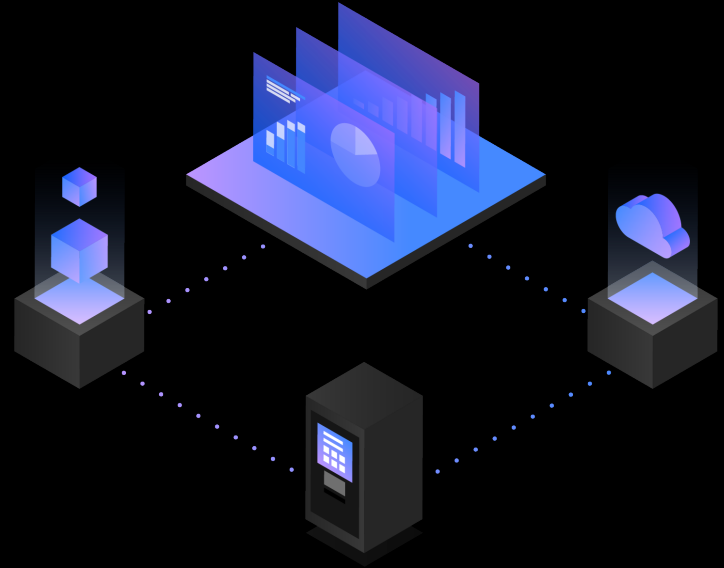


Securing Db2 for z/OS Data: Encryption, and Much More

Central Canada Db2 Users Group
& IDUG Data Tech Summit

September 19, 2023

Robert Catterall, IBM
Principal Db2 for z/OS Technical Specialist



Agenda

- Encrypting Db2 for z/OS data
- Privilege management
- Row permissions and column masks
- Client authentication (for client-server applications)
- Auditing
- Application architecture
- Test data management
- RACF management of Db2-internal security

One more thing before we get going...



Because I'll be covering a lot of ground, I won't be able to go into a lot of depth on everything – the presentation contains links to useful sources of additional information

Encrypting Db2 for z/OS data

One primary area of Db2 data encryption: at rest

- “At rest” generally means, “on disk”
- Recently, huge increase in interest/activity in this area – here’s why:
 - Regulatory requirements
 - z/OS data set encryption functionality (introduced with z/OS 2.3)
 - **Application-transparent**: data **automatically encrypted** when written to data set on disk, **automatically decrypted** when read into memory
 - Enormous improvement (like, 7X) in **CPU efficiency of encryption** with z14 vs. z13 (reason: encryption functionality went from card to chip)
 - Db2 12 function level 502 provided a **new option for enabling z/OS encryption for Db2 data sets** (more on this to come)

More on z/OS data set encryption

- z/OS data set encryption and “pervasive encryption”: not the same thing
 - “Pervasive encryption” is more all-encompassing than data set encryption
- z/OS data set encryption: associate an **encryption key label** (external “handle” associated with an encryption key) with a data set at create time
 - How that can be done:
 - A RACF (or equivalent) data set profile
 - Via the IDCAMS process that creates a data set
 - Via and SMS data class specification
 - **With new option of ALTER/CREATE TABLE and STOGROUP, and a new ZPARM – delivered via function level 502 of Db2 12 for z/OS (see next slide)**
- Helpful IBM redbook: [*Getting Started with z/OS Data Set Encryption*](#)

The Db2 12 FL502 encryption enhancement

- A new ZPARM: [ENCRYPTION_KEYLABEL](#)
 - Encryption key label used for catalog/directory objects, and for archive log data sets [when you archive to disk](#) (data set stays encrypted if HSM-migrated to tape)
- New option, [KEY LABEL](#), for ALTER/CREATE TABLE/STOGROUP
- No matter how a key label gets associated with a Db2 data set, if a Db2 object (e.g., a table space) already exists and contains data, encryption is typically accomplished via execution of online REORG
 - Shadow data sets created by Db2 will have key label associated with them
 - If a key label is associated with a table or STOGROUP, online REORG of table space encrypts [everything](#), including indexes and LOB/XML table spaces (if any)
- More information in [Db2 12](#), [Db2 13](#) online documentation

A little more on z/OS data set encryption for Db2

- What about Db2 active log data sets?
 - Can't convert **existing** unencrypted active log data sets – have to **replace them**
 - One option: dynamically add new encrypted log data sets with NEWLOG option of -SET LOG command, remove old ones with DSNJU003 utility
 - Db2 13 (FL500) option: same as above, but **dynamically** remove old unencrypted log data sets with REMOVELOG option of -SET LOG command
- With z/OS data set encryption, ID of a process needs to be **RACF-permitted to use encryption key**
 - For access to Db2 data sets, IDs of Db2 database services and system services address spaces are **only ones** that need RACF permission to use encryption key(s)
 - Reason: when ID SMITH selects data from table T1, **z/OS perceives that Db2** – not SMITH – is accessing data set

Other main area of Db2 encryption: on the wire

- “On the wire” refers to encryption of data flowing between **DDF-connected application** (DRDA requester or REST client) and Db2 for z/OS server
- Often referred to as “**SSL encryption**” – more accurate term is AT/TLS (application transparent/transport-layer security)
- **Multiple actions required** to enable AT/TLS security for Db2 client-server applications: these relate to Db2 for z/OS, RACF, z/OS, client-side server...
 - Very useful redbook: [IBM DB2 for z/OS: Configuring TLS/SSL for Secure Client/Server Communications](#)
- Relatively recent change: fix for Db2 APAR PH08188 (April 2019) made it possible for Db2 subsystem’s **only SQL listener port** to be **secure port**
 - Effect: **only** clients that support SSL encryption can connect to subsystem

Db2 SSL encryption: a lesson learned

- In my experience, #1 source of client-side problems is **mistakes in connection string** used to request SSL connection to Db2 for z/OS server
 - Lots of ways you can get this wrong
 - One way I have seen these problems overcome: **do not try to specify required information in connection string**
 - Instead, put required information (e.g., name/location of host certificate file) in entry for Db2 for z/OS system in IBM Data Server Driver's **db2dsdriver.cfg file**
 - Then, have client application simply request connection to the named Db2 for z/OS server, with **nothing about SSL** in the connection string
 - Data Server Driver will check entry for Db2 for z/OS system in db2dsdriver.cfg, and **that is where it will find required info**: specification that request is for SSL connection, Db2 system's secure SQL port, host certificate file name/location...

Privilege management

Especially important for a Db2 for z/OS system

- Here's why: with at-rest and on-the-wire encryption in effect, only “in the clear” Db2 data is in z/OS memory (i.e., Db2 buffer pools), and thanks to z/OS key-protected memory, **only way to access that data is through Db2**
- So, only IDs that can access data are those that have requisite Db2 privileges – you need to **make sure those privileges aren't abused**
 - One way that is being done: organizations are **more careful than they used to be in granting SYSADM** (Db2 “super-user” authority) to users
 - For example, DBA who had SYSADM authority might instead get **system DBADM authority**, and system DBADM authority can be granted **WITHOUT DATA ACCESS**
 - In that case, if DBA needs to access data in table T1 to accomplish some task, SELECT ON T1 can be **temporarily granted** to DBA until task is completed
 - Hassle for DBA? Yes, but that can come with **tighter data access** controls

Restricting who can manage privileges

- Besides having fewer SYSADMs, you can reduce scope of SYSADM activity via the Db2 ZPARM parameter `SEPARATE_SECURITY`
 - When parameter set to `YES` (default value is `NO`), an ID with SYSADM authority...
 - `Cannot` create and manage `security objects` (e.g., roles, trusted contexts, column masks, row permissions)
 - `Cannot` grant `privileges` to others
 - Who can do those things, if SYSADM can't? An ID with `SECADM` authority
 - If `SEPARATE_SECURITY=NO`, SYSADM has `implied SECADM` authority

- A blog entry on this topic:

<http://robertsdb2blog.blogspot.com/2021/09/db2-for-zos-separatesecurity-and-secadm.html>

RACF group IDs can simplify privilege management

- RACF group ID is ID to which multiple individual user IDs can be connected
- Db2 privileges and authorities can be granted to RACF group IDs
- Db2-supplied sample connection and sign-on exits make RACF group IDs to which primary authorization ID is connected *secondary authorization IDs*
 - Your Db2 privilege set (term used in Db2 documentation) is union of privileges granted to your primary authorization ID and to all of your secondary auth IDs
 - Authorization for most Db2 actions requires that privilege be in your privilege set
- Granting privileges to RACF group IDs can especially helpful if you follow best practice of granting the minimum privileges required to do a job
 - That can sometimes be a complex mix of privileges – OK, grant that mix to a RACF group ID associated with the job, then connect individual IDs to that group ID

Preventing “hijacking” of Db2 privileges

- **DDF-using applications** often connect to Db2 with ID and password
 - ID is **application’s ID**, and privileges granted to the ID are those required for successful execution of SQL statements issued by application
- ID and password will be known to some people – what prevents person from connecting to Db2 from laptop using **application’s ID and password**?
 - A defense against this: Db2 **role and trusted context** functionality
 - Grant Db2 privileges needed by application **to Db2 role** – not to application’s ID
 - Create **Db2 trusted context** that says, “Role ABC’s privileges can be used by process that connects to Db2 using ID XYZ, and **from this set of IP addresses**”
 - So what if someone connects to Db2 with application’s ID? **ID has no privileges**
- Blog entry: <http://robertsdb2blog.blogspot.com/2019/03/a-case-study-implementing-db2-for-zos.html>

Row permissions and column masks

Often preferable to “security views”

- At one time, about the only way to prevent ID SMITH from seeing data in certain rows or in a certain column of a table was via “security view”
 - SELECT in CREATE VIEW would have predicate that would filter out rows that SMITH shouldn’t see, or would not include “off limits” column in its select-list
 - Then grant to SMITH SELECT on view but not on underlying table
- Approach worked, but created some headaches for developers, DBAs
 - Developer: view could not have same fully-qualified name as underlying table – that could complicate programming
 - DBA: every view another database object to be managed – and, “views on views” could really complicate things (like the game Jenga – “Don’t touch that view!”)
- Row permissions and column masks, introduced with Db2 10 for z/OS, provided an alternative approach that is often a better way to go

Row permissions and column masks

- Example: for table T1, only process with **secondary auth ID XYZ** should see rows with DEPTNUM = 'A01' and actual values in column C1
- CREATE PERMISSION statement: check to see if ID associated with a SELECT from T1 **is associated with group ID XYZ**
 - If yes, data in rows with DEPTNUM = 'A01' is accessible
 - **If no**, predicate **filtering out rows** with DEPTNUM = 'A01' is **automatically added to query**, regardless of whether query is static or dynamic
- CREATE MASK: also check T1-reading process for **secondary auth ID XYZ**
 - If yes, process can retrieve actual data values from column C1
 - **If no**, **data-masking CASE expression** is **automatically applied to C1 when referenced in query's select-list**, regardless of whether query is static or dynamic

Row permissions and column masks: advantages

- For developers: queries just **reference the target table**, and row permission or column mask does the security work based on ID of table-accessing process – no need to remember names of views
- For DBAs: **easier to manage** than a multiplicity of views
- For security team: strong protection, **even against ID with SYSADM authority**
 - If row permission or column mask looks to see if ID associated with query is connected to group ID XYZ (for example), and SYSADM's ID is not so connected, **SYSADM will not be able to see** rows and/or actual data values in column
- A related blog entry:
<http://robertsdb2blog.blogspot.com/2013/04/db2-for-zos-goodbye-security-views.html>

One more thing about column masks

- A DBA might tell me: “We need to **encrypt** data in **column C1** of table T1”
 - My initial response: “If you’re talking about at-rest encryption, just use z/OS data set encryption and **encrypt the whole table space** (and indexes, etc.)”
 - DBA: “We need to make sure that **only certain users can see data** in column C1”
 - Me: “That’s **not necessarily an encryption** situation”
- One option: encrypt at column level, using Db2 built-in function ENCRYPT_DATAKEY, but **that approach is NOT application-transparent**
- Application-transparent solution: **create a mask** for column C1 of table T1
 - Does not involve encryption, but **enables only certain users to see values in C1**
 - Request that DBA got was for data protection – column-level encryption is **a way to achieve objective**, but so is column mask, and latter is **application-transparent**

Client authentication (for client-server applications)

Changing the password for a DDF-using application

- As previously mentioned, DDF-using applications often authenticate to Db2 using an ID and a password
- In times past, not unusual for an application's ID to have a “never expires” password – increasingly **unacceptable for security auditors**
- If the password for an application's ID has to be changed periodically, can that be done **without impacting application availability**?
 - YES – I have seen that done **through the use of two IDs** for an application
 - The privileges needed by the application are granted to **both IDs**
 - When password of ID1 about to expire, application is changed to connect to Db2 **using ID2** (same done in reverse when password of ID2 is about to expire)
 - If application **running on at least two servers**, ID switch does not cause outage

Alternatives to password for authenticating to Db2

- Besides password, how can client-server application authenticate to Db2?
 - One alternative is to use **RACF PassTickets**
 - z/OS doc: <https://www.ibm.com/docs/en/zos/2.5.0?topic=guide-using-passtickets>
 - Another alternative: **certificate-based** client authentication
 - Some folks get this mixed up with SSL encryption
 - SSL encryption involves use of certificate, but usually that is certificate of **host system**, stored on client system and presented to complete “SSL handshake”
 - Application that gets SSL connection to Db2 for system typically authenticates with password, but use of **authentication via client certificate** is an option
 - See redbook [*IBM DB2 for z/OS: Configuring TLS/SSL for Secure Client/Server Communications*](#) – “Client access to Db2 using TLS/SSL client authentication”

Also related to client authentication: TCPALVER

- This ZPARM parameter answers question, “Is client process requesting a TCP/IP connection to Db2 subsystem considered to be already verified?”
- Default is NO – client **must authenticate to Db2** with password or certificate
- At some sites, value of TCPALVER is YES – that means **Db2 does not require client to authenticate**
 - Though not a gaping security hole (RACF will almost certainly require client authentication), TCPALVER=YES is not recommended
 - Sometimes, TCPALVER=YES is a remnant of time when only systems requesting TCP/IP connections to Db2 for z/OS were other Db2 for z/OS subsystems
 - If subsystem DB2A is DRDA requester (over TCP/IP) to DB2B, and DB2A is not sending a password when connecting to DB2B, update DB2A’s **communications database** to send password **before going to TCPALVER=NO for DB2B**

Auditing

Auditing: the complement to privilege management

- Even if privileges managed carefully, **some IDs will have access to sensitive data** – guarding against **misuse of access** is what auditing is about
- Db2 10 delivered **audit policy** functionality – here are things you can track:
 - Data access attempts that **failed due to inadequate authorization**
 - Occurrences of a user **changing SQL ID**
 - **ALTER TABLE** actions
 - Read or update actions targeting **a particular table**
 - **Utility** execution
 - Privilege **GRANT** and **REVOKE** actions
 - **System administration "super-user"** activity (e.g., SYSADM, SYSOPR)
 - **Database/security administration "super-user"** activity (e.g., DBADM, SECADM)

A little more on Db2 audit policy functionality

- Define audit policy with insert into `SYSIBM.SYSAUDITPOLICIES` table
- Audit policy, once defined, put into effect with `-START TRACE` command
- An activated audit policy will cause Db2 to generate trace records (the particular IFCIDs generated will depend on what is being audited)
 - The generated trace records can be [formatted using your Db2 monitor](#) (report might have a name like, “record trace report”)
- Db2 12 FL509 introduced [tamper-proof audit policy](#) functionality – permission to modify or delete tamper-proof policy must be [provided by RACF administrator](#)
- Audit policy information in online Db2 for z/OS documentation:
<https://www.ibm.com/docs/en/db2-for-zos/13?topic=db2-audit-policy>

Application architecture

Application architecture and Db2 data security

- **Security advantage of static SQL:** think about a `SELECT... FROM TABLE_XYZ`
 - If **static query**, auth ID (usually, of application) requires only **execute authority on Db2 package** of which static query is a part – **no table access privileges needed**
 - If query is dynamic, auth ID requires **SELECT privilege** on `TABLE_XYZ`
- What if DBAs want static SQL for security reasons, but application developers want to use a generic SQL interface such as JDBC or ODBC?
 - SQL in JDBC or ODBC form is **dynamic** from Db2 for z/OS perspective
 - An approach that can satisfy DBAs and app developers: put “table-touching” SQL statements in **Db2 stored procedures** that are **called via JDBC or ODBC**
 - This approach enables **dynamic invocation of static SQL** statements
 - Developers who know JDBC or ODBC usually **familiar with stored procedures**

More on application architecture and data security

- Besides enabling use of static SQL in a JDBC/ODBC situation, Db2 stored procedures provide another security benefit: **database schema obfuscation**
 - Because table and column names only referenced in SQL statements coded in stored procedures, client-side developers **do need to know that information**
 - The smaller the number of people who are **familiar with a database schema**, the better from a data security perspective
- Speaking of static SQL, keep in mind that **Db2's REST interface** is about invocation of **static SQL statements** via REST requests
 - What you're REST-enabling is **Db2 package** associated with static SQL statement (SELECT, INSERT, UPDATE, DELETE, TRUNCATE or **CALL [of stored procedure]**)
 - Besides security advantage of static SQL, REST interface extends **programming language options**: if program can issue REST request, program can access Db2

Test data management

Db2 dev/test, and the data security “threat area”

- You probably have sensitive data values in your production Db2 for z/OS environment, and you probably have protections in place for that data
- Question: how do you populate tables in **development/test** Db2 systems?
 - If these tables contain data copied from production (very common), are sensitive data values **as well-protected** in dev/test as they are in production?
 - Maybe not – maybe **more IDs have access** to the data in dev/test
 - Even with comprehensive data protection in effect for dev/test, just having copies of production data increases the **data security “threat area”**
 - Often, best risk-mitigating action is to **mask sensitive data values** in dev/test systems

Options for masking Db2 data in dev/test systems

- Do-it-yourself
 - Could be done **programmatically**, or potentially with **Db2 column masks**
 - **Development effort** could be significant, and some approaches can substantially **increase CPU and elapsed time** for processes that populate dev/test Db2 tables
- **Data masking software** (from IBM, other vendors) can deliver a lot of value:
 - **CPU-efficiency** (some can mask large volumes of data with little overhead)
 - **Time-efficiency** (setting up tool for a masking operation can be pretty quick)
 - **Masking choices**, including (but not limited to):
 - **Generation of values** (e.g., credit card numbers) that look like **“the real thing”**
 - Value **“shuffling”** – actual values are unmasked, but **switched from one row to another** (often, value is sensitive **only in combination** with other values in row)

RACF management of Db2-internal security

There is often confusion about this

- Note: Db2 **external** security is concerned with, “Which IDs can connect to this Db2 system, and how?” while Db2 **internal** security is concerned with, “Once ID has successfully connected to Db2 system, **what can that ID do?**”
- If asked, “Do you use RACF to manage Db2-internal security?” some Db2 DBAs will say, “Yes,” when the answer is, in fact, “No” – **why is that?**
 - Sometimes, it’s because person gets Db2 internal and external security **mixed up**
 - Sometimes, it is because RACF is **involved** with Db2 internal security (“We grant privileges to **RACF group IDs**”)
- Simplest way to figure this out: if you use **GRANT and REVOKE** SQL statements to assign privileges and authorities to IDs, you are using Db2 – **not RACF** – to manage Db2 internal security

How RACF can manage Db2 internal security

- Done by way of Db2-provided exit, `DSNXRXAC`
- When that exit is `activated`, and ID SMITH wants to (for example) SELECT data from table T1, `Db2 drives the exit` and asks RACF if SMITH can do that
 - RACF checks to see if a) it has a `resource defined` that maps to SELECT privilege on T1, and b) if ID SMITH has been `permitted access to that resource profile`
 - If `a and b are true`, RACF tells Db2 that `SMITH can SELECT from T1`
 - If a is true and `b is not`, RACF tells Db2, “SMITH `cannot` do that”
 - If a is not true (no applicable resource defined), RACF defers to Db2
 - Note: when RACF management of Db2 internal security is `fully implemented`, the `authorization tables` in the Db2 catalog (e.g., SYSUSERAUTH, SYSPACKAUTH) can be `empty`, because Db2 is not using them

More on RACF management of Db2 internal security

- Why do some organizations go this route?
 - Often related to desire for **one team** to manage Db2 internal **and** external security
- Does it work?
 - **Absolutely** – if you tell me, “We’re going to do this,” I’ll say, “That’s fine”
- In my experience, **biggest challenge is the transition** – partly because Db2 people and RACF people use **different terminology**
 - Once transition is done, Db2 DBAs **often pretty happy** (one less thing to manage)
- Useful links:
 - Db2 doc: <https://www.ibm.com/docs/en/db2-for-zos/13?topic=db2-managing-security-racf-access-control-module>
 - Blog entry: <http://robertsdb2blog.blogspot.com/2020/01/db2-for-zos-and-racf-part-2-db2.html>

Robert Catterall

rfcatter@us.ibm.com