Introduction Oracle Database Security

Leeland Artra

Cellworks Project

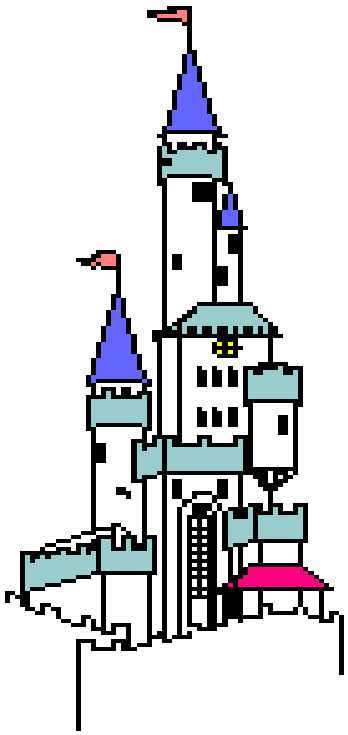
University of Washington

Notes:

This presentation is online at: http://www.sasag.org/1999/03/199905_ora_sec_talk.pdf

You can reach me for questions or comments at:**leeland--AT--GreyDragon-dot.com**

Security & Networking



Did you lock the Gate?

Notes:

Syllabus

- Introduction to database security
- Definitions and nomenclature
- Oracle Privileges
- Oracle security methods
- Common security holes
- Evaluating security on Oracle
- SQL*Net Servers
- Summary

Notes:

Introduction

- Security is available to protect your major asset: DATA
- Many levels of security are available within Oracle
  - Account security
  - Access security
  - System-level security
- This section is mostly from the perspective of DATABASES and Oracle

Notes:

In the last few years a growing trend has emerged. The **data** is the most valuable asset, not the computers the data is stored on. So computer security has been a major area for development.

Oracle, like the rest of the computer industry, has provided a large sampling of security settings for the database. Especially with Oracle7, where they introduced 80 plus attributes that can be controlled individually.

The basic levels of security break down to account, access, and system level security. Account security is the right to connect to the database. The connection right or privilege is identified by an account. What that user or process can do is then governed by the access permissions associated with each object. Finally the overall Oracle system has its own security to control new table creation, or deletion. These combine to present a very complex and configurable security model.

Who Is Attacking

Oddly enough database hacking has not really been done in mass by the hacker communities (YET). You still need to worry about:

- Underpaid accountants
- Free thinking engineers
- Disgruntled employees (who are hiding the fact that they are upset)
- Over competative programmers
- Rival companies
- Unknown cracker/hacker on a random attack

Notes:

It is odd that the hacking community has not yet attacked databases. There is almost no information on hacking databases available publicly or even semi-privately. This is going to change as databases become more and more sophisticated. Additionally as databases grow more and more data that is of interest to the hackers will be placed into the databases. Thus, growing interest in how they work and how they break.

Since databases have not come under heavy attack yet there most likely are hundreds of security holes that we are unaware of today. The best you can do is cover up the ones that are known and watch out for any new ones that may be discovered.

Although the unknown attacker is a low probability today you still have to worry about that underpaid accountant in accounts receivables that sells financial information to competitors. Or how about that engineer who really thinks that everyone should be on a level playing field and is generously sharing product designs. Or that really upset person who didn't quit but is now in product development and selling product designs from within the company to a friend who has connections in the European technology marketplace. Not to mention those less than honorable programmers from an ally company that really are not good at doing their own work and are good at borrowing other peoples' work. Don't forget that new marketing guy came from a competitor, and is secretly still working with that competitor by selling spatial data from the demographics firm back to his old company for a handsome fee.

You can find unscrupulous individuals who will do anything to advance the position of their company and gain them a competitive advantage to do business, just about anywhere. That person could be your co-worker who you go to lunch with every day to gossip about the daily events. It could be your best friend who comes over to your house with his wife and their kids for an afternoon barbecue. The point is that it could be anyone. In today's world, we must trust very little, but have enough confidence in our abilities and practices so that we don't worry about it too much.

Sensitive Data

:Examples of what might be considered sensitive data

Enterprise	Information	Enterprise	Information
Financial:		Medical groups	
Banks	- Customer accounts	**Educational:**	- Students' grades
Savings and loans	- Credit ratings	Universities	- Personal information about students
Insurance		Colleges	
Credit reporting		**Military:**	- Emer gency procedures
Industrial:			- Secret weapons
Manufacturing	- Characteristics and announcement dates of new products		- Numbers and distribution of military personnel
Mining	- Production processes	**Commercial:**	
Medical:		Retail sales	-Mailing lists
Hospitals	Medication data	Distribution	- Selling strategies
Clinics Patients'	health histories	**Airlines:**	- Reservations

Notes:

These are just a few of the types of enterprise sensitive data. Obviously there are a lot more general and site specific types of sensitive data.

Aspects to Security

- Legal, social and ethical aspects
- Physical controls
- Policy questions
- Operational problems
- Hardware controls
- Operating system security
- Database administration concerns

Notes:

There is no doubt that security is a very broad issue. Some of the foundations upon which you must build a security model are:

- Legal, social and ethical aspects - Are the monitoring and authorization plans legal in your state, county, country? Are you violating ethics (reading private email, system usage, web surfing)?
- Physical controls - Is the system site secure. Can someone take the actual disks on which your data is stored easier than trying to break into the system?
- Policies - Does your company have an acceptable use policy? Do you have a service level agreement with the managers that you have to meet?
- Operational problems - are sections of the database in Antartica?
- Hardware controls - is this version of the CPU board secure? Is there a hardware security bug in the ethernet card?
- Operating system security - You should be very familiar with these concerns.
- Database adminstration concerns - Do all the triggers work? Are the grants correct?

OS vs. Database Security

- More objects must be protected in a database
- The lifetime of the data is normally longer in a database
- Database security is concerned with a higher level of granularity, such as table, row, or cell
- In database systems objects can be complex logical structures, a number of which might map to the same physical data objects
- Database security is concerned with the semantics of data, as well as with its physical representation.

Notes:

Operating System Security

- To get local access to the database first must go through the server OS
  - OS security is entirely by-passed when the database is accepting remote connections
- First step in securing the database is securing the server at the OS level:
  - TCP Wrappers
  - Good passwords
  - etc.

Notes:

Obviously the server itself is a target point for any security violations. Hence, make sure it is secure.

Later in this course we'll discuss the remote connection possibilities. For now we are going to explore only a local user connection from the OS.

There are a lot of things that can be done. But, there are small easy steps that cover a lot of ground like: security patches, TCP wrappers, good passwords, no open accounts, etc.

Database Security Layers

* User
* Physical Security
* User Application
- Session Encryption
* Transaction Control Program - Login, Menus
* DBMS User Views - SQL - GRANT, REVOKE
- Operating System - File Management System
- Data Block Encryption
* Database

Notes:

Depending on how paranoid you may be you can have all or some of these layers. In general you will always have the user, physical, application, DBMS user and database layers. The rest are mostly optional.

Starting with the user (without whom we'd all be out of work) we first look at physical security. Physical security consists of doors, locks, card key entries etc. Physical security is to ensure that the computers and peripherals do not just walk off. If I wanted to attack your database I'd first check the simple things, like how secure is your hardware. If I can just walk off with the necessary disks and mount them on my own computer elsewhere I will. Then I would have all the time I needed to analyze your database and figure out how to get what I want out of it.

Next is the application being run. Whether you are using home grown applications or off the shelf products there is some security and integrity controls built into the application. The amount of security in this layer will vary wildly based on what it is.

Next is a possible encryption of the connection to the database (e.g. products like HTTPS or SSH).

Next is the transaction control layer, which is the application and OS layer on the server. Like the user application end the amount of security in this layer will depend on what is being used. Generally this layer is better designed for security then the user application layer.

Next is the DBMS itself. Oracle has extensive security option built in. But, they are only as effective as they are implemented.

Next is the OS again. This layer is the kernel level of the OS. This layer is the controls and access to the actual physical devices. The one that is almost always present is the file system. This layer may not be present if you are using raw devices.

Next is additional encryption. This is not easy to implement. If this layer is present it would offset the security risk of someone walking off with the physical drives only.

Finally is the actual database itself. Security here is dependent on design (layout).

Security Mechanisms

- External procedures: Security clearance of personnel, Protection of passwords, Information classification and security policy formulation, Application program controls, Audit
- Physical environment: Secure areas for files/ processors/ terminals, Radiation shielding
- Data storage: Data encryption, Duplicate copies
- Processor software: Authentication of user, Access control Threat monitoring, Audit trail of transactions
- Processor hardware: Memory protection, States of privilege, Reliability
- Communication lines: Data encryption

Notes:

Just as a few examples of possible mechanisms.

Definitions

It is difficult to talk security without some specific, agreed upon definitions:

- **Information security-** is the protection of information against unauthorized disclosure, alteration or destruction
- **Database security-** is the protection of information that is maintained in a database
- **Privacy-** term used broadly for all the ethical and legal aspects of personal data systems - systems that contain information about individuals

Notes:

Definitions 2

- **Authorization-** is the specification of access rules about who has what type of access to what information
- **Authoriser-** is the person who writes access rules
- **Protection-** is the term that refers primarily to techniques that control the access of executing programs to stored information
- **System integrity-** is the ability of a system to operate according to specifications even in the face of deliberate attempts to make it behave differently

Notes:

Definitions 3

- **Access control-** The process of ensuring that information and other protected objects are accessed only in authorized ways
- **Intentional resolution-** aims at controlling the actions performed on the data once it is legally accessed
- **Information flow control-** aims at preventing security leakage as information flows through the computer system

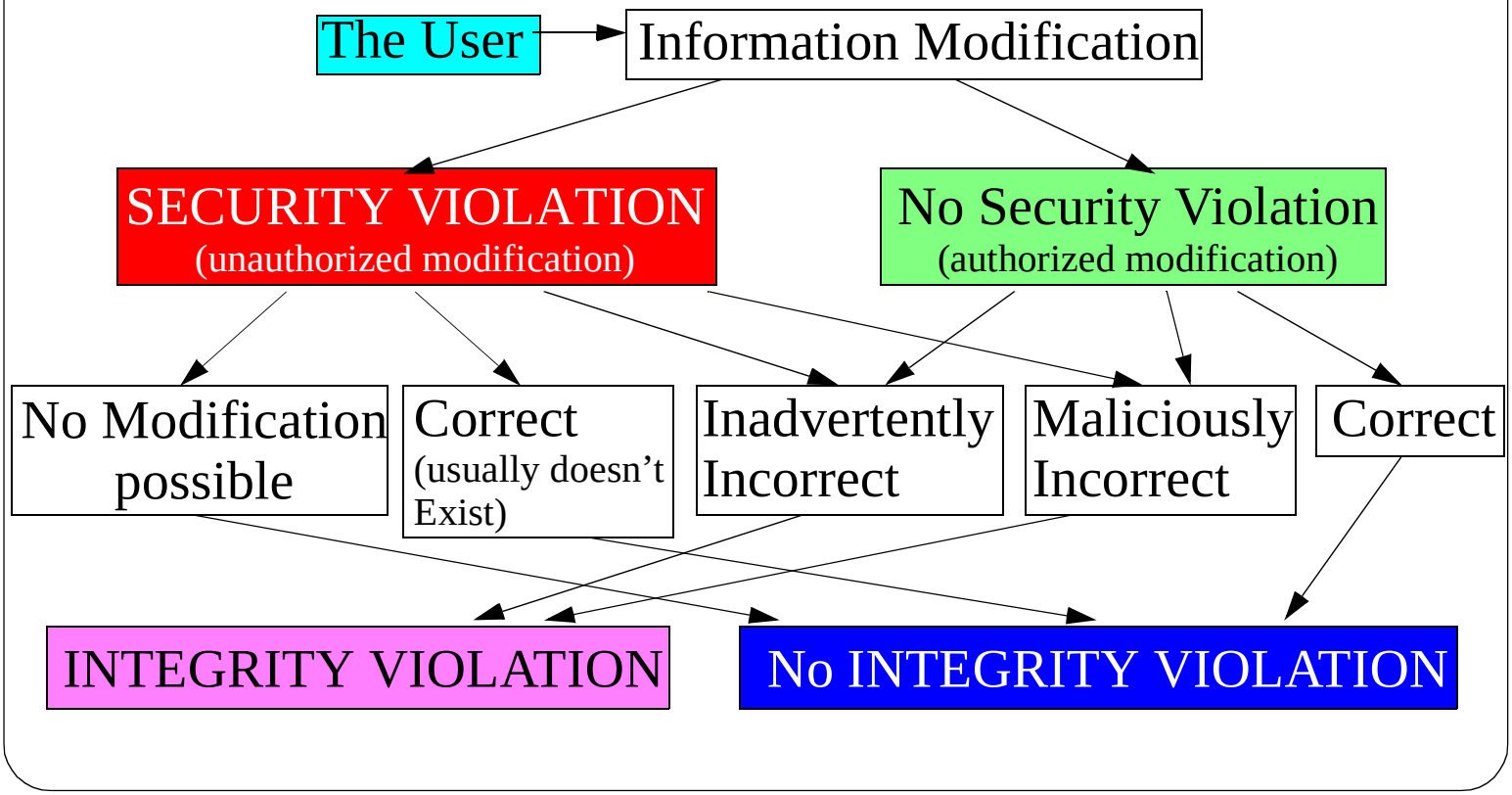
Notes:

Definitions 4

- **Integrity-** term that is also applied to data and to the mechanisms that help to ensure its correctness
- **Semantic integrity-** is the part of data integrity concerned with the correctness of database information in the presence of user modifications
- **Data security and integrity** are also objectives of **auditing**, which is the examination of information by persons other than those who produced the information

Notes:

Security vs. Integrity



Notes:

I have been blind sided by someone asking me to define the relationship between security and integrity. After an hour debate on what the definitions were I was then told that I didn't have a clue and that they were the same. After another hour debate we had made no progress. So I was forced to remove the definitions and use the term "security" generically throught the security policy docuement. (The debate was with my boss. So obviously you knew who was going to win from the beginning. However, I felt obligated to try.)

This diagram was sent to me some time ago by a some I told this story to. I do not remember who sent it. But, I am enternally grateful to whomever originally made it. Also, I am now ready to win the next debate on this subject!

In this diagram the security violations are at the second level. The path from a security violation to a modification is through an error in the security constraints on the database. Hence, although it was a security violation the change was allowed.

Database Security Categories

Oracle database security can be generally classified into two distinct categories:

- System security
- Data security.

Notes:

System security is NOT operating system security. It is specifically the internal database operating system. Remember Oracle is an operating system unto itself.

Data security includes integrity.

DBMS System Security

- DBMS system security includes the mechanisms that control the access and use of the database at the DBMS's system level. DBMS System security includes:
  - valid username/password combinations
  - the amount of disk space available to the objects of a user
  - the resource limits for a user
- DBMS System security mechanisms check:
  - whether a user is authorized to connect to the database
  - whether database auditing is active
  - which system operations a user can perform

Notes:

So if the Oracle is an operating system onto itself, it must have an underlying kernel of OS type instructions. This is where process controls, CPU usage, and security are implemented.

The security mechanisms are present for all operations. This is the same as trying to use UNIX to create a new directory. The UNIX kernel checks if your UID is valid, if your UID or GIDs have access privileges to the directory of the nature to allow the requested modification. It then performs the operation or issues an error (which is semi auditing). Remember that the database is a little more granular than a normal OS.

Data Security

Data security is the mechanism that control the access and use of the database at the object level.

Data security includes:

- which users have access to a specific schema object and the specific types of actions allowed for each user on the object (for example, user ARTRA can issue SELECT and INSERT statements but not DELETE statements on the EMPLOYEE table)
- the actions, if any, that are audited for each schema object

Notes:

This is the normal nature of security which most people are comfortable with and understand pretty well.

Remembering that the database acts similiar to an OS with a high degree of control the data security mechanism is similiar to read and modify access to a file system. The data security mechanisms are present for all operations. This is the same as trying to use UNIX to list a directory. The UNIX kernel checks if your UID is valid, if your UID or GIDs have access privileges to the directory of the nature to allow the requested read operation. It then performs the operation or issues an error (which is semi auditing).

Database Security

- Access controls include individual users as well as the "general public" (all users defined in Oracle)
- Data access is controlled by means of granting permissions on tables, views, and database procedures to a user
- Specific DBA objects (forms, reports and graphs) are public access
- The DBA can grant permissions on any databases, tables, views or database procedures

Notes:

Oracle has sophisticated security mechanisms, which enable the DBA to control access to sensitive data by an assortment of privileges.

The granting of permissions on tables, views, and database procedures to a user is based on the name the user uses to connect to the database.

A useful list of DBA objects and reports (views etc.) are granted select priviliges to PUBLIC. This means that any user that can connect to the database can access these objects. This provides a nice working environment from the start.

The DBA account is like the root account for UNIX, there is little it can not do. Remeber, in Oracle there is more than one DBA account, there is system, sys, and others that might be defined at your site.

Oracle Security

Oracle, a multi-user database system, security features control how the database is accessed and used. Oracle security mechanisms do the following:

- prevent unauthorized database access
- prevent unauthorized access to schema objects
- control disk usage
- control system resource usage (such as CPU time)
- audit user actions

Associated with each Oracle user is a schema by the same name. By default, each database user creates and has access to all objects in the corresponding schema.

Notes:

Oracle is a full multi-user database system. As such is contains numerous security features to control how data is accessed and manipulated. The security mechanisms provide the tools with which you can impliment your security policies.

The user schema, as discussed earlier in this course, is a logical collection of objects (tables, views, sequences, synonyms, indexes, clusters, procedures, functions, packages, and database links) that are associated (owned) by the database user.

Database Users

- User accounts allow access to the database
- They should provide for
  - a secure connection
  - useful access
  - adequate privileges to work
  - proper default settings

Notes:

Users are created in Oracle using the CREATE USER command. It allows for setting the password, default tablespace, temporary tablespace, quota, and profile.

If the optional settings are not specified the default values are used.

Revoking users is as easy as a 'DROP USER artra CASCADE;' The CASCADE can be dangerous since it removes all objects in the database owned by the user. Oracle prevents unowned objects by explicitly not allowing you to drop a user unless that user no longer owns any objects or unless you specify the CASCADE option. Since CASCADE removes all object from the user's schema it might be far more reaching than you might have believed. In short, review the users schema before running a drop user with cascade.

Database Roles

- Named collection of privileges
- May be assigned to a user, or another role
- Roles are NOT users (can not login to database)
- Group logically associated privileges and allow privileges to be passed on by reference
- If role is altered or dropped all users who have the role are emmidiately effected
- Very much like a UNIX group (including role passwords)
- Excellent security mechanism

Notes:

Roles in the Oracle database act something like groups in the UNIX realm. Except that Roles can be granted privileges far in excess of anything a UNIX group could dream of doing.

Oracle provides for easy and controlled privilege management through roles . Roles are named groups of related privileges that are granted to users or other roles. The following properties of roles allow for easier privilege management:

- reduced granting of privileges - Rather than explicitly granting the same set of privileges to many users, a database administrator can grant the privileges for a group of related users granted to a role. Then the database administrator can grant the role to each member of the group.
- dynamic privilege management - When the privileges of a group must change, only the privileges of the role need to be modified. The security domains of all users granted the group's role automatically reflect the changes made to the role.

Roles, like UNIX groups, allow you to require a password to activate the role privileges. But, unlike UNIX the roles contain the privileges as well as the objects having access privileges for specific roles (or groups).

Oracle System Level Roles

- A Role is a job or access privileges
- Oracle 7 expanded CONNECT, RESOURCE & DBA Roles into 80 plus privileges
- Privileges can be grouped together into Roles
- Roles can be granted to users
- So now there are Roles called CONNECT, RESOURCE, & DBA (plus more) that have all the same privileges associated to them
- With GRANT allows blessings to occur

Notes:

For each kind of activity in Oracle there is an associated Privilege. Each Privilege can be individually assigned to a User OR to a Role. Then the Roles can be associated to the Users thus granting a whole slew of Privileges in one association.

If a Role or Privilege is GRANTed to a User or Role there is an optional clause ’with grant’ that can be tacked onto the command. This allows that user or role to grant the privilege or role to other users or roles. The allows for delegation of some duties. Oracle maintains a hierarchical tree mapping of the grants so that the user who granted cannot be altered by someone she granted the privileges to. Hence the DBA account SYS cannot be altered by any other account except for itself.

User Profiles

- Profiles are like Roles except they deal with resources like:
  - SESSIONS_PER_USER
  - CPU_PER_SESSION
  - CPU_PER_CALL
  - CONNECT_TIME
  - IDLE_TIME
  - etc.

Notes:

A user profile is a means to build up resource controlling limits into a single profile that can then be assigned to users. This allows for keeping road hog users under control. It is worth knowing that profiles are reactive not pro-active. So, for instance, a user launches a rather badly formed query. The system will chug away at the poor query until a limit is hit then the query will halt.

Authorisation / Privileges

- A user may have several forms of authorization on parts of the database (object privileges):
  - Select, Insert, Update, and/or Delete authorization
- A user may be granted authorization to modify the database scheme (system privileges):
  - Index, Resource, Alteration, and/or Drop authorization
- A user may be granted the authorization to share the authorization(s) with other users (WITH GRANT OPTION, WITH ADMIN OPTION)

Notes:

A privilege is a right to execute a particular type of SQL statement. Some examples of privileges include the: right to connect to the database (create a session); right to create a table in your schema; right to select rows from someone else's table; and right to execute someone else's stored procedure.

The privileges of an Oracle database can be divided into two distinct categories: System Privileges allow users to perform a particular systemwide action or a particular action on a particular type of object. For example, the privileges to create a tablespace or to delete the rows of any table in the database are system privileges. Many system privileges are available only to administrators and application developers because the privileges are very powerful. Object privileges allow users to perform a particular action on a specific object. For example, the privilege to delete rows of a specific table is an object privilege. Object privileges are granted (assigned) to end users so that they can use a database application to accomplish specific tasks.

There are only 16 Object privileges in Oracle version 8.0.4. A complete list of object privileges can be found in the data dictionary table TABLE_PRIVILEGE_MAP. Object privileges allow a user the right to manipulate the data within the object or, in the case of programs, to execute the program. These privileges are usually refered to as 'grants' and are object specific. A user granted insert on Leeland's mass spectrum table does not automatically have any privileges on any other table owned by Leeland or anyone else.

There are more than 80 system privileges available to authorise users in Oracle. The number of privileges will vary depending on the Oracle version being used. A complete list of privileges can be found in the Oracle data dictionary table called SYSTEM_PRIVILEGE_MAP. System privileges do not convey any rights for data access. Although a user may be able to create any view in any other user's schema, the new view will be owned by the user who's schema was specified. The creating user will still not be able access the view after it is created.

Granting Privileges

- Normally a DBA responsibility
- Grants can range from very general to very specific

The grant statement is used to grant privileges. The general form is:

GRANT *privilege* ON *object* TO *whom*
[WITH GRANT OPTION]

whom can be:

- Individual users
- PUBLIC
- A defined role

Notes:

The granting of privileges (also called permissions, grants, or permits) is normally a DBA responsibility.

Privileges are granted to users so that users can access and modify data in the database. A user can receive a privilege in two different ways:

Privileges can be granted to users explicitly. For example, the privilege to insert records into the EMPLOYEE table can be explicitly granted to the user ARTRA.

Privileges can be granted to roles, and then the role can be granted to one or more users. For example, the privilege to insert records into the EMPLOYEE table can be granted to the role named CLERK, which in turn can be granted to the users ARTRA and ADAMS.

A privilege that is granted using the 'WITH GRANT OPTION' can be passed on to other users and roles by the grantee. If the grant includes 'WITH GRANT OPTION', the grantee can further grant the object privileges to the object privileges to other users. **Object privileges granted WITH GRANT OPTION are revoked when the grantor's privilege is revoked**. A privilege WITH GRANT OPTION cannot be granted to roles (discussed later).

Displaying DBA Object Privileges

Available to DBA	Description
DBA_TAB_PRIVS	**All privileges on objects in the database**
DBA_COL_PRIVS	**All privileges on columns in the database**

Notes:

Displaying User Object Privileges

Available to the User	Description
USER_TAB_PRIVS	**Privileges on objects for which the user is the owner, grantor, or grantee**
USER_TAB_PRIVS_MADE	**All privileges on objects owned by the user**
USER_TAB_PRIVS_RECD	**Privileges on objects for which the user is the grantee**
USER_COL_PRIVS	**Privileges on columns for which the user is the owner, grantor, or grantee**
USER_COL_PRIVS_MADE	**All privileges on columns of objects owned by the user**
USER_COL_PRIVS_RECD	**Privileges on columns for which the user is the grantee**

Notes:

Displaying Object Privileges

Available to ALL users	Description
ALL_TAB_PRIVS	**Privileges on objects for which the user or PUBLIC is the grantee**
ALL_TAB_PRIVS_MADE	**User's privileges and privileges on user's objects**
ALL_TAB_PRIVS_RECD	**Privileges on objects for which the user or PUBLIC is the grantee**
TABLE_PRIVILEGES	**Privileges on objects for which the user is teh grantor, grantee, or owner, or PUBLIC is the grantee**
ALL_COL_PRIVS	**Privileges on columns for which the user or PUBLIC is the grantee**
ALL_COL_PRIVS_MADE	**Privileges on columns for which the user is the owner or grantor**
ALL_COL_PRIVS_RECD	**Privileges on columns for which the user or PUBLIC is the grantee**
COLUMN_PRIVILEGES	**Privileges on columns for which the user is the grantor, grantee, or owner, or PUBLIC is the grantee**

Notes:

Yes it is a bit dense. But, I wanted to get it all onto the one slide. Luckily you can look it up and read it.

Removing (Revoking) Privileges

- DBAs can revoke any privilege(s) from any user
- Grantors can revoke privileges from users to whom they have granted privileges

The revoke statement is used to remove privileges. The general form is:

REVOKE *privilege* ON *object* FROM *whom*
[CASCADE CONSTRAINTS]

whom can be:

- Individual users
- PUBLIC
- A defined role

Notes:

The revoke statement is very similar to the grant statement.

The notable difference is the ending option CASCADE CONSTRAINTS, which is the option to revoke 'references privileges'. This means that the revoke will also walk through all constraints on the tables to make sure than the changes do not have any references if so they too shall be revoked.

If a privilege is revoked from a user and that user had granted the privilege to other users then the other users will also loose the privileges as well.

Tying Database Accounts to the OS

- Oracle is an OS on a OS
- Oracle can use the Server's OS to identify a user
- The Oracle account must be specially named
- The name prefix can be specified (or NULL)
- The OS authentication is based on the client HOST not the Server
- Can open up some security holes

Notes:

Oracle is an OS unto itself. But, it must run on another OS. So users must identify themselves to Oracle with a username and a password. But, Oracle can check with the Server's OS for valid account data. Then the User can just connect.

To set up this connection you create a user account with a special prefix identified by the Oracle parameter OS_AUTHENT_PREFIX. Then Oracle allows that user (who is already connected) to simply place a slash '/' in place of their username/password combination.

However, in creating a user account a password has to be specified. So someone can still connect directly to Oracle using the oracle_username/password combination. This can be bypassed by setting an impossible password (see passwords). Or you can prevent this by specifying the IDENTIFIED EXTERNALLY clause which causes oracle to check the account you are coming from against the user account you are trying to connect to. If they do not match the connection is refused.

Oracle Passwords

- Protect accounts from unauthorized access
- Cryptographically stored
- DBAs can become other USERS
- Set in the CREATE USER and ALTER USER commands
- Can be set for ROLES
- Validated like UNIX passwords
- Can be specifically set using VALUES clause

Notes:

The user account passwords are every bit as important as the Server OS level passwords. They provide for a level of authentication that the person accessing the account has the privilege to do so.

Oracle passwords are encrypted similar to UNIX passwords. The cryptographic hash is stored in the data dictionary.

Knowing how the passwords are stored makes for some interesting abilities. For example the DBA could view the data dictionary table dba_users and save off the password hash. Then they can set the password for that account to anything they want using the ALTER USER <soandso> IDENTIFIED BY 'cheat'; Now they can log onto that account all they want. When they are finished they can reset the password to what it was by issuing the command ALTER USER <soandso> IDENTIFIED BY VALUES 'CRYPTOGRAPHICHASH'; and everything is back to normal.

Passwords may be assigned to ROLES. This causes the password to be required whenever the user attempts to shift to that ROLE for a privilege.

Passwords are collected and encrypted then the resulting hash is compared to the stored value like UNIX password authentication.

The fact that Oracle authenticates passwords like this means, that by using the VALUES clause, impossible passwords can used to replace usable passwords (e.g. 'no way'). This combined with the System OS authentication would cause each user to only connect from the local machine when they are logged in as themselves.

Remote Account Access

- Allowing remote connections opens up potential hazards
- With remote connections the OPS$ account alters flavor
- Account names being the same a remote user can access a local user’s account in Oracle
- If a user is a DBA on a remote host they can ’connect internal’ to your host
- At least turn off SQL*Net OPS$ connections

Notes:

Once you activate the SQL*Net listener you open up a whole slew of additional considerations. You must consider the network connections and their implications.

If you have remote database links, and are using the OPS$ host account linking, a connection’s authentication then comes into question. Should the OPS$ name of the originating account be checked against the local OS or the remote OS. If a user with multiple accounts, all the same name connects through SQL*Net using the ’/’ identifier it will pass without a problem. But, if an entirely different user on another host has the same username as the local user then that alternate user can connect just as easily bypassing all the security.

This gets even more hair raising when you think about a remote user with DBA privileges. That user can use the connect internal clause on their machine against the local host and get a connection with complete DBA authority.

This can be stopped by specifying to the SQL*Net listener ’orasrv dbaoff opsoff’. This will turn of remote DBA connections and remote OPS$ connections.

Using Views for Security

Views can be thought of as a means of providing a user with a "personalized" model of the database.

A view:

- can hide data that a user does not need to see
- is assembled with the permissions of the owner of the view not the user executing the query

A view’s ability to hide data serves both to simplify usage of the system and to enhance security.

Notes:

Oracle definitions of views are assembled with the permissions of the owner of a view. Access to views is the same as access to tables. So if sensitive data needs to be hidden while other data needs to be published a view is an excellent mechanism to meet this requirement. So although a user may be denied direct access to a relation, the user may be able to access part of that relation through a view.

Using Roles for Security

- Protecting objects with roles reduces a large security problem
- 3rd party (or in house) applications are designed to protect data integrity (usually by extra ‘security’)
- Instead of managing security in the application use password protected roles
- Applications can have an internal password or query user for password to set a role:

  SET *role* IDENTIFIED BY *password*
- Permissions end when application terminates

Notes:

Most software products are designed so that a user must provide some user name and password in order to access data within the database. One of the most common mistakes made when deploying a new application to users resides in letting the client-side application code manage privileged access to specific data, as well as maintain much of the data's integrity. This should be a major concern for security conscious administrators. What would happen if a user were to connect to the database from Microsoft Access, SQL*Plus, or some other 3rd party product? Even the best of intentions can wreak awesome havoc. Or even worse, what might happen if someone were to hijack another's username/password combination, and interact with the database outside the bounds of a seemingly “protective” desktop application? The possibilities are quite imaginable for those of us with a little experience. Critical data + vulnerability = Disaster!

It is not good enough to manage user access to database objects simply through a software application. By taking advantage of password protected roles, we can eliminate some of the worry. The idea is that an application will have a standard user login window where the user will supply a name and a password. This will be good enough to connect them to the database. Once the user is connected, and before any SQL on application objects is executed, the application uses an internal or asks for a password that will set a given Role, thus giving them access to the necessary objects. When the user ends the session (i.e. exits the application), they will no longer have access to any of those database objects.

When the role is set, the privileges acquired are only good for the user’s session. Once the user’s session ends, or when a user exits an application, the role is terminated.

To implement this in an application is very simple. Once the user logs into the application via a login name and password (connecting them to the database), a statement is executed to set the role providing a password. When the user exits the application, they will no longer have access to those objects.

This method closes a gaping hole in security as users will attempt to gain access to database objects only to find they must enable a role that must be accompanied by a password.

Common Security Holes

- SYSTEM user password = "manager"
- SYS user password = "change_on_install"
- connect internal/oracle = SYS user access
- Too much access to data dictionary (DBA.ALL_USERS view)
- DBA privileges assigned to development or 3rd party users

Notes:

The SYS and SYSTEM users are the most obvious holes, and should be the first holes to fix. The SYSTEM user has a default password of "manager". That's easy enough to change. What about the SYS user whose password is "change_on_install". I have gotten in a few times through SYS. So if your SYS and SYSTEM users still have those same passwords, change them now and on a regular basis. Keep track of what those passwords are as well. If you forget them, your in some pretty hot water.

The next item of concern is the INTERNAL connection through server manager. I am rarely surprised when a 'connect internal/oracle' gets me in as the user SYS with full privileges to do whatever I want. I've met a lot of DBAs who say "Oh, well, I guess I can't change the Internal password unless I recreate the database.. and by the way, who knows about it anyway?". That is totally wrong (as of Oracle8...;^), you can change the internal password very easily without causing any harm to the database, or network connectivity for clients. To change the Internal password do the following:

- Delete the SID: ORADIM80 -DELETE -SID sid (i.e. sid: PROD, etc.)
- Create the SID again and specify the new Internal password after the keyword INTPWD:
  ORADIM80 -NEW -SID sid -INTPWD password -STARTMODE auto -PFILE <database root>/database/init<sid>.ora
- Log in to server manager and connect... internal/<new_password>

Another very common mistake is that a user has access to much of the data dictionary, especially the view ALL_USERS. By examining that view someone might see other users who might have DBA roles. For example the view may reveal a user like the CONTEXT cartridge user (identified by the CTXSYS user being present). The CTXSYS user needs the DBA role, and has a default password of 'ctxsys' (another common mistake).

Along the same lines are the DBAs who think that one of the ways to keep users/developers from knocking on their door with minor problems or complaints about not enough privileges and how they need them to do this or that, is to grant them the DBA role.

Evaluating Security in Oracle

The grant statement and the permission system allow for data access to be restricted in the following ways:

- Operational restrictions (select, insert, update and delete permissions applied to some or all of the columns of a table)
- Restrictions by data value. These are implemented via views.
- Resource restrictions. These are the optional database privileges.

Notes:

The Oracle Server provides discretionary access control, which is a means of restricting access to information based on privileges. The appropriate privilege must be assigned to a user in order for that user to access an object. Appropriately privileged users can grant other users privileges at their discretion; for this reason, this type of security is called ‘discretionary’.

Oracle manages database security using several different facilities:

- database users and schemas
- privileges
- roles
- storage settings and quotas
- resource limits
- auditing

Think very hard about what privileges and roles you grant your users. Understand what it means to assign roles to users. Most often CONNECT, and RESOURCE will be more than enough on development servers, and usually way more for users who log into the database through applications to insert new spatial data, or post transactions to the cost-accounting systems. There are a very diverse set of ways to skin this cat. With all the options available it can be easy to meet most security and integrity requirements.

However, it is also easy to overlook a mistake that might disclose more privileges than expected. The Oracle security system is complex and diverse, so try to keep the complexity down. The simple rule of ‘keep it simple’ applies very well to Oracle databases.

Questions on Security?

Notes:

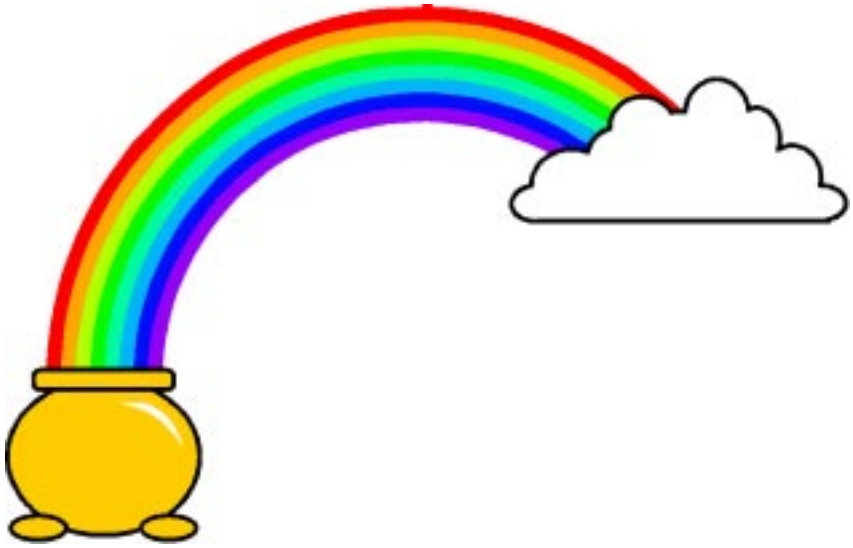
Security Summary

- The DATA in the database is the most important asset
- Security should be used to ensure data consistency and distribution
- User Account, Privileges, Roles and Profiles are available to provide required security controls
- If networking is turned on for the database then ensure its configuration meets the security plan
- Networking only works if the proper listener is turned on

Notes:

And To You

This is really your time to ask what ever question you like on Oracle, your site, etc.



Notes:

Credit where Credit is Due

This talk would not be in the great shape it is in without the generous help of:

- Curtis Presten, Collective Technologies (God of backups be praised!)
- O'Reilly & Associates (my book shelves are quite the menagerie ever since they started!)
- David Evans, Systems Administrator (a true friend indeed)
- Dr. Andrea Evans, Lecturer, Deakin University (who's insightful comments kept me up at night wondering if I should even open my mouth)
- Hal Miller, Senior Systems Administrator (for all the put downs to keep my ego in check)
- Many others who took the time to provide great feedback!

Notes:

Please Contribute

Although this is the end of this presentation, it is not the end of this talk.

Even if you don't have anything specific Please remember to fill out the Speaker evaluation forms

Remember Leeland Artra: 10,10,10,10,10,10

Notes:

Bibliography

This talk was assembled from in no particular order:

- "Oracle DBA Handbook" 7.3 Edition, by Kevin Loney; McGraw Hill; ISBN 0-07-882289-0.
- "Oracle Database Backup" by Curtis Preston; July, 1996 Sys Admin. Magazine.
- "Oracle7: Administrators" course; Oracle Training.
- "Oracle7: Database Backup and Recovery" course; Oracle Training.
- "Oracle Security", by Marlene Theriault & William Heney, O'Reilly 1998.

Notes:

Bibliography 2

- "Oracle7 Administrator’s Reference for UNIX" release 7.3, Oracle Corporation.
- "Oracle7 Installation Guide for MIPS ABI Systems" release 7.3, Oracle Corporation.
- "Oracle Unleashed" Second Edition; Sams Publishing; ISBN 0-672-31148-8.
- And a whole lot of not-so-fun-at-the-time trouble shooting, installations, and other calamities of the school of hard knocks.

Notes: