Defence

Defending the Oracle Database with Advanced Security Features

Mikoláš Panský

Difficulty

There are some actual issues with Oracle Security. There is a new book The Oracle Hacker's Handbook written by David Litchfield. It covers possible methods to attack the Oracle server. Some of the examples shown in that book are based on traffic sniffing, direct access to Oracle's Shared Global Memory, or just accessing the raw data files.

Some of these risks could be prevented by adding Advanced Security features to an already existent database. This does not mean that you would open the box and a perfect unique shield will protect your database. It does not work like this. All these features must be carefully planned and velvety implemented. In the event of a system crash/ configuration error your data may never be recovered otherwise. However if the implementation is carefully planned, and you have an experienced DBA there is way to better defend your database.

Authentication

Authentication means verifying the identity of subject who wants to access database objects. When authentication is successfully passed, authorization processes comes into play. *Authorization* is the process that controls access to database objects. There are different methods of authentication. The most commonly used are authentication by the Operating System, Network, Database, Multi-Tier System, or Secure Socket Layer. When OS Authentication is used there is no need for any further validation. Users can connect to database just by running the database client (e.g. *sqlplus*). In the OS Authentication case, security is traded for comfort resulting in a less secure environment. Network authentication is implemented using Secure Socket Layer or the help of Third-Party Services. These could for example be Kerberos or PKI-based authentications. In my opinion the most commonly used method of authentication is the Oracle Database is the text password. Thus far it achieves decent security, and ease of use. However this is all governed by the complexity of the password as well as it's resistance to social engineering. There is no need to install any other system authentication. Nor is it a simple walk around a poorly secured operating system. Database

What you will learn...

- What is Oracle Wallet,
- What is Transparent Data Encryption,
- What is Oracle Advanced Security.

What you should know...

- Basic knowledge of SSL,
- Basic knowledge of computer Cryptography

authentication is based on the comparison of a given username/password combination. As well as information encrypted and stored in a data dictionary. Users can change their passwords at any time. One of the basic tasks in securing the database should be enabling password encryption while connecting, account locking, password lifetime and expiration, password history and password complexity verification. These requests are explained in detail in the Oracle Database Security Guide. This was my prior reference source for this article. My intentions were not to rewrite the entire manual. I wanted to give you short overview and save you several hours of reading. Other ways to control access to the database is to create Multi-Tier Application. This provides access to database with a layer that handles queries and user controls. Multiple users can access a data server without separate connections for each of them. For these purposes an Oracle Call Interface could be used. It has some advantages but generally it is not recommended due to poor security. Once an account has been compromised the attacker gains full access to the application, and may seek higher account privileges.

Authorization

Post-authentication tasks (that verifies user identity) have to control user's access to database objects. At first we used profiles and identification methods to complete the first task. Now we need to manipulate privileges, roles, profiles and resource limitations. The authorization consists of two main processes. First is to ensure that only certain users can access, process, or alter data. The second is to apply limitations on user access or actions (e.g. *limitations of objects and/or resources*). When speaking of authorization – One of the first questions to ask is what privileges a particular user has? A privilege is a right to execute a particular type of SQL statement or to access another user's object. Privileges could be granted to a user one by one or in groups through roles. Roles are incorporated into the database to simplify the process of administrating users and their rights to do something in the database. There are two main categories of privileges. System privileges should be granted with care. They should never be given to common database users. They should be granted only to administrators and application developers. The SQL statements to use with privileges are GRANT and REVOKE. There is one unique factor about privileges in Oracle. It is possible to grant privileges with the admin option. This option allows the target user to grant or revoke such privileges to/from another person. Object privileges control user's access to tables, views, procedures, functions or packages.

Concept

I must begin at the starting point for exploring the *Oracle Database*. This is a *Conceptual Guide*. The basic idea of Security is to deny or allow users actions. The ideal model of security implementation in Oracle is discretionary access control. This means that privileges are granted to users at the discretion of other users. The database itself stores a list of users. When a user is trying to access a database application a valid username and password must be provided. A security domain exists for each user. A security domain is set of privileges and roles, table space quotas and system resources limits. A privilege is an implementation of access control. Oracle is very flexible and offers precise

Tables that are Used to Build the Views

- `user$` – table of users identified by name, type and number.
- `defrole$` – default roles (columns are user# and role#).
- `objauth$` – table of authorization.
- `sysauth$` – system authorization (system privileges, grantee, options).
- `ts$` – tablespaces.
- `obj$` – Objects. Identifies objects by name, type and owner number.
- `cols$` – Columns.
- `profile$` – Connects profiles and resource privileges.
- `resource_map` – description of resources.
- `system_privilege_map` – description of system privileges.
- `table_privilege_map` – description of auditing privileges.
- `user_astatus_map` – status of password and account status.

Security Related Views

VIEWS RELATED TO PROFILES:

```
DBA_PROFILES, DBA_SQL_PROFILES.
```

VIEWS RELATED TO ROLES:

```
DBA_APPLICATION_ROLES, DAB_CONNECT_ROLE_GRANTEES, DBA_ROLE_PRIVS, DBA_ROLES,
PROXY_ROLES, PROXY_USER_AND_ROLES.
```

VIEWS RELATED TO PRIVILEGES:

```
DAB_COL_PRIVS, DBA_ROLE_PRIVS, DBA_SYS_PRIVS, DAB_TAB_PRIVS.
```

VIEWS RELATED TO USERS:

```
DBA_USERS.
```

Listing 1. *Creating TDE column with and without salt*

```
CREATE TABLE gold_partner (
  partnerID NUMBER ENCRYPT NO SALT,
  name VARCHAR2(128),
  surname VARCHAR2(128),
  ccno NUMBER(16) ENCRYPT USING 'M4g1cW0rD'
);
CREATE INDEX partnerID_idx ON god_partner (partnerID);
```

Listing 2. *Sample sqlnet.ora for server*

```
# sqlnet.ora Network Configuration (Server)
# Example configuration for server to use Oracle Advanced Features
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER= (MD5)
SQLNET.ENCRYPTION_TYPES_SERVER= (3DES168, 3DES112, DES40)
```

Listing 3. *Sample sqlnet.ora for client*

```
# sqlnet.ora Network Configuration (Client)
# Example configuration for client to use Oracle Advanced Features
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT= (MD5)
SQLNET.ENCRYPTION_TYPES_CLIENT= (3DES112, 3DES168, DES40)
```

control of user's privileges. We could divide system privileges into two categories. One of its privileges that is applicable to whole database system. This privilege has a name word *ANY*. These privileges are very powerful. It gives access rights to all objects that are not in the SYS scheme (data dictionary). Access to the data dictionary could be regulated by the system initial parameter `07_DICTIONARY_ACCESSIBILITY`. If this parameter is set to false no privilege could access the data dictionary. The rest of system privileges affect the state of the database. For example these could be privileges to CREATE TABLESPACE, AUDIT SYSTEM, CREATE LIBRARY etc. Most of the system privileges could be found in the SYSTEM_PRIVILEGE_MAP view. We could allow standard access to this table by issuing a query to explore its contents. To view system privileges granted to users and roles we could use the `DBA_SYS_PRIVS` view. In terminology the system privileges view grantee is the person who has the privilege granted to. These views could serve for easy database administration, but it is not recommended that one rely on it 100%. As stated in one of the previous issues of Hakin9 magazine these views could be modified with full access to the database to hide users, processes etc. There are many view used for security purposes.

Some of this views also has ALL and USER versions. It differs in scope from objects that display. The difference could be obtained from view name prefix. We have *ALL*, which displays all objects, USER displays objects that are owned by user and DBA corresponds to the database administrator's objects. These views are in the database to show their user-friendly way following database tables:

As I mentioned above there is another group of privileges. These privileges are object privileges. These are relevant to a specific object. For example when a user wants to view the content of the employee table, he has to issue the SELECT command. However it's not allowed by default for the user to view their privileges. If the administrator would like to allow users to see what access level they have, access could be granted to the view `user_objects_privs` or `user_sys_privs` by making the access to these tables to PUBLIC. However I would definitely not recommend this. The only usage that comes to my mind is database access level debugging. Yet another way to use this view would be to check to see who has the grantable right.

Supervisors

To administer the database, there are two predefined *accounts* in the system – *SYS* and *SYSTEM*. In previous versions (10g and less) there was a default password associated with these accounts. SYS had default password CHANGE_ON_INSTALL and SYSTEM had default password MANAGER. One of the very important tasks during database creation CREATE DATABASE is to use commands USER SYS IDENTIFIED BY password and USER SYSTEM IDENTIFIED BY password to change default passwords for these accounts. There are many hacking tools to reveal default password association. These accounts are really powerful! It is a good idea to create another account to perform daily administration tasks and avoid using these types of accounts. To help simplify administration of privileges associated with each user a group of privileges called Role exists for this purpose. These could be granted with the GRANT clause. Imagine there is inquiry to manage access control for a hundred

Oracle Wallet Registry Keys

DEFAULT:

```
\HKEY_CURRENT_USER\SOFTWARE\
ORACLE\WALLETS\DEFAULT
```

ENCRYPTED WALLET:

```
\HKEY_CURRENT_USER\SOFTWARE\
ORACLE\WALLETS\DEFAULT\
EWALLET.P12
```

OBFUSCATED ORACLE WALLET:

```
\HKEY_CURRENT_USER\SOFTWARE\
ORACLE\WALLETS\DEFAULT \
CWALLET.SSO
```

users from ten different departments. This could become very frustrating without having to associate each user with at least five different privileges. Things would become even worse if you had to change privileges for all users in the department. This is where user roles come into play. For each department a role could be created. Then changing privileges for all departments would be as easy as *grant*/*revoke* privilege from role. Another case of using roles would be necessary when there is need to use several different applications. It's the same principle, however it differs a bit in the reason to create role. Imagine applications that use a table of offers. With the use of roles in the game it's much easier to change the access depending on the user that is logged into the application. For example when regular user is logged into an application there could be a role created for this purpose. However this principle allows the implementation element that changes the access privileges during runtime just by using the SET ROLE clause on the fly. The most powerful role in the system is DBA (stands for Database Administrator). This role is implicitly associated with the *SYS* and *SYSTEM* account. However I must again note that access control to these accounts is a critical task. In obsolete versions of Oracle (8i and older) there was a special user named *INTERNAL* that could access the database whether it was in a *MOUNT* or *NOMOUNT* state. This account had the default password set to *oracle*. This account wasn't maintained in the database data dictionary, but in an Oracle password file. In past versions the INTERNAL mechanism has been replace by the SYSDBA and SYSOPER privilege. SYSDBA privileges allows user to startup, shutdown, backup, recover and create databases. The list of all users who has SYSDBA or SYSOPER privileges could be found in v$defile users. There is a limitation for the SYSDBA role – it cannot be granted to the public. Another issue when creating databases is the default action of creating the role PUBLIC. This role is often used in hacking methods. There are two main reasons. The first is that some people don't even know that it exists. That's because it cannot be seen in dab_roles. Another and even more important reason is because changing the privileges in this role applies to other users as well. This role is created when a new database is created (to create new database Oracle uses the script sql.bsq). To determine the type of account the user is connected to database with could be shown with *SHOW USER* (*SQL*Plus*). There is one more thing in using these accounts. The objects owned by user SYS cannot be exported via standard tools (*exp*, *imp*). Another rule tells us that No objects may be created in SYS schema. The SYS schema has the job of storing data dictionary objects and it is fully managed by the database itself.

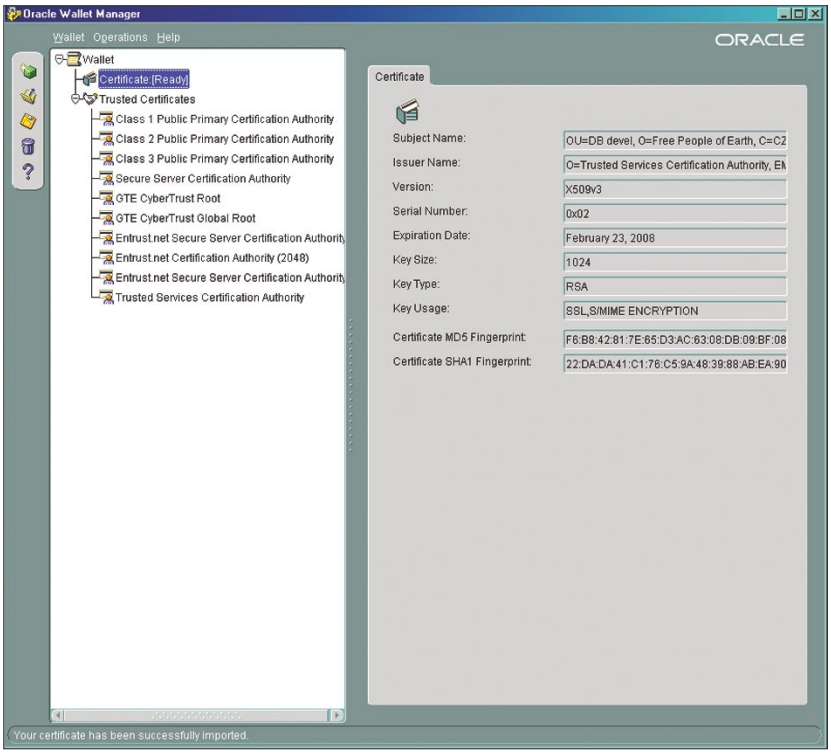

Figure 1. *ORACLE.WALLET.SCREENSHOT*

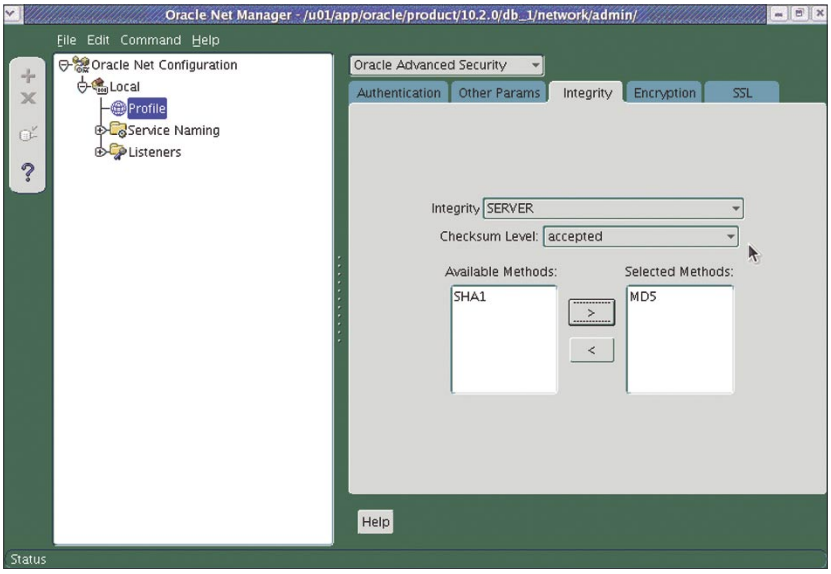

Figure 2. *ORACLE.NETWORK.MANAGER*

Application Security Roles

Application security roles could be enabled only by authorized PL/SQL packages. To ensure higher level of security it is better not to embed passwords in the source code or the table. In order to achieve this it is necessary to create a secure application role that could define which PL/SQL package has sufficient privileges to enable this role. This concept could be enhanced by adding additional checks of conditions for authorization by the application. However implementing authorization on the client side of the application is always tricky. The main reason for this is the fact, that an application could be *skipped* by using sqlplus client or any other tool to connect to the database. Another reason to use security mechanisms on the server side is re-usability. There is no need to implement the security access control twice when we change the application. It is enough to store it once on the server and then reuse it with different applications. Also when using database server side security we could use all the security features that Oracle offers (fine-grained access control with application context), roles, stored procedures and auditing. For this reasons Use of Ad Hoc Tools is a potential security problem.

It is recommended by Oracle to equal application users to database users. This would give us the potential to use all security features that Oracle has to offer. However this is not true for many applications. Most of these applications use one user to connect to the database with higher privileges. It is the so-called One Big Application User model. There are some disadvantages while using this model to access the database. For example there is no way to audit the actions of each user using this application. The database doesn't recognize each user. If we would like to use auditing in the One User Model we must implement our own auditing mechanisms. The second disadvantage of using the one user model is the possible inability to use Advanced Security Authentication. These include SSL, tokens etc. The third reason against this is user access to the database is less effective than with the usage of roles. This restriction could be overcome by using a set role dynamically. And last but not least is to disable the Oracle Identity Manager. The fundamentals in implementing a Secure Application Role are based validating the identity by looking into the context. Application roles could be also used for controlling the value of IP an address, where is the user connecting from? Application roles could be implemented in separate packages. The basic principle of using a secure application role is to associate privileges with User Database Roles. Let's focus on some of the details of using roles. Roles are used to simplify the process of granting and revoking user privileges. A role is a set of privileges that allows a user to access objects (see SQL code). Another interesting query to view could provide information about the default privileges assignment. This could be done by querying the list of all privileges with restrictions only to the PUBLIC grantee. It is highly recommended that privileges be revoked from the PUBLIC. So from this we could derive the most popular method used to hack the database. This is to obtain the highest privileges or the most powerful role in the system.

Oracle Advanced Security

OAS is a collection of security features related to Oracle Net Services, Oracle Database, Oracle Application Server and Oracle Identity Management infrastructure. It provides defenses against most common security threats. Eavesdropping, data theft, data tempering, falsifying user identities and password-related threats. Eavesdropping is the illegal interception of conversations by unintended recipients. This is the method used by an intruder once data is sent over an insecure network (de facto whole Internet). However even in a de-militarized zone network sniffers could be used to capture secret communications. Data Tempering means to compromise data integrity as it is moved between sites. User identify falsifying is an attack vector based on the premise that an attacker can pretend that he/she is someone else. Another type of attack in this group is to hijack the connection of the user.

Oracle Advanced Security Secure Sockets Layer Authentication

Oracle Advanced security supports both *Secure Sockets Layer* (SSL) and *Transport Layer Security* (TLS) protocols. The SSL protocol is authentication and encryption method that enhances TCP clients with secure services. This protocol was originally developed by Netscape Communication Company to secure the HTTP protocol communication between client and server. This still remains its primary usage. The SSL protocol is on based on IETF standard RFC-2246 under name *TLS* (*Transport Layer Security*). Each side in the communication gives proof of identity with a digital certificate (encrypted block of data). A certificate is validated by a trusted third-party which then verifies the communication between identity and a given encrypted key. This third-party is called Certification Authority (see *http://www.openssl.org*) for more details. Using this feature ensures encrypted connections between clients and servers, and it could also be used to validate a secure client/server database connection. This feature

Frequently Used Terms in Cryptography

- Encrypt – Scrambling data to make it unrecognizable.
- Decrypt – Unscrambling data to its original format.
- Cipher – Another word for algorithm.
- Certificate Authority (CA) – third-party, e.g. Verisign, CyberTrust or RSA.
- Digital Certificates – Consists of private key and public key the private key has to be verified by CA.

Public-Key Cryptography Standards (PKCS) Support

Author of this standard is RSA Data Security Inc. [8] RSA is part of EMC Corporation [9] – American manufacturer of software and security management and storage systems. RSA has the patent for RSA asymmetric key algorithm. RSA research resulted in PKCS standard. This defines the industry standard for promoting and facilitating the use of public-key techniques. Today there are fifteen PKCS standards. For our purposes the most relevant are PKCS #15, #12 and #10. PKCS #15 is the standard that enables users to use cryptographic tokens to identify themselves to multiple, standard-aware applications regardless of the application's cryptoki (or other token interface) provider. PKCS #12 is standard of format to store X.509 certificates and private keys. Finally there is PKCS #10 to generate certificate requests. The hardware storage of credentials is conforming to PKCS #11 specification.

is great in case of fraud attempts by sniffing the network communication. It also improves the reliability of the authentication process. There are some basic steps in Oracle SSL communication. There are two main parts of SSL communication. The first is SSL handshake, and second is the actual authentication process. The SSL handshake consists of following steps. The client and the server establish a set of authentication, encryption and data integrity algorithms used for exchanging messages between network nodes. During an SSL handshake, for example, the two nodes negotiate to see which cipher suite they will use when transmitting messages back and forth. The server sends its certificate to the client and the client verifies that the server's certificate was signed by a trusted CA. This ensures proof of the server's identity. If client authentication is required, the client sends its own certificate and server verifies that the client's certificate was signed by a trusted CA. Next client and server exchange key information using public key cryptography. Based on this information, each generates a session key. A session key is shared by at least two parties. It is used for the duration of that particular communication session. This improves the security via cracking the session key due to frequency of session key change. The next part is the authentication process. At first the user initiates an Oracle Net connection to the server by using SSL. SSL performs a handshake between client and the server and if the handshake is successful, the server verifies that the user has the appropriate authorization to access the database. To make this things work there is a Public Key infrastructure (Also Known As PKI) in the Oracle Environment. PKI ensures trusted relations for the entire organization. The PKI that used by Oracle is based on RSA Security, Inc., Public-Key Cryptography Standards. [10] PKI is a robust public key system that was designed to utilize *single-sign-on* feature and provide digital ID. In contrast to private-key or symmetric-key cryptography that requires a single, secret key that is shared by two or more parties public-key cryptography

makes the public key freely available. The public key is used to encrypt messages that can only be decrypted by the holder of the associated private key. The private key is securely stored together with other security credentials in an encrypted container called a wallet. A wallet is a data structure used to store and manage security credentials for an individual entity. A Wallet Resource Locator (WRL) provides all the necessary information to locate the wallet. Public-key algorithm has a weakness that could be exploited in the absence of the communicating party's identity verification. This type of attack is called the man-in-the-middle. It's based on the idea that an intruder captures the public key of the sender. Then uses his/her own public key to send messages to the receiver. When the receiver responds, the intruder is able to re-encrypt the message with public key of sender and forwards the message to the sender. It gives the intruder the possibility to read the message (eavesdropping). To prevent this type of attack, it is necessary to verity the owner of the public key through authentication. This is the point where CA comes to play. The CA issues public key certificate that contains information about the principal entity's security credentials and encrypts a message with private key. This provides an opportunity to verify that the key was issued by the CA. In an Oracle Environment the PKI components include Certificate Authority, Certificates, Certificate Revocation Lists, Wallets and Hardware Security Modules. CA issues the certificate signed with its own private key. To verify the certificate was in fact issued by the CA its public key is used. The certificate is created only in the event that the entity's public key is signed by a trusted CA. Certification Revocation (CLR) lists is a list where CA stores expired or invalid certificates. The server searches for CLRs in the following locations: local file system, Oracle Internet Directory and CRL Distribution Point. Oracle Wallet is used for generating a public-private key pair and create certificate request, store a user certificate that matches with the private key and configure trusted certificates. And finally Hardware Security Modules are devices that stores cryptographic information, such as private keys or to perform cryptographic operations to off load RSA operations from server. There are two types of this device: server-side (stores keys) and client-side (smart card readers). To improve the security, it is possible to use additional authentication methods (RADIUS, Kerberos). SSL brings some issues with using Firewalls. Firewalls that perform packet inspections must have this feature disabled otherwise they are unable to read the packet. In this case Oracle Net Firewall Proxy kit can provide some specific support for database network traffic. U.S. government regulations prohibit double encryption. This is the reason why this will not work concurrently with SSL encryption or another encryption method.

OAS SSL Authentication Practice

When implementing advanced security features there are some options. You can utilize third-party software like Kerberos or RADIUS; it is possible to use Secure Socket Layer (SSL). Oracle has a specific set of tools used to manage certificates, wallets and certificate revocation lists. Oracle's Wallet Manager is an application that stores security credentials in the users Oracle wallets. This Manager can be used to create public and private key pairs, store and manage user credentials, generate certificate requests, store and manage certificate authority certificates, upload and download wallets to and from an LDAP directory and create wallets to store hardware security module credentials. OWM could be found on UNIX in `$ORACLE_HOME/bin/owm`. See Figure 1. Another important tool is Oracle's Net Manager. It is well known common database administration however when oracle's advanced security is installed Oracle Net Manager allows one to configure strong authentication, network encryption and check summing for data integrity. See Figure 2.

Oracle Wallet Manager

Is used as place to store, manage and edit authentication and signing credentials. This includes private keys, certificates and trusted certificated needed by SSL. It could also be used as storage for credentials for a hardware security module. To protect the content a password must be chooses that complies with the Password Management Policy guidelines (min. 8 chars, alphanumeric required). It could be used to store certificates (X.509) under Triple-DES encryption. For optimum wallet access and administration Oracle provides an option to store your user profile in the registry. [6]

OWM enables one to store multiple certificates in each wallet supporting SSL authentication, S/MIME signature, S/MIME encryption, Code-Signing and CA Certificate Signing. The process of obtaining a new certificate consists of several steps. First is to generate a unique private/public key pair. The private key stays in the wallet and the public key is sent with the request to a certificate authority. Once the certificate authority generates and signs the certificate it could then be imported into the wallet that has the corresponding private key. There is X.509 Version 3 Key Usage extension to define Oracle PKI certificate usage. Oracle's Wallet also supports LDAP. This feature allows users to retrieve their wallets from LDAP directory. This allows users to access wallets from multiple locations or devices. Only functional wallets could be uploaded to LDAP. To protect access Oracle's wallets are stored in LDAP there are passwords to access Wallets from LDAP and another to open the Wallet itself. It is recommended that separate password be used where neither one can logically be derived from the other. The description of creating a new wallet could be found in [9 Chapter 9.3]. Here is a short overview of possible actions: Create wallet (standard or stored on a hardware security module [PKCS #11]), *Open*,

References

- [1] Oracle Database Advanced Security Administrator's Guide 10gR2.
- [2] SSH (O'Reilly) – cap. 1.6.6 Secure Socket Layer SSL.
- [3] Chey Cobb, CISSP – Cryptography for Dummies (Wiley) 2004.
- [4] The Oracle Hacker's Handbook.
- [5] Transparent Data Encryption stores key unencrypted in the SGA, *http://www.red-database-security.com/advisory/oracle_tde_unencrypted_sga.html*.
- [6] Oracle Database Platform Guide 10gR2 for MS Windows (32-bit).
- [7] Public Key Cryptography Standard, *http://en.wikipedia.org/wiki/PKCS*.
- [8] RSA Security, *http://en.wikipedia.org/wiki/RSA_Security*.
- [9] EMC Corporation, *http://en.wikipedia.org/wiki/EMC_Corporation*.
- [10] Oracle's special users: SYS, SYSTEM, INTERNAL and PUBLIC, *http://www.adp-gmbh.ch/ora/misc/sys_system_internal.html*.
- [11] Oracle password file (orapwd utility), *http://www.adp-gmbh.ch/ora/admin/password_file.html*.

Close, *Upload/Download to/from LDAP Directory*, *Save*, *Save As*, *Delete*, *Change password*, *Use Auto-Login*, *Manage Certificates*. Let's have a quick look at the last action. There are two types of certificates to manage. There are User certificates and trusted certificates. The first step will be to Request a certificate (there is difference in key length `512b-4096b`). After the Certification Authority processes and approves your request for certificate it is possible to import the certificate into the wallet. =

Transparent Data Encryption

Transparent Data Encryption works to enhance database security at the Operating System Level. This feature can protect the Virtual Private Database [4]. This type of attack is based on raw access to data files. Data encrypted using Transparent Data Encryption could be helpful in a regulatory issue. Also there is no need to request users to store encryption keys. It also simplifies the development process because it is not necessary to make any deep changes in applications that access data. There are some cases where TDE it is not recommended (indexes, BLOB and utilities with raw access to data). While TDE is good it is not the ultimate solution. The master key is stored unencrypted in the SGA [5]. This should serve as a warning that even the most complex security defense is useless when there is someone who has a bright idea and a genius brain. Let's have a look what we already know. The usage of Wallet is to store the master keys. There are specific commands to work with Wallets. For the sake of this article let's assume say there is only one key that opens and allows access to our wallet. There is also a feature to enable auto-login into the wallet. The mkwallet is a command line utility that allows wallet management without a GUI. Just after the wallet is setup and it is opened with master key we can start to create and use the transparent data encryption. To set a new master key issues this command: ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY password. To make the encrypted data accessible just issue ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY password. Right after opening the wallet TDE uses standard DDL e.g. CREATE TABLE table_name (`column_name column_type` ENCRYPT, …), ALTER TABLE table_name MODIFY (`column_name column_type` ENCRYPT,…). Access to all encrypted columns in the database could be done by statement ALTER SYSTEM SET WALLET CLOSE. To get information of what columns in database are encrypted just use following views: `DBA_ENCRYPTED_COLUMNS`, `ALL_ENCRYPTED_COLUMNS` and `USER_ENCRYPTED_COLUMNS`.

Network Data Encryption and Integrity

Let's recap what we have learned in this article. Some time ago when Oracle was in version 8.1.7 there was restrictions on the exportation of cryptosystems from the US. That's why there are several versions (in 8.1.7) these are Domestic, Upgrade and Export. Each version is different in the key length that is uses. There is well known post-attack on Oracle action. This consists of Rootkit deployment upon successful infiltration. Another is just to distract data integrity in two possible ways data modification attack (this mean to alter in some manner values stored in database) and replay attack (multiple usage of normally disallowed transaction). Oracle offers defenses against these types of attack by using checksumming packages. I mean the usage of MD5 (*Message Digest 5*) or SHA-1 (*Secure Hash Algorithm*) SHA-1 to prevent and discover this type of attack. The principle is to use hash algorithms to create a checksum. On the base of this checksum there is possibility to discover if the data integrity has been altered on its way between server and client. This could prevent man-in-the-middle attack). The principle is based on session key by Diffie-Hellman negotiation algorithm. Then OAS combines the shared secret and the Diffie-Helman session key to generate a stronger session key designed to defeat a man-in-the-middle attack. To activate Encryption and Integrity the server selects which algorithm to use from those specified in the sqlnet.ora files. There are four levels of security between client and server. These are REJECTED (minimum amount), ACCEPTED (default), REQUESTED and REQUIRED (maximum amount). As an option the encryption seed can be set. ●