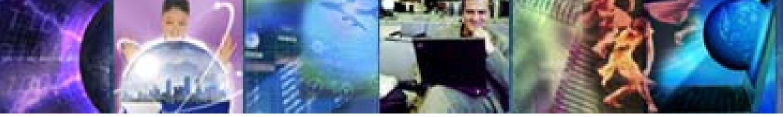
DB2 for LUW Security

Walid Rjaibi
wrjaibi@ca.ibm.com
July 19, 2007

DB2 Information Management Software

ON DEMAND BUSINESS



Agenda

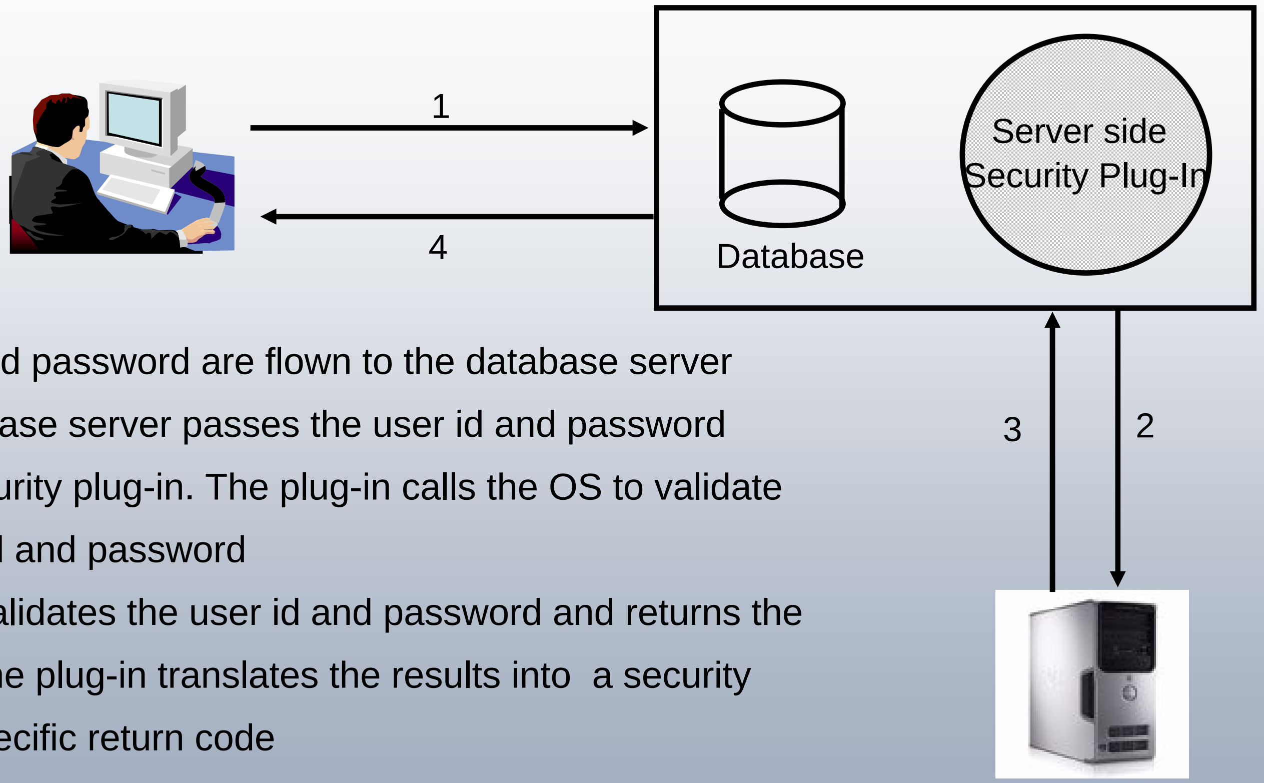
- ***Authentication***
- ***Authorization***
- ***Database Roles*** **(Viper II)**
- ***Label-Based Access Control*** **(Viper & Viper II)**
- ***Trusted Contexts*** **(Viper II)**
- ***Auditing*** **(Viper II)**

Authentication

- ***The first security measure encountered when attempting to access a database***
- ***The process by which a user identity is validated***
- ***Actual user identity validation is performed outside DB2 through an authentication security plug-in***
  - Authentication security plug-ins can be custom-built (by DB2 customers) and provide greater flexibility in accommodating specific authentication needs
  - The default, out of the box, authentication relies on an IBM-shipped OS-based authentication security plug-in

Authentication (Cont.)

Authentication using OS-based security plug-in



1: User id and password are flown to the database server

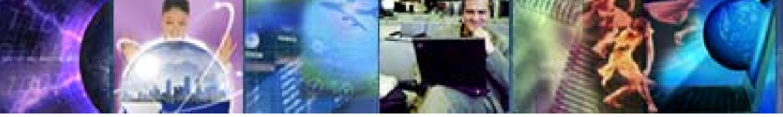
2: The database server passes the user id and password to the security plug-in. The plug-in calls the OS to validate the user id and password

3: The OS validates the user id and password and returns the results. The plug-in translates the results into a security plug-in specific return code

4: The database server denies access if the security plug-in return code indicates that the OS failed to validate the user id and password

Authentication (Cont.)

- ***When a user identity is validated successfully***
  - The user’s groups (if any) are acquired using a group membership security plug-in
    - The default, out of the box, group acquisition relies on an IBM-shipped OS-based group membership security plug-in
  - The user id is mapped to a DB2 authorization id
    - An authorization id is an internal representation of a user or a group to which privileges/authorities can be granted
    - The user id to authorization id mapping is determined by the security plug-in; with the default IBM shipped security plug-in, the authorization id is simply the uppercase of the user id



Agenda

- *Authentication*
- *Authorization*
- *Database Roles*
- *Label-Based Access Control*
- *Trusted Contexts*
- *Auditing*

Authorization

- ***The process of checking whether an authorization id is allowed to execute a database operation***
  - SQL statement
  - Command (or API)
- ***The process involves the acquisition of the set of permissions available to the authorization id***
  - Permissions held by the authorization id itself
  - Permissions held by the authorization id's groups and roles
  - Permissions held by PUBLIC

Authorization (Cont.)

- ***Permissions are divided into authorities and privileges***
  - Authorities
    - System level authorities can be acquired only through membership in a group and are managed at the instance level through DBM configuration parameters
    - Database level authorities can be acquired either directly or through membership in a group or role, and are managed at a database level through GRANT and REVOKE
  - Privileges
    - They can be acquired either directly or through membership in a group or role, and are managed at an object level (e.g., a table) through GRANT and REVOKE

Authorization (Cont.)

- ***Content-based authorization***
  - Views
    - Allow users to read only the rows (and columns) they are authorized for
    - Symmetric views prevent users from updating rows they are not authorized to read
  - Triggers
    - Allows to implement security mechanisms that get activated when certain events happen (e.g., if an INSERT fails a security check an error can be signaled from the trigger on the table subject of the INSERT)

Agenda

- *Authentication*
- *Authorization*
- *Database Roles*
- *Label-Based Access Control*
- *Trusted Contexts*
- *Auditing*

Database Roles

- ***What is a database role?***
  - A database object that may group together one or more privileges, authorities, security labels or exemptions, and may be granted to users, groups, PUBLIC or other roles
- ***What is the advantage of database roles?***
  - They simplify the administration and management of privileges in a database
    - SECADMs can control access to their databases at a level of abstraction that is close to the structure of their organizations (e.g., they can create roles in the database that map directly to those in their organizations)

Database Roles (Cont.)

- ***Grantable privileges and authorities***
  - All database privileges, security labels and exemptions
  - All database authorities, except SECADM
- ***Role membership***
  - Managed by SECADM and could be delegated to others

Example: Delegate the management of membership in the role teller to user Bob

```
GRANT ROLE teller TO USER Bob
WITH ADMIN OPTION
```

Database Roles (Cont.)

- ***Role enablement***
  - All the roles assigned to a user are enabled when that user establishes a connection
    - All the privileges and database authorities associated with those roles are taken into account when DB2 checks for authorization

Database Roles (Cont.)

- ***Roles vs groups***
  - Privileges and database authorities granted to groups are not considered by DB2 when creating views, triggers, MQTs, static SQL and SQL routines
    - DB2 cannot know when membership in groups change so that it can invalidate the database objects (e.g., views) created by users who relied on their group privileges to succeed
  - Roles are managed inside the database and are considered by DB2 when creating views, triggers, MQTs, static SQL and SQL routines
    - Roles granted to groups are not considered for the above

Database Roles (Cont.)

- ***Usage scenario***

BOB and ALICE are members of the DEV department and have the privilege to SELECT from tables SERVER, CLIENT and TOOLS.

One day, the management decides to move them to the QA department and the administrator has to revoke their privilege to select on tables SERVER, CLIENT and TOOLS.

Department DEV hires a new employee, TOM, and the administrator has to grant SELECT privilege on tables SERVER, CLIENT and TOOLS to TOM.

Database Roles (Cont.)

Without database roles:

GRANT SELECT ON TABLE SERVER TO USER BOB, USER ALICE

GRANT SELECT ON TABLE CLIENT TO USER BOB, USER ALICE

GRANT SELECT ON TABLE TOOLS TO USER BOB, USER ALICE

REVOKE SELECT ON TABLE SERVER FROM USER BOB, USER ALICE

REVOKE SELECT ON TABLE CLIENT FROM USER BOB, USER ALICE

REVOKE SELECT ON TABLE TOOLS FROM USER BOB, USER ALICE

GRANT SELECT ON TABLE SERVER TO USER TOM

GRANT SELECT ON TABLE CLIENT TO USER TOM

GRANT SELECT ON TABLE TOOLS TO USER TOM

Database Roles (Cont.)

With database roles:

CREATE ROLE developer

GRANT SELECT ON TABLE SERVER TO ROLE developer

GRANT SELECT ON TABLE CLIENT TO ROLE developer

GRANT SELECT ON TABLE TOOLS TO ROLE developer

GRANT ROLE developer TO USER BOB, USER ALICE

REVOKE ROLE developer FROM USER BOB, USER ALICE

GRANT ROLE developer TO USER TOM

Agenda

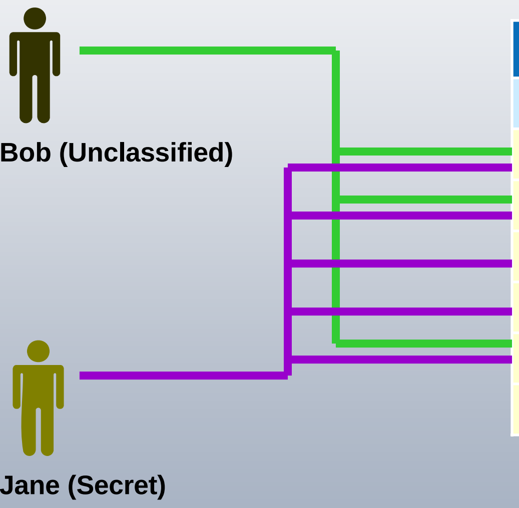
- *Authentication*
- *Authorization*
- *Database Roles*
- *Label-Based Access Control*
- *Trusted Contexts*
- *Auditing*

LBAC

- *A flexible implementation of Mandatory Access Control (MAC)*
- *A security label is associated with both users and data objects*
- *Access control is governed by the predefined LBAC security rules*

Bob U Results	
SERIAL_NUMBER	*TITLE*
SN0000001	***PO-HARRY-1***
SN0000002	***PO-TONY-1***
SN0000005	***PO-BRIAN-1***

Jane S Results	
SERIAL_NUMBER	*TITLE*
SN0000001	***PO-HARRY-1***
SN0000002	***PO-TONY-1***
SN0000003	***Instructions***
SN0000004	***Release Notes***
SN0000005	***PO-BRIAN-1***



Artifact Table (protected table)		
SERIAL_NUMBER	*SECLABEL*	*TITLE*
SN0000001	***Unclassified***	***PO-HARRY-1***
SN0000002	***Unclassified***	***PO-TONY-1***
SN0000003	***Confidential***	***Instructions***
SN0000004	***Secret***	***Release Notes***
SN0000005	***Unclassified***	***PO-BRIAN-1***
SN0000006	***Top Secret***	***MARS-35***

```
select serial_number, title
from artifact
```

LBAC (Cont.)

- ***Why is LBAC flexible?***
  - A security label is not a fixed structure (“level” and “compartments”) as with other MAC implementations
    - Security label structure is specified by the user
    - Security label structure does not require a “level” component
  - Security administrators can protect different tables with different security policies within the same database
  - Addresses more scenarios than those of the traditional US Gov. classified spaces
    - But LBAC is not a universal solution to all row and column level security problems
    - LBAC may be suitable to some non classified scenarios if the requirements can be mapped to security labels

LBAC (Cont.)

- ***What does LBAC add to table protection?***
  - LBAC does not replace the traditional discretionary access control. Rather, it complements it at the row and/or column level
  - The content of a table appears different depending on the identity of the user accessing that table
  - No user has any inherent privileges to access the content of LBAC protected data even if they are DBADM!

LBAC (Cont.)

- ***What's coming up in Viper II?***
  - The ability to alter a security label component (of any type) to add an element
  - The ability to grant security labels to roles and groups
  - The ability to grant exemptions to roles and groups
  - The ability to alter a security policy
  - The ability to combine LBAC and Trusted Contexts to make acquiring security labels and exemptions dependent on where the user is connecting from

Agenda

- *Authentication*
- *Authorization*
- *Database Roles*
- *Label-Based Access Control*
- *Trusted Contexts*
- *Auditing*

Trusted Contexts

- ***Security challenges***
  - Applications servers use of a single user id causes the following:
    - Loss of end user identity within the database server
    - Diminished user accountability
    - Over granting of privileges to a single authorization id
  - The lack of control on when privileges are available to a user can weaken overall security
    - The privileges may be used for purposes other than what they were originally intended for

Trusted Contexts (Cont.)

- ***What is a trusted context?***
  - A database object that defines a trust relationship between the database and an external entity such as an application server
  - The trust relationship is based on the following trust attributes
    - System authorization id
    - IP address (or domain name)
    - Data stream encryption

Trusted Contexts (Cont.)

- **Trusted context definition example**

 CREATE TRUSTED CONTEXT appServer

BASED UPON CONNECTION USING

SYSTEM AUTHID appServerID

ATTRIBUTES (ADDRESS ‘host-name1.dept.organization.com’,

ADDRESS ‘host-name2.dept.organization.com’

ENCRYPTION ‘HIGH’)

DEFAULT ROLE appServerRole

WITH USE FOR PUBLIC WITHOUT AUTHENTICATION,

Alice WITH AUTHENTICATION ROLE mgrRole

ENABLE

Trusted Contexts (Cont.)

- ***Trusted Connection***
  - A database connection whose attributes satisfy the definition of trusted context object is referred to as a trusted connection
  - A trusted connection allows the application to acquire additional capabilities that are not available to it outside the scope of the trusted connection
  - The additional capabilities vary depending on whether the trusted connection is explicit or implicit
    - An explicit trusted connection is a trusted connection explicitly requested by an application whereas an implicit trusted connection is not explicitly requested

Trusted Contexts (Cont.)

- An explicit trusted connection allows the initiator of such trusted connection the ability to:
  - Switch the current user ID on the connection to a different user ID with or without authentication
  - Acquire additional privileges that may not be available to them outside the scope of the trusted connection
  - Application changes are needed to take advantage of explicit trusted connections
- An implicit trusted connection allows only the ability to acquire additional privileges
  - No application changes are required to take advantage of implicit trusted connections

Trusted Contexts (Cont.)

- ***Switching user rules***
  - State
    - Must be an explicit trusted connection
    - Must be on a transaction boundary
  - Authorization
    - User must be allowed by the trusted context definition
    - Authentication token must be provided if required by the trusted context definition
  - Environment reset
    - All traces of the connection environment under the old user are gone. For example, any temporary tables or WITH HOLD cursors open will be totally lost when the user ID on the connection is switched to a new user ID

Trusted Contexts (Cont.)

- ***Trusted context-specific privileges***
  - To gain control on when a privilege becomes available to a user, security administrators should not grant privileges to that user directly or indirectly. Rather:
    - Grant the privilege to a role
    - Assign that role only to a trusted context object
    - A user will have access to the role only and only if they are working within the scope of a trusted connection based on that trusted context

Trusted Contexts (Cont.)

- ***Advantages of trusted contexts***
  - User accountability
    - Knowing the end user identity provides improved data access auditing capability
    - Eliminates shared user id and have each person audited and accountable with their own user id
  - Improved security
    - More control on when privileges are available to users
    - Eliminates the concern about misusing the system authid credentials to access the DB2 server
    - Enforce the least privilege security principle

Agenda

- *Authentication*
- *Authorization*
- *Database Roles*
- *Label-Based Access Control*
- *Trusted Contexts*
- ➡ *Auditing*

Auditing

- ***Audit Policy***
  - A database object that specifies what categories of events are to be audited
  - An audit policy can be applied to:
    - A database
    - A table
    - A trusted context
    - An authorization id (user, role, group)
    - An authority (SYSADM, SYSMAINT, SYSCTRL, SYSMON, DBADM, SECADM)

Auditing (Cont.)

- Audit policies provide a finer granularity of control on what needs to be audited compared to Viper instance level auditing
  - Smaller audit trails
  - Smaller performance overhead

Example: Audit all SQL access to the EMPLOYEE table

```
CREATE AUDIT POLICY employeeTablePolicy
 CATEGORIES EXECUTE STAUS BOTH ERROR TYPE AUDIT


 AUDIT TABLE EMPLOYEE
 USING POLICY employeeTablePolicy
```

Auditing (Cont.)

- ***EXECUTE event category***
  - Generates audit records to show the execution of SQL statements
    - Captures the SQL statement text and the compilation environment
    - Input data values provided for any host variables and parameter markers can optionally be logged (except for LOB, LONG, XML and structured types)

Auditing (Cont.)

- ***Audit log archiving***
  - Archiving of the audit log moves the current audit log to an archive directory while the database server begins writing to a new active audit log
    - Extract must be performed on the archived log, not the active audit log
    - Prevents any performance degradation associated with locking the active audit log when an extract operation is performed
  - The new SYSPROC.AUDIT_ARCHIVE procedure must be used to archive an audit log

Auditing (Cont.)

- ***Audit log on disk***
  - It is now possible to configure the path the audit log is written to
  - The new “datapath” option of the db2audit tool must be used to customize the location of the active audit log
  - The new “archivepath” option of the db2audit tool must be used to customize the location of the archived audit log

Example:

```
db2audit configure datapath /auditlog archivepath /auditarchive
```

Thank you!