29

RACF update

August 2002

In this issue

RACF Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: fionah@xephon.com

North American office

Xephon
Post Office Box 350100
Westminster CO 80035-0100
USA
Telephone: (303) 410-9344

RACF Update on-line

Code from *RACF Update*, and complete issues in Acrobat PDF format, can be downloaded from http://www.xephon.com/racf; you will need to supply a word from the printed issue.

Subscriptions and back-issues

A year's subscription to *RACF Update* (four quarterly issues) costs £190.00 in the UK; $290.00 in the USA and Canada; £196.00 in Europe; £202.00 in Australasia and Japan; and £200.50 elsewhere. The price includes postage. Individual issues, starting with the August 1999 issue, are available separately to subscribers for £48.50 ($72.75) each including postage.

Editor

Fiona Hewitt

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

Contributions

When Xephon is given copyright, articles published in *RACF Update* are paid for at £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above or download a copy of our *Notes for Contributors* from http://www.xephon.com/index/nfc/css/nfc.html

Printed in England.

Finding the current RACF user ID

From an application programming perspective, the most common RACF-related requirement is to determine the current user ID. However, although this is a simple concept, it's almost impossible to code directly in a common programming language like COBOL. This article explores some solutions in common environments. (Note that, to ensure that the information remains practical for use in production applications, solutions requiring APF authorization are not considered.)

ASSEMBLER

Although some environments, such as CICS and ISPF, offer alternative solutions, wading through z/OS Control Blocks is the most practical and reliable approach for Assembler in most environments. Because it's so easy to find – starting at virtual address zero – the best place to start is usually the Prefixed Save Area (PSA).

The PSA contains a pointer (PSATOLD) to the Task Control Block (TCB), which in turn has a pointer (TCBJSCB) to the Job/Step Control Block (JSCB). Although the JSCB has a field (JSCBJCTA) that gets you to the Job Control Table (JCT), it's a Scheduler work area Virtual Address (SVA), not an address pointer like those we've seen up to this point. The SWAREQ macro must be used to translate the SVA into an address pointer to the JCT associated with the calling program.

The JCT actually contains the JCTUSER field where the current RACF User ID can be found. Technically, the documentation says that this is the value of the USER= field on the JCL job statement that started the current job or TSO session. In fact, it's accurate even when the USER= field is not specified. Whatever User ID was associated with the job is stored here in JCTUSER.

But how do you figure all this out? From the MVS Data Areas manuals, currently a five-volume set for Release 3 of z/OS, order numbers GA22-7581-02 through GA22-7585-02, also available on-line for viewing or download at:

http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/Shelves/IEA2BK20

Begin by finding a control block with the data you want. As well as looking for a control block with a name that sounds related to the data, the Index at the back of each volume is an excellent place to look. There are also search functions for the on-line versions. You can even search across a complete Bookshelf of manuals.

Once you've found the Control Block that contains the data you want, read the Heading Information in the Data Areas manual. The 'Pointed to by:' section tells you how to find the Control Block. For JCT, it says:

- JSCBJCTA field (SVA) of the JSCB data area.
- SWBUFPTR field in IEFZB506 upon return from IEFQMREQ macro (preferred method of SVA translation).
- SWBLKPTR field in IEFZB505 on return from SWAREQ macro.

Of course, the first point means that you must find the JSCB by repeating this process. What the second point doesn't tell you is that IEFQMREQ requires APF authorization, while SWAREQ can run without, making SWAREQ the preferred approach for applications. The last thing you want, from a system security/stability point of view, is to have to store some or all of an application in an APF-authorized library.

Put it all together into an Assembler subroutine that returns the user ID to an eight-byte field passed to it as a parameter by a calling program in virtually any programming language, and you have:

```
*  USERIJCT ROUTINE
*  GET FROM JCT (JOB CONTROL TABLE) Z/OS CONTROL BLOCK.
*  CALL WITH AN 8 BYTE AREA AND, UPON RETURN, THE CURRENT USER ID
*  WILL BE INSERTED IN THAT 8 BYTE AREA, WITH TRAILING BLANKS,
*  IF REQUIRED.
R1       EQU   1                  REGISTERS USED IN ROUTINE
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
```

```
PARMS    DSECT ,                  PARAMETER PASSED FROM CALLER
JOBUSER  DS    CL8                JCL JOB STATEMENT USER= VALUE
         IHAPSA ,                 PSA DSECT DEFINING FIRST 4K OF MEMORY
         IKJTCB ,                 TCB USED TO GAIN ACCESS TO JSCB
         IEZJSCB ,                JSCB USED TO GAIN ACCESS TO JCB
         IEFAJCTB ,               JCB USED TO GET JOB USER= VALUE
*  DSECTS REQUIRED BY SWAREQ:
         IEFZB5Ø5 LOCEPAX=YES
         IEFJESCT
         CVT   DSECT=YES
         IEFQMIDS
USERIJCT CSECT
*  STANDARD PROLOGUE
         STM   R14,R12,12(R13)    SAVE CALLER'S REGISTERS
         LR    R12,R15            ESTABLISH ADDRESSABILITY USING R12
         USING USERIJCT,R12       AS BASE REG. FROM R15 ENTRY POINT.
         ST    R13,SAVEAREA+4     OLD SAVEAREA POINTER IN NEW SA
         MVC   8(4,R13),=A(SAVEAREA)    NEW SA POINTER IN OLD SA
         LA    R13,SAVEAREA       START USING NEW SAVE AREA
*  END PROLOGUE
*  REGISTERS USED IN ROUTINE:
BJOBUSER EQU   R4
BPSA     EQU   R2
BTCB     EQU   R3
BJSCB    EQU   R2
BJCT     EQU   R3
BEPAL    EQU   R5
RABEND   EQU   R6
         L     BJOBUSER,Ø(R1)     ESTABLISH ADDRESSABILITY FOR
         USING JOBUSER,BJOBUSER   CALLER'S PARAMETER TO BE RETURNED
*  A(Ø) -> PSA (PSATOLD) -> TCB (TCBJSCB) -> JSCB (JSCBJCTA) ->
*  JCT (JCTUSER) = USER= FIELD OF JCL JOB STATEMENT
         SR    BPSA,BPSA          PSA STARTS AT ADDRESS ZERO
         USING PSA,BPSA
         L     BTCB,PSATOLD       ADDRESS OF TCB IS FOUND IN PSA
         DROP  BPSA
         USING TCB,BTCB           ACCESS THE TCB
         L     BJSCB,TCBJSCB      ADDRESS OF JSCB IS FOUND IN TCB
         DROP  BTCB
*  JCT POINTER IN JSCB IS AN SVA, SO NEED TO TRANSLATE WITH SWAREQ
         L     BEPAL,AEPAL        PARAMETER LIST (EPAL) FOR SWAREQ
         USING ZB5Ø5,BEPAL
         XC    SWAEPAX,SWAEPAX    INITIALIZE THE PARAMETER LIST
         USING IEZJSCB,BJSCB      ACCESS THE JSCB
         MVC   SWVA,JSCBJCTA      JSCB JCT SVA MOVED TO EPAL
         DROP  BJSCB
*  TRANSLATE THE SVA INTO AN ADDRESS (POINTER)
         SWAREQ FCODE=RL,EPA=AEPAL,UNAUTH=YES,MF=(E,SWAPARMS)
         LTR   R15,R15            BE SURE SWAREQ WORKED
         BNZ   ESWAREQ            NON-ZERO IS AN ERROR
         L     BJCT,SWBLKPTR      USE POINTER RETURNED TO ACCESS JCT
```

```
         DROP  BEPAL
         USING INJMJCT,BJCT
         MVC   JOBUSER,JCTUSER    USER ID IN JCT TO CALLER'S PARM
*  RESTORE/SET REGISTERS
         L     R13,4(R13)         CALLER'S SAVE AREA
         LM    R14,R12,12(R13)    RESTORE REGISTERS AS CALLER LEFT THEM
         SR    R15,R15            ZERO THE RETURN CODE FOR CALLER
         BR    R14                RETURN TO CALLING PROGRAM
*  ERROR IN SWAREQ:
ESWAREQ  ST    R15,SWAREQRC       SAVE SWAREQ RETURN CODE IN MEMORY
         LR    RABEND,R15         SAVE SWAREQ RETURN CODE IN REGISTER
         ABEND (RABEND)           ABEND WITH RETURN CODE AS ABEND CODE
SWAREQRC DS    F                  SWAREQ RETURN CODE, IF NON-ZERO
SAVEAREA DS    18F                THIS ROUTINE'S SAVE AREA
AEPAL    DC    A(EPAL)            POINTER TO SWAREQ PARAMETER LIST
EPAL     DS    CL28               SWAREQ (EXTENDED) PARAMETER LIST
SWAPARMS SWAREQ MF=L              ANOTHER REQUIRED SWAREQ PARM LIST
*  AVOID ERRORS WITH ADDRESSABILITY OF ASSEMBLER STORAGE LITERALS
*  E.G. - =A(Ø)
         LTORG
         END
```

REXX

REXX, on the other hand, provides the current user ID in a built-in function named USERID. The only trick is remembering the parentheses that must follow: userid(). A very short program to demonstrate its use would be:

```
/* REXX */
say userid()
```

Technically, USERID is a non-SAA function that works as you might expect in z/VM, but doesn't exist on other platforms – except for z/OS, of course, where REXX can also be run in non-TSO/E address spaces, but the USERID function behaves less predictably. Not only can you replace the USERID module with your own, but, even when you do not, the value is determined by the first true answer to the following questions:

- Is the user ID non-null and no more than seven characters long?
- Does the job step have a name (Stepname)?
- Does the job have a name (Jobname)?

CLIST

Like REXX, CLIST is an easy language in which to code access to the current user ID. &SYSUID is a CLIST control variable that returns the TSO/E ID, as in the following very short CLIST:

```
WRITE &SYSUID
```

If you're writing CLISTs that need to work in MVS/XA and before, where pre-TSO/E TSO might be running, &SYSUID is also supported there.

JCL

JCL is another easy place to find the current user ID. &SYSUID is a System Symbol that the JCL interpreter replaces by the current user ID, but there's one trick: in its most common usage, in order to specify the HLQ of a DataSet Name (DSN) in a JCL DD statement, two periods must follow &SYSUID, as in the following example:

```
//INPUT DD DSN=&SYSUID..PAYMAST.DATA,DISP=SHR
```

&SYSUID is the only System Symbol that can be used in z/OS batch JCL. Conversely, &SYSUID is a JCL System Symbol, which means it can only be used in JCL. This is a shame, as it would have made a simpler solution than USERIJCT above:

```
ASYSUSER CSECT ,
*  EXAMPLE OF USE OF ASASYMBM
*  THIS PROGRAM WILL NOT WORK BECAUSE &SYSUID IS ONLY AVAILABLE
*  FOR USE WITHIN JCL.  &JOBNAME WORKS, HOWEVER.
RØ       EQU   Ø
R1       EQU   1
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
LUSERID  EQU   8
         STM   R14,R12,12(R13)
         LR    R12,R15
         USING ASYSUSER,R12
         ST    R13,SAVEAREA+4
         MVC   8(4,R13),=A(SAVEAREA)
         LA    R13,SAVEAREA
*  END PROLOGUE
         B     START
```

```
*  BEGIN DATA
SAVEAREA DS    18F
*  THE AMPERSAND MUST BE DOUBLED TO PREVENT CONFUSION WITH
*  WITH A MACRO SYMBOL WITHIN ASSEMBLER.
SYSUSER  DC    C'&&SYSUID'
TLENGTH  DS    F                  TARGET (USER ID) LENGTH
RC       DS    F
         ASASYMBP DSECT=NO
*  BUILD PARAMETER VALUES FOR CALL TO ASASYMBM
START    XC    SYMBP(SYMBP_LEN),SYMBP    ZERO ENTIRE AREA
         MVC   SYMBPPATTERN@,=A(SYSUSER)
         MVC   SYMBPPATTERNLENGTH,=AL(L'SYMBPPATTERNLENGTH)(L'SYSUSER)
*  ADDRESS OF CALLER'S FIRST PARAMETER (USER ID TO BE RETURNED)
         L     R1,Ø(R1)
         MVC   Ø(LUSERID,R1),=(LUSERID)C' '   IN CASE IT IS SHORT
         ST    R1,SYMBPTARGET@
*  TARGET (USER ID) LENGTH IS BOTH AN INPUT AND OUTPUT FIELD TO
*  ASASYMBM
         MVC   TLENGTH,=AL(L'TLENGTH)(LUSERID)
         MVC   SYMBPTARGETLENGTH@,=A(TLENGTH)
         MVC   SYMBPRETURNCODE@,=A(RC)
         LINK  EP=ASASYMBM,MF=(E,SYMBP)
*  RESTORE/SET REGISTERS
         L     R13,4(R13)         CALLER'S SAVE AREA
         L     R14,12(R13)
         L     R15,RC             RETURN THE ASASYMBM RETURN CODE
         LM    RØ,R12,2Ø(R13)
         BR    R14
         END
```

Like the JCTUSER field in the JCT control block, you'll often see definitions stating that &SYSUID delivers the value of the USER= parameter of the JCL JOB statement. In fact, if you omit USER=, &SYSUID (and JCTUSER) still delivers the RACF user ID assigned to the batch job.

There are a few places in the JCL of a batch job where you can't use &SYSUID. Most notable are JES2 and JES3 control statements, where &SYSUID can't be used at all. But it's also worth mentioning the in-stream dataset. Although they're not technically JCL, it's still very tempting to try the following:

```
//IDCAMS  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN   DD  *
      DEFINE CLUSTER (NAME(&SYSUID..TESTF.VSAM) -
         KEYS(44 Ø) -
         RECORDS(1ØØØØØ) INDEXED -
         VOLUMES(ISPW89) -
```

```
         RECORDSIZE(80 80))
//
```

It doesn't work. But the following are fairly common uses of &SYSUID which do work:

```
//JONPEAMS JOB  (ACCT#,,JEP),'JON PEARKINS',REGION=4M,
//        LINES=(5,CANCEL),TIME=1,  1 MINUTE, 5000 LINES LIMIT
//        CLASS=A,MSGCLASS=K,NOTIFY=&SYSUID
//SYSUPARM EXEC PGM=SYSUPARM,PARM='&SYSUID'
//ISPPROF DD DSN=&&ISPPROF,LIKE=&SYSUID..ISPF.ISPPROF,
//        DISP=(NEW,DELETE,DELETE)
```

COBOL IN BATCH

Like most other traditional programming languages, COBOL doesn't offer any way within the language to obtain the current user ID. It would even be worth trying to call ASASYMBM from COBOL if &SYSUID was valid outside of JCL. A simpler, though arguably more error-prone, solution is to pass the user ID as a parameter from JCL using &SYSUID in the PARM= field of the EXEC statement:

```
//SYSUPARM EXEC PGM=SYSUPARM,PARM='&SYSUID'
```

To access the PARM= field in a COBOL program requires a knowledge of the format of a z/OS PARM field. It is passed as a single parameter, effectively a record. The first two bytes are a binary length field specifying the length of the string that follows. The COBOL code to access the user ID passed as a PARM field is shown below:

```
       ID DIVISION.
       PROGRAM-ID. SYSUPARM.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
       DATA DIVISION.
       FILE SECTION.
       WORKING-STORAGE SECTION.
       77  USERID PIC X(8).
       LINKAGE SECTION.
       01  PARM-EQUALS.
           05  PARM-LENGTH PIC S9(4) BINARY.
           05  PARM-VALUE.
               10  PARM-CH PIC X
                           OCCURS 0 TO 9999 TIMES
                           DEPENDING ON PARM-LENGTH.
       PROCEDURE DIVISION USING PARM-EQUALS.
           MOVE PARM-VALUE TO USERID.
           DISPLAY USERID.
```

```
        STOP RUN.
```

A much cleaner approach is to call the Assembler USERIJCT routine shown in the Assembler section above. The COBOL code, equivalent to SYSUPARM, would look like this:

```
       ID DIVISION.
       PROGRAM-ID. CALLUJCT.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
       DATA DIVISION.
       FILE SECTION.
       WORKING-STORAGE SECTION.
       77  USERID PIC X(8).
       PROCEDURE DIVISION.
           CALL 'USERIJCT' USING USERID.
           DISPLAY USERID.
           STOP RUN.
```

That, of course, eliminates the need for the PARM= field:

```
//CALLUJCT EXEC PGM=CALLUJCT
//STEPLIB  DD  DSN=&SYSUID..ACTIVE.LOAD,DISP=SHR
//         DD  DSN=CEE.SCEERUN,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//
```

The above assumes you used the Program Binder or Linkage Editor without NCAL specified, including USERIJCT in the load module.

COBOL IN TSO

We'll look specifically at ISPF below, but COBOL in TSO, with or without ISPF, can be addressed in much the same way as COBOL in batch was above. Instead of JCL, you have TSO commands. The JCL System Symbol &SYSUID is not available, but the CLIST control variable of the same name is. The JCL

```
//SYSUPARM EXEC PGM=SYSUPARM,PARM='&SYSUID'
//STEPLIB  DD  DSN=&SYSUID..ACTIVE.LOAD,DISP=SHR
```

becomes

```
CALL ACTIVE.LOAD(SYSUPARM) '&SYSUID'
```

This doesn't mean that you can type that command into TSO/E, either after a READY prompt or in ISPF, in Option 6, or on any command

line preceded by TSO. What it does mean is that you can create it as a one-line CLIST to initiate the COBOL program. If the application already uses a CLIST to initiate the COBOL program, you can just modify the CALL statement in the existing CLIST.

The REXX equivalent is almost as short. REXX's concatenation rules being what they are, the three statements that follow the REXX header all work, but the first statement has the annoying property of inserting a leading blank in front of the user ID and chopping off the last character of 8-character user IDs.

```
/* REXX */
"CALL ACTIVE.LOAD(SYSUPARM) '" userid() "'"
"CALL ACTIVE.LOAD(SYSUPARM) '"userid()"'"
"CALL ACTIVE.LOAD(SYSUPARM) '" || userid() || "'"
```

Calling the Assembler USERIJCT routine from COBOL is another option. In fact, the same CALLUJCT COBOL batch program shown above works just as well in TSO.

You could also call a REXX subroutine from COBOL and have it return the value of its USERID function, but REXX calls require the use of TSOLNK. Unfortunately, it takes quite a few lines of code to use TSOLNK.

ISPF DIALOGUE MANAGER

The ISPF System Variable ZUSER is the user ID. It can be used in Dialogue Tag Language (DTL), as well as in dialogues written in CLIST, REXX, and traditional programming languages. In fact, the IBM-supplied ISPF main menu uses ZUSER; the relevant code is shown below.

ISPF Primary Option Menu Definition:

```
)INIT
.ZVARS = '(ZCMD ZUSER ZTIME ...)'
)END
Master Application Menu DTL Source
<varclass name=vco type='char 7'>
<varlist>
  <vardcl name=zuser varclass=vco>
</varlist>
<panel name=isp@prim help=isp00003 padc=user keylist=isrnsab applid=isr
       width=80 depth=24 menu prime window=no>&panel_title;
```

```
<area dir=horiz>
  <region dir = vert>
    <divider>
    <dtacol pmtwidth=10 entwidth=8>
       <dtafld datavar=ZUSER usage=out>  &status_userid;
    </dtacol>
  </region>
  &ispzprim;
</area>
</panel>
```

CICS

Even if you expected user ID to be an EXEC Interface Block (EIB) field, you'll still find it very easy to obtain the current user ID in Command Level CICS. For example, in COBOL, you would code:

```
EXEC CICS ASSIGN USERID(user-id) END-EXEC.
```

The user-id would be replaced with an 8-character field in working storage, or the linkage section:

```
77  USERID PIC X(8).
```

IMS/TM

Userid is the seventh of ten fields in an I/O Program Communications Block (I/O PCB). It is eight bytes long. But, to know whether or not you actually have the user ID, you must check the last useful field in the I/O PCB, the one-character Userid Indicator field:

- U – user's identification from the source terminal during sign-on.
- L – LTERM name of the source terminal if sign-on is not active.
- P – PSBNAME of the source BMP or transaction.
- O – other name.

Alternatively, you could use the INQY ENVIRON call which returns a large block of character data, including the User Identifier field. In COBOL, the following code would return that large block of characters into a variable you would define as a 01 level record named ENV-DATA, within which you could define the User Identifier field.

```
EXEC DLI INQY ENVIRON AIB-AREA ENV-DATA END-EXEC.
```

For INQY ENVIRON requests, IMS uses the PSTUSID field of the region's Partition Specification Table (PST). PSTUSID is:

- The name of the PSB currently scheduled into the BMP region, for message-driven BMP regions that haven't completed successful GU calls to the IMS message queue, and for non-message-driven BMP regions.
- Usually the input terminal's RACF ID, derived from the last message retrieved from the message queue, for message-driven BMP regions that have completed a GU call successfully and for any MPP region.
- The input terminal's LTERM, if the terminal has not signed on to RACF.

REXX IN IMS

REXX has a number of special interfaces in the IMS environment. One is IMSQUERY.

The REXX function call IMSQUERY('USERID') returns the input terminal's user ID, if available. If running in a non-message-driven region, the value is dependent on the specification of the BMPUSID= keyword in the DFSDCxxx PROCLIB member:

- If BMPUSID=USERID is specified, the value from the USER= keyword on the JOB statement is used.
- If USER= is not specified on the JOB statement, the program's PSB name is used.
- If BMPUSID=PSBNAME is specified, or if BMPUSID= is not specified at all, the program's PSB name is used.

DB2

In DB2, the USER special register provides the current user ID; technically, the primary authorization ID of the process. A special register is a storage area defined for a process by DB2. Wherever its name appears in an SQL statement, the name is replaced by the register's value when the statement is executed. This makes it the SQL

equivalent of a function with no arguments. USER has a data type of CHAR(8).

A simple COBOL program to use USER, storing the value in the COBOL variable USERID, and then displaying it, might look like this:

```
ID DIVISION.
PROGRAM-ID. DB2USQL.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
DATA DIVISION.
FILE SECTION.
WORKING-STORAGE SECTION.
    EXEC SQL INCLUDE SQLCA END-EXEC.
77  USERID PIC X(8).
PROCEDURE DIVISION.
    EXEC SQL SET :USERID = USER END-EXEC.
    DISPLAY USERID.
    STOP RUN.
```

PL/I

Like COBOL, PL/I lacks a simple way to obtain the current user ID within the language itself. So much of what has been said about COBOL applies equally to PL/I, especially calls to the USERIJCT Assembler routine whose source code is shown near the beginning of this article.

In environments where they're relevant, EXEC ... END-EXEC constructs differ only in how they are terminated. COBOL uses END-EXEC with a trailing period dependent on COBOL syntax, ie not in the middle of an IF. PL/I always uses a semi-colon (';') to terminate, without any END-EXEC at all.

It's actually possible, in PL/I, to navigate through Control Blocks as was done in Assembler in USERIJCT. The coding is complicated by the fact that IBM provides DSECTs only for Assembler for its Control Blocks, so you would have to code your own equivalents in PL/I. But there is no obvious solution for the required conversion of SVA to address pointer required in the JSCB to locate the JCT. IBM supplies only the SWAREQ macro for Assembler to translate the SVA.

OTHER LANGUAGES

Much of what was said about COBOL applies to other traditional programming languages running in z/OS and z/VM. The EXEC ... END-EXEC constructs are typically provided for C and Assembler, but with different terminators. The USERIJCT routine shown above is callable across an even wider set of languages, including FORTRAN. Most languages also support the various methods of passing the user ID as a parameter field to the program, including JCL's EXEC PARM='&SYSUID'.

SUMMARY

Determining the current RACF user ID in traditional programming languages is much more difficult than you might expect. The languages themselves do not include the facility, Language Environment (LE) does not provide a common service, and RACF does not provide an interface.

Jon E Pearkins
(Canada)

Inside IBM – IBM mainframe security update

This article follows on from the 'Inside IBM' that appeared in RACF Update *issue 28 (May 2002, pp 21-34). The article begins by reviewing subsequent updates to RACF and related security enhancements. It ends with a look at security now that Linux is on the mainframe.*

The eServer Security Planner is a free Web-based service intended to simplify security set-up across all IBM server product lines, including zSeries. It can be found at:

http://www.ibm.com/security/news/pr_secplanner.shtml

Z/VM RACF AND DIRMAINT

Version 4.3 of z/VM, which became available on 3 May, added RACF as an optional, per-engine priced feature licensed under an International

Program License Agreement (IPLA). This means that RACF is automatically installed with z/VM 4.3, although it must be licensed and enabled before use.

A Recommended Service Upgrade (RSU) tape must be applied before RACF will support new functions of z/VM 4.3. The end result is equivalent to Version 1.10.0 of RACF with RSU service applied.

Before VM had RACF, Directory Maintenance Facility (DirMaint) was the security administration tool of choice. It remains an optional feature of z/VM, gaining engine-based pricing in z/VM 4.1.

TCP/IP FOR Z/VM

Level 430 of TCP/IP for z/VM detects, reports, and prevents more Denial of Service (DoS) attacks. Level 420 protected against:

- *Smurf*. Internet Control Message Protocol (ICMP) Echo Request packets sent to IP broadcast or multicast addresses.
- *Fraggle*. User Datagram Protocol (UDP) Echo Request packets sent to IP broadcast or multicast addresses.
- *Ping-of-Death*. ICMP Echo Request packets that are too large.

Level 430 has expanded the DoS protection to include:

- *Kiss of Death (KOD)*. Depletes the large envelope queue with invalid Internet Group Management Protocol (IGMP) packets.
- *KOX*. A variant of KOD that uses IGMP packets with spoofed IP addresses.
- *Blat*. A variant of the Land attack, which sends a packet with the source host/port the same as the destination host/port, but with a spoofed source IP address.
- *SynFlood*. Creates a large volume of half-open TCP connections, exhausting the stack TCB pool.
- *Stream*. Floods the TCP/IP stack with TCP packets whose header flags are all off.
- *R4P3D*. A variant of Stream that contains an invalid checksum in the pocket header.

IMB

Intelligent Message Broker for z/OS (IMB) was released in April as a z/OS CICS-based gateway that enables and controls the communication between clients and servers, by eliminating direct connections between them. It provides complete transport transparency between client and server, allowing any-to-any communication via WebSphere MQ (formerly MQSeries), SAP R/3, IBM Information Exchange, e-mail, and fax.

IMB uses standard RACF protection of its resources. Internal users must pass additional gateway controls ensuring extended user authentication. All users must be pre-registered, and user access is limited to the specific authorizations that exist for every user.

With the IMB SAP Bridge, any SAP user has immediate access to all mainframe applications. Distributed users have a secure and controlled on-line connection to newly developed SAP servers. IMB also improves the security and auditability of products like WebSphere MQ Integrator.

Note that, despite the fact it is Version 1, IMB is not a new product. It is, in fact, Version 4 of an IBM internal and Global Services (IGS) product that is now being offered as an IBM program product.

WEBSPHERE MQ 5.3

Version 5.3 of WebSphere MQ for z/OS was also announced in April. This is the base messaging product formerly known as MQSeries.

Secure Sockets Layer (SSL) has been added, and you can choose whether to have just partner authentication, or authentication and message encryption. You can also control who is allowed to connect to a receiver channel by matching distinguished name data in a received certificate with data specified in a template, including a wildcard capability.

RACF certificate name filtering can associate a channel user ID with a received certificate by matching the received distinguished name against a template held in RACF or other SAF-compliant product, again with wildcard capability.

In addition, a new attribute, NLTYPE, for the NAMELIST object is needed by the SSL item to clearly indicate that a NAMELIST object

is a list of names of AUTHINFO objects. It also assists error checking for other object types. Defaults have been set to ensure that existing NAMELIST objects (which do not contain the NLTYPE attribute) work without change.

Version 5.3 also adds context security at the queue level, and provides more effective management of the resource-related security control blocks that the WebSphere MQ security manager caches for improved performance.

TIVOLI

A number of Tivoli products have been renamed since last issue, including several security products – each of the new product names begins with 'IBM Tivoli':

- Access Manager for Business Integration is the new name for Tivoli Policy Director for MQSeries.
- Access Manager for Operating Systems combines Tivoli Policy Director-PDOS and Tivoli Security Manager-Unix Security Enforcement.
- Identity Manager combines four products and components: Tivoli Identity Director, Tivoli User Administration for OS/390, Tivoli Security Manager-Provisioning Capability, and Tivoli Security Manager for OS/390.
- Access Manager for e-business combines the base Tivoli (SecureWay) Policy Director product with its Application Servers component. Although it will not run on z/OS (mainframe Linux support is discussed at the end of the article), Access Manager for e-business does provide mainframe services:
  - Web Single SignOn (SSO) and protection for Web servers on the z/OS enterprise server.
  - Use of the SecureWay Directory on z/OS, with the option of authenticating users through RACF.
  - Security for z/OS-based Web applications without forcing an LPAR to be in the DMZ.
  - Access control for servlets and non-HTTP applications on

the z/OS enterprise server.

- NetView Performance Monitor (NPM) for TCP/IP is now NetView for TCP/IP Performance. (That leaves the NPM name strictly for the original NPM, which supports SNA and has not been renamed.)
- Tivoli Manager for Domino and Tivoli Management Solution for Domino have been combined into the new Monitoring for Messaging and Collaboration. But one thing has not changed: Version 2.2 of Tivoli Manager for Domino is the only z/OS release.
- Version 5.1 of Tivoli Storage Manager now supports encryption for greater security of data being backed up. For mobile users, data is encrypted before it enters the network or passes through public phone lines. Supported clients include Windows NT/ 2000/Me/XP. Encrypted files are also now supported on all Windows 2000 clients.

SMALLTALK

z/OS is one of many deployment platforms for applications developed with Version 6 of VisualAge Smalltalk Enterprise. Applications can use Open SSL to heighten security for any transactions running over Socket Communication Interface (SCI). Open SSL can also be used for Smalltalk-to-Smalltalk connections. There is also HTTPS (SSL) support for Web applications.

Meanwhile, VisualAge Smalltalk Server for OS/390 logs all changes in a source change log and captures change management and versioning information in a source code repository that can reside on the client machine or within a network.

LINUX

Unlike other mainframe operating systems, you cannot buy Linux for zSeries directly from IBM; it's sold through distributors or can be obtained free by applying patches from the IBM developerWorks Web site to the Open Source Linux. For more information, see:

http://www.ibm.com/servers/eserver/zseries/os/linux/dist.html

SECURITY AND LINUX

Ironically, the announcement letter for the zSeries Offering for Linux (z800) has only this to say under the heading 'Security, Auditability, and Control': "*The customer is responsible for evaluation, selection, and implementation of security features, administrative procedures, and appropriate controls in application systems and communications facilities*."

But that same announcement discusses the underlying z/VM security, addressing the concern that an insecure platform could compromise an otherwise secure Linux environment, just as software firewalls are often compromised by security holes in the operating systems they run on.

The eServer Security Planner mentioned at the beginning of this article includes special focus on Linux, so is well worth checking out. And, as we also mentioned above, RACF can now be licensed as an IPLA product with z/VM 4.3. This means that RACF can be licensed on Integrated Facility for Linux (IFL) engines.

TCP/IP for z/VM has recently added a new Internet Message Access Protocol (IMAP) server, improved TCP/IP stack security designed to prevent some Denial of Service (DOS) attacks, and support for HiperSockets. VM Guest RSA-Assist improves Linux application security, allowing Linux guest virtual machines to use the PCI Cryptographic Accelerator (PCICA) that's optional on some zSeries models and included with others. This clear-key RSA support enables hardware SSL acceleration.

DIRECTORY SERVER

Version 4.1 of IBM Directory Server runs on a number of Windows and Unix-based platforms, including mainframe Linux. It's available as a free download from the Download link at:

http://www.ibm.com/software/network/directory

Directory Server is an LDAP identity infrastructure compatible with most IBM directories, including those in Lotus Domino, OS/400, z/OS, and Tivoli products. It provides both master/subordinate replication and peer-to-peer replication with large numbers of master

servers. Directory Server requires DB2 and any of five popular (IBM and non-IBM) Web servers.

TIVOLI MONITORING SOLUTIONS

Linux is the only mainframe platform listed as being supported by Version 5.1.0 of the foundation product IBM Tivoli Monitoring. This, despite the fact that it replaces Tivoli Distributed Monitoring for OS/390, as well as Tivoli Web Component Manager and Tivoli Distributed Monitoring.

Details are still sketchy, but Monitoring for Security and Edge Services promises to extend and validate user roles and security policy throughout an organization, as well as monitoring and managing internal and external transactions to help ensure quick, reliable, and secure data transmission.

Overall security is provided through integration of the base Tivoli Monitoring product with Tivoli Management Framework, Enterprise Console, and Business Systems Manager. The Management Framework allows Tivoli Monitor to run across firewalls. You can either configure Bulk Data Transfer (BDT) proxy mechanism and Secure Socket Layer 3 (SSL3) encryption support, or install the Tivoli Management Framework Firewall Security Toolbox. Although not the default, the Web Health Console can use SSL.

TIVOLI ACCESS MANAGER

Only two of the seven components of Version 3.9 of Tivoli Access Manager for e-business can run on the mainframe under Linux: Access Manager WebSEAL and the all-important foundational Access Manager Base. WebSEAL is an authorization engine for Web applications that can authorize and manage an entire Web site, regardless of how many different servers and server types are involved. Major enhancements in Version 3.9 include:

- Reauthentication, forcing the user to log in again, based on a policy setting or a session timeout, but preserving session-specific data across reauthentication steps.
- A plug-in that will cache HTTP POST data, resuming a previous

POST transaction after reauthentication.

- Support for Transport Layer Security (TLS) and HTTP 1.1 protocols.
- Junction fairness.
- Forms-based Web SSO.
- A session ID that is consistent across reauthentications and available to the back-end via the HTTP header.
- Switch user capability, allowing administrators to log on to Access Manager as another user without knowing the user's password.
- Allowing the creation of multiple WebSEAL component server instances on the same platform.
- Integration with non-IBM Customer Relations Management (CRM), content management, supply-chain, and portal applications.

Jon E Pearkins
(Canada)

RACF restructuring: testing

The third article in our four-part series on RACF restructuring concentrates on testing. For part one of this series, see RACF Update *27, February 2002, pp 8-22; for part two, see* RACF Update *28, May 2002, pp35-50.*

WHAT WE'RE DOING TODAY

In this article, we'll review the creation of a full-sized RACF test LPAR, and examine how it should be used to develop the RACF database structures and refinements for all of your other LPARs. You'll actually recreate this LPAR several times – once for each of your other LPARs now in use – in order to fine-tune your specific security requirements for each system. You'll also learn how to

transfer and translate your current (unstructured) RACF databases into the new (structured) format. And, just as importantly, you'll see how to safeguard the test data that you'll need to transfer into the RACF LPAR, and how *not* to corrupt data, programs, and files in your 'live' LPARs.

We'll then look at developing the wide variety of test scripts and test schedules you'll need to ensure RACF database integrity and functionality. You'll see how to rebuild your RACF database over and over again (perhaps many times for each 'live' LPAR) without the pain (or at least, without as much pain as you'd usually have).

Finally, and this is important, we'll discuss how to provide your technical and business management with progress reports that they can both understand and use.

RACF TEST LPAR

Remember the mini-LPAR that was mentioned in the previous article? Well, throw it away. You won't need it after you've confirmed the basic structure of your RACF groups, user IDs, dataset profiles, general resource profiles, etc. You will, however, need all of the JCL you created that built that basic structure. When you've finished your work on the mini-LPAR, download all of your JCL back to a PC. Also keep a back-up tape of that JCL handy. Transfer your PC downloads to a set of diskettes, write-protect them, and send them to at least two off-site storage facilities. If you've got a CD-R writer, you might want to transfer the JCL to one of those as well. Why? Because you just spent a couple of months developing that baseline JCL, and the last thing in the world you want to do is lose it! Yes, you'll be modifying it as you develop each new database for your 'live' LPARs, but you need to ensure you've got a safe copy.

For the purposes of example, let's assume that you've got a single mainframe divided into three LPARs – DEV1 for program development, TST1 for program testing, and PRD1 for production work. You'll now need a fourth LPAR created – RAC1 for RACF database development. You should sit down with your Technical Support Manager and your Auditors (*WARNING: doughnut alert!*) and map out how you want the

RAC1 LPAR to look. The answer to that is deceptively simple yet extremely difficult – you want RAC1 to be a mirror of DEV1, TST1, and PRD1.

Remember, when you get to the actual implementation (covered in part four, next issue) you'll develop each LPAR's RACF database one at a time. So, you'll copy DEV1's settings, programs, data, etc (except, of course, for the RACF database), into RAC1, develop the new database structure, test it, and then roll it into DEV1 itself for implementation. Once that's done, you'll nuke RAC1 and recreate it to look like TST1, do all of the steps again, and then rebuild RAC1 to look like PRD1. You'll need to do this over and over again, depending on how many LPARs your organization uses.

Naturally, if you're running multiple LPARs on a single machine, you're going to start getting some process degradation if your RAC1 LPAR is too large. It's a balancing act for your Technical Support and Resource Management teams, so don't get greedy. You want just enough to be able to test the veracity of the RACF database, not to make a carbon copy of the processing functionality of the original LPAR(s). Try not to go over 15% of total processor capacity – 10% or less is even better. Naturally, the more LPARs you have running on a single machine, the less of the overall processor pie you'll be able to take a slice from. Think of your mainframe as a rather large triple-fudge-brownie-nut cheesecake, and take only a very small slice.

You will, however, need large amounts of DASD if you plan to copy all of the files. A quick tip: if you're tight on DASD space, copy over all of the program and settings files, but have your DASD guru create a one-track set-up for the data files. You're not going to be doing any serious processing on them anyway, so have them copy only a *very* small selection of records for each data file. This should be sufficient to test the security without having to go out and buy bank after bank of new DASD just for a test. (If, however, your Technical Support and Operations Managers are looking to expand/upgrade on-line data storage, you might be able to dovetail your request in to help with the justification. Who knows? They might even buy *you* doughnuts!)

One more important point on the RAC1 LPAR: make sure your Technical Support teams *isolate* this LPAR from all the others. That means no shared DASD, no shared files, etc. You don't want any

chance of someone accidentally corrupting your live data on any other LPAR. The program and data files you copy (even the abbreviated ones) should be more than sufficient for your testing purposes.

SHOTGUN WEDDING

By this time, you should have a very good idea how each of your existing RACF databases are (un)structured. You also, at this point, have a shiny new RACF database structure. So how do you marry the two together?

Generally, you should start with the analyses you did of the current RACF databases (from the first article). Build the RACF database starting with groups, then user IDs, then datasets and general resource profiles. At each stage, run an analysis of the database using IRRUT400 and/or the Pentland Utilities (using both is better). When all of them are in place, begin the big testing.

In the last article, we outlined the development of the group structure, but didn't cover the other areas too much. That's because group structures are relatively easy, especially if you're rewriting from line 10. When you get to the incorporation of your user IDs, datasets, etc, you've got a bit more homework (and a lot more typing) to do. However, as long as you're careful and methodical, you can incorporate the old profiles into the new structure without too much difficulty.

Make yourself a few simple rules for when you're recoding:

1 DOCUMENT EVERYTHING! (even changes you make on the fly).

2 Each profile must have full description and ownership defined installation data field.

3 Each user connects only to a single (shared) group based on department and work duties/job title.

4 Each dataset connects only to a group owner.

   – (Exception: TSO High Level Qualifier IDs).

5 Connect only groups (not user IDs) to datasets and general resource profiles.

– (Exception: CICS started task IDs for CICS region profiles).

6 Simple doesn't always equal structured.

7 Structured equals simpler to work with.

When you've coded all of your profiles (tip: put the JCL in separate PDS libraries and members – it spreads the risk if a PDS is corrupted), you'll want to start incorporating them into your new LPAR. Remember, you'll have only one user ID available to you at the start (IBMUSER), so do your initial set-ups carefully. IBMUSER has a lot of power, and you want to use that power sparingly.

Appendix A provides the general test script for actually creating the new database on the RAC1 LPAR. You may want to add your own tests, but these are a good baseline. Note that you can download Pentland Utilities for free from http://www.NigelPentland.co.uk/RACF.html. *(Editor's note: upcoming issues of* RACF Update *will include a two-part review of the Pentland Utilities and their functions.)*

A friendly suggestion: convert Appendix A into a Microsoft Project 2000 document. Build timeframes, assign work, and schedule your tasks early on. Your IT staff will be much better off knowing things ahead of time. It'll make your work go more smoothly as well. And remember to try to overestimate your time spent on duties by at least 10%. For one thing, it'll make you look good when you get things done early. It also gives you time to sort out any problems which arise.

You'll note that there are a lot of test programs and files that you'll create, and loads of output as well. You'll also note that you're going to save all of that output, in paper and electronic format, at each stage of the process. Why? Rule one – DOCUMENT EVERYTHING! You need that documentary evidence to fall back on if (or, more likely, when) something goes awry. And yes, that includes the changes you make on a terminal (although you should strive to do most, if not all, of your changes via JCL). Your auditors need that information to ensure that you've followed proper change control steps. Your QC people will need that information if any of their testing discovers problems. Your Technical Support staff will absolutely need that information if system processes start to fail due to RACF errors.

The last step on the general test script is *vital*! Your organization's

technical staff and hex gurus may understand what you've been doing, but remember that you've got to sell this concept to management. Not just when you're developing the business case, but all through the system development life-cycle. Avoid much of the technical jargon, but do provide them with the number of new groups, user IDs, etc, created, how many staff hours were involved in the database recreation, any special problems that arose (especially those that can help you justify getting more security reporting and analysis tools), etc. Focus on the successes, though – managers love to hear when things go right. Also 'spread the wealth' a little bit. Remember, this is not a one-person operation. You've got to remember to single out those departments and/or individuals who provided significant support to the project. This is not only fair and honourable, it also pays dividends in the longer term.

Remember, however, that the general test script is only for the creation of the database itself. We haven't even touched on the tests you'll need to do once your RAC1 LPAR is fully up and running. That comes later in this article.

LATER IN THIS ARTICLE...

Wow, that was fast! Hope you had time to get a fresh cup of tea!

Okay, you've tested the database, ironed out all the kinks, combed out the knots, and now it's a fully functional RACF security system, right? WRONG!!! The database may look just peachy-keen, but if the other systems and applications can't work with it then you haven't completed your task. Now we get into the more nebulous world of unit, user, QC, and acceptance testing.

This is where you're going to have to get a number of people involved – from all areas of your organization. This may or may not be a doughnut meeting, depending on how many people/departments are involved and their general mood. If a little bribery is required, so be it. However, only invite those people who will actually be involved in the tests themselves (designing and executing them). It will cut down on confusion and interruption in the meeting, and your doughnut bill will be significantly lower.

One quick note here: there will be a debate on whether to do your

testing in WARN or FAIL mode. If at all possible, opt for FAIL. There are good arguments for trying things in WARN mode first, but I've generally found that it's much easier to miss potential problems if you take this route. FAIL mode may take longer, and be a bit more frustrating at times, but it's a much more reliable indicator of what you'll have to deal with in real-world situations (ie normal operations).

If you've got a QC and/or change management department in your organization, depend on them heavily for the development of the specific test scripts. They've done it before, and it's their area of expertise. You'll need to test the following items:

- Systems (technical support) access, tools, started tasks, etc.
- All TSO/ISPF/SDSF functions.
- Every CICS transaction for every CICS region.
- All batch processing.
- All print processing.
- All external linkages (via OMVS, FTP, etc).
- Anything else I missed.

In other words, test as if you've just installed a new version of OS/390 or z/OS.

Systems access, tools, started tasks, etc

Start with the fundamental systems that drive the rest of the operation – those of the operating system itself, and the support tools. You can determine this one quite easily. Just IPL the mainframe as you normally would, and watch for any failures. Your first few IPLs may fail completely. Don't be disheartened. You can fix these from the system console using the IBMUSER ID. Once again, note any and all changes, and update your JCL to reflect this.

Once you've made it through the IPL process, have your technical support and operations staff bring up each system tool one by one. Have them run various transactions, traces, reports, etc, and note any failures or error messages. Log these errors and correct them. And remember, when you do further tests, new errors may crop up in the

systems arena. Take these in your stride as part of the testing process.

Now move on to your 'production' started tasks. These can be CICS regions, etc. STC start-ups are generally notorious for problems in dataset access, even with the most prodigious planning.

All TSO/ISPF/SDSF functions

This functionality is a bit more difficult to catalogue, mainly because there are some obscure commands that can be done. Try to dig up all of them from the IBM documentation, and then have someone execute each one. If you find some 'dangerous' commands, this is a chance to either shut them out or severely limit access to them.

You'll need to check that every ISPF panel actually displays properly, without error or violation messages spoiling the fun. Once again, a tedious task, but you must ensure that everything works as it does in your 'live' environment. This has the added benefit of weeding out old or unused panels – your developers and technical support staff can give great input on this issue.

As for SDSF, do a cursory review first. You'll need to check further in depth later on when you've got output to review on-line.

Every CICS transaction for every CICS region

Yes, every single one! No cheating here. And you've got to have each one tested by users with varying degrees of authority. They should determine which ones fail because they're not allowed access, and which ones fail because the RACF authority is incorrect. Also have them test the 'dangerous' commands here as well, to ensure a) that they can't get to them and b) that the NOTIFY flag in RACF works properly.

All batch processing

Once you've got through all of that, have your production control and operations staff run a couple of daily cycles for batch. If possible, have them do dummy tests of month-end, quarter-end, and year-end processing as well. Also have them test the nightly back-up processing for off-site storage.

All print processing

A natural outgrowth of the item above. You must ensure that the print spooling mechanisms (JESSPOOL) are working correctly. This is also the time to test SDSF output access authorities.

All external linkages (via OMVS, FTP, etc)

Here's where it starts to get a bit dicey. If you do any of these types of transfer, you must make sure that the OMVS UID and GID segments are functioning properly. You also have to ensure that there are no IP address problems. However, you may encounter some difficulties if you're transferring to/from 'live' systems, in that there's a risk of corrupting your real data. Discuss with your network operations staff and your internal auditors how to avoid these problems.

Anything else I missed

Well, I can't think of everything now, can I? Each environment has its own foibles, special needs, specialist programs or functions, etc. You must tailor your testing program to cater for these situations. It's far better to test for it now than to clean up a (potentially expensive) mess later down the road in a production situation.

Incorporating the changes

Once again, make sure you document *everything* you do, either via JCL or from written notes when you make changes through the RACF ISPF panels. This is your own little insurance policy, to help you track your changes in case anything goes wrong. If you need to make major changes, do so through the JCL. It's much more controlled that way, and you can incorporate those changes into your overall JCL libraries.

When you've done major changes, it's a good idea to repeat steps 54 through 67 to check the overall effect. This also enables you to document what you've done more thoroughly and in a more readable format. One note, however: before you do major changes, do an IRRUT400 back-up of the database and create an IRRDBU00 flat file. That way, if you need to back out any of the changes (or start again from scratch) you can simply recover from your back-ups (or use the Pentland Utilities to generate the JCL you need to do minor back-outs).

SELLING YOUR PROGRESS TO MANAGEMENT

Now that the easy part is over (!), you need to report your results to management. After all, this is a large-scale project, and they were sold on this back in the first article. You've got to keep management apprised of the current situation, of significant progress or setbacks, etc, in order to keep project momentum from becoming project stagnation.

Many project managers have a simple and concise tool for reporting. It's called lying. Don't bother with that tool, though – it's always more trouble than it's worth.

For the more technically versed, you can always do bi-weekly technical progress reports. Keep them short, to the point, and summarize wherever possible. If you've got serious technical issues, e-mail the specifics to the concerned victims – er, parties – and keep a copy of their responses in your documentation file.

One note of warning: if you've got Microsoft Outlook, configure it so that you get a return receipt and a message that the recipient has either read or discarded your e-mail. You wouldn't believe the number of "I never got your e-mail" claims you can get in a project. This at least gives you documented proof that they opened it. It doesn't prove they actually read it, of course, but that's their problem not yours.

When you get to the stages of development where you're doing major testing, you'll need to report your progress to the members of the executive team (VPs, SVPs, EVPs, SEVPs, etc). Keep in mind that these people are busy, so just hit them with the basics in a two-page summary report, then attach any appendices with non-technical details. Also remember that these folks aren't as interested in the technical nuances of what you're trying to do. They're results-oriented. Make that your focus, and you'll have a lot better luck.

Appendix B gives a sample of a progress report. Here's a breakdown of what you should include:

- *Purpose of test.* A simple, one-sentence summary of what you wanted to do (your goal) and the effect of meeting that goal. You'll note that the sample has two statements. One is for when you do the initial creation of the new database for the DEV1, TST1, or PRD1 LPAR on RAC1. The other is for when you do the

full test of the new database with technical support, computer operations, production control, quality control, users, etc.

- *Outcome of test*. There are two options to choose from here: either your test was "successful" or it was a "qualified success". In management reporting situations like this, failure is not an option. If your test does bite the big one, questions will be asked, heads will roll, etc, which is a very good incentive to ensure that things work properly.

  On the off-chance that your testing does bomb completely, and you can't correct the problems, you'll need to go back to the analysis phase (see 'Part one – planning') and see what's gone wrong. You'll have to inform management of this as well, so it might be a good idea to update your CV or resumé before you send out the progress report.

- *Summary*. Just make it simple and brief. Tell management the duration of the test, what you did, what problems you encountered, and how you fixed them, etc. But do it in a single paragraph, no more than four or five sentences, and try to keep the sentences short and to the point.

  Also take the opportunity to thank the participants in the test. Make sure you include them on the overall distribution list (either cc: or bcc:) so they can get that warm fuzzy feeling inside.

- *Details*. Don't make your details too, well, detailed. Short and to the point paragraphs, linked to more detailed appendices if necessary, are all that management really requires. You should consider including a graphic of the new database structure, perhaps in Visio format (managers like graphics – no-one knows quite why). Also think about a brief summary of the number of individual tests performed by each department or area. If you count the testing of every single CICS transaction, dataset, user ID, etc, that can make quite a large number. Managers like large numbers, too, even when they're meaningless to the overall effort performed or results achieved. Also, if you have any major problems you encountered during the testing, list them out in a summary table, and note any items that may still be pending. One caution, though – if you're asking to put the new database into production, there must be *no pending problems*!

- *Participants*. List all of the test participants here – in alphabetical order, otherwise you'll get complaints. Include their job titles and extensions as well, so that if you have an executive manager who wants more details, or just wants to pat the team on the back, they'll have a much easier time doing so.
- *Respectfully submitted*. Executive managers, for some reason, like to think that their employees hold them in esteem. And, since they sign your paycheques, it's best to humour them on this.

IN OUR NEXT EXCITING EPISODE

Be afraid, be very afraid, when you ...

- Inform all users of the upcoming event!
- Change everybody's user IDs!!
- Transfer the new database to production!!!
- Forget about sleeping for the next two months!!!!

Okay, seriously.

Next time, we get to the ultimate goal of RACF restructuring – moving the database from the test LPARs to the 'live' environments. We'll discuss how you go about informing staff of the upcoming changes, including their new user IDs. We'll look at the actual changeover itself – when and how and why it should be done. We'll review the first 72 hours following the changeover, and the essential needs that must be addressed in that critical timeframe. You'll discover why you need to sleep over in the main computer room, and we'll briefly discuss the best options available in sleeping bags (teddy bears optional, of course). And we'll look at the final handover, when everything is running as well as can be expected for your security system in particular, and your operations in general.

APPENDIX A: GENERAL TEST SCRIPT

1 Start – blank database.

2 Run IRRUT400 with INDEX and MAP.

3 Run DSMON.

4 Review output.

5 Create group structure.

6 Run IRRUT400 with INDEX and MAP. Compare with IRRUT400 output from step 2.

7 Run DSMON. Compare with DSMON output from step 3, and with your graphic group tree structure (created in the Analysis phase).

8 Run IRRDBU00 and copy output to your PC.

9 Run the following Pentland Utilities against the IRRDBU00 output from step 8:
- RACF00 (flat file pre-processor)
- RACF01 (summary statistics)
- RACF03 (group tree)
- RACF11 – 1 per group (list group access X-ref)
- RACF87 (list groups with OMVS GID).

10 Compare output with expected results.

11 For any errors found, make corrections to main JCL.

12 Subset corrections into separate JCL member.

13 Run corrections.

14 Ensure corrections have been incorporated into database.

15 Print all output and place into binder.

16 Copy all electronic output, reports, etc, into ZIP archive and place with binder.

17 Create user ID structure.

18 Run IRRUT400 with INDEX and MAP. Compare with IRRUT400 output from step 2 and step 6.

19 Run DSMON. Compare with DSMON output from step 7, and note any IDs defined as started tasks.

20 Run IRRDBU00 and copy output to your PC.

21 Run the following Pentland Utilities against the IRRDBU00 output file from step 20:
- RACF00 (flat file pre-processor)
- RACF01 (summary statistics)
- RACF03 (group tree)
- RACF11 – one per group (list group access X-ref)
- RACF11 – one per user ID (list group access X-ref)
- RACF38 (list all user IDs with additional RACF authorities)
- RACF79 (list all user IDs with OWNER not equal to DFLTGRP)
- RACF87 (list groups with OMVS GID)
- RACF88 (list user IDs with OMVS UID).

22 Compare output with expected results.

23 For any errors found, make corrections to main JCL.

24 Subset corrections into separate JCL member.

25 Run corrections.

26 Ensure corrections have been incorporated into database.

27 Print all output and place into binder.

28 Copy all electronic output, reports, etc, into ZIP archive and place with binder.

29 Create dataset profile structure.

30 Run IRRUT400 with INDEX and MAP. Compare with IRRUT400 output from steps 2, 6, and 18.

31 Run DSMON. Compare with DSMON output from step 19, and note any dataset profile issues.

32 Run IRRDBU00 and copy output to your PC.

33 Run the following Pentland Utilities against the IRRDBU00 output file from step 32:
- RACF00 (flat file pre-processor)
- RACF01 (summary statistics)
- RACF03 (group tree)
- RACF07 SYS (list all dataset access for all profiles having SYS prefix)
- RACF11 – one per group (list group access X-ref)
- RACF11 – one per user ID (list group access X-ref)
- RACF38 (list all user IDs with additional RACF authorities)
- RACF42 (list discrete profiles with ALTER access)
- RACF56 (list all dataset profiles with UACC=NONE)
- RACF58 (list all dataset profiles with NOTIFY set to user ID or ALL)
- RACF68 (process DSMON output to check APF dataset profiles)
- RACF79 (list all user IDs with OWNER not equal to DFLTGRP)
- RACF80 (list discrete dataset profiles, with JCL to convert to generic)
- RACF87 (list groups with OMVS GID)
- RACF88 (list user IDs with OMVS UID)
- RACF90 (generate JCL to list and recreate all dataset profiles).

34 Compare output with expected results.

35 For any errors found, make corrections to main JCL.

36 Subset corrections into separate JCL member.

37 Run corrections.

38 Ensure corrections have been incorporated into database.

39 Print all output and place into binder.

40 Copy all electronic output, reports, etc, into ZIP archive and place with binder.

41 **Create all CICS region profiles (general resource).** Ensure all of your IBM-supplied CICS transaction profiles are identical for each region you create.

42 Run IRRUT400 with INDEX and MAP. Compare with IRRUT400 output from steps 2, 6, 18, and 30.

43 Run DSMON. Compare with DSMON output from step 31.

44 Run IRRDBU00 and copy output to your PC.

45 Run the following Pentland Utilities against the IRRDBU00 output file from step 44:
- RACF00 (flat file pre-processor)
- RACF01 (summary statistics)
- RACF03 (group tree)
- RACF07 SYS (list all dataset access for all profiles having SYS prefix)
- RACF11 – one per group (list group access X-ref)

- RACF11 – one per user ID (list group access X-ref)
- RACF16 (list CICSTRN profiles)
- RACF22 (list TCICSTRN transactions with descriptions)
- RACF33 (list CICS transaction profiles sorted by TRANCODE)
- RACF34 (list duplicate CICS transaction profiles)
- RACF38 (list all user IDs with additional RACF authorities)
- RACF42 (list discrete profiles with ALTER access)
- RACF56 (list all dataset profiles with UACC=NONE)
- RACF58 (list all dataset profiles with NOTIFY set to user ID or ALL)
- RACF68 (process DSMON output to check APF dataset profiles)
- RACF79 (list all user IDs with OWNER not equal to DFLTGRP)
- RACF80 (list discrete dataset profiles, with JCL to convert to generic)
- RACF87 (list groups with OMVS GID)
- RACF88 (list user IDs with OMVS UID)
- RACF90 (generate JCL to list and recreate all dataset profiles).

46 Compare output with expected results.

47 For any errors found, make corrections to main JCL.

48 Subset corrections into separate JCL member.

49 Run corrections.

50 Ensure corrections have been incorporated into database.

51 Print all output and place into binder.

52 Copy all electronic output, reports, etc, into ZIP archive and place with binder.

53 Create all remaining general resource profiles.

54 Run IRRUT400 with INDEX and MAP. Compare with IRRUT400 output from steps 2, 6, 18, 30, and 42.

55 Run DSMON. Compare with DSMON output from step 43.

56 Run IRRDBU00 and copy output to your PC.

57 Run the following Pentland Utilities against the IRRDBU00 output file from step 56:

- RACF00 (flat file pre-processor)
- RACF01 (summary statistics)
- RACF03 (group tree)
- RACF07 SYS (list all dataset access for all profiles having SYS prefix)
- RACF11 – one per group (list group access X-ref)
- RACF11 – one per user ID (list group access X-ref)
- RACF12 (list general resource access for all profiles)
- RACF16 (list CICSTRN profiles)
- RACF19 (list general resource class pairs for all profiles)
- RACF22 (list TCICSTRN transactions with descriptions)
- RACF32 (list general resource profiles with WARNING set)
- RACF33 (list CICS transaction profiles sorted by TRANCODE)
- RACF34 (list duplicate CICS transaction profiles)
- RACF38 (list all user IDs with additional RACF authorities)
- RACF42 (list discrete profiles with ALTER access)
- RACF56 (list all dataset profiles with UACC=NONE)
- RACF58 (list all dataset profiles with NOTIFY set to user ID or ALL)
- RACF58 (list all general resource profiles with NOTIFY set to user ID or

ALL)
- RACF59 (list all general resource profiles with non-default audit attributes)
- RACF65 (list general resource class – all classes)
- RACF68 (process DSMON output to check APF dataset profiles)
- RACF72 (summary of general resource profiles)
- RACF79 (list all user IDs with OWNER not equal to DFLTGRP)
- RACF80 (list discrete dataset profiles, with JCL to convert to generic)
- RACF87 (list groups with OMVS GID)
- RACF88 (list user IDs with OMVS UID)
- RACF90 (generate JCL to list and recreate all dataset profiles)

58 Compare output with expected results.

59 For any errors found, make corrections to main JCL.

60 Subset corrections into separate JCL member.

61 Run corrections.

62 Ensure corrections have been incorporated into database.

63 Print all output and place into binder.

64 Copy all electronic output, reports, etc, into ZIP archive and place with binder.

65 Make IRRDBU00 flat file and copy to your PC.

66 Create full back-up database (IRRUT400).

67 Copy all JCL files, plus files created in steps 65 and 66, to back-up tape.

68 Write summary report to Management. Outline work done, elapsed time required, and brief statistical analysis of new database structure.

APPENDIX B: PROGRESS REPORT TO MANAGEMENT

Memorandum

To: Vice President, Information Technology Division
Vice President, Internal Audit
Vice President, Business Division A, B, C, etc
Vice President, Risk Management
Vice President, Whoever else is left

From: Doc Farmer
Information Security

Date: Monday, 34th Kunagonda, 2002

Subject: RACF Database Restructuring Project – Progress Report

Purpose of test

To recreate the security structure of the development/test/production region in a separate and segregated environment, for purposes of future testing and eventual migration.

OR

To permit technical, operational, quality control, and user testing of the newly-created security structure for the development/test/ production region in a separate and segregated environment prior to migration.

Outcome of test

Successful OR Qualified success

Summary

Testing was performed from 43rd Maktak through 33rd Kunagonda 2002. All participants (see list below) were most helpful in the execution of this test, and all scheduled tests were completed. Testing found some errors in the original RACF database structure, which were documented and corrected. All new/existing naming conventions and ownership identification policies have been followed on all new security profiles within the database.

The new database is now ready for testing by technical support, computer operations, production control, quality control, and user acceptance testing groups.

OR

The new database has been sufficiently tested, and is ready for a scheduled transfer from the segregated environment to the development/test/production region. With management approval, this transfer will occur on Saturday, 14 Wombat 2002.

Information Security would like to thank all participants in this test for their tireless efforts and invaluable service to this project.

Details

Blah blah blah. See Appendix A for a summary of the new security database structure.

Details (continued)

Blah blah blah. See Appendix B for a summary of the individual testing performed by each department.

Blah blah blah. See Appendix C for a detailed list of problems encountered, resolved, or pending.

Participants

List all test participants here (note – only those who actually participated. If any who were scheduled to show up did not, note that under a separate heading – either 'Apologies' or 'Couldn't be bothered'). If possible, sort their names alphabetically, and include their job title, telephone extension, and how many doughnuts they actually ate during the test.

Respectfully submitted,
Doc Farmer, IS Security

Doc Farmer would welcome comments and suggestions on this article. He can be contacted at Doc.Farmer@sbm.net.sa.

Doc Farmer
Manager and Senior IS Security Analyst (Middle East)

ICHPWX01 – an experience

In 1997, our auditors wanted to introduce strict password rules to improve the quality of our company passwords. This demand was a result of a discussion about 'good' and 'bad' – or 'strong' and 'weak' – passwords. They believed that many people use simple passwords that were easy to remember. For example, some use a password stem with a numeric suffix like JOKE01, JOKE02, others use month and/or year – MAR02, APR02 – and yet others are reluctant to change the password they've been using for a long time.

It goes without saying that once it's been noticed while being entered, or by being inadvertently typed into a displayable field, it will take only a few attempts to guess this kind of weak password even months or years later. And forcing users to change their passwords periodically doesn't resolve the problem.

Fortunately, RACF provides a password exit which enables administrators or system programmers to enforce better password behaviour. But does it really work?

It seemed clear to us that if we made our password rules too complex we would end up with complex passwords which were difficult both to create and to remember, and which would be jotted down on pieces of paper and stored under the keyboard. We therefore agreed on the following password rules:

- Passwords should have a minimum length of four characters.
- The first character should be alphabetic.
- There should be no names of the month or their abbreviations, either in our national language or in English.
- They should not use either the current year or the century (eg 02 or 20).
- They should be different from the userid.
- There should be no sequences of four or more characters, either ascending or descending (eg ABCD.., ZYXW.., 0123.., 9876..).
- Not more than three characters of the old password should be in the same position.
- They must not contain company-specific terms and abbreviations.

Before we implemented the exit, the users were informed and the user Help Desk team was given the appropriate training. We left some traces in the exit to help our user Help Desk team identify the cause of an 'invalid new password' failure and give us information about the efficiency of the exit.

Shortly after the implementation we analysed the recent password changes for a specific period of time to review the result.

Out of around 100 users who had to change their passwords:

- 45% succeeded at the first attempt.
- 55% had to try more than once. Among these:

- 1% had a new password containing the year or century.
- 2% had a new password containing a sequence of characters.
- 7% had a new password containing a month name or abbreviation. Note however that this group included some passwords that contained unintended abbreviations of month names. We estimate that the real percentage was around 5%.
- 31% had a new password containing more than three characters of the old password.
- 14% had misspelled the new password, used passwords already in the password history, or failed for other reasons such as missing new password or too few characters in the new password.

At first glance, it seemed that, disregarding the 14% group, around 41% of all users had weak passwords.

Most worrying was the 31% whose new password contained more than three characters of the old password. Although this could have happened by chance, the fact that almost one third of users fell into this group seemed to indicate that many people had a password stem and a consecutively numbered suffix. This would mean that, before the exit was implemented, more than one third of all users had passwords which were potentially easy to guess.

A second analysis of the behaviour of the 55% of users who changed their password more than once showed that 26% of all users circumvented the password rules by:

- Changing password twice within less than two minutes (21%). For example, from ANNA05 to TEMP05 to ANNA06.
- Changing password five times, thus clearing the password history (2%).
- Letting the administrator reset their password (4% – our RACF in 1997 did not record password resets by administrators in the password history.)

CONCLUSION

The good news is that, based on the information above, almost half of all users already had relatively strong passwords. For them there was no need to implement a password exit. In addition, about one quarter of all users got used to using better passwords.

The bad news is that about one quarter of all users were intentionally circumventing the password rules and weakening the security of our system. A new review a few months ago showed that this has not changed significantly.

Although there are of course ways to limit password changes to once a day, people who really want to keep a permanent password or a password with a stem and a numeric suffix will manage to get back to it anyway, in just a few days.

In conclusion, implementing a password exit is a good step towards better system security. But it cannot compensate for carelessness or deviousness on the part of the users.

Karl Reinhard Blatt
Systems Programmer (Germany)

RACF in focus – OPERATIONS attribute and universal access

'RACF in focus' is a new column that will appear regularly in RACF Update. *It will focus on specific aspects of RACF and show how to best exploit its many features and functions. It will also highlight trends and directions in the 'real' RACF world. In this first article, we focus on two topics: the OPERATIONS attribute and universal access.*

FOCUS ON RACF OPERATIONS ATTRIBUTE: ITS USE AND ABUSE

The RACF OPERATIONS attribute is one of the most powerful privileges available to a user in RACF. It provides wide-ranging

powers, not only to all datasets in your installation, but also to entire DASD volumes and tape volumes, and more. Needless to say, it should be used with care.

This attribute was once needed to perform many storage management 'house-keeping' functions, such as doing dataset back-ups and migrations, deleting all datasets on DASD volumes, moving datasets among DASD volumes, and defragmenting DASD volumes. A person or task doing these functions (in many installations, the HSM started procedure) required this attribute to be able to do the job. However, the attribute gives its owners powers that are far more wide-ranging than those required to get the job done, and using the OPERATIONS attribute to perform storage management functions is like using a sledge-hammer to kill a fly!

In the past, IBM has provided several alternative means to accomplish storage management functions, and the need for use of the OPERATIONS attribute is diminishing. These newer methods allow the installation to get the job done without exposing it to unnecessary security risks.

IBM has (over the last few years) introduced profiles in the FACILITY class that allow security administrators to grant storage management functions to users. These profiles start with STGADMIN. And there are several of them. They are described in the *DFSMSdss Storage Administration Reference* manual and other similar DFSMS manuals. Use these profiles instead of giving out the OPERATIONS attribute.

You should also look at the DASDVOL class. You can build profiles here to allow users volume-level functions. Another way to allow HSM to perform storage management house-keeping functions is to make it a TRUSTED started procedure.

So how do we know who is using (or misusing) the OPERATIONS attribute? To find out, I recommend that the OPERAUDIT global RACF option be activated, if it isn't already. This can be done by issuing the following RACF command:

```
SETROPTS OPERAUDIT
```

This will log the activities of all users and started procedures that acquire access to RACF-protected resources by virtue of having the

OPERATIONS attribute. Not only will this keep the auditors happy, but it will also enable you to run reports to see who is getting access because of their OPERATIONS privilege.

In the past, these reports were enormous and difficult to read, because of thousands of extraneous entries generated by HSM's house-keeping activities. But if you use the alternative method mentioned above, HSM's activity will no longer show up, and the reports will show the 'real' users of the OPERATIONS attribute.

Whenever some activity shows up in these reports, ask yourself: "Can this be accomplished using one of the alternative methods?" If so, implement it, and remove OPERATIONS from that user. In the end, the reports should show very little activity.

The reason why the OPERATIONS attribute is still available, even though it's so powerful and isn't really required any more, is that IBM can't pull back a feature it has once introduced, because someone out there could still be using it. Besides, the OPERATIONS privilege can still come in handy – in disaster recovery scenarios and emergency situations requiring system programmer or production support involvement. But it should be assigned only for the duration of the emergency and then removed.

There should be regular monitoring of users with this privilege. This can be accomplished using the DSMON report (which will be covered in a future article.)

FOCUS ON UNIVERSAL ACCESS: WHAT SHOULD IT BE?

To play it safe, the Universal Access (UACC) of most, if not all, of your profiles should be NONE! This applies to dataset profiles as well as resource profiles. Having UACC anything other than NONE may introduce unintended security exposures. This is because of the implications of UACC on jobs running under 'undefined', or 'default' userids. If a job enters your system (usually via a remote node, using RJE, but it can be internally too), and it has no userid to work with, the job will acquire a 'default' userid, as specified in your global RACF options.

Enter the following command:

```
SETROPTS LIST
```

and see what you have specified for JES (UNDEFINEDUSER (DFTUSR)).

In this case, the ‘undefined’ userid for your shop would be DFTUSR. Ideally, this undefined userid will have been granted minimal access, so these types of ‘undefined’ job will not be able to do much on your system. They will, however, have access to all datasets and resources as per the UACC in their profiles. And therein lies your problem – the universal access applies to undefined userids too! This means that if a profile has UACC(READ), these jobs will get READ access to that profile. A profile with UACC(UPDATE) will allow update access to these jobs. And so on.

You may want to give UACC(READ) to all ‘defined’, or identified, RACF users only. In that case, grant UID(*) ACCESS(READ) in the profile, and change the UACC to UACC(NONE).

The example below illustrates this.

Note that the following partial listing of a dataset profile has a potential security exposure:

```
INFORMATION FOR DATASET PROJECT1.PROD.**   (G)

LEVEL   OWNER   UNIVERSAL ACCESS   WARNING   ERASE
ØØ      XYZ     READ              NO        NO


ID              ACCESS
USERØ1          UPDATE
GRP22           UPDATE
SYSPRG          ALTER
```

Changing the profile as follows removes the potential security exposure:

```
INFORMATION FOR DATASET PROJECT1.PROD.**   (G)

LEVEL   OWNER   UNIVERSAL ACCESS   WARNING   ERASE
ØØ      XYZ     NONE              NO        NO
```

```
ID              ACCESS
USERØ1          UPDATE
GRP22           UPDATE
SYSPRG          ALTER
*               READ
```

So, what needs to be done? You may need to review all your profiles – dataset or otherwise – and find all occurrences of UACC that are anything other than NONE. Then, you need to decide which profiles should be changed to UACC(NONE). If you decide to change a profile from UACC(READ) to UACC(NONE), make sure you issue the following command at the same time, so the intent is not lost.

```
PERMIT "profile-name" UID(*) ACCESS(READ)
```

A word of caution: depending on your use of undefined users (in RACF terms) you may need to keep UACC other than NONE for some profiles. So running a batch job to change all UACCs to NONE is not the answer.

Dinesh Dattani would welcome feedback, comments, and queries about this column. He can be contacted at dddattani@rogers.com.

Dinesh Dattani
Independent Security Consultant, Toronto (Canada)

Protecting the RACF database

This article begins by examining ways in which to protect the RACF database. It then uses a REXX EXEC to demonstrate how easy it is to extract live data from the RACF database.

THREATS

It goes without saying that the RACF database is an extremely important asset, and that access to it must be properly controlled. An attacker with READ access can obtain important information. For example:

- He or she can view how sensitive resources are protected.
- He can find out all the userids, try to connect with one of them, or build a denial of service attack against users.
- If he knows default groups, he can try to use the default group as a password (this may work for some userids that have never been used, when no password has been set for them).
- Worst of all, he can obtain encrypted passwords and try to crack them.

The database may also contain other sensitive information, such as secure sign-on keys for Passtickets, session keys for APPC, certificates, etc.

The EXEC presented below shows how easy it is to get information from a live RACF database or from an 'offline' copy of it.

HOW TO PROTECT THE RACF DATABASE

Your protection of the RACF database should start by applying the need-to-know rule: with the exception of the following, no one should normally have access to the RACF database:

- The task or job that backs up the RACF database (via IRRUT200).
- The RACF administrator or auditor who needs to download

information from it (via IRRDBU00, or other utilities).

- The systems programmer (or anyone with the OPERATIONS attribute, such as a storage administrator), who may need to be able to restore a database on the fly, move it to another volume, etc.

In my opinion (though not everyone – especially auditors – will agree), it shouldn't be a problem if the systems programming staff have access (if you don't trust them, you should fire them, since they can bypass security rules through APF programs at any time). The same applies to people with the OPERATIONS attribute. However, it's a good idea to audit all accesses. Ideally, the profile should be defined as follows :

```
addsd 'SYS1.RACFDS' generic  audit(all(read)) uacc(none) erase
```

One site that I know made use of RACF's B1 facilities to better protect the access:

```
rdefine SECDATA SECLEVEL ADDMEM(external/10 internal/100)

altdsd 'SYS1.RACFDS' GENERIC SECLEVEL(internal)

altuser RACFMGR SECLEVEL(internal)
```

where RACFMGR stands for the userid given to any person entitled to manage the RACF database. This can keep some powerful userids (operations) at a distance, although it's no help against special users.

Access to the database is not normally an everyday requirement. However, some RACF add-on products (such as Consul/RACF) offer administration of RACF based on the live data, and do require READ access. Fortunately, these products generally propose a more restricted access mode, via PADS (program access to data set). This means that the product 'filters' the access to RACF profiles according to some rules that it manages on its own (which can be specific RACF profiles in the FACILITY class or a dedicated class).

WAYS TO COMPROMISE YOUR RACF DATABASE

Let's be clear from the start: what follows is not theoretical. I've often noticed the following flaws even in companies that claim to be "very

secure". Note also that the following applies not only to the RACF database, but also to any confidential data that you may have in your organization.

If we ignore trivial flaws such as unprotected datasets (there are too many sites where the RACF database is called SYS1.RACF and is protected only by a SYS1.* profile with UACC READ), or userids with trivial passwords, the most frequent flaws that give access to the database are these:

- *Shared disks*. Although the RACF database is not shared between systems, it is located on a shared disk. So, for all the other systems, it is just another dataset which can be accessed by anybody if it isn't properly protected on this system.
- *The database is protected, but the back-up database is not*. There may be a SYS1.RACF profile with UACC NONE, but the generic profile for the back-up database is still SYS1.* with UACC READ.
- *The database is protected, but tape back-ups are not*. For example, the bypass label facility may be used by everybody (through too permissive JES2 classes), or Ditto can be used to bypass labels, etc. It's easy to restore the database under another name and to access its contents.
- *Broader flaws surreptitiously jeopardize data security*. For example, ADRDSSU is declared as 'bypass RACF' in the program properties table, allowing anyone to use ADRDSSU to read or update datasets. The same can be achieved with STGADMIN profiles in the FACILITY class with an incorrect UACC.
- *Several tools may offer undue access to confidential data*. If you're being paranoid, you should immediately suspect any program with AC=1: its mode of operation should be analysed, and the program should receive adequate protection. But sometimes you don't need a very sophisticated tool to compromise security, you can simply combine two existing flaws. For example:
  - *Flaw 1*. JES2 classes tolerate MVS commands in the JCL stream (// COMMAND 'mvs-command'), and MVS

commands are not controlled by RACF.

– *Flaw 2*. Some members of PROCLIBs, which are associated by the STARTED class or the ICHRIN03 table to an OPERATIONS userid (or to a powerful userid), can be used to submit jobs, and read or copy datasets. How wonderful to be allowed to issue the following command:

```
S COPY,IN='infile-dataset',OUT='outfile-dataset'
```

I'll leave you to guess what this apparently 'innocuous' COPY procedure can do when it runs under the authority of a powerful userid.

A QUICK AND DIRTY WAY TO READ THE RACF DATABASE

The REXX EXEC presented here shows just how easy it is to obtain information from a RACF database, considered as a 'normal' sequential dataset. I used public information contained in the *OS/390 Security Server (RACF) Diagnosis Guide*.

With this EXEC, you can access any profile and display the fields of which it consists. The templates are used to give a description of each field.

Note the following shortcomings that affect this EXEC:

- Only base segments are processed.
- Some 'phantom' entries may be retrieved.
- Repeat fields are displayed in their raw form.
- Profiles with generics are not supported (this is because RACF stores them with an internal format that differs from the external format).

'Phantom' entries are profiles that still exist in the database although they have been deleted (the space has not been reused yet, similar to a deleted member in a partitioned dataset). Since RACF indexes aren't used by this EXEC, such issues are unavoidable. Optimistic people would say that it could be used to retrieve mistakenly deleted information....

I hope this article will encourage you to reconsider the way in which confidential information (and not just the RACF database) is protected in your organization.

REXX EXEC

```
/* REXX for quick and dirty processing of the RACF database       */

ARG dsn prof_type
      /* 1st optional parameter = dataset name of RACF database    */
                  /*  (blank or '*' for current primary database) */
      /* 2nd optional parameter = profile type                     */
            /* (USER, GROUP, DATASET or GENERAL) default is "USER" */

/* SEE SYS1.MODGEN(IRRTEMP1) TO GET A DESCRIPTION OF ALL FIELDS    */

/*-----------------------------------------------------------------*/
/* If no dsname passed, take name of the active primary database   */
/*-----------------------------------------------------------------*/

IF dsn = '' | dsn = '*' THEN DO
            CVT  = STORAGE(10,4)                  /* CVT address  */
            RCVT = ,                        /* CVT+3E0= RCVT address */
            STORAGE(D2X(992+C2D(CVT)),4)  /* load address of RCVT  */
            dsn = ,                         /* RCVT+38=DSN of RACF DB*/
            STRIP(STORAGE(D2X(56+C2D(RCVT)),44)) /* DSN of RACF DB */
            END

SAY 'RACF DATABASE IS' dsn

/*-----------------------------------------------------------------*/
/* Prompt the user (if in foreground mode) to get profile type     */
/*-----------------------------------------------------------------*/

type = 'BASE'              /* scan criterium, denotes base segment */
proftype.1 = 'GROUP'
proftype.2 = 'USER'
proftype.3 = 'CONNECT'     /* obsolete */
proftype.4 = 'DATASET'
proftype.5 = 'GENERAL'     /* all kinds of general resources */
proftypes  = proftype.1 proftype.2 proftype.3 proftype.4 proftype.5

DO WHILE prof_type = '' | pos(prof_type,proftypes) = 0
say 'Enter profile type (' proftypes ' - default = USER)'
IF SYSVAR('SYSENV') = 'FORE'  THEN pull prof_type
IF prof_type = '' THEN prof_type = 'USER'         /* default value */
END
```

```
/*-------------------------------------------------------------------*/
/* Template processing, get all fields for the type of profile     */
/* we are interested in                                              */
/*-------------------------------------------------------------------*/

dsn_template = 'SYS1.MODGEN(IRRTEMP1)'
"FREE F(TEMPLATE)"
"ALLOC F(TEMPLATE) DA('" || dsn_template || "') SHR"
if rc > Ø then do
               say 'Cannot allocate' dsn_template
               exit(16)
               end
"execio * DISKR TEMPLATE (stem templ. finis"     /* read templates */
flag = 'init'
k = 1     /* index for fields related to this type of segment */
say 'Getting templates for profile type' prof_type
do i = 1 to templ.Ø
   if flag = 'ok' &   substr(templ.i,1Ø,3) ='ØØ1' then flag = 'done'
   if flag = 'init' & substr(templ.i,1Ø,3) ='ØØ1' & ,
               strip(substr(templ.i,1,8)) = prof_type ,
               then flag = 'ok'
   if flag = 'ok' & substr(templ.i,1Ø,3) <> 'ØØ1' & ,
               substr(templ.i,1Ø,3) <> 'ØØØ' & ,
               datatype(substr(templ.i,1Ø,3)) = 'NUM' then do
                 k =         substr(templ.i,1Ø,3) + Ø
                 fldname.k = substr(templ.i,1,8)      /* field name */
                 flddesc.k = substr(templ.i,32,4Ø)   /* description */
          /*   say k fldname.k flddesc.k         */
               end
   end i
"FREE F(TEMPLATE)"

/*-------------------------------------------------------------------*/
/* Prompt the user (if in foreground mode) to get profile name     */
/*-------------------------------------------------------------------*/

SAY
SAY 'Enter name of' prof_type 'profile or * to list all entries'
if prof_type = 'GENERAL' then do
 SAY 'Name of general resource must be specified as "class-profile",'
 SAY 'for example : '
 SAY ' FACILITY-BPX.DEFAULT.USER    or '
 SAY ' TSOPROC -ISPFPROC -the class name must be exactly 8-byte long'
end
profin = '*'
if SYSVAR('SYSENV') = 'FORE' then pull profin
if prof_type = 'DATASET' then do    /* special change for DATASET */
   firstdot = pos('.',profin)      /* locate first dot */
   if firstdot > Ø then ,
      profin = left(profin,firstdot-1) || 'Ø1'x || ,
      substr(profin,firstdot+1)
```

```
   end
if pos('*',profin) > 0 & profin <> '*' then do
   say 'Sorry, generics are not supported'
   exit(8)
   end

ADDRESS  TSO  "FREE   F(DD1)"
ADDRESS  TSO  "ALLOC  F(DD1) DA('"DSN"') SHR" /* Allocate RACF DB  */

/*-------------------------------------------------------------------*/
/* Read all blocks of the RACF DB, keep only base segments           */
/*-------------------------------------------------------------------*/

block_nb = 0
data_block_nb = 0
nb_profs = 0

                     /*----------------------------------------*/
                     /* 1st LEVEL LOOP, PROCESS A BLOCK        */
                     /*----------------------------------------*/
DO FOREVER
call read_a_block          /* read a block from the RACF database  */

/* Must be a block containing data, not an index block             */
IF LEFT(BLOCK,1) <> '83'X THEN ITERATE         /* not a data block */
data_block_nb = data_block_nb+1 /* increment number of data blocks */

                       /*-------------------------------------*/
                       /* 2nd LEVEL LOOP, PROCESS A SEGMENT   */
                       /*-------------------------------------*/
   DO WHILE length(block) > 0
   bloctp= left(block,1)                /* X'83' Record identifier */
   IF bloctp  <> '83'X THEN LEAVE              /* not a data block */
   segm_pl= C2D(SUBSTR(block,2,4))  /* physical length of record   */
   segm_l = C2D(SUBSTR(block,6,4))  /* logical length of record    */
   segm_tp=     SUBSTR(block,10,8)  /* segment type (BASE, TSO...) */

      /* if physical record spans several blocks, read some more   */
      do while length(block) < segm_pl          /* not enough data */
         save_block = block
         call read_a_block                   /* read another block */
         block = save_block || block      /* concatenate the block */
         if segm_pl = 65537 then ,
         say 'length now' length(block) '->' segm_pl segm_l segm_tp ,
         'block nb=' block_nb SUBSTR(block,18,30)
         end

   segm   = left(block,segm_l)           /* take segment           */
   block = substr(block,1+segm_pl)  /* shift block for next segment*/
   if segm_pl <= 0 then LEAVE                        /* no segment */
   if strip(segm_tp) <> type then iterate     /* test segment type */
```

```
   prof_l = C2D(SUBSTR(segm,18,2))             /* length of profile*/
   prof = SUBSTR(segm,21,prof_l)                /* profile itself */

   nb_profs = nb_profs + 1
   array. = ''                /* reinit array of fields in segment */
   IF translate(prof) <> profin & profin <> '*' ,
      THEN iterate           /* no match (comparison in uppercase) */

                                          /* match, give details */
   say '------------ profile :' prof
   say '------------ physical length=' segm_pl ', logical=' segm_l ,
       ', block number' block_nb

   segm   = substr(segm,21+prof_l)  /* take segment           */

                               /*---------------------------------*/
                               /* 3rd LEVEL LOOP, PROCESS A FIELD */
                               /*---------------------------------*/
                  /* ALL THE FIELDS OF THE BASE SEGMENT ARE PARSED */

   DO WHILE length(segm) > Ø
      field_id = C2D(SUBSTR(segm,1,1))          /* get id of field */
      field_ln = C2D(SUBSTR(segm,2,1))      /* get length of field */
      field_r  = ' '                        /* repeat field or not */
      IF field_ln > 127 THEN DO                /* if repeat group  */
         field_ln = C2D(SUBSTR(segm,3,3))+3 /* length is different */
         field_r  = '  ** repeat field **'  /* repeat field or not */
         END
      array.field_id = SUBSTR(segm,3,field_ln)  /* get field value */
                                           /* match, display fields*/
      say fldname.field_id  '(' flddesc.field_id ')'
      say '   field-id=' field_id '  field-length=' field_ln ,
              field_r
      say '   value' array.field_id
      say '   (hex)' c2x(array.field_id)

      IF fldname.field_id = 'ENTYPE' then do /*show prof type*/
         k = c2d(array.field_id)
         say '          *** Profile type is' proftype.k '***'
         IF prof_type <> proftype.k THEN LEAVE
         END

      segm = substr(segm,3+field_ln)          /* truncate segment */
      say '         ' length(segm) 'bytes remaining in segment'

      END                                /* next field in segment  */

   END                                   /* next segment in block  */

END                                      /* next block in database */
```

```
  EXIT

  /*-------------------------------------------------------------------*/
  /*  Procedure to read a block from the RACF database                 */
  /*-------------------------------------------------------------------*/

read_a_block:

  'EXECIO 1 DISKR DD1   '                    /* reading the RACF DB */
  IF RC > Ø THEN DO                            /* end of file, stop */
                say 'End of file on' dsn block_nb 'block were read'
                say data_block_nb 'data block were read'
                say nb_profs 'base segments processed'
                EXIT
                END
  parse pull block              /* get current DB block into memory */
  block_nb = block_nb + 1
  if 1ØØØ*(block_nb%1ØØØ) = block_nb then say block_nb 'blocks read'
  RETURN
```

EXAMPLE (SOME FIELDS OMITTED)

```
RACF DATABASE IS SYS1.RACFDS
Enter profile type (GROUP USER CONNECT DATASET GENERAL - default = USER)
 Getting templates for profile type USER

 Enter name of USER profile or * to list all entries
IBMUSER

 ------------ profile : IBMUSER
 ------------ physical length= 768 , logical= 352 , block number 67
 ENTYPE   ( ENTRY TYPE                                  )
    field-id= 2   field-length= 1
    value :
    (hex) Ø2
           *** Profile type is USER ***
           322 bytes remaining in segment
 VERSION  ( TEMPLATE VERSION NUMBER                     )
    field-id= 3   field-length= 1
    value :
    (hex) Ø1
           319 bytes remaining in segment
 AUTHDATE ( USER CREATION DATE                    ‡ØDC )
    field-id= 4   field-length= 3
    value n:"
    (hex) 95157F
           314 bytes remaining in segment
 AUTHOR   ( OWNER OF USER ENTRY                         )
    field-id= 5   field-length= 8
```

```
   value IBMUSER
   (hex) C9C2D4E4E2C5D940
          304 bytes remaining in segment
FLAG2    ( SPECIAL = BIT0                      ‡L8C )
   field-id= 7   field-length= 1
   value :
   (hex) 80
          301 bytes remaining in segment
FLAG3    ( OPERATIONS = BIT0                   ‡L8C )
   field-id= 8   field-length= 1
   value :
   (hex) 80
          298 bytes remaining in segment
DFLTGRP  ( DEFAULT GROUP                       ‡03C )
   field-id= 15   field-length= 8
   value SYS1
   (hex) E2E8E2F140404040
          267 bytes remaining in segment
LJTIME   ( TIME OF RACINIT                          )
   field-id= 16   field-length= 4
   value ::::
   (hex) 11471057
          261 bytes remaining in segment
LJDATE   ( DATE OF RACINIT                     ‡0DC )
   field-id= 17   field-length= 3
   value :::
   (hex) 01233F
          256 bytes remaining in segment

1000 blocks read
2000 blocks read
3000 blocks read
End of file on SYS1.RACFDS 3600 block were read
89 data block were read
1026 base segments processed
```

Thierry Falissard (France)

E-mail alerts

Our e-mail alert service will notify you when new issues of *RACF Update* have been placed on our Web site. If you'd like to sign up, go to http://www.xephon.com/racf and click the 'Receive an e-mail alert' link.

RACF – your questions answered

This is the first article in what will be come a regular feature in RACF Update. *'RACF – your questions answered' provides a forum for you to ask questions, gain insights, or just wonder out loud about some of the foibles and idiosyncrasies of IBM's Resource Allocation Control Facility subsystem. Since this is the first article in this continuing series, we don't have any letters from you yet, so we're going to start off by covering some general subjects. But you're strongly encouraged to contribute questions, comments, or suggestions for future articles – please send them to the Editor, Fiona Hewitt, at any of the addresses on page 2, or contact me directly at Doc.Farmer@sbm.net.sa.*

REPORTING ON SENSITIVE CICS TRANSACTIONS

Q: What's the best way to monitor and report on sensitive CICS transactions?

A: There are certain CICS transactions that should have very limited access and usage. A simple RACFRW report can be drafted and run on a nightly basis to let you (and the owner of those transactions) know who has used such sensitive transactions as CEMT or CECI. It will also let you know who tried to use them and failed, enabling the owner of the transactions to follow up on the attempt. (Generally, the Manager of your technical support team should be the person who decides who is authorized for access to IBM-supplied CICS transactions.)

For successful transaction usage, try the following RACFRW code:

```
RACFRW
 SELECT PROCESS SUCCESSES
  EVENT ACCESS CLASS(TCICSTRN) NAME(REGION1.CEMT REGION2.CEMT
REGION3.CEMT)
   LIST SORT(NAME DATE TIME USER) TITLE('*** CEMT SUCCESSFUL TRANSACTION
USAGE - SEND TO TECHNICAL SUPPORT ***')
END
```

And, of course, you just need to change one word to get a listing of all failed attempts:

```
RACFRW
 SELECT PROCESS FAILURES
  EVENT ACCESS CLASS(TCICSTRN) NAME(REGION1.CEMT REGION2.CEMT
REGION3.CEMT)
   LIST SORT(NAME DATE TIME USER) TITLE('*** CEMT FAILED TRANSACTION
ATTEMPTS - SEND TO TECHNICAL SUPPORT ***')
END
```

You should probably run separate reports for each of the following IBM-supplied CICS transactions: CEMT, CEDA, CEDB, and CECI. There are probably others you'll want to monitor too, but this should be your minimum coverage. Oh, and be sure to keep these reports archived in hard or soft copy – you'll make your auditors very, very happy....

HASP AND NAMING CONVENTIONS

Q: Why are naming conventions so restrictive?

A: The Houston Automated Spooling Program (an early batch manager which preceded JES) had some quite restrictive naming conventions, when looked at from today's perspective. Eight-character naming limits (seven for TSO IDs), no numeric first character for dataset HLQs (or TSO IDs again), and so on. It was developed for the (very) old OS/360 MVT (Multi-programming with Variable number of Tasks).

So why are we still stuck with those naming conventions from the old 8-bit mainframe days now that we're into 31-bit virtual address spaces? The days of punch cards and ferrite cores are gone. Maybe it's time to do a REAL update to the operating system – one that's more in line with the Third Millennium. Just a thought, IBM.

NON-GENERIC DATASET PROFILES

Q: What are the implications of coding a non-generic dataset profile, and should I be using them?

A: No, this isn't a nice option at all, is it? Code one of these puppies and you're just asking for a nice gaping hole in the security profile for that dataset – anyone with ALTER access not only gets to create or delete the dataset, they also have control over the RACF authorizations

for that dataset. Do you really want to give that kind of power to users or technical staff? It's much safer to code all your dataset profiles as generic. Including the explicit (no asterisk) profiles.

A QUESTION FOR YOU ...

Q: How many of you actually use the ERASE feature of RACF – the one that physically erases your scratched datasets?

A: I don't know actually. But I'd like to hear from you and find out a couple of things: first, how much system overhead and/or clock time does it take to perform the erase; and second, is it a simple overwrite of binary zeros or is an option available to allow a DOD seven-pass erasure? If it's the former, then there's still a security exposure, as there are software packages that can 'interrogate' the surface properties of the disk to retrieve supposedly erased data. If your data is so sensitive that you need complete erasure, I'd heartily recommend a third-party package to eliminate this exposure.

Let me know if you've found the ERASE feature to be a security benefit or an operational nightmare.

RACF-L

Q: What Internet resources would you recommend to RACF users?

A: RACF-L is one of the best on-line resources for getting RACF technical questions answered. It offers a wealth of knowledge, from a wide variety of sources, about all the ins and outs of RACF commands, structures, and usage. So if you're not a subscriber yet, don't hesitate: send an e-mail to listserv@listserv.uga.edu with the message 'subscribe RACF-L first-name last-name'.

REQUEST FOR SETROPTS INFORMATION

I'm preparing a full-sized article on SETROPTS 'best practice', and your input would be much appreciated. If you've got views or suggestions on what works best for your environment, please send them to me. If you could send along a text file of your SETROPTS settings themselves, that would be even better. Your information will of course be kept strictly confidential.

FUTURE COLUMN SUBJECTS

Here are some ideas in the pipeline for this column. Let me know what you think:

- Password structure tips.
- Audit – how much is too much?
- SECLEVEL: what's best for you?
- Shared libraries – what are the risks?
- RACF's business continuity issues.
- How to win friends and influence technical support.

But above all, please feel free to make suggestions or pose questions of your own. That's what this column is here for.

Please send your questions, comments, and suggestions for future articles to Doc.Farmer@sbm.net.sa

---

Doc Farmer
Manager and Senior IS Security Analyst (Middle East)

Information point – reviews

SECURITYFOCUS – http://www.securityfocus.com

Although not all free newsletters offer much in the way of valuable information, this one certainly does.

The right half of the home page is labelled ONLINE and is divided into five sections: News, Columnists, InFocus, Vulnerabilities, Advisories. There's also a subsection of news titled From the Wires. Further down the page, a section labelled SecurityFocus Research provides access to some of the company's most useful recent findings. If you were there in early June, you would have found the '2002 Quarter 1 (Q1) Top 10 Attacks and Vulnerabilities' and a Time Survey summarizing the percentages of time that security professionals spend searching for Internet security information, with a link to the full report at:

http://www.securityfocus.com/corporate/research/timesurvey.shtml

In the bottom right corner of the home page is the SecurityFocus Upcoming Events section. Although it lists the company's booth numbers at future conferences, it also provides a useful list of security events with dates and links to each conference's Web site.

While the home page will lead you to all of the material written by the company's own people, it's easy to miss a very comprehensive library of links to relevant research hosted on other sites, even recently completed doctoral theses. In the top right corner of the page just below the Search and Go boxes and buttons, you'll find a Library link. Don't be turned off by the fact that the first thing you see is a book for sale with a Buy Now button. Look to the right and you'll see a list of categories that you can click for access to, amongst many others:

- Access control
- Auditing
- Authentication
- Computer crime
- Confidentiality
- Cryptography
- Hostile code
- Incident handling
- Intrusion detection
- Operating systems
- Security
- Vulnerabilities.

Unfortunately, the Library is also where you'll find one of the problems I found with this site: when you click on some of these Category links, you'll often see the same paper repeated several times in a row.

IT-DIRECTOR.COM – http://www.it-director.com

Bloor Research's portal might have you clicking immediately on Security under Tech Spotlights on the left side of the page. But, if you do, you'll be missing much of what this site offers on the subject of security. The News Analysis, News Briefs, and Today's News sections that occupy the vast majority of the home page include security-relevant items that can keep you up to date on recent events and analysis.

The Columns section on the left side of the page includes Info Assurance and Biz Ethics. And don't overlook the columnists linked by name at the top of the Columns section. They often write about security.

IT-ANALYSIS.COM – http://www.it-analysis.com

At the bottom of the Misc section of links on the left side of the IT-Director.com home page is a link to sister site IT-Analysis.com, also run by Robin Bloor's British IT research organization. However, the News Analysis, News Briefs, and Breaking News are not the same either in sequence or content as those on IT-Director.com. It's therefore worth looking at both sites.

JUSTIFYING A CONFERENCE

http://www.go2vanguard.com/expo2002/2002_RACF_Justification.pdf

Although Vanguard wrote this document specifically for RACF security staff wishing to attend its Enterprise Security Expo, a little imagination can turn it into an internal justification for any security conference or other event. Note, however, that converting .pdf files into Microsoft Word documents for use internally within your organization can be a very painful experience.

Jon E Pearkins
(Canada)

Contributing to *RACF Update*

In addition to *RACF Update*, the Xephon family of *Update* publications now includes *AIX Update*, *CICS Update*, *DB2 Update*, *MQ Update*, *MVS Update*, *TCP/SNA Update*, and *VSAM Update*. Although the articles published are of a very high standard, the vast majority are not written by professional writers, and we rely heavily on our readers themselves taking the time and trouble to share their experiences with others. Many have discovered that writing an article is not the daunting task that it might appear to be at first glance.

They have found that the effort needed to pass on valuable information to others is more than offset by our generous terms and conditions and the recognition they gain from their fellow professionals. Often, just a few hundred words are sufficient to describe a problem and the steps taken to solve it.

If you have ever experienced any difficulties with RACF, or made an interesting discovery, you could receive a cash payment, a free subscription to any of our *Updates*, or a credit against any of Xephon's wide range of products and services, simply by telling us all about it.

More information about contributing an article to a Xephon Update, and an explanation of the terms and conditions under which we publish articles, can be found at http://www.xephon.com/index/nfc/css/nfc.html. Alternatively, please write to the editor, Fiona Hewitt, at any of the addresses shown on page 2, or e-mail her at fionah@xephon.com

MegaCryption/MVS from MegaCryption Labs was designed to protect z/OS data, going beyond RACF's dataset protection, to encryption, signing, integrity checking, and even controlling encryption key usage by RACF. Available in September, Version 5.2.0 adds support for both OpenPGP and IBM cryptographic coprocessors.

For more information contact:
ASPG, 3185 Horseshoe Drive S, Naples, FL 34104, USA.
Tel: (800) 662 6090.
Software Europe, Nibley House, Low Moor Road, Doddington Road, Lincoln LN6 3JY, UK.
Tel: 01522 881300.
URLs: http://www.megacryption.cc
http://www.aspg.com
http://www.software-europe.co.uk

* * *

Vanguard has just released INCompliance, audit software that performs 200 checks for compliance against RACF best practices. As well as a Detail View of any non-compliance, it also provides instant e-mail notification for high exposure changes or exceptions to your policy standards or rules. The Advisor feature demonstrates and explains each compliance check.

For more information contact:
Vanguard Integrity Professionals, 2950 East Flamingo Road, Suite D-1, Las Vegas, NV 89121, USA.
Tel: (800) 939 8266.
URL: http://www.go2vanguard.com

* * *

BMC has enhanced components of its Patrol Version 7 infrastructure to simplify installation, increase security, and add new platforms, including mainframe SuSE Linux Enterprise Server 7. You can now control the level of security for product installations, and define management profiles and classifications of job responsibilities to limit the ability of a group of users to perform tasks within PATROL.

For more information contact:
BMC Software, 2101 CityWest Blvd, Houston, TX 77042, USA.
Tel: (800) 793 4262.
URL: http://www.bmc.com/solutions/patrol7

* * *

Consul/zLock verifies all RACF command keywords against your security policies and procedures to ensure compliance. Non-compliant commands are not allowed to execute, and real-time alerts are generated if certain types of RACF command are issued. Consul/zLock is the latest member of zSecure Pro Suite, which also includes Consul/zAdmin RACF, Consul/zAudit RACF, Consul/zAudit ACF2, Consul/zVisual RACF and Consul/zToolkit.

For more information, contact:
CONSUL risk management, Marshallaan 2, 2625 GZ Delft, The Netherlands.
Tel: (31) 15 2513333
URL: http://www.consul.com/index.php3?cid=92

* * *