Real-Time DB2 Database Activity Monitoring with the CorreLog Agent for z/OS

Database Activity Monitoring (DAM) is defined by Gartner as "… tools that can be used to support the ability to identify and report on fraudulent, illegal or other undesirable behavior, with minimal impact on user operations and productivity…[1]" This paper describes how the CorreLog Agent for z/OS (CZAGENT), in conjunction with the CorreLog Correlation Server or another SIEM console, may be used for real-time DAM with minimum impact on performance.

Privileged User Monitoring

The PCI DSS standard (10.2) requires automated audit trails for all actions taken by any individual with root or administrative privileges, such as a system programmer or database administrator. To use CZAGENT to create a real-time automated audit trail for all privileged users, it is necessary only to turn on IFCID[2] 361 and start CZAGENT with the default configuration. (Full instructions for controlling DB2 traces are found in the CZAGENT reference manual.) This is a minimally intrusive trace type as it is only activated for administrative actions, not routine database accesses. From then on, all actions by privileged users will be logged to your SIEM console. The log message will include the user ID, the text of any command or SQL statement executed, a code for the specific action taken, the job or CICS transaction name, and the name of the specific database object affected. Two examples are as follows:

1. A privileged user displaying the entire list of database tables from the DB2 instance. This action might be benign and it might be a malicious "fishing expedition." (CorrID: RU018BD3 shows the z/OS job name.)

```
DB2: Subsys: DA1L - IFCID: 361 - UserID: RU018B - AuthID: RU018B -
CorrID: RU018BD3 - Auth: SYSADM - Priv: SELECT - ObjType: Table or
view - Cmd: SELECT * FROM SYSIBM.SYSTABLES - SrcQual: SYSIBM - Src:
SYSTABLES
```

2. A privileged user stopping all DB2 traces. This action is extremely suspicious as it indicates someone trying to "cover their tracks."

```
DB2: Subsys: DA1L - IFCID: 361 - UserID: None - AuthID: SYSOPR -
CorrID: 022.CKPA0201 - Auth: SYSADM - Priv: Stop or Start Trace -
ObjType: User Auth - Cmd: -STOP TRACE(*)
```

[1] http://www.gartner.com/it-glossary/database-activity-monitoring-dam/, accessed January 14, 2013.
[2] DB2 SMF trace record types are identified by "IFCID number." IFCID stands for "instrumentation facility component identifier," which is simply another way of saying "trace record type." There are about 400 record types or IFCIDs, numbered between 1 and 499. Each IFCID type record has a specific layout and describes a specific event.

Invalid Logical Access Attempts

The PCI DSS standard requires that you "implement automated audit trails for … invalid logical access attempts." If you enable IFCID 140 as described in the CZAGENT manual and use the supplied CZAGENT configuration file, then you will achieve the real-time auditing of invalid logical access attempts. IFCID 140 is a low overhead trace because it is only invoked for failed accesses, not every access. An example follows; note the logging of the user ID, job name (RU018ADS), and the submitting user's JES2 node, RACF group, and "Port of Entry" (POE, the terminal name or other source of the attempted access). Notice that the specific failed privilege (SELECT) and object type is identified. (Many of these "human-readable" descriptions are also available as numeric codes if that form is preferred.)

```
DB2: Subsys: DA1L - IFCID: 140 - UserID: RU018A - AuthID: RU018A - CorrID:
RU018ADS - Priv: SELECT - ObjType: Table or view - SrcQual: CORE1010 - Src:
NEWPHONE - ExitRet: -1 - Lang1: Dynamic - Lang3: None - Node: JES2SYSB -
Group: RESTRICT - POE: INTRDR - Sql: SELECT * FROM CORE1010.NEWPHONE
```

Creation and Deletion of System-Level Objects

The PCI DSS standard also requires the automated logging of "audit trails for … creation and deletion of system-level objects." You can log the creation and deletion of DB2 databases by enabling IFCID 97. These messages audit DB2's use of IDCAMS commands to create system-level data objects. IFCID 97 is minimally intrusive because it is generated only for the deletion or creation of databases. The example below shows the deletion and creation of the CORED10U.NEWPHONE database.

```
DB2: Subsys: DA1L - IFCID: 97 - UserID: RU018B - AuthID: RU018B - CorrID:
RU018BDL - RC: 0 - Cmd: DELETE DA1LDB.DSNDBC.CORED10U.NEWPHONE.I0001.A001
PURGE;
```

```
DB2: Subsys: DA1L - IFCID: 97 - UserID: RU018B - AuthID: RU018B - CorrID:
RU018BDL - RC: 0 - Cmd: DEFINE
CL(NAME(DA1LDB.DSNDBC.CORED10U.NEWPHONE.I0001.A001 ) NOERASE LIN
OWNER(SY002A ) RUS SPEED CISZ( 4096)) DATA
(NAME(DA1LDB.DSNDBD.CORED10U.NEWPHONE.I0001.A001 ) KB( 00001440 00000720)
OWNER(SY002A ) SHR(3,3) RUS VOL('* ' ));
```

Data Access Monitoring

The PCI DSS standard (10.2) requires the logging of all accesses to cardholder data. To create an automated real-time audit trail for all accesses to specific DB2 tables it is necessary to take two simple actions at the DB2 subsystem. First, turn on IFCIDs 143 and 144 (detailed instructions are in the CZAGENT manual) and, for each DB2 table to be audited, a person with administrative authority must execute the SQL statement

```
    ALTER TABLE database.table AUDIT access
```

Where *database.table* is the table to be audited and *access* is CHANGES to audit only writes, or ALL to audit both reads and writes. This is a fairly low-overhead trace as only specific tables are affected, and because DB2 only generates only one message per *commit*, not one for every SELECT, INSERT or UPDATE.

Once these commands are in effect, CZAGENT will forward a message every time one of the specified tables is read (IFCID 144) or written (IFCID 143) as requested by the ALTER. Following is an example of a message indicating a read of an audited database.

```
DB2: Subsys: DA1L - IFCID: 144 - UserID: RU018B - AuthID: RU018B -
CorrID: RU018BD3 - DBID: 265 - PSID: 4 - OBID: 18
```

Granted, this is not the most user-friendly message in the world. You can see the user ID and the job name (CorrID: RU081BDR) but the database object is identified only by a database ID, page set ID, and object ID. Fortunately, it is not hard to relate those to a specific table. First, you will probably be monitoring only a relatively small number of tables. Second, DB2 keeps the database and object IDs fairly static, and maintains them in an easily accessible table, SYSIBM.SYSTABLES. To relate those ID numbers to an actual table name, it is necessary to refer to the results of the following query (for the relevant instance of DB2):

```
SELECT DBID, OBID, OWNER, NAME FROM SYSIBM.SYSTABLES
```

The results of the query are relatively stable and may be saved for future reference. For example, on the test system, DBID 265 and OBID 18 identify DSN81010.EMP, one of the tables we are auditing.

Digging Deeper

Suppose that you would like more detail: We know from the IFCID 144 message that job RU018BD3 running on behalf of user RU018B read DSN81010.EMP, but what exactly did it do? IFCID 145[3] shows us, for audited tables, the exact SQL statement that was executed:

```
DB2: Subsys: DA1L - IFCID: 145 - UserID: RU018B - AuthID: RU018B -
CorrID: RU018BD3 - Loc: NA01DA1L - Collection: DSNTEP4 - Prog:
DSN@EP4L - Token: 1914f7b21db184f0 - StmtType: SELECT QUERY Lang1:
Dynamic - Lang3: None - Isolate: CS - DBID: 265 - OBID: 18 - Stmt:
SELECT * FROM DSN81010.EMP WHERE EMPNO = 36
```

You can relate the IFCID 144 message back to its preceding IFCID 145 message because both will have the same Correlation ID (CorrID).

Static and Dynamic SQL

If you have been exposed to SQL primarily through ODBC then you are probably used to SQL statements such as the above SELECT. DB2 supports SQL like this, but it may be the exception rather than the rule in mainframe shops. DB2 refers to this type of statement as "dynamic SQL" because the SQL is constructed "dynamically," moments before it is executed. (Notice Lang1: Dynamic in the message above.) On the other hand, much or most

[3] IFCID 145 tracing is economical because a record is written only for audited tables.

of the SQL in mainframe shops is what DB2 terms “static SQL,” which will be discussed below.

The more common type of mainframe SQL is “static SQL.” Static SQL is embedded in a COBOL or other mainframe source language program and compiled into a binary form and fixed – hence the term “static” – for reasons of performance and security when the program is built. DB2 refers to this process of “fixing” SQL as “preparation” and “binding.”

Auditing static SQL is somewhat more complex than auditing dynamic SQL. If the programmer could only write statements like the above in which something like EMPNO = 36 was hard-coded then the resulting program would be inflexible and not very useful. Instead, a COBOL programmer would write a static DB2 SQL statement (inside a COBOL program) something like

```
UPDATE VEMPLP SET PHONENUMBER = :NEWNO WHERE EMPLOYEENUMBER =
:ENO
```

For readers unfamiliar with static SQL, the colon prefixes on :NEWNO and :ENO indicate that they are COBOL program variables rather than SQL tokens, and that the run-time contents of NEWNO and ENO are to be used as though they were part of the SQL statement. DB2 refers to NEWNO and ENO as “host variables.” Adding to the complexity, notice also that the programmer has referred to the monitored table, DSN81010.EMP by means of a DB2 View, VEMPLP.

From a SIEM point of view static SQL presents two major challenges:

1. By the time the RDBMS action is actually performed the original SQL statement is gone, having been compiled previously (much like any other program source code) into an executable binary form – possibly days, weeks, or even years before; and
2. In addition to the text of the SQL UPDATE statement you might want to know the run-time contents of the host variables NEWNO and ENO.

If the appropriate audit and IFCID 145 tracing are in effect at the time the COBOL program is compiled, then when the compile runs CZAGENT, it would log a message something like

```
DB2: Subsys: DA1L - IFCID: 145 - UserID: RU018B - AuthID: RU018B -
CorrID: RU018BDC - Loc: NA01DA1L - Collection: RU018PHN - Prog:
DSN8BC3 - Token: 1958afbc1a3c8a1a - StmtType: 234 - Isolate: S -
Lang1: See Lang3 - Lang3: IBM COBOL - Stmt#: 541 - StmtID: 11860 -
DBID: 265 - OBID: 18 – Stmt: UPDATE VEMPLP SET PHONENUMBER. = : H
WHERE EMPLOYEENUMBER = : H
```

Notice the DBID: 265 and OBID: 18, identifying this statement as accessing DSN81010.EMP, even though it is accessed through a View. Notice that the host programming source language (IBM COBOL) is identified. Also, in the message, the host variables NEWNO and EMPLOYEENUMBER are indicated with the placeholder “: H.”

What about those host variables? Without the contents of the host variables we only know that *some* row of the audited table was updated to *some* value. If you really want the complete story, then it is necessary to also turn on IFCID 247. (Full instructions are in the CZAGENT reference manual.) An example of the two IFCID 247 messages resulting from execution of the above statement follow:

```
DB2: Subsys: DA1L - IFCID: 247 - UserID: RU018B - AuthID: RU018B -
CorrID: RU018BDR - Loc: NA01DA1L - Pkg: RU018PHN - Prog: DSN8BC3 -
Token: 1958afbc1a3c8a1a - Stmt#: 541 - Type: 452 - SQLDA#: 2 - Data:
"000230"
```

```
DB2: Subsys: DA1L - IFCID: 247 - UserID: RU018B - AuthID: RU018B -
CorrID: RU018BDR - Loc: NA01DA1L - Pkg: RU018PHN - Prog: DSN8BC3 -
Token: 1958afbc1a3c8a1a - Stmt#: 541 - Type: 452 - SQLDA#: 1 - Data:
00 "4265"
```

In the above formatted messages we can see the contents of the two host variable: 4265 and 000230. We know which is the first and which is the second host variable because the message includes the “SQLDA#.” So we know that the run-time contents of ENO (the employee number) was 000230 and therefore that the DB2 row for employee 000230 was the one updated to a phone number of 4265.

On a busy system, we can relate the IFCID 247 run-time messages to their corresponding IFCID 145 compile-time messages because both contain the same Token value – a unique value assigned by DB2 – and both reference the same line number of the original COBOL program (Stmt#: 541).

Performance

CorreLog realizes the importance of having a minimal impact on DB2 performance and z/OS system utilization. The above traces were all chosen for the low impact on DB2 performance. In addition, CZAGENT has the unique ability to further reduce the impact of DB2 trace records by instructing SMF, on a record by record basis, to suppress logging installation-specified records to the SMF datasets. So, for example, if you enable IFCID 145 strictly for CZAGENT, and do not require IFCID 145 records in your SMF logs, then you can instruct CZAGENT to tell SMF not to write IFCID 145 records to disk. By not logging these SMF records to disk the performance impact is further reduced.

Additional Auditing Options

Additional real-time auditing options are available in CZAGENT for use as desired. They should be used with care because of the potential overhead involved.

IFCID 3 shows accounting counters by DB2 instance:

```
DB2: Subsys: DA1L - IFCID: 3 - AuthID: LDAPSRV - CorrID: LDAPSRV -
Plan: DSNACLI - OpID: LDAPSRV - UserID: LDAPSRV - Trans: LDAPSRV -
WrkSta: RRSAF - Loc: NA01DA1B - LU: NA01DA1B - Conn: RRSAF - SQL:
{Create Synonym: 1 - Create Store Group: 1 - Drop Index: 1}
```

IFCID 58 audits the completion of every SQL operation. Error and warning conditions are indicated. The IFCID message may be correlated to other message numbers by means of the Token, Stmt#, and StmtID fields.

```
DB2: Subsys: DA1L - IFCID: 58 - UserID: RU018B - AuthID: RU018B -
CorrID: RU018BDR - SQLcode: 0 - SQLerrm: None - SQLerrp: DSN -
SQLwarn: None - Stmt#: 541 - Loc: NA01DA1L - Collection: RU018PHN -
Prog: DSN8BC3 - Token: 1958afbc1a3c8a1a - StmtType: Static - StmtID:
11928
```

IFCID 63 and IFCID 350 audit the SQL text for *all* SQL requests, not just audited tables (dynamic SQL at execution time and static SQL at compile time). IFCID 63 is somewhat easier to parse than IFCID 350, but IFCID 63 messages truncate SQL statements at 5000 bytes, whereas one or more IFCID 350 messages contain the complete text of every SQL statement no matter how long (as do IFCID 145 messages).

```
DB2: Subsys: DA1L - IFCID: 350 - AuthID: RU018B - CorrID: RU018BD3 -
Plan: DSNTP410 - OpID: RU018B - UserID: RU018B - Trans: RU018BD3 -
WrkSta: BATCH - Loc: NA01DA1L - NetID: USASG - LU: NA01DA1L - Conn:
BATCH - StmtType: Dynamic - StmtID: 0 - StmtSeg: Only - Stmt:
DECLARE TELE3 CURSOR FOR SELECT * FROM VPHONE WHERE LASTNAME = : H
AND FIRSTNAME LIKE : H
```

IFCID 90 and 91 messages audit DB2 console commands and their completion return and reason codes. These messages are largely redundant with IFCID 361, Privileged User Monitoring.

```
DB2: Subsys: DA1L - IFCID: 90 - AuthID: RU018B - CorrID:
023.GCSCN602 - Cmd: -STA TRA(AU) C(30) IFCID(247,350)
```

```
DB2: Subsys: DA1L - IFCID: 91 - UserID: None - AuthID: RU018B -
CorrID: 023.GCSCN602 - RC: 0 - Reas: 0
```

IFCID 107 messages audit *every* table open and close, not just audited tables. They are also an additional source of correlation between DBID/PSID pairs and their corresponding database and table names.

```
DB2: Subsys: DA1L - IFCID: 107 - UserID: RU018B - AuthID: RU018B -
CorrID: RU018BDL - Type: Open - DBID: 274 - DB: CORED10U - PSID: 5 -
Obj: NEWPHONE
```

IFCID 239 messages audits plan usage by collection and program name:

```
DB2: Subsys: DA1L - IFCID: 239 - Subsys: DA1B - AuthID: LDAPSRV -
CorrID: LDAPSRV - Plan: DSNACLI - OpID: LDAPSRV - UserID: LDAPSRV -
Trans: LDAPSRV - WrkSta: RRSAF - Loc: NA01DA1B - NetID: USASG - LU:
NA01DA1B - Conn: RRSAF - Pkg: {Collection: DSNAOCLI - Prog:
DSNCLIC1} - Pkg: {Collection: DSNAOCLI - Prog: DSNCLIMS} - Pkg:
{Collection: DSNAOCLI - Prog: DSNCLINF} - Pkg: {Collection: DSNAOCLI
- Prog: DSNCLIC1} - Pkg: {Collection: DSNAOCLI - Prog: DSNCLIMS} -
Pkg: {Collection: DSNAOCLI - Prog: DSNCLINF}
```

Additional Help and Information

CorreLog is pleased to support proof-of-concepts, and provide technology proposals and demonstrations. Visit our website today for more information, or contact us using the information below.

EMEA

Agileise Ltd
Enterprise House
Roundwood Lane
Harpenden
AL5 3BW
UK

+44-1582-380140

info@agileise.com

www.agileise.com

www.agileise.com
bringing agility to your business

North America

CorreLog, Inc.
1004 Collier Center Way, 1st Floor
Naples, Florida, USA 34110

Toll-free: 1-877-CorreLog (267-7356)
International: +1-239-514-3331

support@correlog.com

www.correlog.com

Specifications for CorreLog CZagent for z/OS are subject to change at anytime.