34

RACF update

November 2003

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 *CICS Update* are paid for at the rate of £100 ($160) per 1000 words and £50 ($80) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £20 ($32) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

HLQ security check

In a mainframe shop with z/OS, it’s very important to check the ‘dataset’ type resource definitions – especially those belonging to the base software. Specifically, after the installation of a new version of the OS, you need to check that all target and distribution dataset prefixes are protected by RACF. It goes without saying that, because of the nature and use of the base software, all the prefixes examined refer only to datasets catalogued directly in the master catalog.

We wanted to avoid the situation whereby RACF, if it is active with the ‘ProtectAll’ option, might prevent a new address space from starting. To this end, I wrote the REXX code presented here.

The code should preferably be launched as batch, and produces a brief descriptive report of the installation. It can also be launched on the ‘driving system’ as long as it can access the SMP/E CSI that describes the ‘target system’. The procedure will examine each dataset’s high level qualifier to check that they are defined both in the Master Catalog and in a DDDEF of the target SMP/E.

In a correct installation, all the dataset prefixes that are defined in SMP/E should also be protected by RACF. The final list produced by this code will show which prefix is used by SMP/E and/or the Master Catalog and whether a RACF profile is missing or not.

QUAL1EXA REXX SAMPLE

```
/* REXX ---------------------------------------------------------- */
/* REXX COMMAND : %QUAL1EXA MVS.GLOBAL.CSI_NAME  TARGET_ZONE_NAME  */
/* REXX ---------------------------------------------------------- */
 "PROF NOPREF"
 PARSE UPPER ARG CSI_MVS TGT_ZON TRC_ACT
 IF TRC_ACT = 'ON'
   THEN DO
```

```
  TRACE ALL
 MSG_STATUS = MSG("ON")
      END
     ELSE DO
     MSG_STATUS = MSG("OFF")
      TRACE OFF
          END

  IF TGT_ZON = ' '
      THEN DO
     SAY ">>E>> TARGET ZONE NAME OMITTED "
       EXIT(15)
           END

 AA = OUTTRAP(LINE.)
 "LISTC ENT("CSI_MVS ")"
  IF RC  <> Ø
      THEN DO
     SAY ">>E>> 'MVS.CSI' NAME OMITTED "
       EXIT(25)
           END
 LL = OUTTRAP(OFF)

/* - MASTER CATALOG --------------------- */
"FREE F(ICFQUAL1)"
"ALLOC F(ICFQUAL1) UNIT(VIO) RECFM(F B) LRECL(8)
   SPACE (5) TRACKS NEW DELETE"
   IF RC > Ø THEN EXIT(35)
"EXECIO Ø DISKW ICFQUAL1 (OPEN"

 AA = OUTTRAP(LINE.)
  "LISTC NONVSAM "
   IF RC > Ø THEN EXIT(45)
 QUAL1_PREV = '_$_$_'

      DO I =1 TO LINE.Ø

     DSN_CAT  = SUBSTR(LINE.I,17,1Ø)
     POS_DOT  = INDEX(DSN_CAT,'.')

         IF POS_DOT = Ø
          THEN QUAL1_DSN = DSN_CAT
              ELSE  DO
             QUAL1_LEN = POS_DOT - 1
          QUAL1_DSN = SUBSTR(DSN_CAT,Ø1,QUAL1_LEN)
                   END

       IF QUAL1_DSN <> QUAL1_PREV
            THEN DO
           QUAL1_PREV = QUAL1_DSN
```

```
              ICFQUAL1   = QUAL1_PREV
               PUSH ICFQUAL1
              "EXECIO 1 DISKW ICFQUAL1 "
                     END

         END

  LL = OUTTRAP(OFF)

 "EXECIO Ø DISKW ICFQUAL1 (FINIS"

/* - DDDEF TARGET SMP/E DATASET ---------- */
 MLØØØØ:
 "FREE F(SMPLIST)"
 "ALLOC F(SMPLIST) UNIT(VIO) RECFM(F B A) LRECL(121) BLKSIZE(726Ø)
     SPACE (5 5) TRACKS NEW DELETE"
 "FREE F(SMPOUT)"
 "ALLOC F(SMPOUT)  DUMMY"
 "FREE F(SMPLOG)"
 "ALLOC F(SMPLOG)  DUMMY"
 "FREE F(SMPLOGA)"
 "ALLOC F(SMPLOGA) DUMMY"
 "FREE F(SMPRPT)"
 "ALLOC F(SMPRPT)  DUMMY"
 "FREE F(SMPCSI)"
 "ALLOC F(SMPCSI) SHR DSN("CSI_MVS")"
 "FREE F(SMPCNTL)"
 "ALLOC F(SMPCNTL) UNIT(VIO) RECFM(F B) LRECL(8Ø) NEW DELETE"
 RECNTL = "   SET BOUNDARY("TGT_ZON"). "
 QUEUE RECNTL
 RECNTL = "    LIST DDDEF.           "
 QUEUE RECNTL
 QUEUE ''
 "EXECIO * DISKW SMPCNTL (FINIS"
 ADDRESS "LINKMVS" "GIMSMP"
     IF RC  <> Ø
        THEN DO
       SAY ">>E>> PGM : GIMSMP RETURNED ERROR "
         EXIT(55)
             END

/* -------------------------------------- */
 "FREE F(SMPWORK)"
 "ALLOC F(SMPWORK) UNIT(VIO) RECFM(F B) LRECL(8)
    SPACE (5) TRACKS NEW DELETE"
    IF RC > Ø THEN EXIT(65)
 "EXECIO Ø DISKW SMPWORK  (OPEN"
 "EXECIO Ø DISKR SMPLIST  (OPEN"
         DO FOREVER
         "EXECIO 1 DISKR SMPLIST"
         PULL REC_SMPLIST
          IF RC <> Ø THEN LEAVE

         IF SUBSTR(REC_SMPLIST,12,17) = 'DATASET      ='
               THEN   DO
            SMPLIST_DSN  = SUBSTR(REC_SMPLIST,3Ø,1Ø)
              POS_DOT    = INDEX(SMPLIST_DSN,'.')
               QUAL1_LEN   = POS_DOT - 1
            REC_SMPWORK  = SUBSTR(SMPLIST_DSN,Ø1,QUAL1_LEN)
               PUSH REC_SMPWORK
               "EXECIO 1 DISKW SMPWORK "
                      END
          END
  "EXECIO Ø DISKW SMPWORK  (FINIS"
  "EXECIO Ø DISKR SMPLIST  (FINIS"

 /* - ICETOOL - START --------------------- */
  "FREE  F(SMPQUAL1)"
  "ALLOC F(SMPQUAL1) UNIT(VIO) RECFM(F B) LRECL(8)
     SPACE (5) TRACKS NEW DELETE"
     IF RC > Ø THEN EXIT(75)
  "FREE  F(TOOLMSG)"
  "ALLOC F(TOOLMSG) DUMMY"
  "FREE  F(DFSMSG)"
  "ALLOC F(DFSMSG)  DUMMY"
  "FREE  F(TOOLIN)"
  "ALLOC F(TOOLIN) UNIT(VIO) RECFM(F B) LRECL(8Ø) NEW DELETE"
  REC_TOOLIN = " SELECT FROM(SMPWORK) TO(SMPQUAL1) ON(Ø1,8,CH) FIRST "
  QUEUE REC_TOOLIN
  QUEUE ''
  "EXECIO * DISKW TOOLIN (FINIS"

  ADDRESS  "LINKMVS" "ICETOOL"
     IF RC  <> Ø  THEN EXIT(85)

 /* - ICETOOL - END ---------------------- */
  "EXECIO Ø DISKR SMPQUAL1  (OPEN"
  "EXECIO Ø DISKR ICFQUAL1  (OPEN"
  CALL READ_SMP
  CALL READ_ICF
  CALL PRINT_HEAD

  DO FOREVER
  END_FILE_ALL = SUBSTR(QUAL1_SMP,1,2)SUBSTR(QUAL1_ICF,1,2)

   SELECT
  WHEN END_FILE_ALL = '9999'
  THEN LEAVE

 WHEN QUAL1_SMP = QUAL1_ICF
   THEN DO
  CALL PRINT_PAGE
  CALL READ_SMP
  CALL READ_ICF
       END

 WHEN QUAL1_ICF > QUAL1_SMP
   THEN DO
  CALL PRINT_PAGE
  CALL READ_SMP
       END

 WHEN QUAL1_ICF < QUAL1_SMP
   THEN DO
  CALL PRINT_PAGE
  CALL READ_ICF
       END
  END
 END
 SAY " * ---------------------------------------------------- * "

/* - END ------------------------------- */
 EXIT(ØØ)
/* ------------------------------------ */

/* - INTERNAL ROUTINE - READ SMP --------- */
READ_SMP:
 IF QUAL1_SMP <> '99SMP99'
   THEN DO
  "EXECIO 1 DISKR SMPQUAL1"
      IF RC >  Ø THEN DO
         QUAL1_SMP = '99SMP99'
         "EXECIO Ø DISKR SMPQUAL1  (FINIS"
                    END
         ELSE DO
            PULL QUAL1_SMP
             END
       END
 RETURN

/* - INTERNAL ROUTINE - READ ICF --------- */
READ_ICF:
 IF QUAL1_ICF <> '99ICF99'
   THEN DO
  "EXECIO 1 DISKR ICFQUAL1"
```

```
      IF RC >  Ø THEN DO
          QUAL1_ICF = '99ICF99'
         "EXECIO Ø DISKR ICFQUAL1  (FINIS"
                    END
         ELSE      DO
             PULL QUAL1_ICF
                    END
       END
RETURN

/* - INTERNAL ROUTINE - PRINT PAGE ------- */
PRINT_PAGE:

PRINT_ICF = ''
PRINT_SMP = ''
PRINT_RACF = ''

SELECT

WHEN QUAL1_ICF = QUAL1_SMP
     THEN DO
   MY_QUAL1 = "('"STRIP(QUAL1_ICF,'T')".*')"
     "LD DA"MY_QUAL1
      MY_RC = RC
    PRINT_ICF = QUAL1_ICF
    PRINT_SMP = QUAL1_SMP
          END

WHEN QUAL1_ICF < QUAL1_SMP
     THEN DO
   MY_QUAL1 = "('"STRIP(QUAL1_SMP,'T')".*')"
     "LD DA"MY_QUAL1
      MY_RC  = RC
    PRINT_ICF = QUAL1_ICF
    PRINT_SMP = '------- '
          END

WHEN QUAL1_ICF > QUAL1_SMP
     THEN DO
   MY_QUAL1 = "('"STRIP(QUAL1_ICF,'T')".*')"
     "LD DA"MY_QUAL1
      MY_RC  = RC
    PRINT_ICF = '------- '
    PRINT_SMP = QUAL1_SMP
          END

OTHERWISE NOP

END
```

```
 IF MY_RC = Ø THEN PRINT_RACF = ' YES '
   ELSE PRINT_RACF = '  NO '

 P_A1 = ' *        '
 P_A2 = PRINT_ICF '     ' PRINT_SMP
 P_A3 = '    ' PRINT_RACF
 SAY P_A1 P_A2 P_A3 "      *"

 RETURN

/* - INTERNAL ROUTINE - PRINT HEAD ------- */
 PRINT_HEAD:
 P_BL = '        '
 W_DA = DATE('S')
 X1 = SUBSTR(W_DA,5,2)
 X2 = SUBSTR(W_DA,7,2)
 X3 = SUBSTR(W_DA,1,4)
 P_DA = X1"/"X2"/"X3
 P_TI = TIME()
 P_RA = SYSVAR("SYSLRACF")
 P_IP = MVSVAR('SYSNAME')
 P_MS = CSI_MVS
 P_LL = P_MS P_BL P_BL
 P_MI = SUBSTR(P_LL,1,44)
/* - CVT ------------------------------- */
 CVT  = C2D(STORAGE(1Ø,4))
/* - AMCBS ----------------------------- */
 A_1  = C2D(STORAGE(D2X(CVT + 256),4))
/* - ACB ------------------------------- */
 A_2  = C2D(STORAGE(D2X(A_1 + 8),4))
/* - CAXWA ----------------------------- */
 A_3  = C2D(STORAGE(D2X(A_2 + 64),4))
/* - MASTER CATALOG NAME ---------------- */
 P_MT = STORAGE(D2X(A_3 + 52),44)

 SAY "*-----------------------------------------------------*"
 SAY " * " "TIME: " P_TI "  DATE: " P_DA P_BL P_BL " *"
 SAY " * " "SYSTEM ID: " P_IP P_BL P_BL P_BL P_BL "*"
 SAY " * " "LVL/RACF : " P_RA P_BL P_BL P_BL P_BL "*"
 SAY " * " "MVS MCAT : " P_MT "*"
 SAY " * " "MVS CSI  : " P_MI "*"
 SAY "*-----------------------------------------------------*"
 SAY " *         MASTER         SMP/E          RACF        * "
 SAY "*-----------------------------------------------------*"

 RETURN
_
```

SAMPLE JCL TO RUN QUAL1EXA

```
//...... JOB ......,......,CLASS=.,MSGCLASS=.,REGION=ØK
//** ---------------------------------------------- **
//MYREXX  EXEC PGM=IKJEFTØ1,DYNAMNBR=2ØØ,
// PARM='%QUAL1EXA your.mvs_GLOBAL_CSI your.mvs_target_zone'
//** ---------------------------------------------- **
//SYSPRINT DD  DUMMY
//SYSTSPRT DD  SYSOUT=your.output_class
//SYSEXEC DD  DISP=SHR,DSN=your.jcllib
//SYSTSIN DD  DUMMY
```

PRINTOUT RESULT

```
*-------------------------------------------------------*
 *  TIME:  xx:xx:xx  DATE:  xx/xx/xxxx                  *
 *  SYSTEM ID:  IPOX                                    *
 *  LVL/RACF :  xxxx                                    *
 *  MVS MCAT :  CATALOG.MCAT......                      *
 *  MVS CSI  :  .......GLOBAL.CSI                       *
*-------------------------------------------------------*
 *        MASTER         SMP/E          RACF            *
*-------------------------------------------------------*
 *         AOP           AOP            YES             *
 *         ASM           ASM            YES             *
 *         ASU           ASU            YES             *
 *         BFS           BFS            YES             *
 *         CDS           CDS             NO             *
 *         CEE           CEE            YES             *
 *        -------        CICSTS          NO             *
 *         CSF           CSF            YES             *
 *         DIT           DIT            YES             *
 *         DRL           DRL            YES             *
 *         DVG           DVG            YES             *
 *        -------         IMS           YES             *
 *         IMW           IMW            YES             *
 *         IOA           IOA            YES             *
 *         IOE           IOE            YES             *
 *         ISF           ISF            YES             *
 *         ISP           ISP            YES             *
 *        -------         JAVA          YES             *
 *        NETVIEW        NETVIEW        YES             *
 *        OMVS           -------        YES             *
 *        REXX           REXX           YES             *
 *        -------        SMPZ13         YES             *
 *        SOMMVS         SOMMVS          NO             *
 *        SYS1           SYS1           YES             *
 *        TCPIP          TCPIP          YES             *
 *        TCPIVP         -------        YES             *
 *         TIVOLI        TIVOLI        YES        *
 *--------------------------------------------------*
```

Massimo Ambrosini
(Italy)

Business continuity and RACF

This article reviews the steps that you as RACF administrators can take, in conjunction with your technical support and business continuity departments, to ensure that your IT recovery site can continue processing with an adequate level of security. It discusses the options for RACF recovery from a variety of situations:

- Recovery at a third-party contingency site
- Recovery at your own separate contingency site
- LPAR-to-LPAR database recovery.

Let’s face it: in a disaster situation, security is probably one of the last things people are really thinking about. Yet it can be one of the most important items required to make your recovery plans work properly. Not only that, it can also be the difference between the life and death of your company.

THE BASICS

Let’s start with the simple stuff. What’s your business continuity position in a disaster? Hiding under your desk in the foetal position, sucking your thumb, doesn’t count – I’m talking about your company’s position:

- Do you have a business continuity department? It could be called any of the following:
  - business continuity
  - business continuation

  - business-as-usual
  - contingency planning
  - disaster recovery
  - incident response.
- Is there an overall plan for recovery from a disaster situation?
  - are there specific, detailed technical recovery plans?
- Does the plan include an operating system recovery plan?
- Does the operating system recovery plan address RACF?
- Have the technical plans ever been tested?

If you have answered positively to all of the above, you're in business. If you haven't, you could be out of business if disaster strikes. It's that simple, folks.

For the purposes of this article, we'll assume that you have the items above. Now, how do you recover your security database? Well, that depends on your recovery options.

OPTION 1 – THIRD-PARTY RECOVERY SITE

The third-party recovery site option is very simple and straightforward. Back up the RACF database along with all of the other OS/390 system files. Recover them onto the DASD at the third-party site, and you're in business, right?

Well, no – wrong actually. You need to ask yourself some hard questions:

- How often is the RACF database backed up?
- What do you do if your back-ups run on Sunday at 04.00 and your system gets fried on Saturday at 22.00?
- Do you back up both the primary and secondary RACF databases?
- How long does it take to recover those files onto the third-party site's DASD?

- How do you handle DASD volume differences between sites?
- How do you handle IP address security between sites?

If at all possible, you should make sure that your RACF databases (primary and secondary) are backed up every night, preferably after the overnight production run. This back-up should then be sent along with your other off-site back-ups to a secure location. Please note, secure location does *not* mean in the same building as you made the back-ups, nor in a storage shed 20 feet away from the building where you made the back-ups. Try to ensure that your back-up tapes are stored at least one mile (1.6 km) away from your data centre. If you're in an area prone to earthquakes, make that a much longer distance, or transfer your files by secured high-speed data link.

Do you know how long it will take for your database to be recovered from tape onto the third-party system? Don't guess. When your company has a test of its recovery site, wangle a way to go with the rest of the team. You'll probably have to go anyway, just to fix the problems that invariably crop up. However, assuming that you've got all the back-up tapes, and all of them are readable (sometimes they're not, so a second set of back-ups is always a good idea), recovery of the operating system shouldn't take more than two to three hours.

Once the restore has been run and the operating system is up and running, you'll want to do two things. First, run any JCL you created before the test to update the DASDVOL settings within RACF, as well as any IP addresses you've hard-coded into the database. You should be able to get that information directly from your recovery site's technical support staff. Once this is done, run an IRRUT400 with INDEX and MAP to check the integrity of the database. The report is a bit long-winded, but it should give you some idea of any potential problems early on in the test.

As you progress through the recovery test, always make notes of any changes you make to the RACF database – access permissions, ID sign-on problems, etc. You'll need this to enhance your recovery JCL, so that you can install all of those fixes early in the recovery process for the next test (or, of course, the real thing). Also, it would probably be a good thing to bring along a log of all of the updates you made to the RACF database in the days before the recovery test. It'll help you keep track of items that may have been missed if you're using a back-up from a few days past.

OPTION 2 – RECOVERY AT YOUR OWN SEPARATE CONTINGENCY SITE

Recovery at your own separate contingency site doesn't really differ much from the first option if your separate contingency site is a warm site (equipment there, but not loaded with operating system or software). If it's a hot site (fully loaded with programs, and just awaiting data), you need only restore the most recent RACF database from your production site and do some RVARY SWITCH and RVARY ACTIVE commands to replace the old database. You'll still need to stay on top of any operational security problems, of course, but if your company can afford its own contingency site, you'll save precious hours and minutes. That could easily make the difference between life and death for your organization – and that's no exaggeration.

OPTION 3 – LPAR-TO-LPAR DATABASE RECOVERY

One of my friends in technical support came up with this little jewel for LPAR-to-LPAR database recovery, so I can't claim full credit for the code presented here. However, I've found that it works very well, and it keeps the databases on separate LPARs in sync like a treat. It's also easy to run, and can be set up to execute the first part automatically on a daily basis (through automated production control software).

This option does require a couple of things, however. First, your systems must have a copy of Connect:Direct on board to use the JCL shown below. It'll probably also work with MQSeries or a simple ftp transfer as well, but these are options you'll need to discuss with your technical support and network gurus. It also requires your LPARs to be in separate physical locations (ie city A and city B).

If you need to keep RACF databases in, let’s say, a development and test LPAR in sync in a single box, this process will work. But the main focus of this article is recovery from a catastrophic loss of computer services. If you keep your production and recovery LPARs on the same piece of iron, you might as well tack a ‘kick me’ sign on your back. You’re much safer having your recovery LPAR on a separate computer at a separate facility in a separate city. City-wide blackouts have been known to happen, so it’s better to be safe than sorry. Besides, when the technical and operations staff do a test of their recovery plans, it gives you a good excuse to get out of town for a couple of days.

Anyway, back to the JCL. Run the first job on a daily basis in the early morning (my favourite time is 04:00) when your major production processing has finished. This job requires only one point of operator intervention, and that’s just to ensure that Connect:Direct is up and running. After that point, the transfer goes on its merry way without the interference of mere mortals.

The second set of JCL is the one you submit yourself, from your own TSO library. Actually, before you submit the job, split your screen using F2 (if you’re using IBM’s PComm software) and have one frame showing the SDSF system log. You’ll need it.

Submit the job, and then switch to the log. The mainframe will generate console messages asking for the RVARY ACTIVE and RVARY SWITCH commands. At the top of the SDSF log, type:

```
/nn,xxxxxxxx
```

where *nn* is the console message number and *xxxxxxxx* is the password. You’ll alternate between the VARY and SWITCH passwords, entering each one of them twice. The last step of the JCL performs a simple RVARY LIST, which you should use to ensure that your primary and back-up databases are in the right positions.

SOME FINAL COMMENTS ON BUSINESS CONTINUITY

- Some operations like to create ‘recovery’ userids for their contingency sites. You should avoid this like the plague! Those ids are usually of the ‘super’ variety, having access to everything. They also probably have OPERATIONS and SPECIAL capabilities as well. This may be a great shortcut, but it also short-circuits your overall security.

- Keep copies of your recovery plan off-site. Keep a copy at home. Keep it up to date. In the event of an emergency, you want to provide the fastest possible response to minimize recovery time. Digging around for hours in a storage facility, trying to find a particular plan, is a waste of time.

- Remember the old adage – ‘Failure to plan on your part does not constitute a crisis on *my* part’. If other areas don’t plan for disaster situations, that doesn’t mean that you shouldn’t as well. And if things do go wrong, you always have the option of saying those four little words that mean so much – “*I told you so!!!*”

JCL EXAMPLES

RACF database unload and transfer

```
//RACFDR  JOB SYS1,'SEND RACFDB TO DR',MSGCLASS=X,MSGLEVEL=(1,1),
//*     TYPRUN=HOLD,CLASS=A,NOTIFY=DFARMER,USER=PRODCTL
/*ROUTE XEQ PRD1
/*ROUTE PRINT PRD1
//*-----------------------------------------------------------------*
//*   This job copies the RACF database to a DFDSS backup for       *
//*   transmission to a separate LPAR via Connect:Direct            *
//*                                                                 *
//* This is the sequence of events:                                 *
//*                                                                 *
//* Step1: Copy the RACF database to a separate file.               *
//*                                                                 *
//* Step2: Send message to Console Operator to ensure Connect:Direct *
//*        is up and running.                                       *
//*                                                                 *
//* Step3: Send the file to the receiving LPAR.                     *
//*                                                                 *
//*-----------------------------------------------------------------*
```

```
//*        COPY PRIMARY RACF DATABASE                                  *
//*--------------------------------------------------------------------*
//STEP1  EXEC PGM=IRRUT4ØØ,
//     PARM='DUPDA FREE(3Ø) ALIGN NOLOCKINPUT'
//SYSPRINT DD SYSOUT=*
//INDD1  DD DSN=SYS1.RACF.RACFPRIM,DISP=OLD
//OUTDD1  DD DSN=SYS1.RACF.DR.BACKUP,DISP=OLD
//*--------------------------------------------------------------------*
//**********************************************************************
//*
//MSG1   EXEC IPOWTO
//SYSIN   DD  *
             ***********************
       ***********************************
 ****************************************************
*******************************************************
* *         %#%#%#%#%#%#%#%#%#%#%#%#%#%#%          * *
* *          FILE TRANSMIT TO BKP1                 * *
* *         %#%#%#%#%#%#%#%#%#%#%#%#%#%#%          * *
* *                                                * *
* *    ..........................................  * *
* *     ENSURE CONNECT DIRECT IS RUNNING ON        * *
* *        DR LPAR BKP2                            * *
* *    ..........................................  * *
****    PLEASE ENSURE CD IS UP AND RUNNING     ****
****     ISSUE D A,CD ON BKP2 CONSOLE          ****
****                                           ****
****     THEN ENTER C TO CONTINUE              ****
****                                           ****
*******************************************************
 ****************************************************
       ***********************************
             ***********************
//*--------------------------------------------------------------------*
//*       TRANSMIT THE BACKUP TO THE DR SITE                           *
//*--------------------------------------------------------------------*
//       IF (STEP1.RC = Ø) THEN
//IEBVIO  EXEC PGM=IEBGENER
//SYSIN   DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSUT2  DD DSN=&&TEMP,DISP=(,PASS),UNIT=VIO,SPACE=(TRK,1)
//SYSUT1  DD *
 SIGNON NODE=RACF.MVS.PRD1
  SUB PROC=COPYNET                     -
     SNODE=RACF.MVS.BKP1               -
    &DSN1=SYS1.RACF.DR.BACKUP          -
    &DSN2=SYS1.RACF.DR.BACKUP          -
    &ASCB=@@@@@@@@ &ECB=########
 SIGNOFF
/*
```

```
//CDWAIT  EXEC PGM=CDWAIT,PARM='3Ø:3Ø,YYLLYYY'
//DMPUBLIB DD DSN=SYST.CD.PROCESS,DISP=SHR
//DMNETMAP DD DSN=SYST.CD.NETMAP,DISP=SHR
//DMMSGFIL DD DSN=SYST.CD.MSG,DISP=SHR
//NDMCMDS  DD SYSOUT=*
//DMPRINT  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN   DD DSN=&&TEMP,DISP=(OLD,DELETE)
/*
//       ENDIF
//$J EXEC PGM=$A,COND=(Ø,LE),PARM='RAC3Ø527ØØØ1ØØØØØØØØ'
```

RACF restore job

```
//RACFREST JOB SYS1,'RACF DB RESTORE',MSGCLASS=X,MSGLEVEL=(1,1),
//     CLASS=A,NOTIFY=DFARMER,TYPRUN=HOLD
/*ROUTE XEQ BKP1
/*ROUTE PRINT BKP1
//*------------------------------------------------------------------*
//*   This job restores the RACF database from a DFDSS backup        *
//*                                                                  *
//* This is the sequence of events:                                  *
//*                                                                  *
//* Step1: Issue RACF Command to list the state of the RACF datasets *
//*        before we start                                           *
//*                                                                  *
//* Step2: Will switch to the backup RACF database, a reply to       *
//*     confirm the switch at the master console is required.        *
//*                                                                  *
//* Step3: Restores the Primary RACF database from a backup          *
//*                                                                  *
//* Step4: Activates the newly restored RACF database, a reply to    *
//*     confirm the activation at the master console is required.    *
//*                                                                  *
//* Step5: Switches back to the Primary RACF database, a reply to    *
//*     confirm the switch at the master console is required.        *
//*                                                                  *
//* Step6: Lock and copy the primary to the backup database          *
//*                                                                  *
//* Step7: Unlocks the Primary after the copy has completed          *
//*                                                                  *
//* Step8: Activates the backup RACF database                        *
//*                                                                  *
//*------------------------------------------------------------------*
//*      Query the status of RACF databases                          *
//*------------------------------------------------------------------*
//STEP1   EXEC PGM=COMMAND
//INFILE  DD *
 RACF RVARY LIST
/*
```

```
//*-------------------------------------------------------------*
//*     Issue the switch command (from Primary database)        *
//*-------------------------------------------------------------*
//      IF (STEP1.RC = Ø) THEN
//STEP2  EXEC PGM=IKJEFT1A
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
 RVARY SWITCH DATASET(SYS1.RACF.RACFPRIM)
/*
//       ENDIF
//*-------------------------------------------------------------*
//*     Restore the Primary RACF database from the backup       *
//*-------------------------------------------------------------*
//      IF (STEP2.RC = Ø) THEN
//STEP3  EXEC PGM=IRRUT4ØØ,PARM='NOLOCKINPUT,FREESPACE(2Ø)'
//SYSPRINT DD SYSOUT=*
//INDD1  DD DSN=DSYS.PRD1BKUP.RACFPRIM,DISP=OLD
//OUTDD1  DD DSN=SYS1.RACF.RACFPRIM,DISP=OLD
//       ENDIF
//*-------------------------------------------------------------*
//*      Activate the newly restored RACF database              *
//*-------------------------------------------------------------*
//      IF (STEP3.RC = Ø) THEN
//STEP4  EXEC PGM=IKJEFT1A
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
 RVARY ACTIVE DATASET(SYS1.RACF.RACFPRIM)
/*
//       ENDIF
//*-------------------------------------------------------------*
//*      Switch back to the Primary RACF database               *
//*-------------------------------------------------------------*
//      IF (STEP4.RC = Ø) THEN
//STEP5  EXEC PGM=IKJEFT1A
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
 RVARY SWITCH DATASET(SYS1.RACF.RACFBACK)
/*
//       ENDIF
//*-------------------------------------------------------------*
//*     Lock and copy the primary to the backup database        *
//*-------------------------------------------------------------*
//      IF (STEP5.RC = Ø) THEN
//STEP6  EXEC PGM=IRRUT4ØØ,PARM='LOCKINPUT,FREESPACE(2Ø)'
//SYSPRINT DD SYSOUT=*
//INDD1  DD DSN=SYS1.RACF.RACFPRIM,DISP=OLD
//OUTDD1  DD DSN=SYS1.RACF.RACFBACK,DISP=OLD
//*
//       ENDIF
//*-------------------------------------------------------------*
```

```
//*       Unlock the primary                                    *
//*-------------------------------------------------------------*
//      IF (STEP6.RC = Ø) THEN
//STEP7  EXEC PGM=IRRUT4ØØ,PARM='UNLOCKINPUT'
//SYSPRINT DD SYSOUT=*
//INDD1  DD DSN=SYS1.RACF.RACFPRIM,DISP=OLD
/*
//       ENDIF
//*-------------------------------------------------------------*
//*      Activate the backup RACF database                      *
//*-------------------------------------------------------------*
//      IF (STEP7.RC = Ø) THEN
//STEP8  EXEC PGM=IKJEFT1A
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
 RVARY ACTIVE DATASET(SYS1.RACF.RACFBACK)
/*
//       ENDIF
//*-------------------------------------------------------------*
//*      Query the status of RACF databases                     *
//*-------------------------------------------------------------*
//      IF (STEP8.RC = Ø) THEN
//STEP9  EXEC PGM=COMMAND
//INFILE  DD *
 RACF RVARY LIST
/*
//       ENDIF
```

Doc Farmer
Senior IS Security Analyst (USA)

PassTicket generator

A RACF PassTicket is a one-time password which calculates a password based on the name of the application, the time, the userid, and a secret key. PassTickets are valid for approximately 10 minutes and generally can't be reused in that time. PassTickets are often used for Secure Single Sign-on procedures where a machine issues a log-on command to an application on behalf of an already authenticated user, without user intervention.

This article begins by explaining how PassTickets work and examining some possible pitfalls. It ends by presenting the coding for a small ISPF PassTicket generator application.

PASSTICKETS PRIMER

Apart from the userid itself, a PassTicket has the following components.

The secret key is kept in the RACF database in a profile in the PTKTDATA class and consists of a 64-bit DES key that can be masked or encrypted if a cryptographic product like Integrated Cryptographic Service Facility (ICSF) is installed. If one of the systems is not RACF-controlled – an NT workstation, for instance – the place where the common secret, the DES key, is stored should be carefully checked. Once the secret key for an application is known, an intruder needs only a userid and the name of the application to generate a valid PassTicket.

The time is based on Greenwich Mean Time (GMT), nowadays called Universal Time Coordinates (UTC). For a PassTicket to function, the machines must be synchronized.

The application can be any VTAM application that passes the APPL= on the RACROUTE REQUEST=VERIFY,ENV=CREATE macro:

- TSO/E, for instance, does not, but there is a bypass hardcoded in the PassTicket algorithm.
- Pre-OS/390 V2R10 releases can use TSO<SMFID> as the application name.
- Starting from OS/390 V2R10, the VTAM Generic Resource Name (VGN) is used if defined in the GNAME= parameter in the TSOKEY*xx* member of the PARMLIB concatenation. Without a GNAME specification, TSO<SMFID> is used.
- For a batch job, the application name becomes MVS<SMF id>.

You should check the documentation of the application carefully before attempting to get PassTickets working. Sometimes a bypass can be implemented in the RACROUTE REQUEST=VERIFY(X) pre-processing exit routine ICHRIX01.

More information on PassTickets can be found in *Security Server (RACF) Security Administrator's Guide* (SC28-1915). The complete calculation algorithm is described in *SecureWay Security Server RACF Macros and Interfaces* (SC28-1914). ICHRIX01 is described in *SecureWay Security Server RACF System Programmer's Guide* (SC28-1913).

USABILITY

In our shop, we use PassTickets for Secured Single Sign-on from the session manager to a number of applications both inside and outside the sysplex configuration that contains the network front-end machine. We also autoconnect trusted non-z/OS machines to the mainframe. Finally, we have an emergency system whereby after some operator commands an SOS RACF userid with extreme authorities gets resumed. The password can be generated with the application described below, which I also found very useful while implementing and testing PassTicket implementations. I've also seen PassTickets used to send a temporary password by e-mail to users who have forgotten their normal one – they then have about ten minutes to set a new password of their own.

RACF IMPLEMENTATION

In order to start using PassTickets for an application, the PTKTDATA class must be activated and RACLISTED, some profiles must be defined, and the synchronization of the clocks must be checked. See *SecureWay Security Server RACF Command Language Reference* (SC28-1919) for further details.

For the application TSOT and the user JEDSP00, a good start would be to execute the following RACF commands:

```
setropts classact(ptktdata)
setropts raclist(ptktdata)
rdefine ptktdata tsot uacc(none) ssignon(keymasked(0123456789abcdef))
setropts refresh raclist(ptktdata)
```

PTKTDATA PROFILES

PTKTDATA profiles can be defined in four different ways:

1 <application>.<group>.<user>

2 <application>.<user>

3 <application>.<group>

4 <application>

where <application> corresponds to the profile in the APPL class. The qualifier <group> is the actual connect group of the user. Some applications – such as TSO/E for instance – allow a user to pass a group in the log-on panel. The name is passed with the GROUP= parameter of the RACROUTE REQUEST=VERIFY,ENV=CREATE macro. This information is then stored in the ACEE field ACEEGRPN, and is the only one that the PassTicket algorithm will check – other groups to which the user is connected are ignored.

The first three profiles are said to be qualified, and only the last, non-qualified, profile is used to generate a PassTicket – we were very confused until we discovered this. For verification, on the other hand, if the string sent as a password doesn't correspond to the password in the RACF database, the PTKTDATA profiles are checked (see Figure 1).

Note that if profile 1 exists, profiles 2 and 3 aren't tested – this also confused us a lot.

We wanted to minimize the impact of a lost key in an insecure environment. We thought that giving different users different keys would ensure that only one could get compromised in the case of a breach. However, this didn't work out well in a RACF-RACF situation. Since the generation of the PassTicket uses only the <application> profile, this comes down to defining all users from RACF1 who want to access an application protected by RACF2 in one group in the RACF2 database. I don't think it's a good idea to create RACF groups solely for PassTicket verification in a RACF-RACF situation – the APPL class should take care of this.

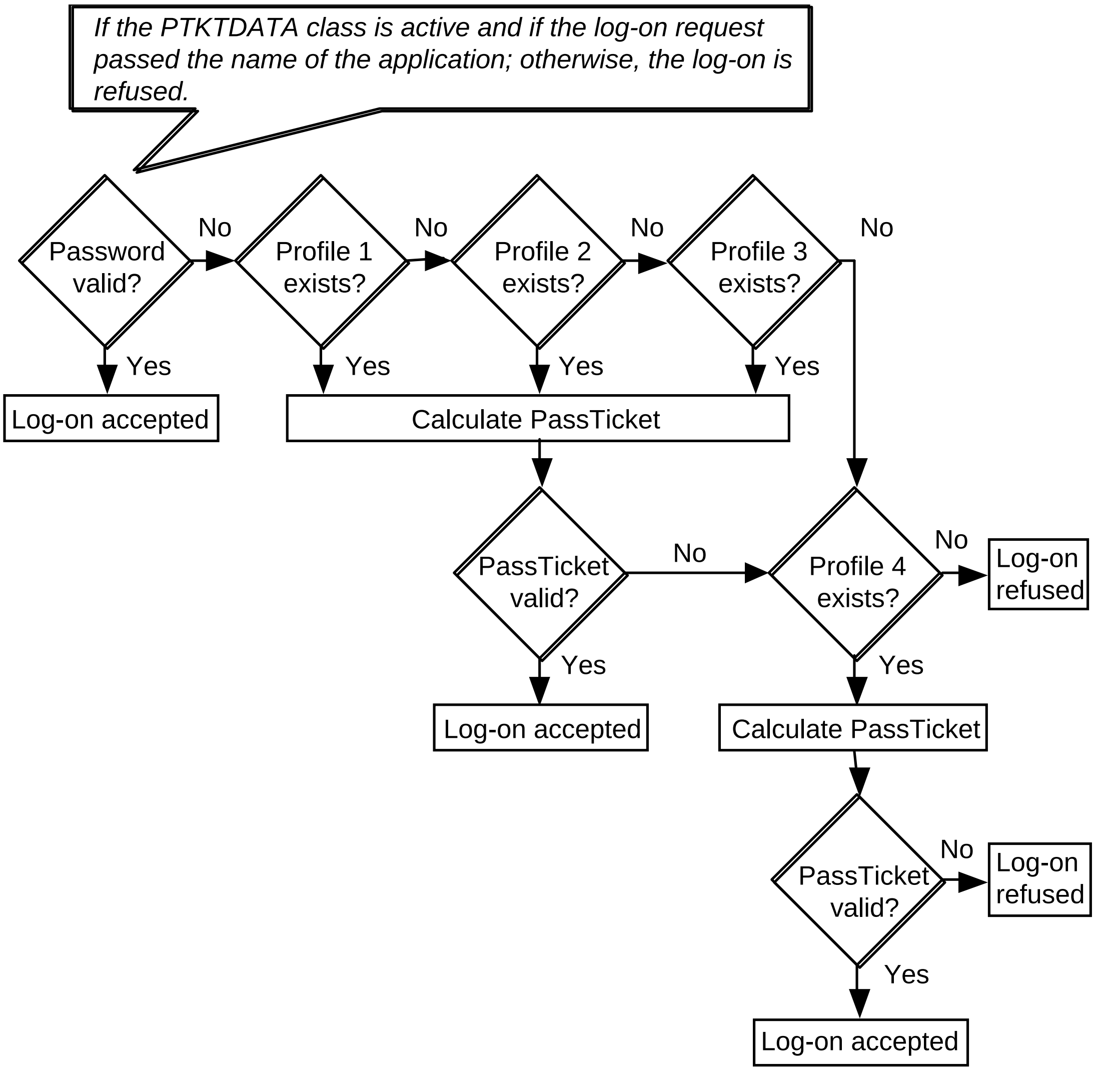


Figure 1: Checking PTKTDATA profiles

THE GENERATOR

The PassTicket generator application consists of an Assembler program, a REXX program, and a small ISPF dialog. A REXX (POTEMKIN) program which displays the panel below calls the generator program:

```
----------------------------------------
----------<Passticket Generator>----------
----------------------------------------
```

```
 VTAM Application   ===> TSOT
 User Identification ===> JEDSPØØ


 PF3 to cancel
 Enter to calculate
```

The initial values for the two variables are taken from the user ISPF profile and are later restored there. After some formatting, the REXX program will call the POTEMKIN authorized Assembler program, which will do the following.

Before calculating a PassTicket, the program will check whether the user has SPECIAL RACF authority. If this is the case, a PassTicket is always generated. Otherwise, a comparison is made between the actual userid and the requested one, and the decision is made, based on RACF profiles in the PTKTDATA class (we thought it convenient to keep them there but the class can easily be changed if needed). The profiles are formatted in one of the two following ways:

- GENERATE.<application>.<group>
- GENERATE.<application>.<user>

READ is sufficient to generate a PassTicket for your own userid, but for another user UPDATE access is required. We check this for all the groups to which the target userid is connected.

Suppose that we have RACF ‘systems’ and ‘security’ groups, and we issue the following commands:

```
rdefine ptktdata generate.tsot.* uacc(none) owner(security) +
data('Generate a PassTicket for application TSOT')
permit generate.tsot.* class(ptktdata) access(read) id(*)
permit generate.tsot.* class(ptktdata) access(update) id(security)
rdefine ptktdata generate.tsot.systems uacc(none) owner(security) +
data('Generate a PassTicket for application TSOT for a user connected to the group
systems')
permit generate.tsot.systems class(ptktdata) access(update) id(systems)
setropts refresh raclist(ptktdata)
```

With these rules in place, everybody can calculate a PassTicket for the application TSOT, but only for their own userid. This could be an acceptable situation for a session manager to pass a log-on through to TSOT after the user has been authenticated. The ‘security’ group can generate a PassTicket for TSOT for everybody, and the ‘systems’ group can generate PassTickets for everybody who is connected to the ‘systems’ group. Note that there’s no need for the KEY segment, since the profile isn’t used by the PassTicket mechanism itself but only by the generator application.

If the user passes all the tests, the following message is displayed:

```
The PassTicket generated for user JEDSPØØ and VTAM application TSOT is KDYMMJ2G. This
will be valid for 10 minutes and can only be used once.
```

Our operators use the application if they have to generate a PassTicket in an emergency. We therefore took care to generate as many clear messages as possible, so that no time would be lost in debugging. This made the coding rather lengthy.

THE DRIVING REXX PROGRAM (POTEMKIN)

```
/* REXX to display a panel with the appl and userid fields. It then
  tries to calculate a PassTicket.
*/

address "ISPEXEC"

"VGET (APPL USER) PROFILE"
user = strip(user)
appl = strip(appl)

"ADDPOP"
do forever
  "DISPLAY PANEL(PTKNØØØ)"
  if strip(pf3) = 'END' ! appl = '' ! user = '' then leave
  if Get_Ptk() = Ø then leave
   end

"REMPOP"
"VPUT (APPL USER) PROFILE"

exit
/* ------------------------------------------------------------------
Get_Ptk tries to calculate a PassTicket by calling the program
     POTEMKIN. It analyses the return code (see ISPMLIB) and
     returns it to its caller.
-----------------------------------------------------------------*/
Get_Ptk:

upper appl
upper user

l_user = length(user)
l_appl = length(appl)

user = left(user, 8)
appl = left(appl, 8)
parm = l_user user l_appl appl

address "TSO"

x = outtrap('passticket.')
"POTEMKIN" parm
retcode = rc
x = outtrap('OFF')

address "ISPEXEC"

if length(passticket.1) > length('PASSTICKET.1') then do
  safrc   = strip(left(passticket.1, 7), 'L', 'Ø')
  safrc   = safrc !! substr(passticket.1, 8, 1)
  x       = substr(passticket.1, 9, 8)
  racfrc  = strip(left(x, 7), 'L', 'Ø')
  racfrc  = racfrc !! substr(x, 8, 1)
  x       = substr(passticket.1, 17, 8)
  racfrea = strip(left(x, 7), 'L', 'Ø')
  racfrea = racfrea !! substr(passticket.1, 8, 1)
  racrtype = strip(substr(passticket.1, 25, 9))
  if racrtype = 'STAT' then ,
    msgpre = 'The PTKTDATA class or RACF is not active.'
  else msgpre = ''
  end

select
  when retcode =  Ø then do
    pkt = passticket.1
    "SETMSG MSG(PTKNØ2Ø)"
      end
  when retcode =  8 then "SETMSG MSG(PTKNØØØ)"
  when retcode = 12 then "SETMSG MSG(PTKNØØ1)"
  when retcode = 16 then "SETMSG MSG(PTKNØØ2)"
  when retcode = 2Ø then "SETMSG MSG(PTKNØØ3)"
  when retcode = 24 then "SETMSG MSG(PTKNØØ4)"
  when retcode = 28 then "SETMSG MSG(PTKNØØ5)"
  when retcode = 32 then "SETMSG MSG(PTKNØØ6)"
```

```
 when retcode = 36 then "SETMSG MSG(PTKN007)"
 when retcode = 40 then "SETMSG MSG(PTKN008)"
 when retcode = 44 then "SETMSG MSG(PTKN009)"
 when retcode = 48 then do
   intyrc = strip(left(passticket.1, 7), 'L', '0')
   intyrc = intyrc !! substr(passticket.1, 8, 1)
   "SETMSG MSG(PTKN010)"
     end
 when retcode = 52 then "SETMSG MSG(PTKN011)"
 when retcode = 56 then do
    cb = 'ASCB'
   "SETMSG MSG(PTKN012)"
     end
 when retcode = 60 then do
    cb = 'ASXB'
   "SETMSG MSG(PTKN012)"
     end
 when retcode = 64 then do
    cb = 'TCB'
   "SETMSG MSG(PTKN012)"
     end
 when retcode = 68 then "SETMSG MSG(PTKN013)"
 when retcode = 72 then do
    cb = 'ACEE'
   "SETMSG MSG(PTKN012)"
     end
 when retcode = 76 then "SETMSG MSG(PTKN014)"
 when retcode = 80 then "SETMSG MSG(PTKN003)"
 when retcode = 84 then "SETMSG MSG(PTKN015)"
 when retcode = 88 then "SETMSG MSG(PTKN016)"
 when retcode = 92 then do
    cb = 'CVT'
   "SETMSG MSG(PTKN012)"
     end
 when retcode = 96 then do
    cb = 'RCVT'
   "SETMSG MSG(PTKN012)"
     end
 when retcode = 100 then "SETMSG MSG(PTKN017)"
 when retcode = 104 then "SETMSG MSG(PTKN018)"
 when passticket.1 = 'PASSTICKET.1' then "SETMSG MSG(PTKN019)"
 otherwise nop
  end

return(retcode)
```

THE ASSEMBLER COMMAND PROCESSOR (POTEMKIN)

In order to use the PassTicket callable service, a program must run in key zero. This is a problem if you want to use an ISPF interface. What’s more, REXX cannot trap output from a TPUT macro, ISPF services are difficult to use, the REXX ATTACHMVS and LINKMVS environments abend if authorized functions are used, and the generated PassTicket is eight bytes, four bytes too long to be returned in register 15.

We decided to resolve these problems by writing a command processor. This allows the Assembler program to issue PUTLINE messages that can easily be trapped in REXX. An authorized command processor must be defined in the AUTHCMD macro of the IKJTSO*xx* member of the PARMLIB concatenation. A dynamic refresh is possible with the TSO/E PARMLIB UPDATE(*xx*) command. See also *TSO/E System Programming Command Reference* (SC28-1972) and *TSO/E Programming Services* (SC28-1971).

We link-edited the program in the LINKLIST concatenation with AC(1) as reentrant with AMODE 31 and RMODE ANY.

The RACROUTE REQUEST=EXTRACT allocates storage in subpool x'E5' which is private fetch protected storage. If you’re using a FREEMAIN macro, no authorization is required. However, the STORAGE RELEASE macro requires storage key 0 or supervisor mode. See ‘SecureWay Security Server RACF Macros and Interfaces’ (SC28-1914) for information on the fields that can be retrieved.

The Assembler macro M#REGS equates the registers (RF is R15), and I imagine EYECATCH and AMODE31 are obvious.

```
TITLE '*** POTEMKIN: CALCULATE A PASSTICKET         JANX
            DE DECKER ***'
*-------------------------------------------------------------------
*  JAN DE DECKER JED:SP NV            JAN.DE.DECKER@TISCALI.BE
*-------------------------------------------------------------------
*
* NAME:      POTEMKIN TSO/E COMMAND PROCESSOR
*
* PARAMETERS:  R1 -> CPPL
*           CPPLCBUF -> COMMAND BUFFER
*          COMMAND BUFFER: DISP  PARAMETER LENGTH
*                            Ø   L(BUF)     2
*                                2   N/A        2
*                                4   COMMAND    8
*                                D   L(USERID)  1
*                                F   USERID     8
*                               18   L(APPLID)  1
*                               1A   APPLID     8
*
* PURPOSE:    TSO COMMAND PROCESSOR TO CALCULATE A PASSWORD
*
* SYSTEM:     OS/39Ø V2R1Ø
*
* LINK:       AMODE 31
*             RMODE ANY
*             REENTRANT
*             AC(1)
*             LINKLIST (REFRESH)
*
* INSTALL:    LINK-EDIT INTO A LINKLIST LIBRARY AND REFRESH LLA
*             ADD TO IKJTSOXX AS AN AUTHORIZED COMMAND
*             RACF PTKTDATA CLASS ACTIVE AND GENERIC
*
* USE:        SEE POTEMKIN REXX SAMPLE
*
* LOGIC:      CALLED AS A TSO COMMAND WITH FIXED PARAMETERS
*             CHECKS - SEE RETURN CODES
*             IF CALLING USER HAS SPECIAL, CALCULATES PASSTICKET
*             IF CALLING USER IS TARGET USER, CHECKS READ ACCESS
*             OTHERWISE UPDATE. RULES THAT DETERMINE THE RIGHT
*             TO GENERATE A PASSTICKET ARE IN THE PTKTDATA CLASS
*             GENERATE.<APPL>.<USER OR A CONNECT GROUP>
*             IF SAF RETURNS ZERO, CALCULATES AND DISPLAYS
*             A PASSTICKET, OTHERWISE RETURNS
*
* RETURN CODES:   Ø ALL OK                                PTKNØ2Ø
*                 8 NOT APF AUTHORIZED                    PTKNØØØ
*                12 STORAGE OBTAIN FAILURE                PTKNØØ1
*                16 COMMAND BUFFER LENGTH ERROR           PTKNØØ2
*                2Ø PTKTDATA CLASS OR RACF INACTIVE       PTKNØØ3
*                24 LENGTH USERID > 8                     PTKNØØ4
*                28 LENGTH USERID = Ø                     PTKNØØ5
*                32 LENGTH APPL > 8                       PTKNØØ6
*                36 LENGTH APPL = Ø                       PTKNØØ7
*                4Ø USER IS UNDEFINED                     PTKNØØ8
*                44 USER IS REVOKED                       PTKNØØ9
*                48 ICHEINTY ERROR                        PTKNØ1Ø
*                52 ICHEINTY ERROR + PUTLINE ERROR        PTKNØ11
*                56 ASCB EYECATCHER ERROR                 PTKNØ12
*                6Ø ASXB EYECATCHER ERROR                 PTKNØ12
*                64 TCB EYECATCHER ERROR                  PTKNØ12
*                68 NO ACEE FOUND                         PTKNØ13
*             72 ACEE EYECATCHER ERROR                 PTKNØ12
*             76 EMPTY ACEE USERID                     PTKNØ14
*             8Ø RACROUTE ERROR                        PTKNØØ3
*             84 NOT AUTHORIZED TO GENERATE PASSTICKET PTKNØ15
*             88 RACROUTE ERROR + PUTLINE ERROR        PTKNØ16
*             92 CVT EYECATCHER ERROR                  PTKNØ12
*             96 RCVT EYECATCHER ERROR                 PTKNØ12
*            1ØØ PASSTICKET GENERATION ERROR           PTKNØ17
*            1Ø4 PASSTICKET PUTLINE ERROR              PTKNØ18
*
* AUTHOR:     JAN            DATE: Ø6/2ØØ2
*
* SAMPLE:     N/A
*
* MODIFICATION:
*
*-----------------------------------------------------------------
         EJECT
POTEMKIN CSECT
POTEMKIN AMODE 31               31 BIT ADDRESSING
POTEMKIN RMODE ANY              PROGRAM CAN RESIDE ANYWHERE
         M#REGS .               REGISTER EQUATES
         PRINT NOGEN            DONT PRINT MACRO EXPANSIONS
         BAKR  RE,Ø             SAVE REGISTERS
         LR    RC,RF            RC -> START OF POTEMKIN
         USING POTEMKIN,RC      ADDRESS POTEMKIN WITH RC
         LR    RA,R1            KEEP PARAMETER POINTER
         EYECATCH .             NAME, DATE & TIME OF ASSEMBLY
         AMODE31 .              CHANGES RØ AND R1
         EJECT
*
* START OF PROCESSING: CHECK AUTHORIZATION AND OBTAIN WORKING STORAGE
*
         TESTAUTH FCTN=1        ARE WE AUTHORIZED?
         LTR   RF,RF              OK?
         BZ    LØØØØ            YES -> CONTINUE
         LA    RF,8             NO --> SET RC = 8
         PR    .                AND RETURN TO CALLER
LØØØØ    DS    ØH
         STORAGE OBTAIN,          ASK FOR STORAGE              X
               LENGTH=L_WORK,     FOR THIS LENGTH              X
               COND=YES         CONDITIONALLY
         LTR   RF,RF            STORAGE OBTAIN OK?
         BZ    LØØ1Ø            YES -> CONTINUE
         LA    RF,12            NO --> SET RC = 12
         PR    .                AND RETURN TO CALLER
LØØ1Ø    DS    ØH
         LR    RB,R1            RB -> WORK AREA
         LR    R4,R1            R4 -> WORK AREA
         LR    R6,R1            R6 -> WORK AREA
         L     R7,=AL4(L_WORK)        R7 = L(WORK AREA)
         XR    R5,R5                  R5 = Ø
         MVCL  R6,R4                  ZERO OUT WORK AREA
         USING D_WORK,RB              RB ADDRESSES THE WORK AREA
         EJECT
*
* INITIALIZE SOME FIELDS
*
         LA    RD,SAVEAREA            RD -> SAVEAREA
         MVC   SAVEAREA+4(4),=C'F1SA' LINKAGE STACK INDICATOR
         MVC   RET_CODE,=F'Ø'         DEFAULT RETURN CODE = Ø
         MVC   EXT_RA,=F'Ø'           RACROUTE EXTRACT RETURN AREA
         EJECT
*
* ENTIRE PROGRAM: RB ADDRESSES OUR WORKAREA
*                 RC ADDRESSES OUR CSECT
*                 RD -> OUR SAVEAREA
* PREPARE THE MACRO AND I/O PARAMETER LIST ADDRESSES FOR PUTLINE
*
         USING CPPL,RA                RA ADDRESSES THE CPPL
         LA    R2,IOPLADS             R2 -> I/O PLIST ADDRESSES
         USING IOPL,R2                R2 ADDRESSES THE IOPLADS
         MVC   IOPLUPT,CPPLUPT        KEEP @(USER PROFILE TABLE)
         MVC   IOPLECT,CPPLECT        KEEP @(ENV. CONTROL TABLE)
         LA    R3,PUTL_ECB            R3 -> PUTLINE ECB
         ST    R3,IOPLECB             SET @(EVENT CONTROL BLOCK
         DROP  R2                     FORGET THE IOPLADS
         MVC   D_PUTL,S_PUTL          STATIC LIST MACRO TO DYNAMIC
         EJECT
*
* CHECK LENGTH PARAMETER LIST IN COMMAND BUFFER
*
         L     RA,CPPLCBUF            RA -> COMMAND BUFFER
         DROP  RA                     FORGET THE CPPL
         USING P_LIST,RA              RA ADDRESSES THE COMMAND BUFFER
         CLC   L_CMDBUF,=AL2(L_P_LIST) CHECK LENGTH PARAMETER LIST
         BE    LØØ2Ø                  OK -> CONTINUE
         MVC   RET_CODE,=F'16'        SET RETURN CODE
         B     LØ31Ø                  GO RETURN TO CALLER
         EJECT
*
* CHECK IF THE CLASS PTKTDATA IS ACTIVE, IF NOT RETURN
*
LØØ2Ø    DS    ØH
         MVC   D_RACSTA,S_RACSTA      COPY STATIC TO DYNAMIC MACRO
         LA    R2,D_RACAUT            R2 -> RACROUTE MACRO
         USING SAFP,R2                R2 ADDRESSES THE RACF PLIST
         MVC   SAFPRRET,=F'Ø'         PRESET RACF RETURN CODE
         MVC   SAFPRREA,=F'Ø'         PRESET RACF REASON CODE
         DROP  R2                     FORGET THE RACF PLIST
         RACROUTE REQUEST=STAT,      REQUEST RACF INFORMATION     X
               ENTRY=@_CDTENT,       WILL -> CDT ENTRY            X
               RELEASE=2.6,          RACF RELEASE                 X
               WORKA=RACFWORK,       R7 -> RACF WORK AREA         X
               MF=(E,D_RACSTA)       EXECUTE POINTING TO DYNAMIC
         LTR   RF,RF                 CLASS ACTIVE?
         BZ    LØØ3Ø                 YES -> CONTINUE
         ST    RF,X_SAF_RC           KEEP SAF RC
         LA    R2,D_RACEXT           R2 -> RACROUTE PARAMETER LIST
         MVC   RET_CODE,=F'2Ø'       SET RETURN CODE
         MVC   TEXT,BLANKS           BLANK OUT ERROR MESSAGE
         MVC   RACRTYPE,=CL8'STAT'   SET RACROUTE TYPE
         B     LØ26Ø                 GO PRINT SAF AND RACF CODES
         EJECT
*
* THE LENGTH OF THE COMMAND BUFFER IS OK
* RA ADRESSES THE COMMAND BUFFER
* COPY AND CHECK THE PARAMETER LIST
*
LØØ3Ø    DS    ØH
         MVC   L_USERID,L_PARAM1     GET L(USERID)
         NI    L_USERID,X'ØF'        REMOVE FIRST HALF BYTE
         MVC   USERID,PARAM1         GET USERID
         MVC   L_APPLID,L_PARAM2     GET L(APPLID)
         NI    L_APPLID,X'ØF'        REMOVE FIRST HALF BYTE
         MVC   APPLID,PARAM2         GET APPLID
         DROP  RA                    FORGET THE  COMMAND BUFFER
         CLI   L_USERID,X'Ø8'        CHECK MAXIMUM LENGTH
         BNH   LØØ4Ø                 NOT TOO HIGH, CONTINUE
         MVC   RET_CODE,=F'24'       SET RETURN CODE
         B     LØ31Ø                 GO RETURN TO CALLER
LØØ4Ø    DS    ØH                    L(USERID) <= 8
         CLI   L_USERID,X'ØØ'        ZERO ?
         BNE   LØØ5Ø                 NO --> CONTINUE
         MVC   RET_CODE,=F'28'       SET RETURN CODE
         B     LØ31Ø                 GO RETURN TO CALLER
LØØ5Ø    DS    ØH                    Ø < L(USERID) <=8
         NI    L_APPLID,X'ØF'        REMOVE FIRST HALF BYTE
         CLI   L_APPLID,X'Ø8'        CHECK MAXIMUM LENGTH
         BNH   LØØ6Ø                 NO --> CONTINUE
         MVC   RET_CODE,=F'32'       SET RETURN CODE
         B     LØ31Ø                 GO RETURN TO CALLER
LØØ6Ø    DS    ØH                    L(APPLID) <= 8
         CLI   L_APPLID,X'ØØ'        ZERO ?
         BNE   LØØ7Ø                 NO --> CONTINUE
         MVC   RET_CODE,=F'36'       SET RETURN CODE
         B     LØ31Ø                 GO RETURN TO CALLER
LØØ7Ø    DS    ØH                    Ø < L(APPLID) <=8
         OC    USERID,BLANKS         TO UPPER CASE
         OC    APPLID,BLANKS
         EJECT
*
* TEST WHETHER OR NOT THE USER EXISTS AND IS NOT REVOKED
*
         MVC   L_WORK_R,=AL4(L'RACFWORK) LENGTH WORKAREA PREFIX
         MVC   D_INTY,S_INTY        COPY STATIC TO DYNAMIC MACRO
         ICHEINTY LOCATE,           LOCATE A RACF PROFILE          X
               TYPE='USR',          OF A USER ENTITY               X
               ENTRY=USER,          POINTED TO BY R2               X
               WKAREA=L_WORK_R,     USE THIS WORK AREA             X
               TESTS=REVOKED,       DO THE TEST AT LABEL REVOKED   X
               RELEASE=2.6,         RACF RELEASE                   X
               MF=(E,D_INTY)        TARGET IS THE DYNAMIC ICHEINTY
         LTR   RF,RF                LOCATE OK?
         BZ    LØ1ØØ                YES -> CONTINUE
         C     RF,=XL4'C'           RETURN CODE 'C' = UNDEFINED
         BNE   LØØ8Ø                NO --> DEFINED
         MVC   RET_CODE,=F'4Ø'      SET RETURN CODE
         B     LØ31Ø                GO RETURN TO CALLER
LØØ8Ø    DS    ØH                   USER IS DEFINED
         C     RF,=XL4'34'          RETURN CODE '34' = REVOKED
         BNE   LØØ9Ø                NOT UNDEFINED, NOT REVOKED
         MVC   RET_CODE,=F'44'      SET RETURN CODE
         B     LØ31Ø                GO RETURN TO CALLER
         EJECT
*
* WEIRD ICHEINTY RETURN CODE: PRINT ANG GO HOME
*
LØØ9Ø    DS    ØH
         MVC   RET_CODE,=F'48'      SET RETURN CODE
         ST    RF,X_SAF_RC          KEEP ICHEINTY RC
         LA    RF,X_SAF_RC          RF -> ICHEINTY RC
         ST    RF,@_F_HEX           STORE IN PLIST
         LA    RF,S_SAF_RC          RF -> RECEIVE FIELD
         ST    RF,@_S_HEX           STORE IN PLIST
         LA    R1,P_PRHEX           R1 -> PARAMETER LIST
         CALL  F#PRHEX              CALL FUNCTION TO PRINT FULLWORD
*
         MVC   PUTL_ECB,=F'Ø'       ECB = Ø
         MVC   TEXTADS,=H'12'       L(OUTPUT LINE)
         PUTLINE PARM=D_PUTL,       PUTLINE LIST FORMAT            X
               OUTPUT=(TEXTADS,TERM,SINGLE,DATA),                  X
               MF=(E,IOPLADS)       PARAMETER LIST
         LTR   RF,RF                PUTLINE OK?
         BZ    LØ31Ø                GO RETURN TO CALLER
         MVC   RET_CODE,=F'52'      SET RETURN CODE
         B     LØ31Ø                GO RETURN TO CALLER
         EJECT
*
* COMPARE THE PASSED USERID WITH THE USERID THAT ISSUED THE COMMAND
* IF THE USERIDS ARE THE SAME WE WILL CHECK READ ACCESS ON THE
* GENERATE.<APPLICATION>.<USER OR CONNECT GROUP> RESOURCE
* IN THE PTKTDATA CLASS, OTHERWISE WE WILL CHECK UPDATE.
* THE AUTHORITY REQUEST LEVEL WILL BE KEPT IN R6
*
LØ1ØØ    DS    ØH
         XR    R6,R6                R6 = Ø
         LA    R6,2                 DEFAULT ATTR IS READ
         XR    R2,R2                R2 = Ø
         USING PSA,R2               R2 ADDRESSES THE PSA
         L     R2,PSAAOLD           R2 -> ASCB
         L     R3,PSATOLD           R3 -> TCB
         DROP  R2                   FORGET THE PSA
         USING ASCB,R2              R2 ADDRESSES THE ASCB
         CLC   =C'ASCB',ASCBASCB    EYECATCHER OK?
         BE    LØ11Ø                YES -> CONTINUE
         MVC   RET_CODE,=F'56'      SET RETURN CODE
         B     LØ31Ø                GO RETURN TO CALLER
LØ11Ø    DS    ØH                   R2 -> ASCB
         L     R2,ASCBASXB          R2 -> ASCB EXTENSION
         DROP  R2                   FORGET THE ASCB
         USING ASXB,R2              R2 ADDRESSES THE ASCB EXTENSION
         CLC   =C'ASXB',ASXBASXB    EYECATCHER OK?
         BE    LØ12Ø                YES -> CONTINUE
         MVC   RET_CODE,=F'6Ø'      SET RETURN CODE
         B     LØ31Ø                GO RETURN TO CALLER
LØ12Ø    DS    ØH                   R2 -> ASXB
         L     R2,ASXBSENV          R2 -> ACEE
         DROP  R2                   FORGET THE ASCB EXTENSION
         LTR   R2,R2                IS THERE A POINTER?
         BNZ   LØ16Ø                YES -> CONTINUE
LØ13Ø    DS    ØH                   NO --> CHECK TCB FOR POINTER
         LTR   R3,R3                SRB MODE? (PROBABLY YES)
         BZ    LØ15Ø                YES -> NO ACEE POINTER FOUND
         USING TCB,R3               R3 ADDRESSES THE TCB
         CLC   =C'TCB',TCBTCBID     EYECATCHER OK?
         BE    LØ14Ø                YES -> CONTINUE
         MVC   RET_CODE,=F'64'      SET RETURN CODE
         B     LØ31Ø                GO RETURN TO CALLER
LØ14Ø    DS    ØH                   R2 -> TCB
         L     R2,TCBSENV           R2 -> ACEE
         DROP  R3                   FORGET THE TCB
         LTR   R2,R2                IS THERE A POINTER
         BNZ   LØ16Ø                YES -> CONTINUE
LØ15Ø    DS    ØH                   NO ACEE COULD BE LOCATED
         MVC   RET_CODE,=F'68'      SET RETURN CODE
         B     LØ31Ø                GO RETURN TO CALLER
LØ16Ø    DS    ØH                   R2 -> ACEE
         USING ACEE,R2              R2 ADDRESSES THE ACEE
         CLC   =C'ACEE',ACEEACEE    EYECATCHER OK?
         BE    LØ17Ø               YES -> CONTINUE
         MVC   RET_CODE,=F'72'     SET RETURN CODE
         B     LØ31Ø               GO RETURN TO CALLER
LØ17Ø    DS    ØH                  R2 -> ACEE
         CLC   ACEEUSRI,=CL8' '    BLANK USERID?
         BNZ   LØ18Ø               NO --> CONTINUE
         MVC   RET_CODE,=F'76'     SET RETURN CODE
         B     LØ31Ø               GO RETURN TO CALLER
LØ18Ø    DS    ØH                  NON-BLANK USERID IN THE ACEE
         CLC   USERID,ACEEUSRI     ACEE USERID = PASSED USERID
         BE    LØ19Ø               YES -> LEAVE READ ATTR
         LA    R6,4                NO --> SET ATTR TO UPDATE
         EJECT
*
* HAS THE COMMAND ISSUING USER THE RACF ATTRIBUTE SPECIAL?
*
LØ19Ø    DS    ØH
         TM    ACEEFLG1,ACEESPEC   CALLING USER HAS SPECIAL?
*        BO    LØ27Ø               YES -> CALCULATE PASSTICKET
         DROP  R2                  FORGET THE ACEE
         EJECT
*
* GET ALL THE GROUPS THE TARGET USER IS CONNECTED TO
*
         MVC   L_ENT_B,=H'8'       LENGTH ENTITYX BUFFER
         MVC   L_ENT_P,=H'Ø'       LENGTH ENTITYX PROFILE NAME
         MVC   ENT_PROF,BLANKS     BLANK OUT PROFILE
         XR    R3,R3               R3 = Ø
         IC    R3,L_USERID         R3 = L(USERID)
         BCTR  R3,Ø                --R3 (FOR EX)
         LA    R4,ENT_PROF         R5 -> PROFILE ENTITYX
         LA    R5,USERID           R4 -> USER ID
         EX    R3,MVC1             MOVE IN APPLICATION ID
*
         MVC   D_RACEXT,S_RACEXT   COPY STATIC TO DYNAMIC MACRO
         LA    R2,D_RACEXT         R2 -> RACROUTE MACRO
         USING SAFP,R2             R2 ADDRESSES THE RACF PLIST
         MVC   SAFPRRET,=F'Ø'      PRESET RACF RETURN CODE
         MVC   SAFPRREA,=F'Ø'      PRESET RACF REASON CODE
         DROP  R2                  FORGET THE RACF PLIST
         RACROUTE REQUEST=EXTRACT,    EXTRACT MACRO TYPE               X
               TYPE=EXTRACT,          REQUIRE AN EXTRACT               X
               ENTITYX=ENTITYX,       R5 -> ENTITY FIELD               X
               FIELDS=EXFIELDS,       FIELDS TO EXTRACT                X
               WORKA=RACFWORK,        R7 -> RACF WORK AREA             X
               RELEASE=2.6,           RELEASE                          X
               MF=(E,D_RACEXT)     EXECUTE POINTING TO DYNAMIC
         LR    R2,R1               R2 -> RESULT AREA
         LTR   RF,RF               RACROUTE OK?
         BZ    LØ2ØØ               YES -> CONTINUE
         EJECT
*
* ERROR EXTRACTING THE CONNECT GROUPS OF THIS USER
*
         ST    RF,X_SAF_RC          KEEP SAF RC
         LA    R2,D_RACEXT          R2 -> RACROUTE PARAMETER LIST
         MVC   RET_CODE,=F'8Ø'      SET RETURN CODE
         MVC   TEXT,BLANKS          BLANK OUT ERROR MESSAGE
         MVC   RACRTYPE,=CL8'EXTRACT' SET RACROUTE TYPE
         B     LØ26Ø                GO PRINT SAF AND RACF CODES
         EJECT
*
* CONNECT GROUPS EXTRACT OK
*
LØ2ØØ    DS    ØH
         ST    R2,EXT_RA            KEEP ADDRESS RESULT AREA
         USING EXTWKEA,R2           R2 ADDRESSES FIXED RESULT AREA
         AH    R2,EXTWOFF           R2 -> VARIABLE RESULT AREA
         DROP  R2                   FORGET THE FIXED RESULT AREA
*
* R2 -> +Ø   L(CONGRPCT DATA) = 4   L(CONNECT GROUP COUNT)
*       +4   CONGRPCT DATA          CONNECT GROUP COUNT
*       +8   L(CONGRPNM)            L(ALL CONNECT GROUPS)
*       +C   L(CONGRPNM 1)   = 8    L(FIRST CONNECT GROUP)
*       +1Ø   CONGRPNM 1            FIRST CONNECT GROUP
*     +1Ø +L(CONGRPNM 1) L(CONGRPNM 2)
*
* R8 IS THE LOOP COUNTER : ALL THE CONNECTED GROUPS
* R9 -> CONNECTED GROUP ENTITY
*
         L     R8,4(R2)             R8 = #(GROUPS)
         LA    R8,1(R8)              + USERID
         LA    R9,12(R2)            R9 -> FIRST GROUP DESCRIPTION
*
* PREPARE THE RACROUTE ON GENERATE.<APPL>.<USERID>
*              OR GENERATE.<APPL>.<GROUP> IN CLASS PTKTDATA
*
         MVC   L_ENT_B,=H'26'       MAXIMUM BUFFER LENGTH
         MVC   L_ENT_P,=H'Ø'        PROFILE LENGTH
*
LØ21Ø    DS    ØH
         MVC   ENT_PROF,BLANKS      BLANK OUT PROFILE
         MVC   ENT_PRE,=CL9'GENERATE.' MOVE IN PREFIX
         XR    R3,R3                R3 = Ø
         IC    R3,L_APPLID          R3 = L(APPLID)
         BCTR  R3,Ø                 --R3 (FOR EX)
         LA    R4,ENT_REM           R4 -> ENTITY PROFILE POSTFIX
         LA    R5,APPLID            R5 -> APPLICATION ID
         EX    R3,MVC1              MOVE IN APPLICATION ID
         AR    R4,R3                R4 -> LAST BYTE APPLICATION ID
         LA    R4,1(R4)            R4 -> AFTER APPLICATION ID
         MVI   Ø(R4),C'.'          INTER QUALIFIER
         LA    R4,1(R4)            R4 -> AFTER '.'
         XR    R3,R3               R3 = Ø
         IC    R3,L_USERID         R3 = L(USERID)
         BCTR  R3,Ø                --R3 (FOR EX)
         LA    R5,USERID           R5 -> USER ID
         EX    R3,MVC1             MOVE IN USERID
LØ22Ø    DS    ØH
         MVC   D_RACAUT,S_RACAUT   COPY STATIC TO DYNAMIC MACRO
         LA    R2,D_RACAUT         R2 -> RACROUTE MACRO
         USING SAFP,R2             R2 ADDRESSES THE RACF PLIST
         MVC   SAFPRRET,=F'Ø'      PRESET RACF RETURN CODE
         MVC   SAFPRREA,=F'Ø'      PRESET RACF REASON CODE
         DROP  R2                  FORGET THE RACF PLIST
         EJECT
*
* CALL RACF IN CLASS PTKTDATA WITH PROFILE GENERATE.<APPLID>.<USERID>
*
         RACROUTE REQUEST=AUTH,      REQUEST AUTHORITY           X
               ATTR=(R6),            ACCESS REQUIRED             X
               ENTITYX=(ENTITYX,NONE), FOR THIS PROFILE          X
               RELEASE=2.6,          RACF RELEASE                X
               WORKA=RACFWORK,       WORKAREA                    X
               MF=(E,D_RACAUT)     EXECUTE FORMAT, TARGET IN R3
         LTR   RF,RF               SAF OK?
         BZ    LØ27Ø               YES -> CONTINUE
         ST    RF,X_SAF_RC         KEEP SAF RC
         C     RF,=F'8'            SAF CODE > 8?
         BH    LØ25Ø               NO --> GO PRINT SAF RACF CODES
         EJECT
*
* LOOP FOR GROUPS
* WE ASSUME THAT THE LENGTH OF A GROUP NAME IS ALWAYS 8
*
         LTR   R8,R8               COUNTER Ø?
         BZ    LØ24Ø               YES -> STOP
         BCT   R8,LØ23Ø            LOOP FOR ALL GROUPS
         B     LØ24Ø
LØ23Ø    DS    ØH
         MVC   Ø(8,R4),4(R9)       GET NEW GROUP
         LA    R9,12(R9)           R9 -> NEXT GROUP
         B     LØ22Ø
         EJECT
*
* NO AUTHORIZATION TO CALCULATE A PASSTICKET
*
LØ24Ø    DS    ØH
         MVC   RET_CODE,=F'84'     SET RETURN CODE
         B     LØ31Ø               GO RETURN TO CALLER
         EJECT
*
* SAF RETURN CODE > 4 FROM RACROUTE REQUEST = AUTH
*
LØ25Ø    DS    ØH
         MVC   RET_CODE,=F'8Ø'         SET RETURN CODE
         MVC   TEXT,BLANKS             BLANK OUT ERROR MESSAGE
         MVC   RACRTYPE,=CL8'AUTH'     SET RACROUTE TYPE
         LA    R2,D_RACAUT             R2 -> RACROUTE PARAMETER LIST
         EJECT
*
* WEIRD. RETURN ERROR SAF_RC RACF_RC RACF_REASON RACROUTE_TYPE
* R2 IS EXPECTED TO POINT TO THE RACROUTE PARAMETER LIST
*
LØ26Ø    DS    ØH
         USING SAFP,R2                 R2 ADDRESSES THE RACF PLIST
         LA    RF,X_SAF_RC             RF -> SAF RETURN CODE
         ST    RF,@_F_HEX              STORE IN PLIST
         LA    RF,S_SAF_RC             RF -> RECEIVE FIELD
         ST    RF,@_S_HEX              STORE IN PLIST
         LA    R1,P_PRHEX              R1 -> PARAMETER LIST
         CALL  F#PRHEX                 CALL FUNCTION TO PRINT FULLWORD
*
         LA    RF,SAFPRRET             RF -> RACF RETURN CODE
         ST    RF,@_F_HEX              STORE IN PLIST
         LA    RF,RACF_RC              RF -> RECEIVE FIELD
         ST    RF,@_S_HEX              STORE IN PLIST
         LA    R1,P_PRHEX              R1 -> PARAMETER LIST
         CALL  F#PRHEX                 CALL FUNCTION TO PRINT FULLWORD
*
         LA    RF,SAFPRREA             RF -> RACF REASON CODE
         ST    RF,@_F_HEX              STORE IN PLIST
         LA    RF,RACF_REA             RF -> RECEIVE FIELD
         ST    RF,@_S_HEX              STORE IN PLIST
         LA    R1,P_PRHEX              R1 -> PARAMETER LIST
         CALL  F#PRHEX                 CALL FUNCTION TO PRINT FULLWORD
         DROP  R2                      FORGET THE SAF PLIST
*
         MVC   PUTL_ECB,=F'Ø'          ECB = Ø
         PUTLINE PARM=D_PUTL,          PUTLINE LIST FORMAT             X
               OUTPUT=(TEXTADS,TERM,SINGLE,DATA),                      X
               MF=(E,IOPLADS)          PARAMETER LIST
         LTR   RF,RF                   PUTLINE OK?
         BZ    LØ31Ø                   GO RETURN TO CALLER
         MVC   RET_CODE,=F'88'         SET RETURN CODE
         B     LØ31Ø                   GO RETURN TO CALLER
         EJECT
*
* CALCULATE THE PASSTICKET
*
```

```
LØ27Ø    DS    ØH
         XR    R2,R2                R2 = Ø
         USING PSA,R2               R2 ADDRESSES THE PSA
         L     R2,FLCCVT            R2 -> CVT
         DROP  R2                   FORGET THE PSA
         USING CVTMAP,R2            R2 ADDRESSES THE CVT
         CLC   =C' CVT',CVTCVT      EYECATCHER OK?
         BE    LØ28Ø                YES -> CONTINUE
         MVC   RET_CODE,=F'92'      SET RETURN CODE
         B     LØ31Ø                GO RETURN TO CALLER
LØ28Ø    DS    ØH                   R2 -> CVT
         L     R2,CVTRAC            R2 -> RCVT
         DROP  R2                   FORGET THE CVT
         USING RCVT,R2              R2 ADDRESSES THE RCVT
         CLC   =C'RCVT',RCVTID      EYECATCHER OK?
         BE    LØ29Ø                YES -> CONTINUE
         MVC   RET_CODE,=F'96'      SET RETURN CODE
         B     LØ31Ø                GO RETURN TO CALLER
LØ29Ø    DS    ØH                   R2 -> RCVT
         LA    RA,USER              RA -> USER FIELDS
         ST    RA,P_USER            STORE IN PLIST PTC
         LA    RA,APPL              RA -> APPL FIELDS
         ST    RA,P_APPL            STORE IN PLIST PTC
         MODESET KEY=ZERO           STORAGE KEY ZERO
         LA    R1,P_PTC             R1 -> PLIST PASSTICKET CALC.
         L     RF,RCVTPTGN          RF -> PASSTICKET GENERATOR
         BALR  RE,RF                CALCULATE PASSTICKET
         DROP  R2                   FORGET THE RCVT
         LR    RA,RF                KEEP RETURN CODE
         STM   RØ,R1,TEXT           KEEP PASSTICKET
         MODESET KEY=NZERO          TCB STORAGE KEY
         LTR   RA,RA                PASSTICKET SERVICE CALL OK?
         BZ    LØ3ØØ                YES -> CONTINUE
         MVC   RET_CODE,=F'1ØØ'     SET RETURN CODE
         B     LØ31Ø                GO RETURN TO CALLER
         EJECT
*
* WRITE THE PASSTICKET TO THE TERMINAL USING PUTLINE
*
LØ3ØØ    DS    ØH
         MVC   PUTL_ECB,=F'Ø'       ECB = Ø
         MVC   TEXTADS,=H'12'       L(OUTPUT LINE)
         PUTLINE PARM=D_PUTL,       PUTLINE LIST FORMAT             X
               OUTPUT=(TEXTADS,TERM,SINGLE,DATA),                   X
               MF=(E,IOPLADS)       PARAMETER LIST
         LTR   RF,RF                PUTLINE OK?
         BZ    LØ31Ø                YES -> CONTINUE
         MVC   RET_CODE,=F'1Ø4'     SET RETURN CODE
         B     LØ31Ø                GO RETURN TO CALLER
         EJECT
```

```
*
* CLEAN UP THE ENVIRONMENT: STORAGE OBTAINED BY RACF EXTRACT
*                           STORAGE OBTAINED FOR OUR WORK AREA
*
LØ31Ø    DS   ØH
         L    RA,EXT_RA              RA -> RACROUTE EXTRACT RETURN
         LTR  RA,RA                  IS THERE ONE?
         BZ   LØ32Ø                  NO --> CONTINUE, DON'T RELEASE
         USING EXTWKEA,RA            RA ADDRESSES THE RETURN AREA
         XR   R8,R8                  R8 = Ø
         IC   R8,EXTWSP              R8 = SUBPOOL ALLOCATED BY RACF
         XR   R9,R9                  R9 = Ø
         ICM  R9,B'Ø111',EXTWLN      R9 = L(EXTRACT RETURN AREA)
         DROP RA                     FORGET THE RETURN AREA
         FREEMAIN R,                 FREE UP THE RETURN AREA      X
               SP=(R8),              SUBPOOL NUMBER               X
               LV=(R9),              FOR THE GIVEN LENGTH         X
               A=(RA)                FROM THIS ADDRESS
LØ32Ø    DS   ØH                     WORK AREA CLEAN-UP
         L    RA,RET_CODE            RA = RETURN CODE
         STORAGE RELEASE,            FREE UP THE WORK AREA        X
               LENGTH=L_WORK,        FOR THE GIVEN LENGTH         X
               ADDR=(RB)             FROM THIS ADDRESS
         DROP RB                     FORGET OUR WORK AREA
         EJECT
*
* END OF PROCESSING
*
THE_END  DS   ØH                     MY ONLY FRIEND, THE END
         LR   RF,RA                  LOAD RETURN CODE
         PR   .                      RETURN TO CALLER
         EJECT
*
* PR_HEX FUNCTION EXPECTS R5 TO POINT TO A FULL WORD AND R6 TO A CL8
* STRING. THE FULWORD WILL BE PRINTED IN HEX FORMAT IN THE STRING
*
         BAKR RE,Ø                   SAVE REGISTERS
         PR   .                      RETURN TO CALLER
         EJECT
*
* EXECUTE TARGETS, VARIABLES AND CONSTANTS
*
MVC1     MVC  Ø(Ø,R4),Ø(R5)          MOVE LAST QUALIFIER
BLANKS   DC   133C' '
PTKTDATA DS   ØF                     CLASS ENTITY RACROUTE REQ=AUTH
         DC   X'8'                   SINGLE BYTE LENGTH
         DC   CL8'PTKTDATA'          CLASS NAME
BITØ     DC   X'8Ø'                  BIT IN ICHETEST MACRO
EXFIELDS DS   ØF                     FIELDS FOR RACROUTE REQ=EXTRACT
         DC   A(2)                   2 FIELDS
         DC    CL8'CONGRPCT'           GROUP COUNT
         DC    CL8'CONGRPNM'           GROUP NAMES
         EJECT
*
* MACRO'S IN LIST FORMAT
*
S_PUTL   PUTLINE OUTPUT=(,TERM,SINGLE,DATA),                           X
               MF=L
L_S_PUTL EQU   *-S_PUTL                L(STATIC PUTLINE)
*
S_RACAUT RACROUTE REQUEST=AUTH,        REQUEST AUTHORITY               X
               ATTR=READ,              FOR READ ACCESS                 X
               CLASS=PTKTDATA,         IN THIS CLASS                   X
               ENTITYX=,               FOR THIS PROFILE                X
               LOG=NOSTAT,             NO LOGGING                      X
               RELEASE=2.6,            RACF RELEASE                    X
               MF=L                    LIST FORMAT
L_RACAUT EQU   *-S_RACAUT              LENGTH OF RACROUTE MACRO
*
S_RACEXT RACROUTE REQUEST=EXTRACT,     EXTRACT MACRO TYPE              X
               TYPE=EXTRACT,           REQUIRE AN EXTRACT              X
               CLASS='USER',           USER PROFILE                    X
               ENTITYX=,               ENTITY FIELD                    X
               FIELDS,                 FIELDS TO EXTRACT               X
               WORKA=,                 RACF WORK AREA                  X
               RELEASE=2.6,            RELEASE                         X
               MF=L                    LIST_FORMAT
L_RACEXT EQU   *-S_RACEXT              LENGTH OF RACROUTE EXTRACT MACRO
*
S_RACSTA RACROUTE REQUEST=STAT,        REQUEST RACF INFORMATION        X
               CLASS='PTKTDATA',       CLASS NAME                      X
               ENTRY=,                 RETURN ADDRESS -> CDT ENTRY     X
               RELEASE=2.6,            RACF RELEASE                    X
               WORKA=,                 RACF WORK AREA                  X
               MF=L                    LIST_FORMAT
L_RACSTA EQU   *-S_RACSTA              LENGTH OF RACROUTE STAT MACRO
*
S_INTY   ICHEINTY LOCATE,              LOCATE A PROFILE                X
               TYPE='USR',             OF A USER ENTITY                X
               ENTRY=,                 POINTED TO BY R2                X
               WKAREA=,                USE THIS WORK AREA              X
               RELEASE=2.6,            RACF RELEASE                    X
               TESTS=,                 DO THE TEST AT LABEL REVOKED    X
               MF=L
L_INTY   EQU   *-S_INTY                LENGTH OF ICHEINTY MACRO
*
REVOKED  ICHETEST FIELD=FLAG4,                                         X
               FLDATA=(1,BITØ),                                        X
               COND=ZEROS
         EJECT
```

```
*
* LITERAL POOL
*
         LTORG
         EJECT
*
* EQUATES
*
        PRINT NOGEN
*
* DYNAMIC WORK AREA DSECT
*
D_WORK  DSECT
SAVEAREA DS    18F              SAVE AREA
RET_CODE DS    F                RETURN CODE
X_SAF_RC DS    F                SAF RC
EXT_RA   DS    F                RACROUTE EXTRACT RETURN AREA
@_UPT    DS    F                KEEPS @(UPT)
@_ECT    DS    F                KEEPS @(ECT)
@_CDTENT DS    F                WILL KEEP @(CDT ENTRY PTKTDATA)
*
* PARAMETERS FOR THE PUTLINE MACRO
*
TEXTADS  DS    H                TEXTBUFFER LENGTH
         DS    H                RESERVED
TEXT     DS    ØCL32            PASSTICKET OR RC'S
S_SAF_RC DS    CL8              SAF RC STRING
RACF_RC  DS    CL8              RACF RC STRING
RACF_REA DS    CL8              RACF REASON CODE STRING
RACRTYPE DS    CL8              RACROUTE REQUEST=
*
IOPLADS  DS    4F               PUTLINE PARAMETER LIST
PUTL_ECB DS    F                PUTLINE ECB
D_PUTL   DS   XL(L_S_PUTL)      PUTLINE DYNAMIC FORMAT
         EJECT
*
* DYNAMIC RACF FIELDS
*
ENTITYX  DS    ØF               RACROUTE ENTITYX
L_ENT_B  DS    H                BUFFER LENGTH
L_ENT_P  DS    H                PROFILE LENGTH IF KNOWN
ENT_PROF DS    ØCL26            PROFILE
ENT_PRE  DS    CL9              'GENERATE.'
ENT_REM  DS    CL17             REMAINDER OF PROFILE
*
D_RACAUT DS   XL(L_RACAUT)      DYNAMIC RACROUTE REQUEST=AUTH
D_RACEXT DS   XL(L_RACEXT)      DYNAMIC RACROUTE REQUEST=EXTRACT
D_RACSTA DS   XL(L_RACSTA)      DYNAMIC RACROUTE REQUEST=STAT
D_INTY   DS   XL(L_INTY)        DYNAMIC ICHEINTY
*
```

```
* PARAMETERS FOR THE PASSTICKET CALCULATOR SERVICE ROUTINE
*
P_PTC    DS   ØF                PLIST PASSTICKET CALCULATOR
P_USER   DS   F                   -> USER FIELDS
P_APPL   DS   F                  -> APPLICATION FIELDS
*
USER     DS   ØH                KEEP INFORMATION TOGETHER
L_USERID DS   X                   L(USERID)
USERID   DS   CL8               USERID PADDED WITH BLANKS
APPL     DS   ØH                KEEP INFORMATION TOGETHER
L_APPLID DS   X                   L(APPLID)
APPLID   DS   CL8               APPLID PADDED WITH BLANKS
*
* PARAMETERS FOR THE PRINT A HEXADECIMAL NUMBER FUNCTION
*
P_PRHEX  DS   ØF                START OF PARAMETER LIST
@_F_HEX  DS   F                   -> FULLWORD
@_S_HEX  DS   F                  -> PRINTABLE STRING
*
* LAST ADDRESSABLE LABEL
*
L_WORK_R DS   F                 L(RACF WORK AREA) - ICHEINTY
RACFWORK DS   XL4Ø96              RACF WORK AREA
*
* END OF WORK AREA
*
L_WORK   EQU  *-D_WORK          LENGTH OF THE WORKAREA
         EJECT
*
* PARAMETER LIST: THIS IS THE TSO/E COMMAND BUFFER
*           SINCE THE PARAMETER LIST IS CONSTRUCTED BY THE
*           CALLING REXX, WE DON'T TEST TOO MUCH IN THE PROGRAM
*
P_LIST   DSECT .                 PARAMETER LIST
L_CMDBUF DS   H                 L(COMMAND BUFFER)
         DS   H                SEE TSO/E PROGRAMMING SERVICES
COMMAND  DS   CL8                 OUR TMP
         DS   X                   BLANK
L_PARAM1 DS   X                   L(USERID)
         DS   X                   BLANK
PARAM1   DS   CL8                 USERID
         DS   X                   BLANK
L_PARAM2 DS   X                   L(APPL)
         DS   X                   BLANK
PARAM2   DS   CL8                 APPLID
L_P_LIST EQU  *-P_LIST          L(PARAMETER LIST)
         EJECT
*
* SYSTEM DSECTS
*
```

```
         PRINT GEN
     CVT  DSECT=YES,PREFIX=YES   COMMUNICATION VECTOR TABLE
         EJECT
       IHAPSA DSECT=YES          PREFIXED STORAGE AREA
         EJECT
        ICHSAFP .                RACROUTE PARAMETER LIST
         EJECT
         ICHPRCVT .                RACF CVT
         EJECT
        IKJCPPL .                COMMAND PROCESSOR PLIST
         EJECT
        IKJIOPL .                I/O PARAMETER LIST ADDRESSES
         EJECT
        IHAASCB .                ADDRESS SPACE CONTROL BLOCK
         EJECT
         IHAASXB .                 ASCB EXTENSION
         EJECT
        IHAACEE .                ACCESS CONTROL ENVIRONMENT EL.
         EJECT
         IKJTCB .                  TASK CONTROL BLOCK
         EJECT
        IRRPRXTW .               RACROUTE EXTRACT RESULT AREA
         END
         EJECT
*-----------------------------------------------------------------
     TITLE '*** JED:SP REPORTS: PRINT A HEXADECIMAL NUMBER   JANX
               DE DECKER ***'
*-----------------------------------------------------------------
* JED:SP                                 JAN.DE.DECKER@TISCALI.BE
*-----------------------------------------------------------------
*
* NAME:       F#PRHEX
*
* PURPOSE:    PRINT A HEXADECIMAL NUMBER
*
*
* PARAMETERS:  R1 ->  @(FULLWORD)
*                   @(PRINT FULLWORD)
*
* LINK:       CAN BE LINKED REENTRANT
*
* SYSTEM:     OS/39Ø V2R1Ø
*
* MODIFICATION:
*
*-----------------------------------------------------------------
F#PRHEX CSECT
F#PRHEX  AMODE 31                31 BIT ADDRESSING
F#PRHEX  RMODE ANY               PROGRAM CAN RESIDE ANYWHERE
         M#REGS .                 REGISTER EQUATES
         EJECT
         PRINT NOGEN              DONT PRINT MACRO EXPANSIONS
         BAKR  RE,Ø                SAVE REGISTERS
         LR    RC,RF               LOAD BASE REGISTER
         USING F#PRHEX,RC          RC IS BASE REGISTER
         LR    RA,R1               RA -> PARAMETER LIST
         EYECATCH .                EYECATCHER
         AMODE31 .                 SWITCH TO AMODE 31
         EJECT
*
* START PROCESSING
*
         LM    R5,R6,Ø(RA)         R5 -> FULLWORD
*                                  R6 -> PRINTABLE FULLWORD
         XR    R2,R2               R2 = Ø
         LA    R4,4                R4 = 4
LØØØØ    DS    ØH
         IC    R2,Ø(R5)            LOAD BYTE INTO R2
         SRL   R2,4                SHIFT 4 BITS TO THE RIGHT
         LA    R3,2                R3 = 1
LØØ1Ø    DS    ØH
         CH    R2,=H'1Ø'           R2 >= X'A' ?
         BL    LØØ2Ø               NO --> GO ADD FØ
         AH    R2,=H'183'          YES -> ADD CØ
         B     LØØ3Ø               NEXT HALFBYTE
LØØ2Ø    DS    ØH
         AH    R2,=H'24Ø'          ADD X'FØ'
LØØ3Ø    DS    ØH
         STC   R2,Ø(R6)            STORE IN RECEIVE FIELD
         LA    R6,1(R6)            POINT TO NEXT BYTE IN RECEIVE F
         IC    R2,Ø(R5)            TAKE THE SAME BYTE
         N     R2,=X'ØØØØØØØF'     MAKE DISAPPEAR THE FIRST HALFB.
         BCT   R3,LØØ1Ø            AND JUMP 1 TIME TO LØØ1Ø
         LA    R5,1(R5)            POINT TO NEXT FULLWORD BYTE
         BCT   R4,LØØØØ            JUMP 3 TIMES
         EJECT
*
* END OF PROCESSING
*
THE_END  DS    ØH                  MY ONLY FRIEND, THE END
         XR    RF,RF               RC = Ø
         PR    .                   RETURN TO CALLER
         EJECT
*
* LITERAL POOL
*
         LTORG
*
         END
```

THE ISPF PANEL (PTKN000)

```
)ATTR DEFAULT(]#{)
£  TYPE(PT)
]  TYPE(NT)
{  TYPE(NEF) PADC(USER)
}  TYPE(RP)
)BODY EXPAND($$) WINDOW(43,11)
£-$-$-
£-$-$-<Passticket Generator>-$-$-
£-$-$-
£
]  VTAM Application   ===> {APPL     ]
]  User Identification ===> {USER     ]
]
]
}  PF3 to cancel
}  Enter to calculate
]
)INIT
 .CURSOR = APPL
)PROC
 &PF3 = .RESP
)END
```

THE ISPF MESSAGES

PTKN00 member

```
PTKN000 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                                  +
'The POTEMKIN program is not authorized. Check APF authorization,' +
' and the settings of IKJTSOxx. POTEMKIN must be in the AUTHCMD'   +
' list to be invoked from ISPF.'

PTKN001 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                                  +
'The STORAGE OBTAIN SVC failed.'

PTKN002 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                                  +
'The command buffer passed to the generator has the wrong length.'

PTKN003 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                                  +
'&MSGPRE The RACROUTE'                                             +
' REQUEST=STAT macro returned &SAFRC with a RACF return code'      +
' &RACFRC and a RACF reason code &RACFREA..'
```

```
PTKNØØ4 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                           +
'The passed user identification is longer than 8 characters.'

PTKNØØ5 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                           +
'The passed user identification has a zero length.'

PTKNØØ6 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                           +
'The passed VTAM application name is longer than 8 characters.'

PTKNØØ7 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                           +
'The passed VTAM application name has a zero length.'

PTKNØØ8 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                           +
'The userid &USER is unknown to RACF.'

PTKNØØ9 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                           +
'The userid &USER is REVOKED.'
```

PTKN01 member

```
PTKNØ1Ø .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                             +
'The ICHEINTY macro returned unexpectedly (hex): &INTYRC..'

PTKNØ11 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                             +
'The ICHEINTY macro returned an unexpected return code but'   +
' unfortunately also the PUTLINE macro used to return this to' +
' the driving REXX program failed.'

PTKNØ12 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                             +
'There was an error in the eyecatcher of the control block: &CB..'

PTKNØ13 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                             +
'An ACEE could not be located.'

PTKNØ14 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                             +
'The ACEE userid contained only blanks.'

PTKNØ15 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                             +
'You are not authorized by the RACF profiles GENERATE.&APPL'       +
' postfixed by &USER or a CONNECT GROUP in the class PTKTDATA'     +
' to generate a PassTicket for the user &USER and'                 +
' the VTAM application &APPL..'

PTKNØ16 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                                  +
'There was an non-zero return code from a RACROUTE macro.'         +
' Unfortunately there was also an error while trying to'           +
' print the SAF and RACF return and reason codes using '           +
' the TSO/E PUTLINE macro.'

PTKNØ17 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code 56: '                                        +
'The Passticket generation failed for user &USER and VTAM'         +
' application &APPL..'

PTKNØ18 .ALARM=YES .TYPE=ACTION .WINDOW=LR
'POTEMKIN Return code &RETCODE: '                                  +
'The PassTicket generation was a success but'                      +
' unfortunately there was also an error while trying to'           +
' print it using the TSO/E PUTLINE macro.'

PTKNØ19 .ALARM=YES .TYPE=ACTION  .WINDOW=LR
'Unknown error. Please contact your systems programmer.'           +
' System programmers action: correct the error and rerun the job.'
```

PTKN02 Member

```
PTKNØ2Ø .ALARM=YES .TYPE=WARNING .WINDOW=LR
'The PassTicket generated for user &USER and VTAM application &APPL ' +
'is &PKT.. This will be valid for 1Ø minutes and can only be used '   +
'once.'
```

Jan De Decker
Senior Systems Engineer JED:SP NV
(Belgium)

How RACF handles passwords

This article discusses the issues surrounding password authentication. It starts by reviewing the principles behind authentication and encryption, and then develops this theme in order to arrive at a clearer understanding of how RACF handles passwords. Finally, it compares RACF and Windows NT, to bring out the main cross-platform issues associated with synchronizing RACF and NT passwords in a single-sign-on scenario.

Note that whenever I use the terms 'RACF' and 'Windows' in this article, they can be taken to mean RACF or the more recent OS/390 Security Server, and Windows NT/2000/XP.

PASSWORD AUTHENTICATION

The topic of passwords is clouded in mystique and misconceptions, as can be seen by the questions and discussions that periodically arise on racf-l[1], the Internet discussion list for RACF. So first, let's see if we can sort out some basics.

Authentication is a means of being able to arrive at some level of confidence that the person you are communicating with is indeed who they say they are. Humans use many ways to authenticate people – for example, the sound of the person's voice, visual recognition, etc. Computers are much more limited in the ways in which they can try to authenticate who they are communicating with. Typically, they use a combination of userid and password, and, increasingly these days, some form of software or hardware token. Tokens are generally used as a means of implementing one-time passwords (see below). There are three primary opportunities to attack passwords:

- When they are entered
- When they are transmitted
- When they are stored.

Entering passwords

When a password is entered, you're basically at the mercy of the device through which you're entering it. If it's a secure entry device, it should be secure. If it's a general-purpose PC which could potentially have a key-logger installed, it's not so secure.

Transmitting passwords

Two strategies can be used to protect against the interception of transmitted passwords. One is to encrypt the communications channel (eg SSL, tn3270e), and the other is to use a one-time password so that, even if someone does intercept the password, it doesn't matter because they can't re-use it.

Storing passwords

Last, and I think generally the most important, is the question of how passwords are stored. This is particularly important because an attack on a password when it's entered or transmitted is an attack on just one password. Attacking the store, by contrast, means being able to attack the entire user population. This makes it a very attractive target.

Passwords should never be stored on a system in such a way that they can be retrieved. Instead, they should be encrypted using a one-way function to arrive at an encrypted value. The sign-on process should then go through exactly the same process and compare the encrypted result with the stored encrypted value to see if they match.

This is fundamental, and most operating systems do a reasonable job, but many applications do not!

Don't forget that although I'm concentrating on the stored repository of encrypted passwords, equally vulnerable is any mechanism used to store them, eg the ICHPWX01 password exit. If someone can install their own code in here, it's pretty much 'game over' anyway.

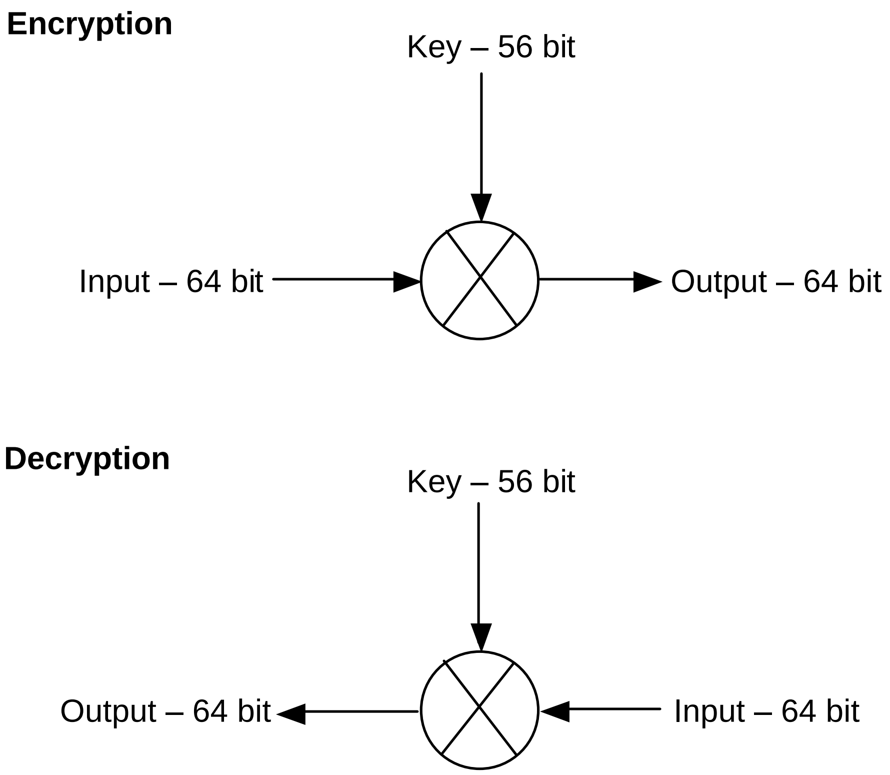


Figure 1: DES encryption and decryption

THEORY OF ENCRYPTION

The Data Encryption Standard[2] (DES) has been around since 1977[3] and has been the base building block for many systems, including RACF. It's relatively simple in theory, even if the precise implementation can get tricky.

From the outside, it works like a black box that has two inputs and one output. The inputs are a 64-bit block of data and a 56-bit key. The black box uses the key to encrypt the data into a 64-bit output data block. To decrypt, you do the inverse – that is, you use the same key but reverse the algorithm (see Figure 1).

DES is normally described starting from the outside and then working in. Here, however, I'll focus simply on the core mechanism used by DES, and then move outwards enough to set it in context. This should, hopefully, be enough to give an appreciation of its properties.

At the core of the DES algorithm are a series of eight 'S' boxes

S1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	11	4	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Figure 2: S box in detail

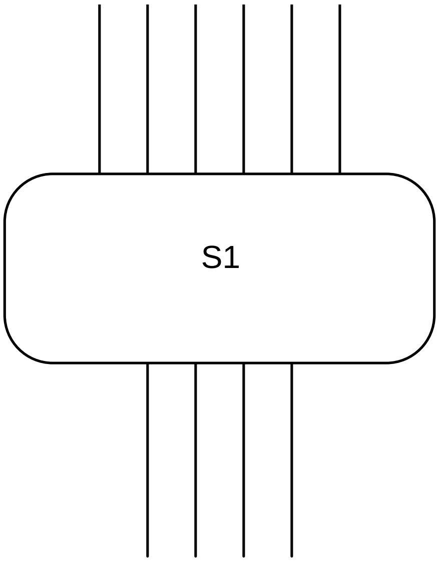
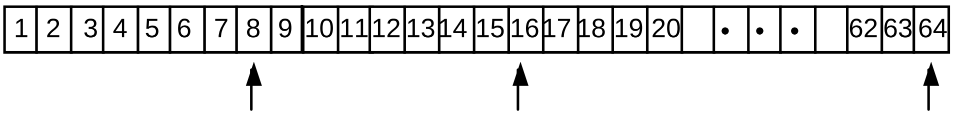


Figure 3: S box concept

(see Figure 2). The 'S' box operates by acting as a look-up table, where the input value is a 6-bit number. Two of the 6 bits are used to decide the row, and the remaining 4 bits are used to determine the column. This then gives an output number between 0 and 15, ie a 4-bit number. This is generally represented as shown in Figure 3.

The other seven 'S' boxes are similar tables, but with the values rearranged to give a different set of look-up values, otherwise just a variation on a theme.

If you understand the 'S' box concept, you almost understand DES, because DES simply takes this principle and places it in a series of nested loops, so that to try to work it back becomes computationally infeasible – that is, it will require as many



DES parity bits which are only used during key entry and are dropped once key has been loaded

Figure 4: DES parity bits

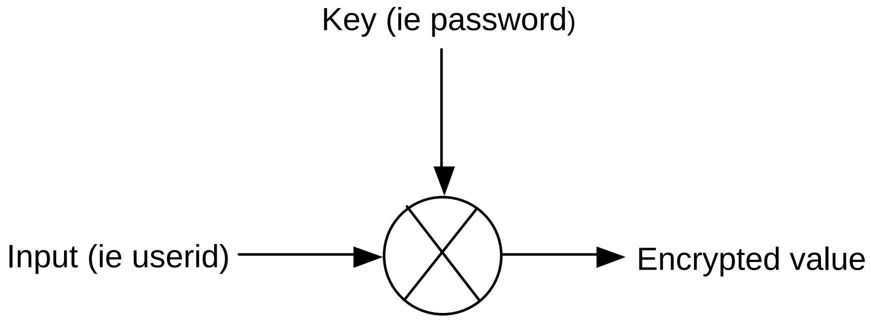


Figure 5: RACF and DES

computations as trying every permutation of the 56-bit key. So even if you know the userid and the encrypted value, the only way to attack it is to use either a dictionary attack or brute force.

In cryptographic terms, the whole point of DES is to protect the key. A cryptographic attack would typically involve obtaining some input data along with some associated encrypted output data, and then trying to deduce the key. DES can either be used as a method of encrypting data so that it can be decrypted at a later time, or as a one-way function – ie throwing away the key once it has been used to encrypt some input data.

Incidentally, in case you're wondering why DES uses a 56-bit key rather than a 64-bit key, allow me to explain. It all dates from the time when keys were manually input to secure devices through keyboards or keypads. They would typically take a 64-bit value as input, but the device would then break it down into 8-bit blocks where each block contains seven data bits and one parity bit (see Figure 4). As this was generally keyed as 16 hexadecimal digits, this gave a rudimentary level of error checking to detect keying errors.

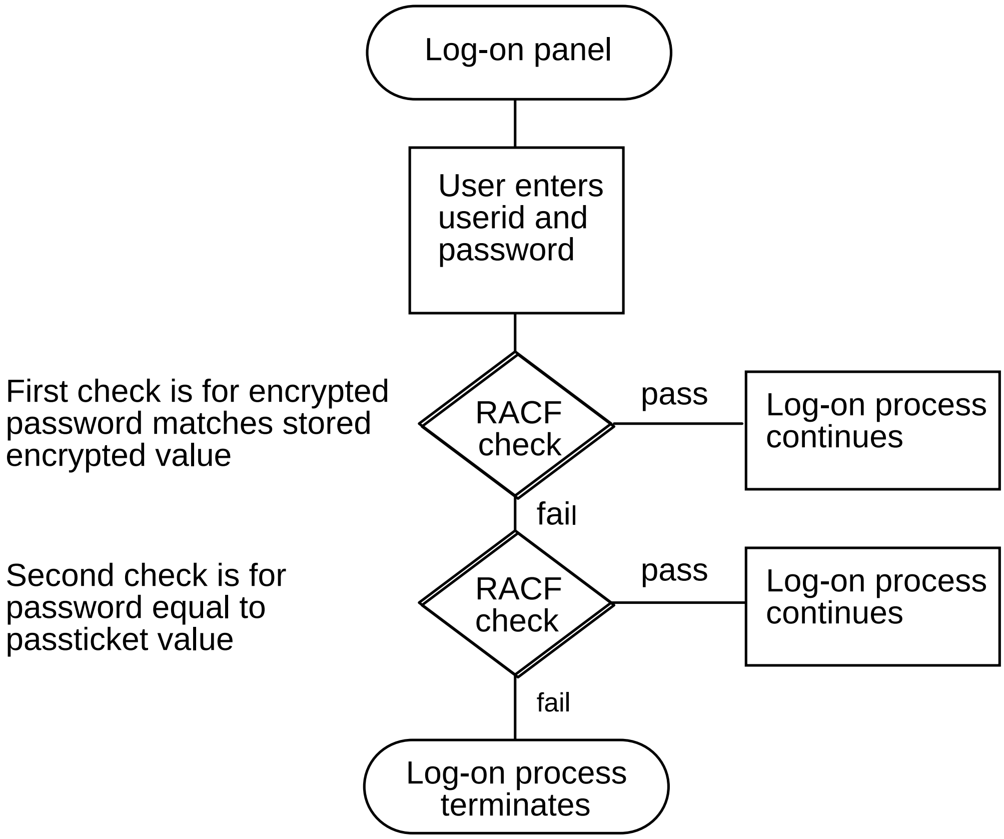

Log-on panel
User enters userid and password
First check is for encrypted password matches stored encrypted value
RACF check
pass
Log-on process continues
fail
Second check is for password equal to passticket value
RACF check
pass
Log-on process continues
fail
Log-on process terminates

Figure 6: What actually happens during log-on

RACF PASSWORDS

I'm pleased to say that RACF uses DES as a one-way function, as shown in Figure 5. Note that although IBM uses DES as per the standard, it has to take an 8-character password and convert it into a 56-bit key. Understanding how it does this requires us to look down a level at the binary bits involved.

As we've seen, in DES the right-most bit of each byte is lost as a parity bit. Now consider the EBCDIC characters, hex values, and bit patterns involved:

```
'A' = C1 = 11000001
'B' = C2 = 11000010
...
'8' = F8 = 11111000
'9' = F9 = 11111001
```

plus three special country characters. In other words, the left-most bit is almost always a 1, and hence virtually redundant, leaving 7 useful bits. Yet again, there's no rocket science here, just good sound logic.

RACF passwords are stored in the RACF database datasets as identified by the RVARY LIST command, so it's not difficult for any user to find out where the encrypted passwords are. These datasets must therefore be protected against unauthorized reading.

Figure 6 shows what actually happens during log-on. PassTicket is techno jargon for an IBM proprietary method for one-time passwords using a software implementation of a token – that is, software used to generate a one-time password based on a secret key configured into the software, the userid, and, of course, a function of the date and time. If you want to delve any deeper, I'd recommend Thierry Falissard's *The RACF PassTicket Page*[4] as an excellent starting point.

PassTickets are of particular benefit if connecting to a system across an untrusted network where there's a possibility of someone trying to 'sniff' userids and passwords. However, this advantage has to be weighed against the fact that the client system needs to have user credentials stored locally. The risk here becomes a combination of the accessibility of the client system and the manner in which the credentials are stored – eg in the clear or encrypted.

By now it should be becoming clear that there's no rocket science involved here: anyone who can go and look up the DES on the Internet and apply it to RACF can carry out a dictionary attack.

Several RACF password crackers are now freely available, in addition to the genuine security administration and audit tools which contain password crackers. The earliest cracker was from Kurt Meiser (now marketed by Peter Goldis[5]). Then came my CRACF[6], followed by another from Thierry Falissard, both available over the Internet. Most recent is an evolution from CRACF called WEAKWORD[7]. CRACF displayed any cracked passwords but was very restrictive in what it tried to crack. WEAKWORD, on the other hand, doesn't display the password but just flags it as weak; however, it does allow a dictionary to be defined. Interestingly, I received more requests for a version which didn't display the cracked passwords than I did for a full-blown cracking version.

There are several ways in which we can mitigate the risk of attack to RACF passwords:

- Apply password rules or use the password exit, to reduce the likelihood of guessable passwords.
- Ensure password history is used to prevent simple recycling of passwords.
- Ensure passwords are not static and are changed periodically.
- Protect the encrypted password storage from unauthorized read access.

PASSWORD SYNCHRONIZATION

As we've seen, RACF doesn't actually store encrypted passwords, but rather encrypted userids, where the password was used as the key and then discarded. This means that if two userids both have the same password, the encrypted values stored in the RACF database will be different. However, this is not true of all systems and is certainly not true of Windows NT.

Windows also uses a much less computationally intensive algorithm for encrypting passwords, making it a very attractive target for hackers. Put simply, it's quicker to carry out an attack on an encrypted Windows password than on an encrypted RACF password, and, more importantly, it takes virtually the same time to attack one Windows password as it does to attack an entire Windows population of users. Compare this with the much older RACF, where every password must be individually attacked.

I don't mean to delve too deep into Windows passwords, but I do feel it's worth pointing out how they're processed/stored. In particular, I'm referring to the fact that they're split into blocks of seven characters before they're encrypted and stored. This means that a 14-character password is not much better than a seven-character one. If you can brute-force-attack a seven-character password, you can brute-force any length of Windows password.

What's more, if your password is, say, nine characters long, then the attack analyses the second (two-character) block first, and will crack that with very little effort. This then leaves the initial seven characters, along with the additional clue of knowing what the last two characters are. Worse still is if your password is, say, a seven-character word, followed by two numbers. My advice with Windows passwords is to use seven-character passwords and be sure to include at least one punctuation character[8].

Incidentally, Windows also has the equivalent of the ICHPWX01 password exit. Any dynamic link library located in the %systemroot%\system32 directory and referenced in the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Notification Packages registry key will be called during password change. It's very much the equivalent of the RACF exit, only the RACF exit is easier to control as it's only in one place. The Windows exit relates only to the local password database – ie if the exit is on a workstation it relates only to the local passwords held on that workstation. Likewise, if the exit is on a Primary Domain Controller, it relates to every password stored locally on that domain controller.

Microsoft supplies a sample exit called PASSFILT.DLL9 for Windows NT which applies its suggested rules for strong password validation – ie there are no configurable options, and you either use it or write your own. The functionality of PASSFILT.DLL has been incorporated into Windows 2000 and XP.

I can't talk about Windows passwords without the obligatory reference to the *de facto* industry standard cracker, L0phtcrack[10]. This is a very powerful and impressive tool which all information security professionals should be aware of. If you've never seen it, go and try it – you'll learn something!

Most recently, in July 2003, there has been some very interesting work[11] coming out of LASEC[12] (Security and Cryptography Laboratory) which exists within EPFL[13] (École Polytechnique Fédérale de Lausanne). It has developed a very fast password cracker called Advanced Instant NT Password Cracker. To demonstrate how fast, it provided an on-line interface and invited visitors to submit encrypted passwords for cracking. They managed to crack 1,845 passwords with an average crack time of 7.7 seconds!

I hope I've managed to convince those who weren't already convinced that there are some very significant differences in the strengths of design between the ways different operating systems handle passwords. This makes me very wary about the potential of synchronizing systems such as a password reset on a Windows platform, which can be propagated to an associated RACF userid, or vice versa.

CONCLUSIONS

In summing up, I'd like to start by quoting from Bruce Schneier, an academic cryptographer cum real-world security practitioner[14]: "You can't memorize good enough passwords any more, so don't bother. Create long, random passwords, and write them down. Store them in your wallet, or in a program like Password Safe. Guard them as you would your cash. Don't let Web browsers store passwords for you. Don't transmit passwords (or PINs) in unencrypted e-mail and Web forms. Assume that all PINs can be easily broken, and plan accordingly."

Password Safe[15] is a freeware utility which started life at Counterpane under the direction of Bruce Schneier. It's been around for many years unchanged. However, it's now being actively developed as a SourceForge Open Source Project[16]. It's a small application dedicated to storing userids and passwords securely.

Let me say that I agree whole-heartedly with Bruce Schneier's statement. I for one use Password Safe and actively encourage its use wherever and whenever I can.

The only caveat with Password Safe is that it is only as secure as the password used to secure it in the first place, and there is a password safe cracker[17] by Joe Smith freely available on the Internet which can be used to carry out a dictionary attack on Password Safe. This means that the initial password used must not be something which is ever likely to appear in any password hacking dictionary.

Because Password Safe is a small, simply run program without requiring libraries etc, it lends itself to being kept on USB removable media, giving an added level of physical security.

Finally, just to recap on the points touched upon earlier, here is my checklist:

- Use Password Safe or an equivalent.
- Use password expiry to enforce regular password changing.
- Use setropts rules to enforce improved password quality.
- Use password history in conjunction with setropts password rules.
- Ensure that the RACF database datasets, primary and back-up, have fully qualified dataset profiles that have no unnecessary read access.
- If authenticating across an insecure network, consider using PassTickets as an alternative to conventional passwords.
- Keep passwords different across different platforms, at least for privileged users.
- Know about any password exits in use within the environment, whatever the platform.
- Use seven-character Windows passwords with at least one punctuation character.

REFERENCES

1 http://www.listserv.uga.edu/archives/racf-l.html

2 Federal Information Processing Standards Publication 46-2 1993 December 30 Specifications for DATA ENCRYPTION STANDARD (http://www.itl.nist.gov/fipspubs/fip46-2.htm)

3 Applied Cryptography Second Edition: protocols, algorithms, and source code in C / Bruce Schneier

4 http://os390-mvs.hypermart.net/passtkt.htm

5 http://www.goldisconsulting.com/

6 http://www.nigelpentland.co.uk/cracf.htm

7 http://www.nigelpentland.co.uk/weakword.htm

8 http://www.nigelpentland.co.uk//Password%20Guidelines.pdf

9 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/security/passfiltdll.asp

10 http://www.atstake.com/research/lc/

11 http://lasecpc13.epfl.ch/ntcrack/

12 http://lasecwww.epfl.ch/

13 http://www.epfl.ch/

14 http://www.counterpane.com/crypto-gram-0105.html#8

15 http://www.counterpane.com/passsafe.html

16 http://passwordsafe.sourceforge.net/

17 http://members.aol.com/jpeschel3/PasswordSafeCracker.zip

Nigel Pentland
Information Security Analyst (UK)

RACF in focus – understanding OS/390 Unix security

'RACF in focus' is a regular column focusing on a specific RACF topic. Here, we examine RACF security for OS/390 Unix.

OS/390 Unix was initially known as OpenEdition MVS, but that name is no longer used. It is now commonly referred to as OS/390 Unix System Services (USS), or simply OS/390 Unix.

Security checking for OS/390 Unix is done in RACF, and in order to use Unix services a user must have a RACF userid. The RACF security administrator therefore needs to understand RACF security as it relates to OS/390 Unix, and implement at least portions of it to protect Unix resources.

HOW OS/390 UNIX SECURITY WORKS

Unix data is stored in files whose structure is more like the one used in Windows-based PCs than the datasets used in the MVS environment. This file system is known as the Hierarchical File System (HFS). Security information, consisting of flags, is kept within the file system in File Security Packets (FSP). Within the FSP, the UID represents the file owner, and the GID represents the group owning the file. Access is given based on UID and GID information stored in the FSP.

On the RACF side, there's an OMVS segment in the RACF userid profile (much like the CICS and TSO segments) that defines various Unix attributes for the user, including the UID, and an OMVS segment in the group profile that defines the GID. Before a RACF user can use Unix services, he or she must have an OMVS segment (or inherit a default, as described below). The user also needs to be connected to a group with an OMVS segment (or inherit a default).

When Unix resources are accessed, security checking is done by calling RACF. RACF compares the UID and GID values of the Unix resource to the UID and GID values assigned to the user. The UID and GID values assigned or inherited by a user therefore play an important part in the Unix world.

PLANNING FOR OS/390 UNIX SECURITY

If your installation is going to deploy Unix applications in the IBM mainframe environment, you'll obviously need to plan for and implement RACF security for OS/390 Unix.

However, there are a number of other reasons why you might need to implement some portions of RACF/Unix security, even if you're not actually implementing Unix applications. For example, if you want to do file transfers between the mainframe and other platforms using ftp, you'll need OS/390 Unix services, and, therefore, RACF security. TCP/IP and LDAP implementations are other examples that require OS/390 Unix and therefore RACF security.

It's important to have a plan that defines basic RACF security for a Unix environment, so you'll be well positioned to exploit its features more fully if and when required. Amongst other things, you'll need to define policies and procedures to administer the Unix environment, and decide who is to be allowed to use OS/390 Unix, what UIDs and GIDs to assign, how you'll track assigned values, how you'll monitor and audit the UnixC environment, and so on. Without a plan, you may end up having undesirable results. For example, ftp usage is quite common these days, and RACF administrators are often asked to assign OMVS segments to users wanting to use ftp processes. Without a plan, you may assign OMVS segments without paying much attention to the UID and GID values. This may have negative implications in the future, and some of the work will need to be re-done.

UNIQUE UIDS AND GIDS RECOMMENDED

Although UIDs and GIDs can be shared among users and groups, this isn't recommended. Sharing doesn't really make sense because file protection in the HFS is done at the UID and GID level, and you lose accountability.

Ideally, each user should have a unique UID assigned, and each group a unique GID. The only exception is UID(0), which gives the user 'superuser' powers. You may have more than one user who needs this.

The RACF class UnixMAP keeps tabs on who is assigned which UIDs and GIDs. It can be used to ensure that RACF users and groups have unique UIDs and GIDs. It's best to activate this class before you start assigning UIDs and GIDs at your installation.

If you can't do this, you'll have to take the action described in the *OS/390 Security Server (RACF) System Administrator's Guide*. This involves running a REXX EXEC to 'populate' the UnixMAP class from the RACF database and then activating the UnixMAP class to keep tabs on future changes to UIDs and GIDs.

If the UnixMAP class is active, you can query it to find out who is using a particular UID or GID, as follows:

```
RL UnixMAP G555 all
```

It's important to note that profiles in the UnixMAP class are auto created – you don't need to maintain the access list.

You may not want to assign OMVS segments to all userids and all groups, even though they require Unix services. In this case, you can specify defaults to be used by individuals who don't have any OMVS segments. This is done via the FACILITY class profile BPX.DEFAULT.USER. It contains, in the appldata field, the default RACF userid and default RACF groupid, to be used for OMVS segment look-up purposes. You have to define the default userid and the default groupid to RACF. Note that the defaults are used only in cases where no OMVS segment is found for the user. If an OMVS segment is found, information from that segment is used, not from the default userid's segment.

SUPERUSER POWERS

In the Unix world, UID(0) is used to denote a superuser. This is a very powerful attribute, and can be compared to someone having the RACF attributes OPERATIONS, SPECIAL, and

AUDITOR all at once. Note that the RACF attributes OPERATIONS, SPECIAL, and AUDITOR give no special powers in the Unix world; similarly UID(0) gives no special powers for MVS purposes.

If you have the UnixMAP class active, you can see how many users have UID(0) assigned by entering the following command:

```
RL UnixMAP U0 ALL
```

Since superuser is such a powerful authority, you may not wish to give this to many people. Some users may need only a subset of the superuser powers. There is a RACF class called UnixPRIV that you can use to specify more granular levels of special powers.

AUDITING OS/390 UNIX

The following classes are available for auditing OS/390 Unix: DIRACC, DIRSRCH, FSOBJ, FSSEC, IPCOBJ, PROCACT, and PROCESS. These classes don't need to be activated, nor do you need to create profiles within these classes.

In order to audit OS/390 Unix, you need to specify your audit options using either of the following commands for the above classes:

- SETROPTS LOGOPTIONS(CLASSNAME)
- SETROPTS AUDIT(CLASSNAME)

SUMMARY

This article has looked only very briefly at OS/390 Unix security, and contains enough information to get you started. If your installation is developing OS/390 Unix applications, you'll need to do further research into the workings of OS/390 Unix. Finer control mechanisms are available in OS/390 to address more complex Unix configurations.

Dinesh Dattani
Security Consultant, Toronto (Canada)
dddattani@rogers.com

RACF news

Critical Path has announced its Critical Path Password Management application for centrally administering passwords across systems and applications. The integrated software provides self-service resets of forgotten passwords, centralized definition and enforcement of password policies, dynamic password synchronization across systems for reduced sign-on, and auditing of all password change activities. It enables passwords for various user applications, databases, and operating systems to be administered centrally. Users can set their own passwords via a Web-based interface or through existing systems such as RACF or Windows.

URL: http://www.criticalpath.net/solutions/enterprise/passwordManagement/

* * *

OpenNetwork Technologies has announced the Universal Identity Platform (Universal IdP), designed to take advantage of Microsoft technology and to extend its value as an identity infrastructure to J2EE and mainframe environments. The software centralizes and unifies the management of identities and security policies, secures access to protected resources, and delivers automated workflow and provisioning. Specifically, it integrates Microsoft Identity Integration Server (MIIS) with mainframe systems such as RACF, ACF2, and TopSecret.

URL: http://www.opennetwork.com/news/press/2003/2003-07-02_UIdP.php

* * *

IBM has announced Tivoli Workload Scheduler Version 8.2, designed to help reduce the complexity of managing the workload on mainframes and open systems and automating many operator activities. Enhancements include improved security through the addition of SSL-based authentication and encryption.

URL: http://www-3.ibm.com/software/tivoli/products/scheduler-apps/

* * *

e-Security has announced the release of e- Security Version 4, with new functionality for managing enterprise security, including enhanced usability, incident management, performance, and correlation capabilities.

URL: http://www.esecurityinc.com/Company/Press_Releases/Dynamic.asp?PR_ID=26

* * *

Computer Associates and SteelCloud have announced an agreement under which the companies will deliver a family of hardened, ready-to-deploy enterprise-class security appliances based on CA's eTrust family of security solutions.

URLs:
http://www3.ca.com/press/PressRelease.asp?CID=45782
http://www.steelcloud.com/appliances/default.asp

* * *