Database Administration:

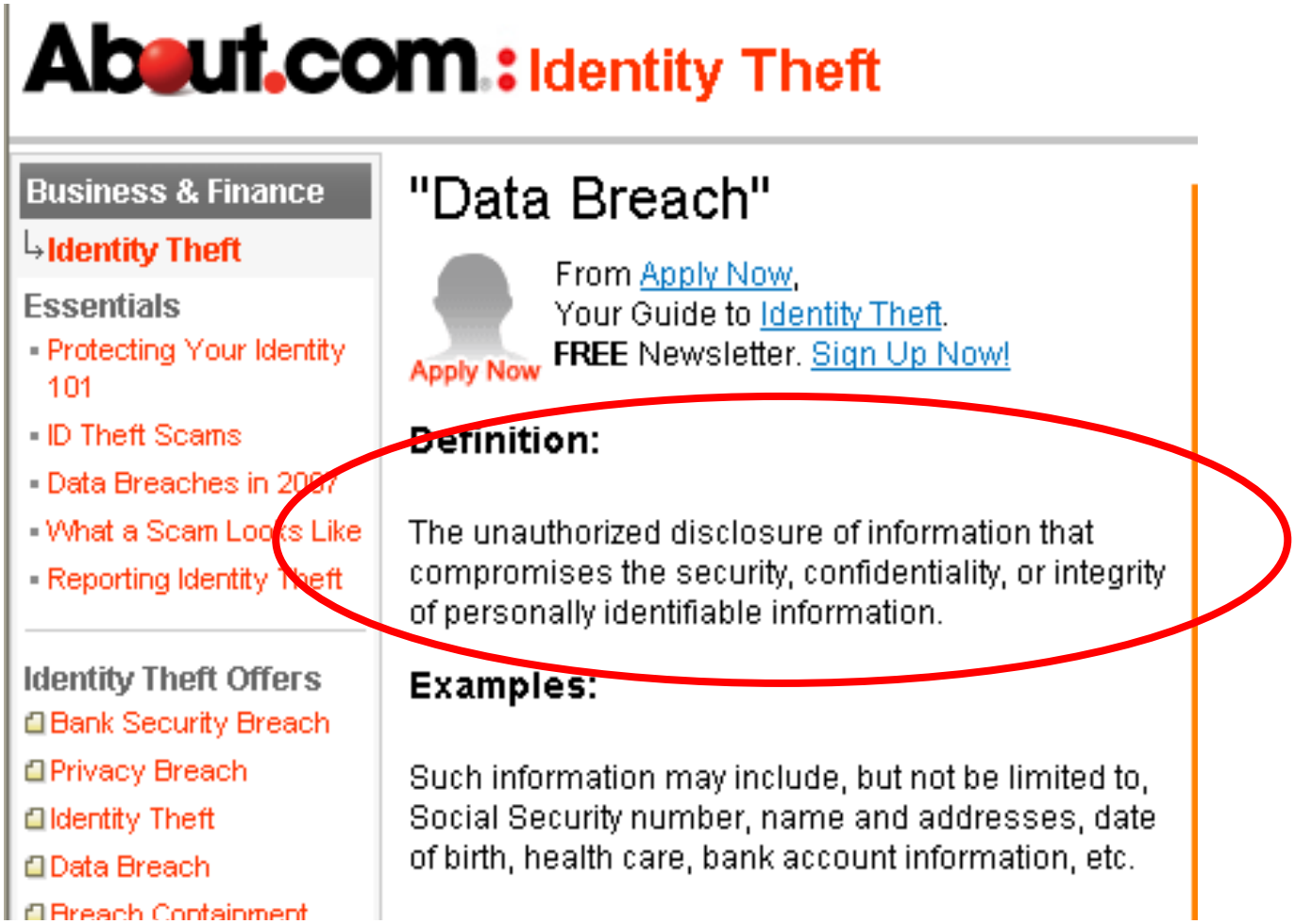
The Complete Guide to Practices and Procedures

Database Security

Agenda

- Data Breaches
- Database Security Basics
- Granting & Revoking Authority
- Authorization Roles & Groups
- Other Database Security Mechanisms
- Encryption
- SQL Injections Attacks
- Auditing
- External Security
- DBMS Fixpacks & Maintenance
- Questions

What is a Data Breach?



Data Breaches

- Data breaches dominate the IT newscape
- According to The Privacy Rights Clearinghouse, more than 544 million records have been breached between January 10, 2005 and early 2012.
  - Over the course of almost 3,000 separate events.

http://www.privacyrights.org/data-breach

The Cost of a Security Breach

- Forrester Research pegs the cost of the average security breach at between $90 and $305 per lost record.
- Ponemon Institute (sixth annual U.S. Cost of a Data Breach Study) indicates that data breach cost increased to $214 per compromised record in 2010.
  - This is a significant increase from their 2006 report which pegged the average cost per lost customer record at $182.
- Or roll your own online using the data loss calculator provided by Darwin Professional Underwriters, Inc.
  - http://www.tech-404.com/calculator.html

A Data Breach Example

- On February 27, 2008 Health Net Federal Services reported thousands of doctors in eleven states had their personal information (including Social Security numbers) openly posted on a company website.
- According to the Privacy Rights Clearinghouse, the total number of records involved was 103,000.
- So what did that cost?
  - At the low end, the cost is $9.3 million, but at the high end it balloons to over $31.4 million.
  - Using the Ponemon estimate, which is sort of in the middle, the cost ranges from $18.7 million to $22 million.

How Prevalent is this Problem?

- **68%** of companies are losing sensitive data or having it stolen out from under them six times a year
- An additional **20%** are losing sensitive data 22 times or more per year.

Sources: eWeek, March 3, 2007
IT Policy Compliance Group

http://www.eweek.com/c/a/Desktops-and-Notebooks/Report-Some-Companies-Lose-Data-Six-Times-a-Year/

Database Security Basics

- All database resources are controlled by the DBMS.
- For a user to be able to perform any DBMS operation or function, one of the following conditions must exist:
  - The user has been granted the ability to perform that function or operation, or
  - That operation or function has been granted generically to all users.

The Database Security Challenge

- Setting up and managing database authorization requires technical expertise and elevated privilege.
- Many aspects of database security require different utilities, system procedures, and commands to implement.
- When users require access to multiple databases, on multiple servers distributed across different physical locations, database security administration becomes quite complicated indeed.
- The commands must be repeated for each database, and there is no central repository for easily modifying and deleting user security settings on multiple databases simultaneously.

Who is Responsible for Security?

- The DBA typically is responsible for administering database security
  - Some organizations have transferred this task to a separate security administration function that controls all of the IT security for the company.
  - However, even in shops where security administration is a separate entity, database security is still handled by the DBA group…
    - …because database security is handled differently than a typical IT authorization scenario.
  - When the security administration group handles security policies, this group usually relies on third-party security software

Database Security *in a Nutshell*

At a high level, database security boils down answering four questions:

- Authentication
  - Who is it?
- Authorization
  - Who can do it?
- Encryption
  - Who can see it?
- Audit
  - Who did it?

Authentication

- Strong authentication is the cornerstone of any security implementation plan.
- It is impossible to control authorization and track usage without it.
- Before authorization to use database resources can be granted, a login needs to be established for each user of the DBMS.
  - Logins are sometimes referred to as *accounts*, or user IDs.
  - The login will have a password associated with it such that only those who know the password can use the login ID.
  - Some DBMSs use the operating system login ID and password as the DBMS login ID and password; others require an additional login ID and password to be created specifically for database access and security.

Login Details

When the DBMS controls the addition of logins, the DBA is required to provide certain information about the login when it is created:

- *Password*—the key phrase, word, or character string associated with the new login that must be provided by the user before access to the database is permitted
- *Default database*—the name of the database to which the user will initially be connected during login
- *Default language*—the default language assigned to the login when using the DBMS if multiple languages are supported
- *Name*—the actual full name of the user associated with this login
- *Additional details*—additional details about the user for which the login has been created: e-mail, phone number, office location, business unit, and so on. This is useful for documentation purposes

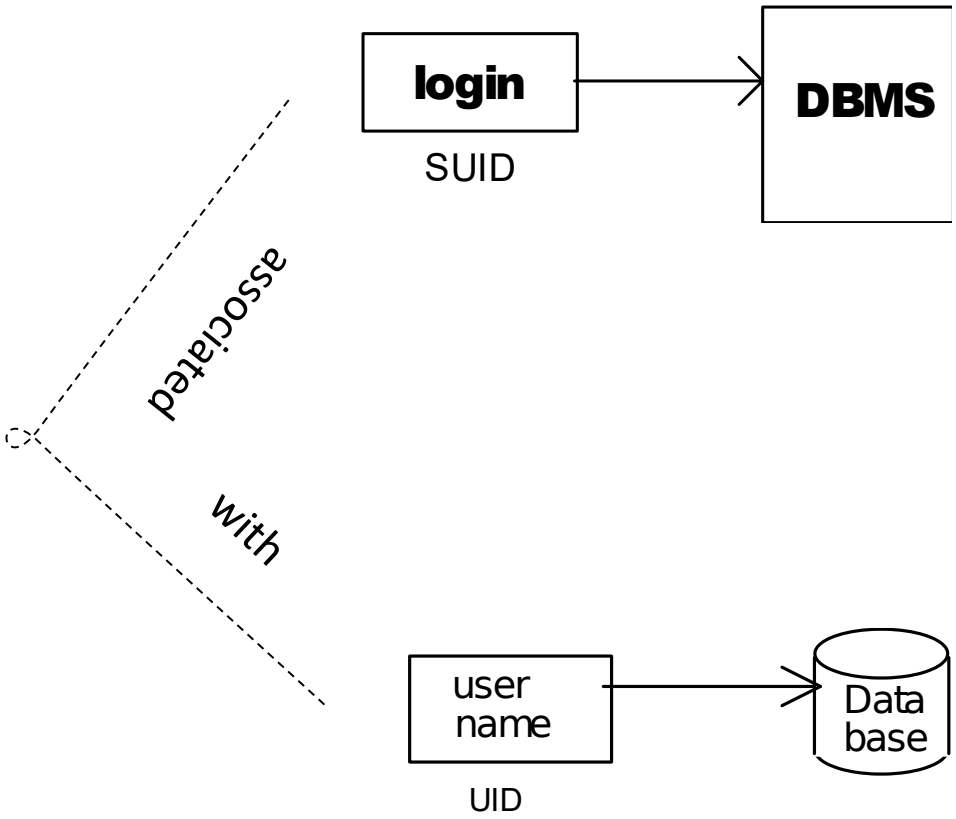
Password Guidance

- Avoid passwords that are too short. Each password should be at least six characters long, more if possible.
- Each password should consist of at least a combination of alphabetic characters and numeric characters. Using other allowable symbols makes the password harder to guess.
- Avoid creating a password that is a complete word (in either the native language of the user or any foreign language).
- Do not embed personal statistics in the password. Street addresses, social security numbers, phone numbers, and the like are easily guessed and do not belong in passwords.
- Consider concatenating two unrelated words with a symbol or number between them. For example, “toe3star” is a viable password.
- Use mnemonic devices to help you remember passwords.
- Avoid common and weak password architypes such as sports teams and sports celebrities.

Login Guidance

- When a user no longer requires access to the DBMS, or leaves the company, the DBA should drop his login from the system as soon as possible.
  - This could become a complicated task—a login cannot be dropped if the person is currently using a database, or if the user owns any database objects
  - Limit the database users who can create database objects to DBAs only, especially in a production environment.
- In lieu of dropping a login, the DBMS may provide an option to lock the login.
  - Locking a login prohibits the user from accessing the DBMS, but it does not actually drop the login from the system.

Database Users



Granting and Revoking Authority

- The DBA controls database security and authorization using Data Control Language (DCL).
- DCL statements comprise two basic types:
  - *GRANT* assigns a permission to a database user.
  - *REVOKE* removes a permission from a database user.

Granting Authority

- The GRANT statement is issued with a list of privileges to be assigned to a list of users.
- The WITH GRANT OPTION allows a user to pass the authority to grant privileges along to others.
  - Generally, the use of this clause depends on whether an installation practices centralized or decentralized administration of privileges.
    - *Decentralized administration* is generally easier to establish, but more difficult to control. As more and more users obtain the authority to grant privileges, the scope of authority is widened and becomes unwieldy.
    - *Centralized administration* is generally easier to administer, but places a burden on the centralized administrator as the sole arbiter of privileges within the environment.

Types of Privileges

- *Table:* to control who can access and modify the data within tables
- *Database object:* to control who can create new database objects and drop existing database objects
- *System:* to control who can perform certain types of systemwide activities
- *Program:* to control who can create, modify, and use database programs
- *Stored procedure:* to control who can execute specific functions and stored procedures

Table Privileges

Table privileges are granted to enable users to access tables, views, and columns within tables and views. The following privileges can be granted for tables and views:

- SELECT: to enable the user to select from this table/view
- INSERT: to enable the user to insert rows into this table/view
- UPDATE: to enable the user to update this table/view
- DELETE: to enable the user to delete rows from this table/view
- ALL*:* to enable the user to select, insert, update, and delete using this table/view

Database Object Privileges

- *Database object privileges* control which users have the permission to create database structures.
- The actual privileges that can be granted will depend on the DBMS and the types of database objects supported.
- Generally, the DBMS will provide options to grant CREATE privileges on each type of database object, including:
  - Databases
  - Tablespaces
  - Tables
  - Indexes
  - Defaults
  - Triggers
  - User-defined data types

System Privileges

- *System privileges* control which users can use certain DBMS features and execute certain DBMS commands.
- The system privileges available will vary from DBMS to DBMS but may include:
  - The ability to archive database logs
  - Shut down and restart the database server
  - Start traces for monitoring
  - Manage storage
  - Manage database caches

Program & Procedure Privileges

- Granting the EXECUTE privilege gives the user permission to execute a program or a stored procedure.
- Granting privileges to users on programs and procedures is easier to control than granting privileges on individual tables and columns.
- The procedural logic in the program and procedure controls which specific tables and columns can be modified.
- The DBA can better maintain the integrity of production data if the only way it can be changed is programmatically.

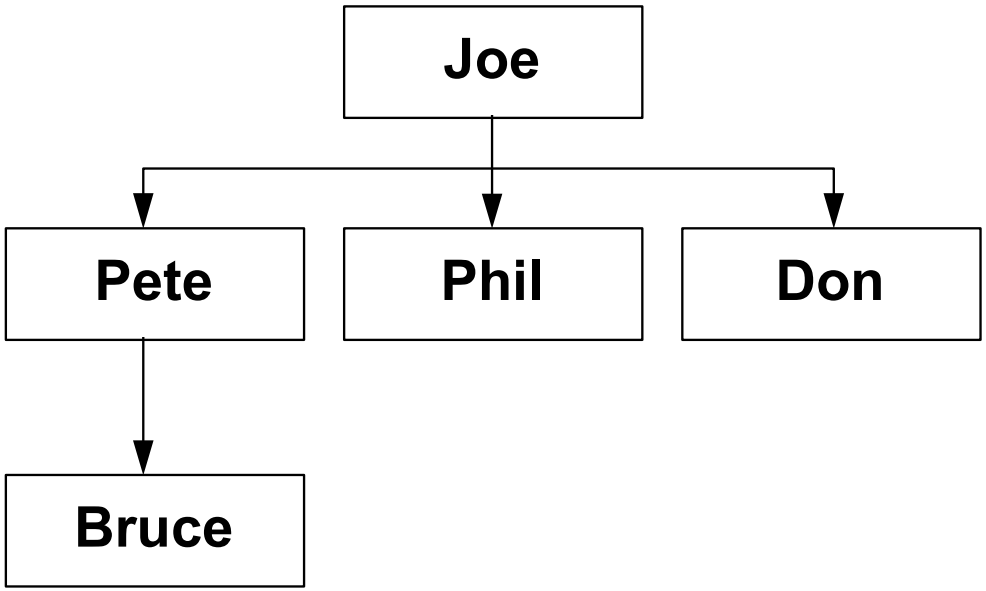
PUBLIC Authority

- The DBA can choose to grant a particular authorization to PUBLIC.
- When authorization is granted to PUBLIC, the DBMS will allow anyone who can log in to the DBMS that particular authority.
- Exercise caution when granting any privileges to PUBLIC.
- PUBLIC access can open your databases to hacking.

Revoking Authority

- Revoke is the inverse of Grant
- Using Revoke, a DBA can remove privileges from a user that has previously been Granted those privileges
- Proceed with caution if WITH GRANT OPTION was used
  - Cascading revokes can occur

Cascading Revokes



grant x to pete wgo
grant x to phil wgo
grant x to don

Chronology and Revokes

- The timing of a GRANT or REVOKE statement may have a bearing on its impact.
- For example, in some DBMSs it is possible to grant a privilege to all users except a specific user by issuing the following statements:

```
GRANT DELETE on titles to public;
COMMIT;
REVOKE DELETE on titles from userx;
```

- The DBA needs to understand exactly how GRANTs and REVOKEs work for each DBMS being administered in order to properly manage privileges on database objects and resources.

Label Based Access Control (LBAC)

- LBAC provides for a more granular security scheme, specifying who can read and modify data in individual rows and/or columns.
- For example, you might want to set up an authorization scenario such that employees can see their own data but no one else's. Or your authorization needs might dictate that each employee's immediate manager is able to see his payroll information as well as all of his employee's data, and so on up through the org chart.
- Setting up such a security scheme is virtually impossible without LBAC.

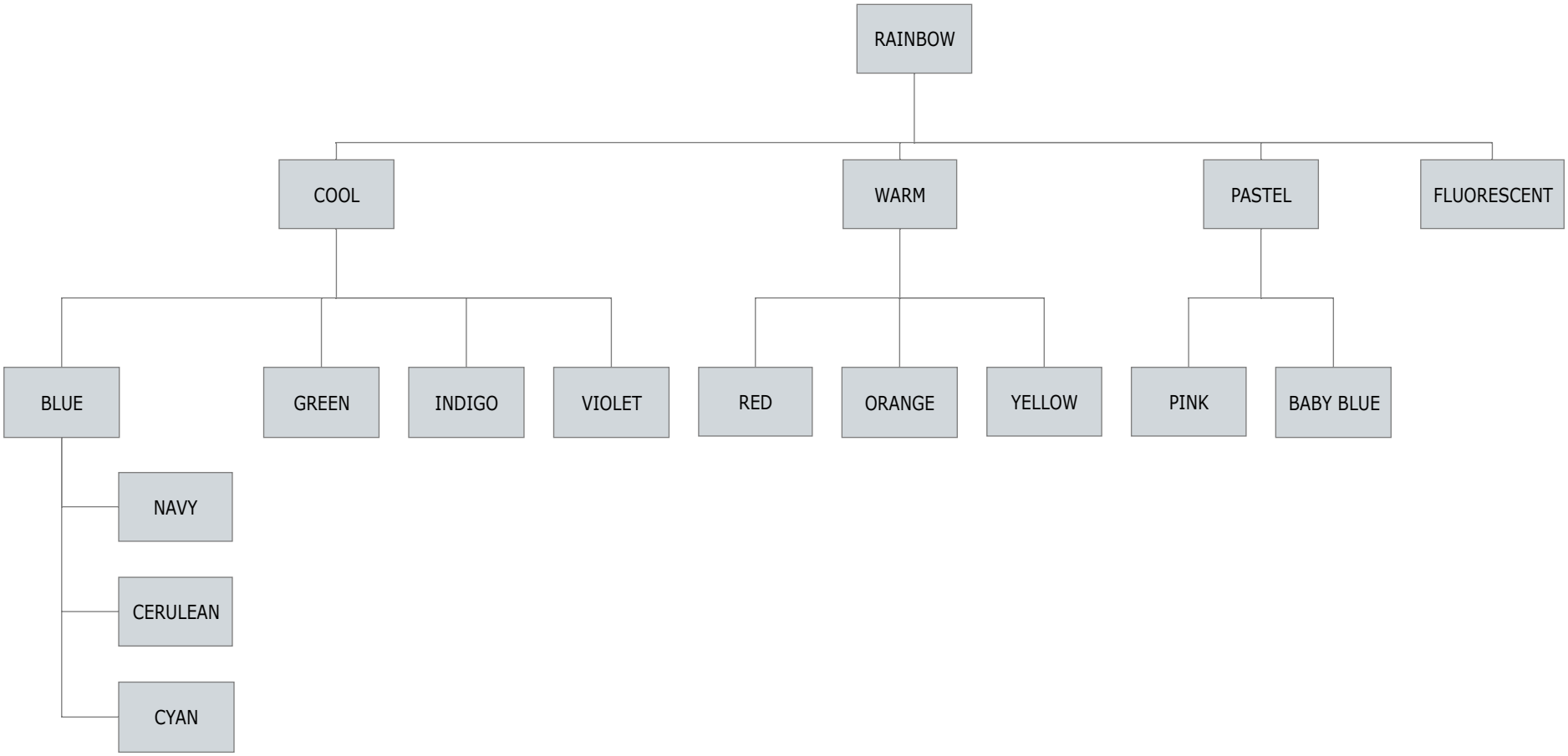
LBAC Configuation

- An administrator must configure the LBAC system creating security label components
- These are database objects used to represent the conditions determining whether a user can access a piece of data.
- A security policy is used to describe the criteria to be used for determining who has access to what data.
- Once created, a security label can be associated with individual columns and rows in a table to protect the data held there.
- An administrator allows users access to protected data by granting them security labels.
- When a user tries to access protected data, that user's security label is compared to the security label protecting the data. The protecting label will block some security labels but not others.

LBAC in Action

- Any individual user can hold security labels for multiple security policies. For any given security policy, however, a user can hold at most one label for read access and one label for write access. A security administrator can also grant exemptions to users. An exemption allows you to access protected data that your security labels might otherwise prevent you from accessing.
  - Together your security labels and exemptions are called your LBAC credentials.
- Any attempted access to a protected column when your LBAC credentials do not permit that access will fail.
  - If you try to read protected rows not allowed by your LBAC credentials, the DBMS simply acts as if those rows do not exist.
  - Even aggregate functions, such as COUNT(*), will ignore rows when your LBAC credentials do not allow read access.
  - This is important because sometimes even having knowledge that the data exists (without being able to access it) must be protected.

LBAC Example



Authorization Roles and Groups

- In addition to granting privileges to individual users, the DBMS may provide the capability to assign
  - Specific privileges to a role, which is then granted to others, or;
  - Specific built-in groups of privileges to users.

Roles

- Once defined, a *role* can be used to grant one or more preassigned privileges to a user.
- A role is essentially a collection of privileges.
- For example:

```
CREATE role MANAGER;
COMMIT;
GRANT select, insert, update, delete on employee to MANAGER;
GRANT select, insert, update, delete on job_title to MANAGER;
GRANT execute on payroll to MANAGER;
COMMIT;
GRANT MANAGER to user1;
COMMIT;
```

Groups

- Group-level authority is similar to roles.
- Each DBMS provides built-in groups that cannot be changed. The following groups are common among the major DBMSs:
  - *System administrator*. Sometimes abbreviated SA or SYSADM, the system administrator group is the most powerful within the DBMS.
  - *Database administrator*. Sometimes abbreviated as DBADM or DBA, the database administrator group gives all privileges over a specific database, plus the ability to access, but not modify, data in tables within that database.
  - *Database maintenance*. Sometimes abbreviated as DBMAINT, the database maintenance group includes the specific database privileges for maintaining database objects (such as the ability to run utilities and issue commands).
  - *Security administrator*. The security administrator role, aka SSO or SECADM, has the privilege-set permitting the granting and revoking of database security across the DBMS. Any database security-related activities can be performed by the security administrator.
  - *Operations control*. Sometimes referred to as OPER or SYSOPR, the operations control role has the authority to perform operational database tasks such as backup and recovery, or terminating runaway tasks.

Limit the Number of SA Users

- A single organization should limit the number of users who are assigned the SA role or group-level authority.
  - A user with SA capabilities is very powerful.
    - Only corporate DBAs and systems programmers should be granted this level of authority.
    - End users, managers, and application development personnel do not need SA authority to do their jobs.
- This is a particularly vexing problem with purchased third-party applications.
  - Many such applications rely on SA authority because it is easier.
  - However, it is not a secure practice.

Groups and Cascading Revokes

- Some users who have been assigned group-level privileges can grant privileges to other users.
  - If the group-level authority is revoked from that user, any privileges that user granted will also be revoked.
  - This is similar to the cascading REVOKEs that occur as a result of the WITH GRANT option.
- Before revoking a group-level authorization from a user, be sure to ascertain the impact of cascading REVOKEs, and be prepared to reapply the required privileges that will be removed due to the cascading effect.

Using Views for Security

- A view can be created that omits sensitive information.
  - By creating the view without the sensitive columns, users can be granted the SELECT privilege on the view and will be able to access employee information that is deemed appropriate.
  - For example, this view omits salary details:

```
CREATE view emp_all
AS
SELECT first_name, last_name, middle_initial,
       street_address, state, zip_code
FROM   employee;
```

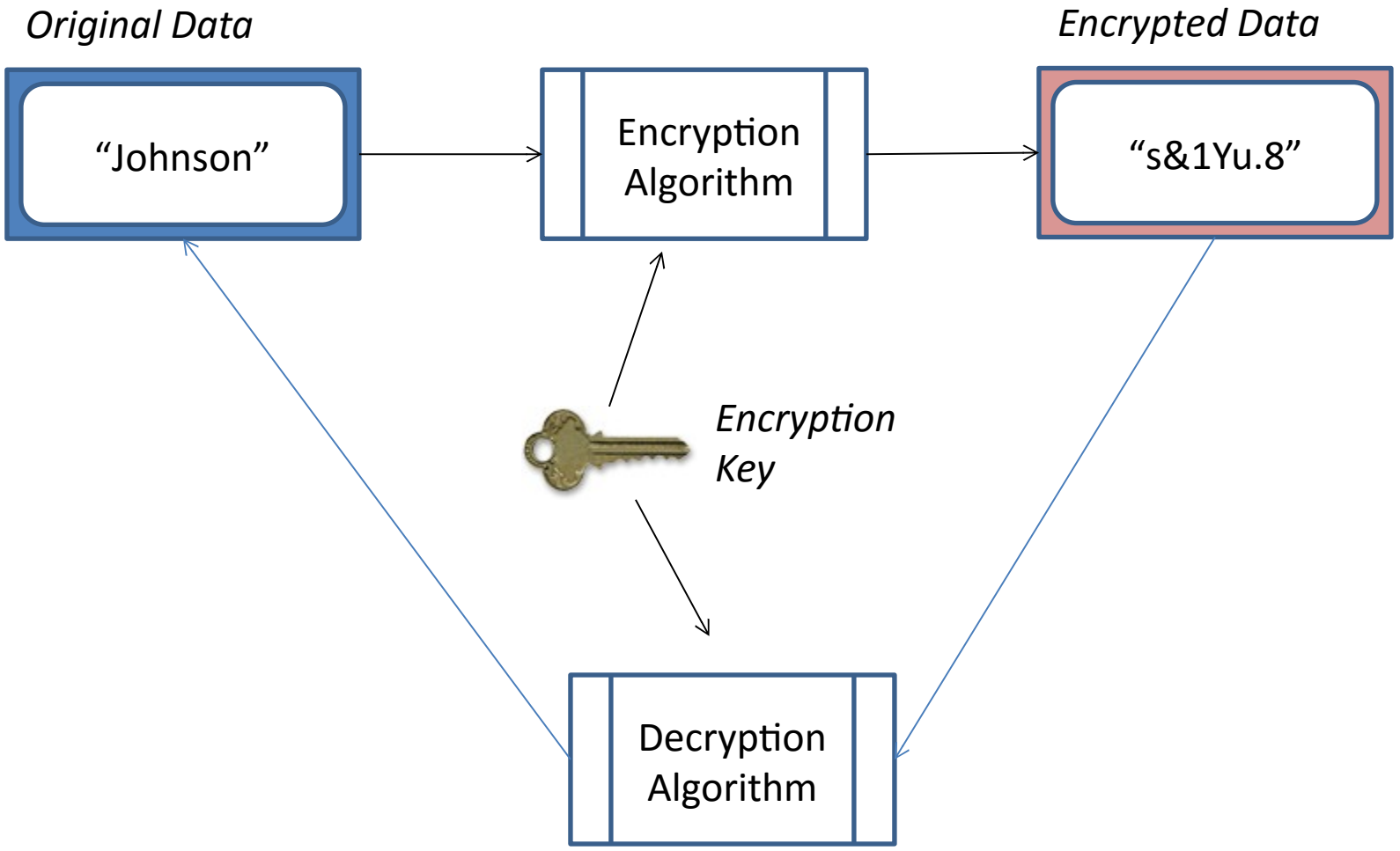
Vertical and Horizontal Restriction in Views

- Vertical Restriction
  - When a view eliminates columns from a base table
  - Using column listing
- Horizontal Restriction
  - When a view eliminates rows from a base table
  - Using WHERE clause(s)

Using Stored Procedures for Security

- Stored procedures can be used to provide an additional level of security.
- Stored procedures can be coded that access only row- and/or column-level subsets of data.
- The ability to execute these stored procedures can then be granted to users.
- If no privileges on the underlying base tables are granted, the users will be able to access the data only by executing the stored procedure, thereby providing the requisite security.

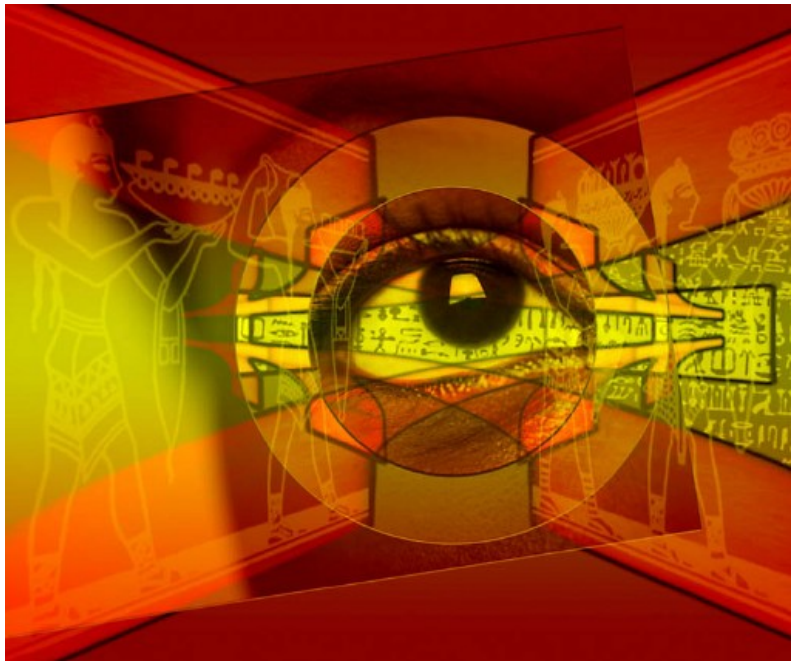
Encryption



Encryption transforms data rendering it unreadable to anyone without the decryption key.

Type of Encryption

- There are two types of encryption
  - At Rest
  - In Transit



“At Rest” Encryption

- Encrypting data at rest is undertaken to prohibit “behind the scenes” snooping for information.
- Consider a database containing a top secret military plan.
  - Of course, this data should be protected using traditional database security and authorization methods.
  - But what if the data is accessed outside the control of the DBMS?
- By encrypting the data at rest, even if a hacker surreptitiously gains access to the data behind the scenes, without the decryption key the data will be meaningless.
- Other examples of data at rest encryption in database systems include encrypting backup data and encrypting userids and passwords.

“In Transit” Encryption

- Encrypting data in transit is undertaken to prohibit network packet sniffing.
  - If the data is encrypted before it is sent over the network and decrypted upon reception at its destination, then it is protected along its journey.
  - Anyone nefariously attempting to access the data en route will receive only encrypted data.
  - Without the decryption key, the data cannot be read.
- Data in transit encryption most commonly is supported using DBMS system parameters and commands or through an add-on encryption product.

SQL Injection

- SQL injection is a form of web hacking whereby SQL statements are specified in the fields of a web form to cause a poorly designed web application to dump database content to the attacker.
- SQL injection is a favorite technique of web hackers and stories abound in the news of the technique being used for nefarious purposes.

SQL Injection Attacks
"In The News"

- On January 13, 2006, Russian hackers broke into a Rhode Island government web site and obtained credit card data http://www.xiom.com/whid-2006-3).
- On June 29, 2007, a hacker defaced the Microsoft website in the U.K. using a SQL injection attack (http://www.cgisecurity.com/2007/06/hacker-defaces.html).
- In January 2008, tens of thousands of PCs were infected by an automated SQL injection attack that exploited a vulnerability in Microsoft SQL Server (http://www.pcworld.com/businesscenter/article/146048/mass_sql_injection_attack_targets_chinese_web_sites.html).
- On September 19, 2010 during the Swedish election, a voter attempted an injection by hand writing SQL statements as part of a write in vote (http://alicebobandmallory.com/articles/2010/09/23/did-little-bobby-tables-migrate-to-sweden).
- Other websites attacked by SQL injection in 2011 included:
  - Barracuda Networks
  - mysql.com
  - Lady Gaga's website
  - Nokia's developer site
  - and Canon, among others

SQL Injection Example

- A web-based application using dynamic SQL requires users to login with their e-mail address and a password.
- Perhaps the SQL looks something like this:

```
SELECT userid, password

FROM   uid_pwd_table

WHERE  field = '$EMAIL';
```

- A savvy hacker can attempt a SQL injection attack by entering:

```
anything' OR '1'='1
```

The SQL Injection Bad News?

- If the application does not check the input properly the injection causes the SQL to now look like this:

```
SELECT userid, password

FROM   uid_pwd_table

WHERE  field = 'anything' OR '1'='1';
```

Another Type of SQL Injection

- Another form of SQL injection relies upon improper typing, for example not checking whether data that should be numeric is actually numeric.
- Consider, for example:

```
statement := "SELECT * FROM userinfo WHERE id = " + in_var + ";"
```

- The hacker can enter:

```
4;DROP TABLE customer
```

SQL Injection Prevention

- Using well designed query language interpreters and coding applications appropriately can prevent SQL injection attacks.
  - When possible use static SQL.
  - Always validate user input by testing type, length, format, and range.
  - The program should make absolutely no assumptions about the size, type, or content of the data that is received.
  - Avoid concatenating user input that has not been validated.
  - Analyze input and reject anything that contains special characters such as the semi-colon (;), the string delimiter ('), comment delimiters (--, /*…*/), V$ (the beginning of Oracle DBA views), and xp_ (the beginning of SQL Server catalog stored procedures).
  - For Microsoft SQL Server users, beware of the Transact-SQLprocedure xp_cmdshell. This procedure spawns a Windows command shell and passes in a string for execution.

Auditing

- Auditing is covered in more detail in the next section on regulatory compliance and database administration.
- But here is a short overview:
  - Auditing enables DBAs to track the use of database resources and privileges.
    - When auditing is enabled, the DBMS will produce an audit trail of database operations.
  - Each audited database operation produces an audit trail of information including what database object was impacted, who performed the operation, and when.
  - Depending on the level of auditing supported by the DBMS, an actual record of what data actually changed may also be recorded.
  - Tracking who does what to which data when is important because there are many threats to the security of your data.

External Security

- System catalog data files
- Active and archive log files
- User data sets for tablespaces
- User data sets for indexes
- Audit data files
- Performance trace files
- Program and script files (both source and executable code)

Job Scheduling and Security

- Most organizations schedules tasks to be run at predetermined times, and when those tasks involve database access, authority must be granted to the scheduler.
- It is not a very good idea to grant SYSADM authority to the job scheduler.
- Doing so would permit any job to perform any database task—creating potentially severe security problems.
- Instead, determine how to grant individual authorization to specific jobs using the facilities of the scheduling package and the DBMS.

Non-DBMS DBA Security

- The DBA will need to possess a fairly high level of operating system authority in order to perform the job of administering and managing the organization's databases and data.
- For example, in the UNIX environment some installation tasks require root authority.
- This situation can be handled in two ways:
  - Grant the DBA root authority to do the installation, or
  - Turn the specific installation tasks requiring root authority over to the UNIX system administrator.

DBMS Fixpacks and Maintenance

- Another important consideration with regard to database security is the development and implementation of a DBMS maintenance plan.
- The DBMS itself must be periodically patched to remedy security defects and other software bugs in the DBMS code.
  - Failing to apply patches as they are delivered from the DBMS vendor via fixpacks, downloads, and other maintenance streams can cause security issues in your databases.

Questions