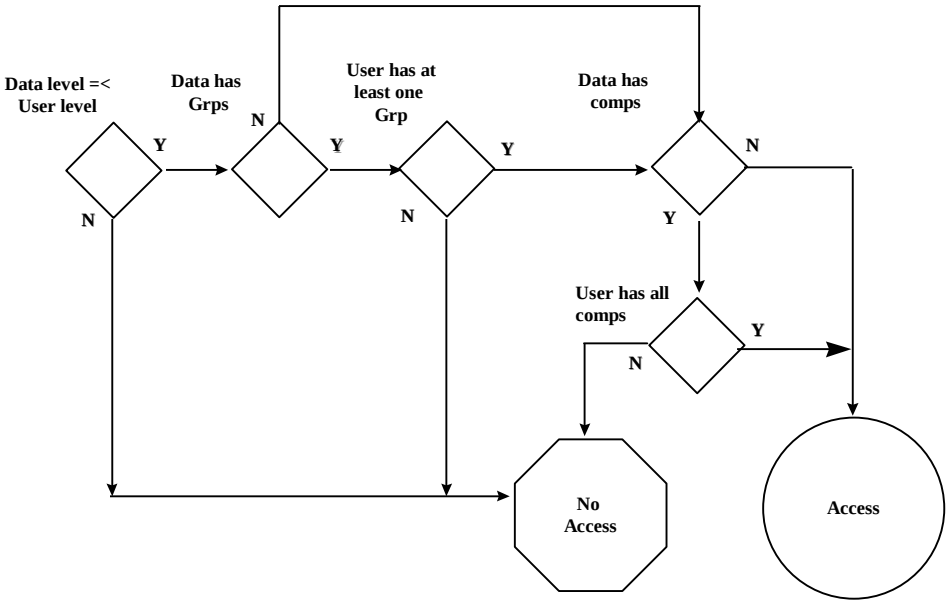

Figure 2 - Read Algorithm

THE WRITE ALGORITHM

1. The level in the data label must be greater than or equal to the user's minimum level and less than or equal to the user's session level. Oracle Label Security establishes a session label for all users in Oracle8*i*. The session level is the level component of the session label. For example, a user might have a session label of *Secret: A, B, C.* The session level is *Secret.*
2. When groups are present, the user's label must include at least one of the groups with write access which appear in the data label. In addition, the user's label must include all the compartments in the data label.
3. When no groups are present, the user's label must have write access on all of the compartments in the data label.

Note that with *Read* and *Write* algorithms, the label evaluation process proceeds from level to groups to compartments. A user cannot write any data below her authorized minimum level, nor above her current session level. The user can always read below her minimum level.

ORACLE LABEL SECURITY — LABEL FUNCTIONS

Label functions can be defined in the Oracle8*i* database and referenced in an Oracle Label Security policy definition. Label functions compute the label value which should be assigned to application data during INSERT and UPDATE statements. Labeling functions enable you to consider, in your rules for assigning labels, information drawn from the Virtual Private Database application context. For example, you can use as a labeling consideration the IP address to which the user is attached. A labeling function is called in the context of special before-row trigger. This enables the label function designer to pass in the old and new values of the data record, as well as the old and new labels.

Label functions can be written in PL/SQL and assigned to Oracle Label Security policies through the Oracle Policy Manager (OPM), a graphical user interface. Label functions are an extremely powerful feature of Oracle Label Security. A simple example is included in Appendix B.

ORACLE LABEL SECURITY — SQL PREDICATES

A SQL predicate can be used to provide extensibility for selective enforcement of data access rules. Oracle Label Security allows SQL predicates to be easily added to Oracle Label Security policies via the OPM or API. For example, the following SQL predicate could be added to the READ algorithm:

```
AND my_function(col1) = 1
```

Another example is as follows:

```
OR SYS_CONTEXT ('USERENV','SESSION_USER') = employee_name
```

The SQL predicate allows basic user defined Virtual Private Database policies to be easily applied to individual application tables or entire applications.

ORACLE LABEL SECURITY API

Oracle Label Security has a comprehensive application programmatic interface (API) which can be used by developers and administrators. The API is documented in the Oracle Label Security Administrator's Guide.

PERFORMANCE

Oracle Label Security has highly optimized algorithms for achieving performance during access mediation. An internal label cache stores user label authorizations for quick access during query execution. In addition, Oracle Label Security privileges can be selectively granted to individual users who have special access or need to perform administrative functions. For example, if a special report needs to be processed which requires access to all data in an application, the READ authorization privilege could be granted to the report generation user. The READ privilege would allow read-only access to all data, but would still enforce access mediation on update, insert and delete statements, protecting the integrity of the application data.

SUMMARY

The Virtual Private Database is key enabling technology opening mission-critical systems to partners and customers over the Internet. Fine-grained access control, with secure application contexts, will enable organizations to secure data in the Oracle8*i* server, and ensure that, no matter how a user gets to the data (through an application, a report writing tool or SQL*Plus) the same access control policy will be enforced. The Virtual Private Database can help banks ensure that customers see their own accounts (and nobody else's), that telecommunications firms can keep customer records safely segregated, and that human resources applications can support their complex rules of data access to employee records. The Virtual Private Database also enables you to lower cost of development, by building security once, in the data server, instead of in every application that accesses the data.

Oracle Label Security provides a convenient method of enabling application hosting in legacy applications and newly-designed applications. Oracle Label Security is an out-of-the-box Virtual Private Database product. Oracle Label Security can also be used to develop applications for governments and defense organizations worldwide. The label attribute allows highly sophisticated access control mediation at the individual data row completely transparent to the application data itself. Oracle Label Security can be used with the Virtual Private Database to provide a comprehensive, fine-grained access control solution.

APPENDIX A - APPLICATION HOSTING EXAMPLE USING ORACLE LABEL SECURITY

SAMPLE IMPLEMENTATION OF ORACLE LABEL SECURITY

This section describes a step-by-step implementation of Oracle Label Security in a payroll application. A fictitious company called *Z3 Hosting Inc.* decided to join the ASP market. The application *Z3 Hosting Inc.* is hosting is a financial payroll system and it has signed up several subscribers for its offering. What distinguishes *Z3 Hosting Inc.* from other hosting companies is that it allows individual hosting subscribers to restrict payroll reports based on management levels within their organization. For example, the Senior VP of one subscriber can generate payroll reports for all employees; however, individual department managers are completely unaware of the overall payroll numbers. The first subscriber *Z3 Hosting Inc.* signs up is the *ACME Corporation.*

STEP 1. CREATE THE PAYROLL APPLICATION HOSTING POLICY.

An ACME Inc. administrator uses the Oracle Label Security GUI tool — Oracle Policy Manager — or issues the following SQL statement:

```
SA_SYSDBA.CREATE_POLICY ('PAYROLL','Subscriber','READ_CONTROL,
                         WRITE_CONTROL');
```

The call to CREATE_POLICY creates a policy called *Payroll* and tells Oracle to use the name *Subscriber* as the database table column to store the label information. The call also tells the new policy to enforce two enforcement options – READ_CONTROL and WRITE_CONTROL. With Read Control, the policy will protect the payroll application during all application reads. With Write Control, the policy will protect the payroll application during updates, deletes and inserts. Note that Oracle Label Security does not replace the standard Oracle8*i* discretionary access control privileges (SELECT, INSERT, UPDATE, DELETE). Oracle Label Security provides an additional access mediation check at the individual row level using a data label.

STEP 2. DEFINE THE VALID LABEL COMPONENTS FOR THE POLICY.

An ACME Inc. administrator could use the Oracle Label Security GUI tool — Oracle Policy Manager — or issues the following SQL statement.

```
SA_COMPONENTS.CREATE_LEVEL ('PAYROLL','Business-On-line',1000);
```

The call to CREATE_LEVEL creates the label level component *Business-On-Line* and assigns the numeric value 1000 to the level. The first parameter is the Label Security policy name. A typical hosting policy will only have a single level. In government and defense environments, levels may include *Confidential*, *Secret* and *Top Secret.*

```
SA_COMPONENTS.CREATE_COMPARTMENTS('PAYROLL','ACME',50);
```

The call to CREATE_COMPARTMENT create the label compartment components *ACME* for the ACME Corporation subscriber and assigns the numeric identifier 50 to the compartment.

```
SA_COMPONENTS.CREATE_GROUPS('PAYROLL',15,'SVP','Senior VP');
SA_COMPONENTS.CREATE_GROUPS('PAYROLL',20,'MGR','Manager',
                            'Senior VP');
```

The first call to CREATE_GROUPS creates the group *Senior VP* and assigns the numeric identifier 15 to the group. Note that *SVP* is the short name associated with the group. The second call to CREATE_GROUPS creates the group *Manager*, assigns the numeric identifier 20 to the group and specifies it as a child of the *Senior VP* group. Note that the numeric identifier values are not used as a hierarchical ranking, but solely as a identifier for the group.

STEP 3. PROTECT THE PAYROLL APPLICATION

An ACME Inc. administrator uses the Oracle Label Security GUI tool — Oracle Policy Manager — or issues the following SQL statement:

```
SA_POLICY_ADMIN.APPLY_SCHEMA_POLICY ('PAYROLL' ,'PAY_APPLICATIONS');
```

The call to APPLY_SCHEMA_POLICY instructs Oracle Label Security to apply the PAYROLL policy to all application tables in the PAY_APPLICATIONS database schema. By default, the policy options specified when the PAYROLL policy was created will be enforced on all tables in the schema.

Oracle Label Security policies can be applied to entire application schemes or to individual application tables. The following SQL statement could be issued to protect an individual table:

```
SA_POLICY_ADMIN.APPLY_TABLE_POLICY('PAYROLL','PAY_APPLICATIONS',
                                   'SALARY_HISTORY');
```

The call to APPLY_TABLE_POLICY instructs Oracle Label Security to apply the PAYROLL policy to the SALARY_HISTORY table in the PAY_APPLICATIONS schema. Note that the APPLY_SCHEMA_POLICY and APPLY_TABLE_POLICY procedures have additional optional parameters for customizing policy enforcement options.

STEP 4. ACME HOSTING INC. ADMINISTRATOR'S GRANT ACCESS PRIVILEGES TO SELECTED ACME EMPLOYEES.

An ACME Inc. administrator uses the Oracle Label Security GUI tool — Oracle Policy Manager — or issues the following SQL statement:

```
SA_USER_ADMIN.set_user_labels('PAYROLL','EMP1',
'BUSINESS-ON-LINE:ACME:SVP');

SA_USER_ADMIN.set_user_labels('PAYROLL','EMP2',
'BUSINESS-ON-LINE:ACME:MGR');
```

In this example, the ACME employees EMP1 and EMP2 are assigned to the PAYROLL hosting policy, granted access to the ACME data and assigned to the *Senior VP* and *Manager* groups respectively. When the user EMP1 connects to the Oracle8*i* based application, the user will be authorized for access to the ACME Corporation data and to the *Senior VP* group. Since the *Manager* group is a child group of the *Senior VP* group, EMP1 will be able to access all data from the *Senior VP* group downward.

APPENDIX B - LABEL FUNCTION EXAMPLE

OPTIONAL LABEL FUNCTION EXAMPLE:

In this example, the function gen_emp_label computes a label based on the job and total_salary of a user. Oracle Label Security label functions override labels entered explicitly by users during insert and update operations. This functionality allows a consistent label policy to be applied. The example also uses the Virtual Private Database application context functionality.

```
CREATE OR REPLACE FUNCTION gen_emp_label
                 (Job varchar2,
                  Deptno number,
                  Total_sal number)
  Return LBACSYS.LBAC_LABEL
as
  i_label  varchar2(80);
Begin
  /* Set Class Level */
  i_label := 'business-on-line:';

  /* Set Compartment */

  i_label := i_label || sys_context('hosting_ctx','COMPANY_ID')
             || ':';
  /* Set Group Component */
  if job = 'Senior VP' then
     i_label := i_label ||'SVP';
  elsif job = 'Manager' then
     i_label := i_label || 'MGR';
  else
     i_label := i_label || 'STAFF';
  end if;

  return to_lbac_data_label ('PAYROLL', i_label);
End;
/
```

APPENDIX C - EXAMPLE LABEL COMPONENTS BY INDUSTRY

INDUSTRY	*LEVELS*	*COMPARTMENTS*	*GROUPS*
Defense	Top Secret Secret Confidential Unclassified	Alpha Beta Sigma	UK NATO US
Financial Services	Acquisitions Corporate Client Operations	Insurance Equities Trusts Commercial Loans Consumer Loans	Client Trustee Beneficiary Management Staff
Judicial	National Security Sensitive Public	Civil Criminal	Administration Defense Prosecution Court
Healthcare	Primary Physician Patient Confidential Patient Release	Pharmaceutical Infectious Diseases	CDC Research Nursing Staff Hospital Staff