WHAT IS TRANSPARENT DATA ENCRYPTION IN DB2 AND WHY DO I WANT IT?

There

have been several high visibility data exposures in the news over the last year or so. As a Db2 DBA, one of your responsibilities is the protection of the data your business needs to run. Your auditors, internal and external, may be demanding action to protect your valuable Db2 data from prying eyes. None of us want the cost, potential fines, or negative publicity associated with a potential or real data exposure. IBM has provided a partial solution in Transparent Data EncryptionTM (or "TDE") for Db2, which is part of a wider effort they call zSystem Pervasive EncryptionTM (or "zSPE"). Transparent Data Encryption provides a mechanism to encrypt your data at rest, on disk.

TDE has been provided for both Db2 11 and Db2 12. For Db2 12, there is additional function, which has been provided via Continuous Delivery as part of Function Level 502.

First, let's look at the encryption mechanism itself, how it is implemented, and what you will need to deploy it. Encryption exploits a hardware instruction which is executed in an attached Crypto processor, to minimize the cost of the instruction. IBM decided to implement the feature in the media manager, which is a low-level access mechanism (part of DFSMS) used by BSAM, QSAM, VSAM, and by Db2. It is worth pointing out that this encryption technique does not support tape. z/OS allows three methods of enabling encryption; in priority order, these are: in a RACF dataset profile, as an attribute of the dataset (specified in JCL with DSKEYBL syntax or KEYLABEL in IDCAMS), and as an attribute in a SMS DATACLAS. To be encrypted, a dataset must have the Extended attribute. SMS encryption is available in z/OS 2.2 (if the PTF for APAR OA50569 has been applied) or z/OS 2.3. You will need the attached Crypto processor.

There is a misconception that a z14 is necessary to implement encryption. This is not true, but the Crypto 6-S processor is much faster than the Crypto 5-S, and the 6-S is available only on a z14. So, it is true that you need a z14 for acceptable performance of encryption (more on performance below).

Because encryption and decryption are done at I/O time, they are almost completely "transparent" to Db2. Tablespaces or indexes, or both, can be encrypted; the Db2 catalog and directory can be encrypted, as well as user data. You can also choose to encrypt the BSDS, active logs, archive logs (on disk), image copies, and utility work files. The first and third methods of enabling encryption for Db2 datasets are supported in both Db2 11 and 12. The second can be used for image copies and utility work files as early as Db2 11, but will be made available through DDL for Db2 data only in Db2 12 (as part of Function Level 502). Note that, after encryption has been enabled for a tablespace or index, the next Reorg of the object will encrypt the data.

The actual key to encrypt and decrypt the data is stored in the CKDS ("Cryptographic Key Data Set"). The key itself is system generated, but users access it through the key label, a 64 bit value used to access the actual key. Note that RACF (or equivalent) authorization is needed to access the key label, so to access an encrypted file, you must be authorized for the key label as well as the dataset itself. Be aware that, if you plan a disaster recovery test or to migrate data from one system to another, you must ensure that the CKDS datasets at both locations are synchronized.

Note that, because encryption is implemented in the I/O routines in Db2, the data is available in the clear in the buffer pool, and SQL sees only decrypted data. You may ask why you would want to implement encryption; the answer in most cases, I think, will be to placate your auditors. This feature will not prevent hacking attempts like SQL injection or password stealing for unauthorized SQL access, but it does further close the window against insider accesses to the disk data (bypassing SQL).

In Db2 12, IBM is providing DDL control of encryption as part of Function Level 502. This will allow you to specify "KEY LABEL" as part of the syntax of the CREATE/ALTER TABLE and the

CREATE/ALTER STOGROUP commands. The TABLE syntax can be used only with single table tablespaces, and automatically enables encryption for associated indexes, LOB and XML spaces, and clones. Archive and history tables must be encrypted separately.

You may be curious about the overhead of encryption and decryption. There is some of course, since you are adding encryption to disk writes and decryption to reads, but it is not onerous. I recommend you run some tests in your environment to see how high your overhead is. It does raise the desirability of keeping data pages in the buffer pool (to avoid the overhead incurred during I/O). You need to be aware that utility execution will also cost more. The lower cost utility suite from BMC Software may be worth looking at, to help you offset some of the cost increase incurred with encryption.

There are many choices for Transparent Data Encryption, starting with whether to deploy it, and then choosing the technique to use. We recommend that you choose one method and use it consistently. That will avoid the confusion of figuring out conflicting specifications. Remember to check with third party vendors to see what level of support they provide for TDE. Please note that all of BMC Software's Db2 products support the feature today; you can ask your BMC account team for details.

Talk to your BMC account team or a BMC Sales person about the use of BMC Next Generation Technology utilities, that can help you alleviate the costs associated with encryption. BMC provides full suites of performance, administration, and recovery tools for Db2 on z/OS.