39

RACF update

February 2005

In this issue

A tc PUBLICATION

RACF Update

Published by
Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor
Trevor Eddolls
E-mail: trevore@xephon.com

Publisher
Colin Smith
E-mail: info@xephon.com

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 $290.00 in the USA and Canada; £190.00 in the UK; £196.00 in Europe; £202.00 in Australasia and Japan; and £200.50 elsewhere. The price includes postage. Individual issues, starting with the August 2000 issue, are available separately to subscribers for $72.75 (£48.50) each including postage.

Disclaimer

Contributions

When Xephon is given copyright, articles published in *RACF Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) 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.

Filtering RACF SMF records by userid

The *Security Server RACF Auditor's Guide* describes how to use two IBM SMF unload utility programs to filter, parse, and format RACF SMF data. Programs IRRADU00 and IRRADU86 format security information held in SMF type 30, 80, 81, and 83 records.

The input parameters available to IFASMFDP allow a certain amount of granularity, ie allowing date, time, and system ID filtering. I wanted to extend filtering to allow the extraction of records by RACF userid as well.

To do this, I wrote my own SMF unload exit, LSGADU01. This delivered the ability to filter on a userid prefix.

By writing the code to run as part of the unload process it removed the need to post-process the output file.

Passing the ID into LSGADU01 proved a bit of a problem. I eventually wrote the code to run through control blocks and retrieve the JCL parameter from the SCTX control block. SMF unload exits are passed a workarea to use by IFASMFDP. This area is already used by the IBM utility programs. This meant that my code couldn't save information to be used by subsequent calls. Which in turn means my code scans control blocks for every call (of LSGADU01) and reduces efficiency. This hasn't caused any noticeable performance issues.

I've found this added functionality very useful when I've had to audit the use of a specific ID.

The code has to be linked to an authorized (APF) library, but can be linked AC=0 (not authorized).

JCL TO RUN SMF UNLOAD

```
//SEAUDP   JOB (,IS),'EXTRACT RACF SMF',CLASS=F,MSGCLASS=X,
//         NOTIFY=&SYSUID
//*
```

```
//         SET AUDITUSR=ID123
//*
//STEPØØØØ EXEC PGM=IEFBR14
//DDNAMEØ1  DD DISP=(MOD,DELETE),DSN=SG.TEMP.&AUDITUSR..SMFDUMP,
//          SPACE=(TRK,(1,1)),DSORG=PS
//*
//STEPØØØ1 EXEC PGM=IFASMFDP,PARM='&AUDITUSR'
//SYSPRINT DD  SYSOUT=*
//ADUPRINT DD  SYSOUT=*
//OUTDD     DD  DISP=(,CATLG),DSN=SG.TEMP.&AUDITUSR..SMFDUMP,
//     SPACE=(CYL,(15ØØ,1ØØ),RLSE),DSORG=PS,LRECL=32756,RECFM=VB
//SMFDATA   DD DISP=SHR,DSN=PJ.PROD.SMF8Ø81(Ø)
//          DD DISP=SHR,DSN=PJ.PROD.SMF3Ø(Ø),UNIT=AFF=SMFDATA
//SMFOUT    DD  DUMMY
//SYSIN     DD  *
     INDD(SMFDATA,OPTIONS(DUMP))
     OUTDD(SMFOUT,TYPE(ØØØ:255))
     ABEND(NORETRY)
     USER1(LSGADUØ1)
     USER2(IRRADUØØ)
     USER3(IRRADU86)
/*
```

LSGADU01

```
LSGADUØ1 AMODE 31
LSGADUØ1 RMODE 24
LSGADUØ1 RSECT
         DS    ØH
         B     BEGIN-LSGADUØ1(,15)
PROGRAM  DC    CL8'LSGADUØ1',C': &SYSDATE &SYSTIME '
         DS    ØH
BEGIN    EQU   *
         BAKR  14,Ø
         LR    12,15
         LR    1Ø,1
         USING LSGADUØ1,12
*
         L     9,4(,1Ø)
         SLR   4,4
         USING SMFHEADR,9
*
         CLI   SMFRTYPE,X'5Ø'
         BE    PROCESS_TYPE8Ø
*
         CLI   SMFRTYPE,X'1E'
         BNE   ENDIT
*
PROCESS_TYPE3Ø EQU *
```

```
         ICM   1,15,SMF3ØIOF
         BZ    ENDIT
         AR    1,9
         LA    1Ø,1Ø8(,1)
         B     RUN_CHAIN
*
PROCESS_TYPE8Ø EQU *
         LA    1Ø,SMF8ØUSR
*
RUN_CHAIN EQU  *
         USING PSA,Ø
         USING TCB,1
         USING IEZJSCB,2
         L     1,PSATOLD
         SLR   2,2
         ICM   2,7,TCBJSCBB
         L     3,JSCBJCT
         LA    3,16(,3)    ADD ON PREFIX
         DROP  Ø,1,2
         USING INJMJCT,3
         SLR   4,4
         SLR   5,5
         ICM   4,7,JCTSDKAD
         DROP  3
         USING INSMSCT,4
         LA    4,16(,4)    ADD ON PREFIX
*
CHECK_PGM EQU  *
         CLC   =C'IFASMFDP',SCTPGMNM
         BE    PGM_FOUND
         ICM   4,7,SCTANSCT
         BZ    ABEND2ØØØ
         LA    4,16(,4)    ADD ON PREFIX
         B     CHECK_PGM
*
PGM_FOUND EQU  *
         USING SCTXIN,5
         ICM   5,7,SCTXBTTR
         LA    5,16(,5)    ADD ON PREFIX
*
         CLI   SCTXIDNT,X'ØC'
         BNE   ABEND2ØØ1
*
WORK_OUT_PARM_LENGTH EQU *
         LA    1,SCTXPARM
         L     2,=A(L'SCTXPARM)
*
PARM_LENGTH_LOOP EQU *
         CLI   Ø(1),X'ØØ'
         BE    WORK_OUT_PARM_LENGTH_END
         CLI   Ø(1),C'*'
         BE    WORK_OUT_PARM_LENGTH_END
         LA    1,1(,1)
         BCT   2,PARM_LENGTH_LOOP
         B     ABEND2ØØ2
*
WORK_OUT_PARM_LENGTH_END EQU *
         L     1,=A(L'SCTXPARM)
         SR    1,2
         SLR   4,4
*
COMPARE_USERID_WITH_PARM EQU *
         LA    3,SCTXPARM
         BCTR  1,Ø
         EX    1,CLC_
         BE    ENDIT
         LA    4,4
*
ENDIT    DS    ØH
         LR    15,4
         PR    ,
*
CLC_     CLC   Ø(Ø,3),Ø(1Ø)
*
ABEND2ØØØ ABEND 2ØØØ,DUMP
ABEND2ØØ1 ABEND 2ØØ1
ABEND2ØØ2 ABEND 2ØØ2
*
         LTORG ,
*
SMFHEADR DSECT
SMFRDW   DS    XL4
SMFSID   DS    XL1
SMFRTYPE DS    XL1
SMFTIME  DS    XL4
SMFDATE  DS    PL4
SMFSYSID DS    CL4
* TYPE 8Ø STUFF
SMF8ØDES DS    XL2
SMF8ØEVT DS    XL1
SMF8ØEVQ DS    XL1
SMF8ØUSR DS    CL8
* TYPE 3Ø STUFF
         ORG   SMFRDW+32
SMF3ØIOF DS    XL4
SMF3ØILN DS    XL2
SMF3ØION DS    XL2
         ORG   ,
*
         PUSH PRINT
         PRINT OFF
         IHAPSA
         IKJTCB
         IEZJSCB
         IEFAJCTB
         IEFASCTB
         IEFSCTX
         POP PRINT
*
         END
```

Calum Reid
Systems Programmer (UK)

RACF 101 – understanding RACF terms: part 2

RACF 101 is a regular column for newcomers to the RACF world. It presents basic RACF topics in a tutorial format. In the last issue we looked at some common RACF terms used in the industry. In this issue we will build on this and discuss a few more RACF and related topics that a beginner to RACF should know. Knowing these concepts will add to your RACF knowledge and help you better understand some of the idiosyncrasies of RACF.

In this second part on RACF terms we will expand our knowledge not only of RACF but also of concepts external to RACF, such as SMF records and SYSLOG. These should be included in a beginner's repertoire even though they are not directly related to RACF. Without them, your knowledge of RACF is only partial. Such terms are often heard from more senior RACF colleagues, but RACF manuals only mention them in passing, if at all, since they are not part of RACF security. In other words, we will look at the broad spectrum of the MVS world (now known as OS/390 or z/OS) as it relates to security.

RACF REFRESH

An inquisitive beginner to RACF may wonder: why do I need to refresh RACF profiles? Isn't it enough to make the change to RACF by using commands? Even though they may know how to do a refresh, and do one when asked to by someone, they may not know the underlying reason for using a refresh.

The reason is simple enough – RACF administration commands update the RACF database directly, but RACF, for various reasons, stores some of the security information in main memory and references this information first. Where there is a discrepancy between information in main memory and the RACF database, the information in main memory takes precedence. Therefore, to effect changes for profiles kept in main memory, a RACF refresh is required.

RACF CLASS DESCRIPTOR TABLE

The class descriptor table, also known as the CDT, is a table (or list) that defines all possible RACF classes used at your installation. Usually your systems programmer modifies this table. In addition to RACF classes provided by IBM, a site has the option to define its own unique RACF classes. This capability allows an installation to build classes for its own purposes and use. A good example of installation-specific classes is CICS. Typically, all shops have their own unique set of CICS classes for their various CICS regions.

Other than knowing of its existence, a beginner rarely needs to know the details or the contents of the CDT, and most likely will never need to modify it!

GLOBAL OPTIONS

RACF global options, as the name implies, are options that apply to the entire RACF database and to every userid. For example, they may specify that the password must be changed every 90 days.

The RACF global options can be viewed by entering the command:

```
SETROPTS LIST
```

Needless to say, changing global options is a serious matter involving input from more senior colleagues, and sometimes even the systems programmers. As a beginner, you should not attempt to change anything using the SETROPTS command, even if you have the power to do so (the SPECIAL attribute).

STARTED TASKS

Started tasks are also known as started procedures. This is the name given to, usually, processes or tasks related to the operating system. Those familiar with the Unix operating system can think of them, roughly, as the daemons of the MVS operating system.

Started tasks have their own peculiarities, and, for authentication purposes, RACF requires that they all be associated with a RACF userid to be used for authentication and validation. Another thing about them is that they can have the TRUSTED or the PRIVILEGED attribute – both of which are very powerful. As mentioned, these are system tasks, and powerful attributes are desired in most (but not all) cases of started tasks. TRUSTED and PRIVILEGED are to be found only with started tasks because there is no way to make an ordinary userid trusted or privileged.

RACF provides a separate class, called STARTED, to define and assign attributes to started tasks. This provision for a separate class is a relatively new feature of RACF, and some installations may still be using the older method of defining started tasks to RACF – namely defining them in the ICHRIN03 table. In fact, even to this day, the ICHRIN03 table may be used in conjunction with the STARTED class.

DSMON

DSMON, sometimes pronounced 'DESMON', is an audit tool that provides a number of useful reports for audit purposes. Quite often, the auditor at your installation may ask for these reports. DSMON produces reports to show you such things as: how important MVS libraries (such as authorized libraries) are protected; userids with special powers; RACF exits (see below for a fuller explanation) in effect at the installation; and other such information useful for audit purposes.

To produce a DSMON report you not only need the capability (the AUDIT attribute), but also some knowledge of JCL. Producing the report, however, does no harm. If allowed, you should try to produce one – it does not change any RACF security.

WARN MODE

When you create RACF profiles, by default they are not in WARN mode – they are in NOWARN mode. This means that the security specified in the profile is in effect.

But you can put a profile in WARN mode with the command:

```
ALTDSD dsn_name WARN
```

Or, for resource profiles, the command is:

```
RALTER classname profilename WARN.
```

When a profile is in WARN mode, the security specified by the profile is not in effect, and anyone can do anything to the resources covered by that profile, including deleting them! The only thing that will happen is that when someone tries to access something they are not allowed to, they will get a warning message telling them that temporary access has been allowed.

So, what is the purpose of WARN mode? The intention is to build security gradually, and it works very well when introducing new security controls at an installation where none existed before.

Needless to say, WARN mode is dangerous – it allows full access, including delete (alter) access to the resources covered by the profile! Use it with care, and only for the less sensitive data at your installation.

OMVS

OMVS stands for OpenEdition MVS, the former name of USS (Unix System Services). Although the name ‘OpenEdition’ is now obsolete, it is still being used; for example the Unix segments in userid profiles are still referred to as OMVS segments. As a beginner it is enough to know that any reference to OMVS can safely be substituted by USS for learning purposes.

RACF EXITS

RACF exits are provided by IBM to modify, or overrule, decisions made by normal RACF processing. They provide a way for an installation to tailor RACF processing for its own needs. An example of this would be where you want special password controls above and beyond what RACF normally provides.

RACF exits are not common, and it is quite possible that your installation does not have any. To see whether there are any RACF exits implemented at your installation, run the DSMON report explained above. If you see any active RACF exits, you may want to try to find out what they do and why they are there.

ASSORTED MVS/SECURITY TERMS

To make any noteworthy progress in the RACF world, a beginner needs to know at least the basics of some non-RACF terms. These are usually MVS-system type concepts and they play an important role in security, and therefore RACF. Here are some of the common ones:

- *SMF (System Management Facility) records*. These come in various flavours, and they are important from a security viewpoint for two reasons. First, if their contents are compromised, there is a big security risk. Secondly, RACF itself stores its daily events (violations, loggings, etc) in SMF records.

SMF records are stored in datasets generally with names like SYS1.MAN*x* or some variation thereof. (The MAN part is confusing, but the reasons for naming them thus are historical.)

- *IPL (Initial Program Load)*. This refers to the ‘booting’ of the MVS operating system. It is important from a security perspective because, after an IPL, all RACF profiles are newly brought into main memory from the RACF database.

  If you had modified a profile that needed a refresh, and were waiting for some clarification before doing the refresh, watch out. The IPL, over which RACF folk have no control, will do the refresh for you by its very nature – whether you like it or not!

- *SYSLOG (System Log)*. This is the log where the on-going MVS activities, RACF and otherwise, are recorded. These activities are time-stamped, to tell you when the event occurred. If someone logs on to TSO, for example, and they get a RACF violation, you can go to the SYSLOG and look at the exact error message. This is about the only use of SYSLOG for our purposes – to view RACF messages when required. To view SYSLOG, see SDSF below.

- *SDSF (System Display and Search Facility)*. This is a facility for viewing job output using TSO. It is a useful thing to know, since output from RACF batch jobs is also viewed using this facility. In addition, SYSLOG (mentioned above) can be viewed with this facility. There are other products in the market that perform the same functions as SDSF, so you may not have this at your site but, rather, one of the competing products.

IN CONCLUSION

With last issue's article and this one, we are well on our way to building a foundation of RACF knowledge. We not only know what the terms mean – that is easy enough to find out – but we are also building a historical and broader perspective of our understanding.

It would help to put in practice lessons learned here. For example, you can produce the DSMON report and examine its contents. Or you can list profiles, and see whether they are in WARN mode. If you find any, you may want to challenge someone about their applicability!

Dinesh Dattani (dinesh123@rogers.com)
Independent Consultant
Toronto (Canada)

Vanguard RACF/Security Conference 2004

I have long been a fan of SHARE, arguing that, for each technical topic that interests you, you can often get what you need out of a 1–3 hour SHARE session, without the expense of a 2–3 day course. But only if you work at it: taking careful notes on the handouts, doing a little reading of manuals, and trying a few things out on your own system is potentially more valuable than a lab in a course, which would be on someone else's machine. And you will be well rested for each session.

VANGUARD

Vanguard's annual RACF/Security Conference patterns itself after SHARE. Yes, it offers sessions about its products, but in the same technical, non-marketing approach that IBM does on its products at SHARE. The majority of the sessions are on security, without a mention of Vanguard products, whether they are conceptual, on IBM software, or user sessions describing user experience in implementing security for a non-IBM software product such as BMC's Control automation products.

Vanguard Integrity Professionals started out with mainframe ISPF and batch software that made RACF easier to use, and now has expanded to other platforms, from Windows GUIs into the mainframe to administer security for non-mainframe platforms.

I attended June 2004's conference in Reno. Saturday comprised full- and half-day sessions on Vanguard products, Sunday was full-day sessions on a variety of topics, and Monday to Thursday noon was 75-minute sessions. Most days had a keynote speaker, four regular sessions, and a lunch session. There were 12 different sessions to choose from for each regular session slot.

Here is what I learned from the sessions I attended.

SATURDAY

Effective use of Vanguard Administrator (full day)

- Administrator can also look at live RACF data, not just last night's extract.
- The Rebuild feature avoids recovering the RACF database, by just recovering the lost data.
- For performance, avoid any secondary extents for the extract datasets.
- Administrator does a complete delete; RACF does not.
- Administrator has a transfer function to move someone within an organization.
- Recommendation: only clone from models, not from real user IDs.

- VRASCHED allows you to schedule a change for the future.
- Use ID=*, not UACC, because UACC includes users without a RACF ID, such as public Internet users.
- Administrator contains a lite version of Vanguard Analyzer.
- A couple of subsets of Administrator exist for Help Desk and group administrators, without the need to grant them RACF SPECIAL attributes.
- Hard Revoke prevents the Help Desk from resuming a user ID.
- QuickGen is a really good way to quickly generate the same set of RACF commands for multiple entries, and is being expanded into a custom report generator in future releases.

SUNDAY

Hacking techniques and tools (full day)

- Speaker Mark Adams of Deloitte & Touche is very good at hacking topics, from both a knowledge and a practical experience perspective, and would be an ideal person should we ever need a high-end test of our security.
- There are on-line databases on the Internet for all (or most) possible values of hashed passwords, including Windows NT's; find the hash value, look up the hash value, and you have the password.
- Google is one of the best research tools for a hacker.
- Web sites like Metasploit exist to allow anyone to simply click on a link to test all known security holes for a given platform on a specified site.
- Many of the best hacker tools are available free on the Internet.

- IDs and passwords are very frequently sent without encryption on the Internet and, even more frequently, on a LAN; hackers can plug into either and watch the traffic that passes.
- Alerting a security staff member whenever an Admin account is created on any Windows server would be a good first step against hackers.
- The easiest way to hack into many corporations is to sit in the company parking lot and look for laptops with wireless NIC cards that are searching for the employee's wireless network that he uses at home; the speaker got into three laptops in 20 minutes in BMC's parking lot not long ago.

MONDAY

Mainframers' introduction to TCP/IP (double session)

- z/OS Unix (USS – Unix System Services, formerly known as OpenEdition) and TCP/IP running on it are relatively easy to make much more secure than non-mainframe Unix because security vulnerabilities, like */etc/passwd*, do not exist and newly-discovered ones are removed promptly by IBM.
- By setting up with a bit of care and using SSL where appropriate, z/OS is the most secure environment to host Web servers, both intranet and Internet.

RACF Level 2 service update – a diagnostic approach

- SAFTRACE debuted with z/OS 1.2 and is the tool of choice for low-level customer debugging of RACF.
- Unlike DB2, RACF development has very few hipers (high impact PTFs) per year.
- The speaker went through the most important RACF PTFs.

- Although different versions of RACF can be run at will, the latest version of Templates and Dynamic Parse must always be run.

VPN 101

- Split tunnelling is the major security threat.
- Creeping trust is also an issue, which can be solved by banning split tunnelling and multiple simultaneous VPN connections.
- I did not understand the speaker's assertion that a virus already in memory when VPN starts can be an issue, even when split tunnelling is addressed.
- The speaker also stated that best practice is corporate-only workstations as remote users.

TUESDAY

How to break into z/OS (double session)

- Run RACF IRA as soon as you install z/OS 1.4.
- The most common virus overwrites the boot sector: impossible in z/OS.
- Risk: UACC or ID(*) greater than READ for any APF-authorized library.
- Risk: dial-up modem (connected to phone line) on a workstation connected to a LAN could answer war dialler calls from hackers, allowing them to bypass the firewall.
- Every user ID used for started tasks should be protected (ie have no password).
- Without RACF Erase on Scratch, even the most junior programmer can read data in a deleted dataset that s/he has no RACF access to.
- The speaker was very good, but I was disappointed in his

suggestion (when I asked him privately) for change control for system libraries: get one person to do all the changes.

Getting more out of Vanguard Advisor

- Replaced RACF Report Writer, which IBM never intended to release in the first place.
- Reads security-related SMF records, both live and historical.

The eight rules of security

- The speaker was author of *Inside the Security Mind*.
- Lots of thought went into his eight rules, but clearly they apply only to the world of servers, as the result of applying his rules is unlimited repetitions of ‘buy another server’.
- Given that background, his rules do create a very secure environment.
- Temporary/development/back-up servers, workstations, network links, and unauthorized applications are the easiest target for hackers because of unapplied security patches, poor IDs and passwords, unmonitored, etc.
- Consider one-way external links instead of just defaulting to two-way.

WEDNESDAY

Sunrise meet the developers’ breakfast

- Voted on what goes into the next release of each Vanguard product.
- Helped design QuickGen’s new Report Generator.

RACF tips and tricks

- The RACF subsystem address space allows you to fix

RACF when TSO starts up, for example, a broken RACF database.

- Recommendation: when typing RACF commands, always use quotes for anything related to datasets (profiles, etc).
- Because SYS1 has to be the top of the RACF group tree, all Level 2 (just below SYS1) groups must be owned by other users; if they are owned by SYS1, they will inherit SPECIAL.
- Recommendation: always specify SUPGROUP and OWNER in an ADDGROUP, whether RACF panel or command, because you won't like the defaults.
- You can revoke a user from a group on a specified date.
- Protected user IDs can still be revoked automatically for inactivity.
- The most specific profile is always used, and RACF works left-to-right to determine 'most specific'.
- Recommendation: always use CONTROL instead of UPDATE, because it allows the reading of VSAM files directly by control interval.
- Handy: DSNS parameter of LISTDSD lists all datasets protected by the profile.
- Vendor software install tapes are a problem because of their unpredictable dataset HLQs.
- ICHRDSNT can automatically do a REFRESH on other LPARs.
- Group class is often called 'grouping class' to differentiate it from a group of users.
- RACF POSIT numbers can really hurt you in CICS, especially if you deactivate a class.
- RACF's return code of 4 to a caller gets interpreted differently: CICS takes it as a No; Tivoli takes it as a Yes.

- Recommendation: never define a facility named ‘*’ (asterisk) because RLIST uses the asterisk to mean ‘anything’.
- RDELETE is ignored if the VOLSER is wrong on the General Resource Profile being deleted.
- Recommendation: SETROPTS NOADDCREATOR.
- Recommendation: SETROPTS GENERIC(*).
- Recommendation: if creating USS default user and group, specify NOPASSWORD on the ADDUSER command.

Defining BMC CONTROL products to RACF

- Once you have security set up properly for CONTROL products, it is low maintenance.
- Speaker repeatedly said that BMC Support seems to have no idea whatsoever about security.
- Naming conventions for IOA prerequisite conditions are very important; they determine profile names.

THURSDAY

Enterprise identity mapping

- IBM middleware to map user IDs across multiple platforms, enterprise-wide.
- Requires software on top to do anything useful.

Optimizing your use of SecurityCenter

- Does not require Vanguard Administrator, or any other Vanguard product, to run.
- Mainframe system licensed, based on number of installed MIPS, with unlimited number of Windows client workstations running it.

- z/OS started task is the server, with a dataspace of RACF information, updated dynamically on both the server and workstation clients.
- Includes DB2 internal security (GRANT/REVOKE).
- Issue: need to check whether defaults can be customized globally, rather than having to customize each user, one by one, because some defaults are critical to the confusion levels of a first-time user.
- Generates RACF commands that you can view before executing and whose time of execution can be customized, but the product acts as if they have already been executed.
- Built to copy and paste easily into Excel.
- Marks Redundancies, such as users in an Access List who are already in one of the groups in the Access List.
- Flags what it terms ‘ghosts’: deleted users or groups still defined in RACF, such as an access list; also has a ghost wizard.
- Recommendation: start the z/OS SecurityCenter started task at IPL time to avoid delays on first use.
- Users do not require a TSO segment.
- Honours RACF access rights, as to what each user can see within RACF, just as RACF panels do.
- Help Desk can use it for password resets, without RACF SPECIAL being defined for each user.

IDEAS

On my return to the office, I listed 15 ideas that might have merit in our own environment that were sparked by the conference. Reviewing them, I see that they would apply to most mainframe environments.

1 Vanguard SecurityCenter is a Windows GUI-based replacement for Vanguard Administrator, not an add-on.

Most organizations will find that SecurityCenter does everything they need and can replace Administrator.

2 Vanguard has seemingly more robust alternatives to Tivoli Identity Manager, which integrate all platforms with RACF.

3 Our focus has been on high availability, as a major selling point of the mainframe as a platform, but z/OS can be much more secure, too, especially its Unix (including TCP/IP) environment. In fact, set up with a bit of care and using SSL where appropriate, z/OS is the most secure environment to host Web servers, both intranet and Internet. Plus, the mainframe has dedicated encryption hardware that effectively eliminates the processing penalty associated with encryption.

4 We could delete the RACF user ID of each ex-employee without deleting their datasets by simply:

- Renaming all the datasets to another high-level qualifier (eg X. instead of Ennnnnn.).
- Deleting the RACF user ID.
- Renaming all the datasets (back to Ennnnnn.), which may require redefining the ALIAS

This should satisfy the Auditor's recommendation.

5 If a RACF audit is ever required, I learned enough to get it done, depending on the requirements – either doing myself from the conference notes and referenced manuals or selecting and managing a consultant.

6 Subsets of most Vanguard products can be given to the Help Desk, allowing them to do password resets and more, without SPECIAL or other RACF extraordinary attributes.

7 We should be monitoring to prevent RACF database full, which means trouble.

8 Alerting a security staff member whenever an Admin

account is created on any Windows server would be a good first step against hackers.

9 Disallowing wireless NICs on all workstations, including laptops, plugged into the corporate network, including remote users coming in with VPN, is an even more important hacker prevention tactic. The easiest way to hack into many corporations is to sit in the company parking lot and look for laptops with wireless NIC cards that are searching for the employee's home wireless network.

10 The hacking and VPN speakers both agreed that the form of VPN that prevents split tunnelling, coupled with up-to-date anti-virus and the NAT firewall in even the cheapest Internet router, will fully protect remote workstations.

11 Ensure that procedures are in place to prevent dial-up modems on (LAN-connected) corporate workstations from answering the phone line they are attached to.

12 Explore mainframe terminal emulators that encrypt mainframe passwords so they cannot be 'sniffed' off the LAN.

13 Investigate the security requirements for and performance issues against using RACF Erase on Scratch to delete the data when a dataset is deleted.

14 Look at the need for each external link to be two-way. For example, a one-way link to an external agency would prevent a hacker on the other end from accessing your network.

15 Never reassign RACF user IDs to another person. When reassigning a deleted RACF user ID to the same person, be aware that they may inherit some or all of their past access: membership of groups, etc. Vanguard products can prevent this, but who knows if the user ID was deleted properly?

Jon E Pearkins
Adiant (Canada)

Monitoring available free space in our RACF database

We experienced a situation recently in which our RACF database ran out of free space. This in turn stopped our RACF administrators defining new profiles.

IBM's utility program IRRUT200 can be used to report on how much free space is available. To create an automated reporting process would have meant reading then parsing the output from this program. I decided it'd be a lot easier (and faster) retrieving and reporting on free space with my own program.

I created program LSGRABAM to read the RACF database directly and process the information in the Block Availability Mask (BAM) blocks. At this point I found the *Security Server RACF Diagnosis Guide* manual invaluable. It gave a breakdown of the structure of the database and allowed me to translate information in the BAM block.

The program uses BDAM to step through the database, reading only the first record in the database, the ICB, and then each BAM block.

The program allows you to pass a threshold value via a parameter setting. This is a numeric value, in the range of 1 to 99, which is used as a percentage free threshold.

If there is a threshold exception, too little free space, the program issues a WTO. In our site we use the information in the WTO to generate an incident record in our problem tracking system.

You can code the SYSRACF DD statement, in the JCL, to point to any RACF database. This could allow you to monitor the back-up database as well as the primary database.

The return code from this code is always set to 0 (unless of course there is an abend).

The program runs in problem state because it doesn’t use (or need) any special system services. It’s been coded to run in 31-bit addressing mode. I’ve had to getmain below the line storage for some of the I/O control blocks. Code has been written as re-entrant but doesn’t need to be link-edited as RENT to function.

JCL TO RUN THE PROGRAM

```
//MVAUDP35 JOB (,IS),'RACF-DB-SPACE',CLASS=4,MSGCLASS=X
//*
//STEPØØØ1 EXEC PGM=LSGRABAM,PARM=2Ø
//SYSRACF  DD DISP=SHR,DSN=SYS1.RACF
//SYSPRINT DD SYSOUT=J,LRECL=8Ø,RECFM=FB
```

The parameter is a percentage value between 1 and 99

Note: the ID assigned to the batch job needs read access to the RACF dataset.

OUTPUT FROM THE PROGRAM

This is an example of output written to DDname SYSPRINT:

```
   TOTAL BAM BLOCKS             ØØØØ2484
   FREE BAM BLOCKS              ØØØØ1846
   FREE 256 BYTE SLOTS          ØØØ3Ø11Ø
   PERCENTAGE FREE BAM BLOCKS       Ø74%
   PERCENTAGE FREE BAM BLOCKS LIMIT Ø2Ø%
```

If the threshold (set in the JCL parm statement) is exceeded, the following WTO is generated:

```
   LSGRABAM - RACF DB FREE SPACE WARNING, <Øxx% FREE
```

LSGRABAM

```
LSGRABAM AMODE 31
LSGRABAM RMODE ANY
LSGRABAM RSECT
         DS    ØH
         B     BEGIN-LSGRABAM(,15)
PROGRAM  DC    C'LSGRABAM: '
         DC    C'&SYSDATE &SYSTIME '
         DS    ØH
```

```
BEGIN    EQU   *
         BAKR  14,Ø
         LR    12,15
         LR    5,1
         USING LSGRABAM,12
         USING WORKAREA,11
         USING IHADCB,9
         L     2,=A(WORK_AREA_LENGTH)
         STORAGE OBTAIN,LENGTH=(2)
         LR    11,1
         ST    11,GETMAIN_ADDRESS
         LA    13,SAVEAREA
         MVC   SAVEAREA+4(4),=C'F1SA'
         MVC   EYECATCHER,PROGRAM
*
         L     2,=A(BELOW_DSECT_LENGTH)
         STORAGE OBTAIN,LENGTH=(2),LOC=BELOW
         LR    1Ø,1
         ST    1Ø,BELOW_STORAGE_SAVEAREA_ADDRESS
         USING BELOW_DSECT,1Ø
         MVC   EYECATCHR2,PROGRAM
         MVC   WTO_TEXT_,WTO_TEXT
         MVC   WTO_,WTO#
*
         L     5,Ø(,5)
         LH    9,Ø(,5)
         CH    9,=H'Ø'
         BNH   EXIT_POINT
         CH    9,=H'3'
         BNL   EXIT_POINT
*
         LA    1,L'PARMVALUE
         SR    1,9
         MVC   MOVE_CHAR_,MOVE_CHAR#
         STC   1,MOVE_CHAR_+3
         XC    PARMVALUE,PARMVALUE
         BCTR  9,Ø
         LA    4,PARMVALUE
         EX    9,MOVE_CHAR_
*
         XC    PACK_DOUBLE_WORD(8),PACK_DOUBLE_WORD
         PACK  PACK_,PARMVALUE
         XC    REQD_PERC_FREE,REQD_PERC_FREE
         MVC   REQD_PERC_FREE+1(3),PACK_
*
*
         MVC   OPENO_,OPENO#
         MVC   OPENI_,OPENI#
         MVC   CLOSE_,CLOSE#
         MVC   OUTPUTDCB_,OUTPUTDCB#
         MVC   INPUTDCB_,INPUTDCB#
         MVC   EOF_,EOF#
         LA    1,EOF_
         STCM  1,B'Ø111',INPUTDCB_+33
*
OPEN_OUTPUT_FILE EQU *
         OPEN  (OUTPUTDCB_,OUTPUT),MODE=31,MF=(E,OPENO_)
         LA    9,OUTPUTDCB_
         LA    15,CHECK_OPEN_ROUTINE
         BALR  14,15
         LTR   15,15
         BNZ   EXIT_POINT
         MVC   RECORD_LENGTH,DCBLRECL
*
OPEN_INPUT_FILE EQU *
         OPEN  INPUTDCB_,MODE=31,MF=(E,OPENI_)
         LA    9,INPUTDCB_
         LA    15,CHECK_OPEN_ROUTINE
         BALR  14,15
         LTR   15,15
         BNZ   EXIT_POINT
*
CHECK_FOR_ICB EQU *
         XC    BLOCK_ADDR,BLOCK_ADDR
         LA    15,READ_ROUTINE
         BALR  14,15
*
         CLC   =F'Ø',BLOCK_DATA
         BE    FOUND_ICB
*
         DC    F'Ø'
*
FOUND_ICB EQU  *
         ICM   6,15,BLOCK_DATA+4
         MVC   BAM_HWM,BLOCK_DATA+28
*
CHECK_FOR_BAM_INIT EQU *
         ICM   5,B'Ø111',DCBREL
         LR    2,5
         A     2,=F'2'
         MVC   BLOCK_ADDR,=XL3'Ø1'
*
         ZAP   BAM_256_FREE,PACKEDØ
         ZAP   BAM_4Ø96_FREE,PACKEDØ
         SLR   8,8
*
CHECK_FOR_BAM EQU *
         LA    15,READ_ROUTINE
         BALR  14,15
*
```

```
         CLI   BLOCK_DATA,X'ØØ'
         BNE   POINT_TO_NEXT_RECORD
*
         LTR   6,6
         BZ    CLOSE_RACF_DATABASE
*
         BCTR  6,Ø
*
*        SLR   1,1
*        ICM   1,B'Ø111',BLOCK_ADDR
*        M     Ø,=A(L'BLOCK_DATA)
*
FOUND_BAM EQU  *
         SLR   Ø,Ø
         SLR   1,1
         ICM   Ø,B'ØØ11',BLOCK_DATA+6
         ICM   1,B'1111',BLOCK_DATA+8
         D     Ø,=A(L'BLOCK_DATA)
         LR    4,1
*
FORMAT_BAM EQU *
         SLR   3,3
         ICM   3,B'ØØ11',BLOCK_DATA+18
         LA    7,BLOCK_DATA+2Ø
         AR    8,3
*
COUNT_FREE_SPACE EQU *
         SLR   1,1
         ICM   1,B'ØØ11',Ø(7)
         BZ    COUNT_FREE_SPACE_LOOP
*
         CLC   =XL2'FFFF',Ø(7)
         BNE   COUNT_FREE_SPACE_256
*
COUNT_FREE_SPACE_4Ø96 EQU *
         AP    BAM_4Ø96_FREE,PACKED1
*
COUNT_FREE_SPACE_256 EQU *
         AP    BAM_256_FREE,PACKED1
*
         SRL   1,1
         LTR   1,1
         BNZ   COUNT_FREE_SPACE_256
*
COUNT_FREE_SPACE_LOOP EQU *
         LA    7,2(,7)
         BCT   3,COUNT_FREE_SPACE
*
COUNT_FREE_SPACE_END EQU *
         LTR   4,4
         BZ    CLOSE_RACF_DATABASE
*
         B     CHECK_FOR_BAM_NEXT
*
*
POINT_TO_NEXT_RECORD EQU *
         LR    4,2
         SR    4,5
*
CHECK_FOR_BAM_NEXT EQU *
         STCM  4,B'Ø111',BLOCK_ADDR
         XC    DECB(DECB_LENGTH),DECB
         BCT   5,CHECK_FOR_BAM
*
*
*
CLOSE_RACF_DATABASE EQU *
         CLOSE INPUTDCB_,MODE=31,MF=(E,CLOSE_)
*
OUTPUT_REPORT EQU *
         CVD   8,BAM_ALLOC
         ZAP   BAM_PERC_FREE,PACKED1ØØ
         MP    BAM_PERC_FREE,BAM_4Ø96_FREE
         DP    BAM_PERC_FREE,BAM_ALLOC
*
         XC    PWORK8,PWORK8
         UNPK  PWORK8,BAM_ALLOC
         MVZ   PWORK8+7(1),FØ
*
         LA    15,PUT_LINE
         BALR  14,15
         MVC   Ø(L'LINE_1,3),LINE_1
         MVC   29(8,3),PWORK8
*
         XC    PWORK8,PWORK8
         UNPK  PWORK8,BAM_4Ø96_FREE
         MVZ   PWORK8+7(1),FØ
*
         LA    15,PUT_LINE
         BALR  14,15
         MVC   Ø(L'LINE_2,3),LINE_2
         MVC   29(8,3),PWORK8
*
         XC    PWORK8,PWORK8
         UNPK  PWORK8,BAM_256_FREE
         MVZ   PWORK8+7(1),FØ
*
         LA    15,PUT_LINE
         BALR  14,15
         MVC   Ø(L'LINE_3,3),LINE_3
         MVC   29(8,3),PWORK8
*
         XC    PWORK3,PWORK3
         UNPK  PWORK3,BAM_PERC_FREE+6(2)
         MVZ   PWORK3+2(1),FØ
*
         LA    15,PUT_LINE
         BALR  14,15
         MVC   Ø(L'LINE_4,3),LINE_4
         MVC   33(3,3),PWORK3
*
         XC    PWORK3,PWORK3
         UNPK  PWORK3,REQD_PERC_FREE+2(2)
         MVZ   PWORK3+2(1),FØ
*
         LA    15,PUT_LINE
         BALR  14,15
         MVC   Ø(L'LINE_5,3),LINE_5
         MVC   33(3,3),PWORK3
         MVC   WTO_TEXT_+4Ø(3),PWORK3
*
CLOSE_OUTPUT_FILE EQU *
         CLOSE OUTPUTDCB_,MODE=31,MF=(E,CLOSE_)
*
CHECK_PERC_FREE EQU *
         CP    BAM_PERC_FREE+6(2),REQD_PERC_FREE+2(2)
         BH    EXIT_POINT
*
         L     1,=AL4(L'WTO_TEXT)
         STH   1,WTO_TEXTL_
         LA    2,WTO_TEXTL_
         WTO   TEXT=((2)),MF=(E,WTO_)
*
EXIT_POINT DS  ØH
         L     2,=A(BELOW_DSECT_LENGTH)
         L     3,BELOW_STORAGE_SAVEAREA_ADDRESS
         STORAGE RELEASE,LENGTH=(2),ADDR=(3)
         L     2,=A(WORK_AREA_LENGTH)
         L     3,GETMAIN_ADDRESS
         STORAGE RELEASE,LENGTH=(2),ADDR=(3)
         SR    15,15
         PR    ,
*
READ_ROUTINE EQU *
         ST    14,SAVE_GPR14
         READ  DECB,DIF,(9),BLOCK_DATA,L'BLOCK_DATA,Ø,BLOCK_ADDR,MF=E
         CHECK DECB
         L     14,SAVE_GPR14
         BR    14
*
```

```
READ_ERROR EQU *
         SLR   Ø,Ø
         SLR   15,15
         ICM   Ø,B'Ø111',BLOCK_ADDR
         ICM   15,B'ØØ11',DECB+1
         DC    F'Ø'
*
*
*
CHECK_OPEN_ROUTINE EQU *
         SR    15,15
         TM    DCBOFLGS,DCBOFOPN
         BO    CHECK_OPEN_ROUTINE_END
         LA    15,8
CHECK_OPEN_ROUTINE_END EQU *
         BR    14
*
PUT_LINE EQU   *
         ST    14,SAVE_GPR14
         PUT   OUTPUTDCB_
         LR    Ø,1
         LR    3,1
         LH    1,RECORD_LENGTH
         LA    14,Ø
         LA    15,C' '
         SLL   15,24
         MVCL  Ø,14
PUT_LINE_END EQU *
         L     14,SAVE_GPR14
         BR    14
*
OUTPUTDCB# DCB DSORG=PS,MACRF=PL,DDNAME=SYSPRINT
OUTPUTDCB#_LENGTH EQU *-OUTPUTDCB#
*
INPUTDCB# DCB  DSORG=DA,RECFM=F,                                      X
               DDNAME=SYSRACF,                                        X
               MACRF=(RIC),                                           X
               OPTCD=RF,                                              X
               SYNAD=READ_ERROR,                                      X
               EODAD=CLOSE_RACF_DATABASE
INPUTDCB#_LENGTH EQU *-INPUTDCB#
*             MACRF=(RISC),                                          X
*             BUFL=4Ø96,                                             X
*             BUFNO=2,                                               X
*
OPENI#   OPEN  INPUTDCB#,MODE=31,MF=L
OPENI#_LENGTH EQU *-OPENI#
*
OPENO#   OPEN  (OUTPUTDCB#,OUTPUT),MODE=31,MF=L
OPENO#_LENGTH EQU *-OPENO#
```

```
*
CLOSE#   CLOSE INPUTDCB#,MODE=31,MF=L
CLOSE#_LENGTH EQU *-CLOSE#
*
EOF#     B    CLOSE_RACF_DATABASE
EOF#LENGTH EQU *-EOF#
*
FØ       DC    XL1'FØ'
PACKED1  DC    P'1'
PACKED1ØØ DC   P'1ØØ'
PACKEDØ  DC    P'Ø'
*
LINE_1   DC    C'TOTAL BAM BLOCKS             XXXXXXXX'
LINE_2   DC    C'FREE BAM BLOCKS              XXXXXXXX'
LINE_3   DC    C'FREE 256 BYTE SLOTS          29XXXXXX'
LINE_4   DC    C'PERCENTAGE FREE BAM BLOCKS       33X%'
LINE_5   DC    C'PERCENTAGE FREE BAM BLOCKS LIMIT 33X%'
*
WTO#     WTO   TEXT=,MF=L
WTO_LENGTH EQU *-WTO#
WTO_TEXT DC    C'LSGRABAM - RACF DB FREE SPACE WARNING, <4ØX% FREE'
*
*
MOVE_CHAR# MVC Ø(Ø,4),2(5)
MOVE_CHAR_LENGTH EQU *-MOVE_CHAR#
*
         LTORG ,
*
*
WORKAREA DSECT
SAVEAREA DS    18F
EYECATCHER DS  CL8
GETMAIN_ADDRESS DS F
BELOW_STORAGE_SAVEAREA_ADDRESS DS F
SAVE_GPR14 DS  F
*
OPENO_   DS    CL(OPENO#_LENGTH)
OPENI_   DS    CL(OPENI#_LENGTH)
CLOSE_   DS    CL(CLOSE#_LENGTH)
*
RECORD_LENGTH DS H
*
BAM_HWM  DS    CL6
*
BAM_256_FREE DS D
BAM_4Ø96_FREE DS D
BAM_PERC_FREE DS L
BAM_ALLOC DS   D
PWORK3    DS   CL3
PWORK8    DS   CL8
```

```
*
         DS    ØF
MOVE_CHAR_ DS  CL(MOVE_CHAR_LENGTH)
*
PARMVALUE DS   F
PACK_DOUBLE_WORD DS ØD
         DS    CL5
PACK_    DS    PL3
*
REQD_PERC_FREE DS F
*
WTO_     DS    CL(WTO_LENGTH)
WTO_TEXTL_ DS  H
WTO_TEXT_ DS   CL(L'WTO_TEXT)
*
BLOCK_DATA DS  CL4Ø96  ***  KEEP AT END OF WORKAREA (ADDR. ISSUES)
*
WORK_AREA_LENGTH EQU *-WORKAREA
*
*
BELOW_DSECT DSECT
EYECATCHR2 DS  CL8
         READ  DECB,DIF,MF=L
DECB_LENGTH EQU *-DECB
INPUTDCB_ DS   CL(INPUTDCB#_LENGTH)
OUTPUTDCB_ DS  CL(OUTPUTDCB#_LENGTH)
EOF_     DS    CL(EOF#LENGTH)
         DS    ØF
BLOCK_ADDR DS  FL3
BELOW_DSECT_LENGTH EQU *-BELOW_DSECT
*
         PRINT OFF
         DCBD  DSORG=(DA,PS)
         END
```

Calum Reid
Systems Programmer (UK)

RACF in focus – SYS1.UADS dataset

This is a regular column focusing on specific aspects of RACF. In this issue we will discuss the SYS1.UADS dataset, its historical importance, and what role it plays in the present day

RACF environment. SYS1.UADS is also known as the User Attribute Data Set, or UADS for short.

The SYS1.UADS dataset exists in all MVS (or OS/390 or z/OS) installations – the name is fixed; it cannot be called anything else. It looks like an operating system dataset, and indeed the prefix SYS1 implies that. And it looks like it has nothing to do with security – many security folk are not aware of this dataset's existence, much less its significance.

But as we shall see it plays an important security role, even to this day, and works together with RACF. Admittedly, its use comes into play only when logging on to TSO, but this in itself is significant because once inside TSO you can do any number of things.

We will first look at SYS1.UADS from an historical perspective, then see how it works in conjunction with RACF, and, finally, list some recommendations on maintaining this important dataset so security is not compromised.

A BRIEF HISTORY OF SYS1.UADS

The origins of SYS1.UADS date back to pre-RACF days! There was SYS1.UADS even when there was no RACF (or other security product such as ACF2 and Top Secret).

At that time, it was used to control access to TSO, by authenticating and validating userids and passwords for TSO users only. It also validated the use of TSO resources such as TSO account numbers, TSO region sizes, TSO log-on procedure names, and so on. In this sense SYS1.UADS played the role of RACF. But, unlike RACF, it played no part in access control to datasets and other resources – just validation of TSO users and their TSO attributes.

There were several disadvantages to using this as your primary means of security:

1 The passwords were stored in clear text.

2 People with TSO account authority could view everyone's passwords, leaving open the possibility of misuse.

3 If you wanted to change your password, you had to go to someone with the TSO ACCOUNT authority, and they would do it for you.

When RACF was introduced it provided the functions of SYS1.UADS, and the idea was to migrate information contained in SYS1.UADS to TSO segments of RACF userid profiles. This placed all security in one central place and negated the need to maintain SYS1.UADS in addition to RACF.

Now one would think that with the introduction of RACF, SYS1.UADS would become obsolete, and it would go down in history as the old method of administering TSO security. But that is not the case. SYS1.UADS still works, but in conjunction with RACF, as follows. In typical IBM fashion, the features of SYS1.UADS were not removed; rather, they were replaced with better methods.

Although SYS1.UADS would still work, it is hard to believe anyone would want to go to the trouble of maintaining all TSO information for all users in SYS1.UADS, and use that to validate TSO users. Most installations did the migration of TSO information from SYS1.UADS to RACF (or other security product). But some may have forgotten to remove the old SYS1.UADS entries, leaving open a potential security exposure. If you want to see how your installation is doing, just browse SYS1.UADS (it is a partitioned dataset) and see how many entries you find!

HOW SYS1.UADS WORKS WITH RACF

In RACF, the use of TSO is controlled by the TSO segment of the userid profile – if the userid has a TSO segment, it can log-on to TSO; otherwise not, with some exceptions, as described below.

Essentially, if RACF is active, and a userid attempts to log on to TSO, then SYS1.UADS comes into play in one of two situations:

1 If the userid is defined to RACF but does not have a TSO segment, SYS1.UADS is checked to see whether the userid is defined to SYS1.UADS. If the userid is defined there, TSO log-on continues.

2 If the userid is not defined to RACF, SYS1.UADS is checked to see whether an entry exists for this userid. If it does, TSO log-on continues.

In other words, SYS1.UADS is referenced in cases where appropriate TSO information is not found in RACF.

If RACF is inactive, and a user tries to log on to TSO, the log-on will succeed if the password and other TSO attributes specified at log-on match those found in the SYS1.UADS entry for that userid.

This can be very useful in emergency situations, because it allows someone to log on to TSO even when RACF is not active. And TSO would typically be needed to 'fix' the RACF problem.

KEEPING SYS1.UADS CURRENT

IBM provides the TSO ACCOUNT command to administer and maintain the SYS1.UADS dataset. With the ACCOUNT command in TSO, you can list, add, delete, and change userid information in SYS1.UADS. ACCOUNT has subcommands to perform its functions (list, add, delete, and change). We will not describe the subcommands of the ACCOUNT command because they are readily available by entering HELP once inside the ACCOUNT command.

Although you could write some exit code to disable SYS1.UADS checking altogether, it is desirable to keep its features available for emergency use.

But you should note that not keeping SYS1.UADS current is one of the indirect ways in which the security at your installation is compromised.

RECOMMENDATIONS

Here are some recommendations:

1 For emergency situations, including disaster recovery, it is recommended that IBMUSER and the systems programmers' userids be correctly defined in SYS1.UADS. Assign appropriate passwords, and keep in a secure place a hardcopy listing of all the details, such as userid and password, of these few individuals. This will come in handy in emergency situations.

2 Delete all other entries from SYS1.UADS to prevent compromising the system. But before you delete these, make sure you have defined appropriate TSO segments for the userids you are deleting. You never know whether users were logging on to TSO by virtue of these supposedly obsolete SYS1.UADS entries!

3 Specify a TSO segment in RACF for valid SYS1.UADS entries (IBMUSER and the systems programmers' userids). This will prevent information from SYS1.UADS being used when RACF is available.

4 Perform a periodic review of the SYS1.UADS dataset to make sure there are no redundant entries, and that the TSO attributes specified for IBMUSER and the systems programmers' userids are still valid.

5 Limit the number of people having the TSO ACCOUNT authority. It is a powerful authority, and should be treated just as you would treat other powerful RACF authorities.

Dinesh Dattani would welcome feedback, comments and queries about this column. He can be contacted at dinesh123@rogers.com.

Dinesh Dattani
Security Consultant
Toronto (Canada)

DB2 V7 to RACF utility

This DB2R REXX procedure converts DB2 GRANT statements to the equivalent RACF profiles and commands. The RACF commands provide access control to the DB2 V7 objects.

The procedure generates a RACF command for the following DB2 objects:

- Collection privilege – defines the privilege to use the BIND subcommand to create packages in the designated collections.
- Database privilege – defines the database administration, control, and maintenance authority. Defines the privilege to create new tables and new table spaces. Defines the START, DISPLAY, STOP, DROP, or ALTER database authority. Defines the privilege to run COPY, MERGECOPY, QUIESCE, MODIFY, LOAD, RECOVER, REPORT, REORG, REPAIR, RUNSTATS, and DIAGNOSE utilities.
- Function and procedure privilege – defines the privilege to run the user-defined functions and stored procedures.
- Package privilege – defines the privilege to use the BIND and REBIND subcommands for the packages and the privilege to run application programs that use the packages.
- Plan privilege – defines the privilege to use the BIND, REBIND, and FREE subcommands for the identified plans and the privilege to run programs that use the identified plans.
- Schema privilege – defines the privilege to create, alter, and drop stored procedures, user-defined functions, distinct types, and triggers in the designated schemas.
- Resource privilege – defines the DB2 subsystem privilege. The system commands are: ARCHIVE LOG and SET LOG, BINDADD commands to create plans and packages; BINDAGENT commands to BIND, FREE PACKAGE, or

REBIND plans and packages; BSDS command to run RECOVER bootstrap dataset; CREATEDBA to create database statement; CREATEALIS to create an alias statement, CREATESG to create new storage groups; CREATETMTAB to create a global temporary table; DISPLAY status and description of DB2 system objects; and STOSPACE to run the STOSPACE utility, SYSADM authority, SYSCTRL, and SYSOPR authority.

- Table, View, Trigger privilege – defines the ALTER, DELETE, INDEX, INSERT, SELECT, REFERENCES, TRIGGER, and UPDATE authority to the table.
- Bufferpool, Stogroup, Tablespace privilege – defines the authority to use particular buffer pools, storage groups, and table spaces.

The procedure does not execute any RACF commands, it only generates them and writes them to the ISPF skeleton file.

The procedure uses the following DB2 V7 objects and RACF class names:

- Collection – MDSNCL
- Database – MDSNDB
- Stored procedure – MDSNSP
- User-defined function – MDSNUF
- Package – MDSNPK
- Plan – MDSNPL
- Schema – MDSNSC
- System resource – MDSNSM
- SYSADM, SYSOPR, SYSCTRL – DSNADM
- Bindagent – MDSNSM
- Table, view, trigger – MDSNSG
- Storage group – MDSNSG

- Table space – MDSNTS
- Bufferpool – MDSNBP.

The components of DB2R are as follows:

- DB2R – the driver procedure
- DB2RM0 – main menu panel
- DB2RM1 – selection result panel
- DB2RM2 – message panel
- RACFS – ISPF skeleton.

DB2R

```
/* REXX */
/* DB2 and RACF Utility                                      */
/* sqlid: NADI                                               */
/* trace r */
 zpfctl = 'OFF'
 address ispexec 'vput (zpfctl) profile'
 Y=MSG("OFF")
 CUR='F1'
 address ispexec "display panel(db2rmØ) cursor("CUR")"
 sqlid='NADI'
 date=date()
 time=time(c)
 user=userid()
 do while rc=Ø
    if kurs='F1' | kurs='FIELD1' then do
         Call Connection
         Clas='MDSNCL'
         Db2c='CREATEIN'
         Call Init
         Call Collection
         if Result=Ø then Call Skeleton
         CUR='F1'
    end
    if kurs='F2' | kurs='FIELD2' then do
         Call Connection
         Call Dba_Privileges
         if Result=Ø then Call Skeleton
         CUR='F2'
    end
    if kurs='F3' | kurs='FIELD3' then do
         Call Connection
```

```
          Db2c='EXECUTE'
          Call Init
          Call Routine_Auth
          Call Routine_Ralt
          Call Routine_Opt
          if Result=Ø then Call Skeleton
          CUR='F3'
     end
     if kurs='F4' | kurs='FIELD4' then do
          Call Connection
          Call Package_Privileges
          if Result=Ø then Call Skeleton
          CUR='F4'
     end
     if kurs='F5' | kurs='FIELD5' then do
          Call Connection
          Call Dba_Plan
          if Result=Ø then Call Skeleton
          CUR='F5'
     end
     if kurs='F6' | kurs='FIELD6' then do
          Call Connection
          Call Schema_privileges
          if Result=Ø then Call Skeleton
          CUR='F6'
     end
     if kurs='F7' | kurs='FIELD7' then do
          Call Connection
          Call System_Privileges
          if Result=Ø then Call Skeleton
          CUR='F7'
     end
     if kurs='F8' | kurs='FIELD8' then do
          Call Connection
          Call Tab_Privilege
          if Result=Ø then Call Skeleton
          CUR='F8'
     end
     if kurs='F9' | kurs='FIELD9' then do
          Call Connection
          Call Use_Privilege
          if Result=Ø then Call Skeleton
          CUR='F9'
     end
     address ispexec "display panel(db2rmØ) cursor("CUR")"
  end
  Exit

  /* Collection privileges                                            */
  Collection:
```

```
SQLSTMT= "SELECT DISTINCT STRIP(NAME)                              ",
"FROM SYSIBM.SYSRESAUTH                                          ",
"WHERE OBTYPE='C'                                                ",
"  AND QUALIFIER='        '                                      ",
"  WITH UR                                                       "

Call Cursor
address dsnrexx "execsql fetch c1 into :dbitem"

row = '/* '||db2c||' Package privilege */'
address ispexec 'tbadd "rlist"'
do while(sqlcode=0)
   row='RDEF '||clas||' '||ssid||'.'||dbitem||'.'||db2c||,
       ' UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
   address ispexec 'tbadd "rlist"'
   row='PERMIT '||ssid||'.'||dbitem||'.'||db2c||,
       ' CLASS('||clas||') RESET'
   address ispexec 'tbadd "rlist"'
   address dsnrexx "execsql fetch c1 into :dbitem"
end
address dsnrexx "execsql close c1"

SQLSTMT= "SELECT DISTINCT STRIP(NAME)                              ",
"FROM SYSIBM.SYSRESAUTH                                          ",
"WHERE OBTYPE='C'                                                ",
"  AND QUALIFIER='        '                                      ",
"  AND GRANTEE IN ('PUBLIC','PUBLIC*')                           ",
"  AND USEAUTH¬=' '                                              ",
"  WITH UR                                                       "

Call Cursor
address dsnrexx "execsql fetch c1 into :dbitem"

address ispexec 'tbadd "rlist"'
do while(sqlcode=0)
   row='RALT '||clas||' '||ssid||'.'||dbitem||'.'||db2c||,
       ' UACC(READ)'
   address ispexec 'tbadd "rlist"'
   address dsnrexx "execsql fetch c1 into :dbitem"
end
address dsnrexx "execsql close c1"

SQLSTMT= "SELECT DISTINCT STRIP(NAME), STRIP(GRANTEE),             ",
"CASE(USEAUTH)                                                   ",
"  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                             ",
"END                                                             ",
"FROM SYSIBM.SYSRESAUTH                                          ",
"WHERE GRANTEE NOT IN ('PUBLIC','PUBLIC*')                       ",
"  AND QUALIFIER='        '                                      ",
"  AND OBTYPE='C'                                                ",
"  AND NAME='*'                                                        ",
"  AND USEAUTH IN ('Y','G')                                            ",
"  WITH UR                                                             "

Call Cursor
address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"

do while(sqlcode=0)
   row='PERMIT '||ssid||'.'||dbn||'.'||db2c||,
       ' CLASS('||clas||') ACC('||ucc||') ID('||gra||')'
   address ispexec 'tbadd "rlist"'
   address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"
end
address dsnrexx "execsql close c1"

SQLSTMT= "SELECT DISTINCT STRIP(NAME)                                          ",
"FROM SYSIBM.SYSRESAUTH                                                ",
"WHERE OBTYPE='C'                                                      ",
"  AND QUALIFIER='        '                                            ",
"  AND NAME¬='*'                                                       ",
"  AND EXISTS(SELECT 1 FROM SYSIBM.SYSRESAUTH                          ",
"             WHERE NAME='*'                                           ",
"               AND OBTYPE='C' AND QUALIFIER='        ')               ",
"  WITH UR                                                             "

Call Cursor
address dsnrexx "execsql fetch c1 into :dbn"

do while(sqlcode=0)
   row='PERMIT '||ssid||'.'||dbn||'.'||db2c||,
       ' CLASS('||clas||') FROM('||ssid||'.*.'||db2c||')'
   address ispexec 'tbadd "rlist"'
   address dsnrexx "execsql fetch c1 into :dbn"
end
address dsnrexx "execsql close c1"

SQLSTMT= "SELECT DISTINCT STRIP(NAME), STRIP(GRANTEE),                         ",
"CASE(USEAUTH)                                                         ",
"  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                                   ",
"END                                                                   ",
"FROM SYSIBM.SYSRESAUTH                                                ",
"WHERE GRANTEE NOT IN ('PUBLIC','PUBLIC*')                             ",
"  AND QUALIFIER='        '                                            ",
"  AND OBTYPE='C'                                                      ",
"  AND NAME¬='*'                                                       ",
"  AND USEAUTH IN ('Y','G')                                            ",
"  WITH UR                                                             "

Call Cursor
address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"
```

```
 do while(sqlcode=0)
    row='PERMIT '||ssid||'.'||dbn||'.'||db2c||,
        ' CLASS('||clas||') ACC('||ucc||') ID('||gra||')'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"
 end
 address dsnrexx "execsql close c1"
Return 0
/* End Collection privileges                                    */

/* Database privileges                                          */
Dba_Privileges:
 SQLSTMT= "SELECT NAME,                                          ",
 "          REPLACE(REPLACE(NAME,'AUTH',''),'STOP','STOPDB')     ",
 "FROM SYSIBM.SYSCOLUMNS                                         ",
 "WHERE TBCREATOR='SYSIBM'                                       ",
 "  AND TBNAME='SYSDBAUTH'                                       ",
 "  AND NAME LIKE '%AUTH%'                                       ",
 "  AND COLNO > 8                                                ",
 "  ORDER BY NAME                                                ",
 "  WITH UR                                                      "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
 Call Init

 do while(sqlcode=0)
    address ispexec 'tbadd "slist"'
    address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
 end
 address dsnrexx "execsql close c1"
 address ispexec 'tbtop "slist"'
 address ispexec 'tbdispl "slist" panel(db2rm1)'
 if rc=8 then do
    address ispexec 'tbend "slist"'
    address ispexec 'tbend "rlist"'
    address ispexec "tbend "messdb""
    call create_messg
    return 99
 end
 if rc<8 & cmd='S' then do
    Clas='MDSNDB'
    do while ztdsels > 0
       if db2c='DBADM' | db2c='DBCTRL' | db2c='DBMAINT' then do
          Call Database
          Call Dba_Detail
          if db2c='DBADM' then Call Dba_Opt
       end
       else do
          Call Db_Util
          Call Db_Util_Pub
          Call Db_Util_Det
       end
       if ztdsels = 1 then ztdsels=0
       if ztdsels > 0 then address ispexec 'tbdispl "slist" '
    end
    address ispexec 'tbtop "rlist"'
 end
Return 0

Database:
 SQLSTMT= "SELECT DISTINCT STRIP(NAME)                              ",
 "FROM SYSIBM.SYSDBAUTH                                            ",
 "  WITH UR                                                        "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbitem"

 row = '/* '||db2c||' privilege */'
 address ispexec 'tbadd "rlist"'
 do while(sqlcode=0)
    row='RDEF '||clas||' '||ssid||'.'||dbitem||'.'||db2c||,
        ' UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
    address ispexec 'tbadd "rlist"'
    row='PE '||ssid||'.'||dbitem||'.'||db2c||' CLASS('||clas||') RESET'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbitem"
 end
 address dsnrexx "execsql close c1"
Return 0

Dba_Detail:
 /* Select Dbadm  authority */
 SQLSTMT= "SELECT DISTINCT STRIP(NAME)                              ",
 "FROM SYSIBM.SYSDBAUTH                                            ",
 "WHERE "db2obj" ¬=' '                                             ",
 "  AND GRANTEE ='PUBLIC'                                          ",
 "  WITH UR                                                        "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbn"
 do while(sqlcode=0)
    row='RALT '||clas||' '||ssid||'.'||dbn||'.'||db2c||' UACC(READ)'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbn"
 end
 address dsnrexx "execsql close c1"

 SQLSTMT= "SELECT DISTINCT STRIP(NAME), STRIP(GRANTEE),             ",
 "CASE("db2obj")                                                   ",
 "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                               ",
 "END                                                               ",
 "FROM SYSIBM.SYSDBAUTH                                             ",
 "WHERE "db2obj" IN ('Y','G')                                       ",
 "  AND GRANTEE¬='PUBLIC'                                           ",
 "  WITH UR                                                         "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbn, :gra, :acc"
 do while(sqlcode=Ø)
    row='PERMIT '||ssid||'.'||dbn||'.'||db2c||,
        ' CLASS('||clas||') '||'ACC('||acc||') '||'ID('||gra||')'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbn, :gra, :acc"
 end
 address dsnrexx "execsql close c1"
Return

Db_Util:
 SQLSTMT= "SELECT DISTINCT STRIP(NAME)                              ",
 "FROM SYSIBM.SYSDBAUTH                                             ",
 "  WITH UR                                                         "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbitem"

 row = '/* '||db2c||' privilege */'
 address ispexec 'tbadd "rlist"'
 do while(sqlcode=Ø)
    row='RDEF '||clas||' '||ssid||'.'||dbitem||'.'||db2c||,
        ' UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
    address ispexec 'tbadd "rlist"'
    row='PERMIT '||ssid||'.'||dbitem||'.'||db2c||,
        ' CLASS('||clas||') RESET'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbitem"
 end
 address dsnrexx "execsql close c1"
Return Ø

Db_Util_Pub:
 SQLSTMT= "SELECT DISTINCT STRIP(NAME)                              ",
 "FROM SYSIBM.SYSDBAUTH                                             ",
 "WHERE GRANTEE IN ('PUBLIC','PUBLIC*')                             ",
 "  AND "db2obj" ¬=' '                                              ",
 "  WITH UR                                                         "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbitem"
 address ispexec 'tbadd "rlist"'
 do while(sqlcode=Ø)
    row='RALT '||clas||' '||ssid||'.'||dbitem||'.'||db2c||' UACC(READ)'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbitem"
 end
 address dsnrexx "execsql close c1"
Return Ø

Db_Util_Det:
 SQLSTMT= "SELECT DISTINCT STRIP(NAME), STRIP(GRANTEE),             ",
 "CASE("db2obj")                                                    ",
 "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                               ",
 "END                                                               ",
 "FROM SYSIBM.SYSDBAUTH                                             ",
 "WHERE "db2obj" IN ('Y','G')                                       ",
 "  AND GRANTEE NOT IN ('PUBLIC','PUBLIC*')                         ",
 "  WITH UR                                                         "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbn, :gra, :acc"
 do while(sqlcode=Ø)
    row='PERMIT '||ssid||'.'||dbn||'.'||db2c||,
        ' CLASS('||clas||') '||'ACC('||acc||') '||'ID('||gra||')'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbn, :gra, :acc"
 end
 address dsnrexx "execsql close c1"
Return Ø

Dba_Opt:
 SQLSTMT= "SELECT DISTINCT STRIP(GRANTEE),                          ",
 "CASE("db2obj")                                                    ",
 "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                               ",
 "END                                                               ",
 "FROM SYSIBM.SYSDBAUTH                                             ",
 "WHERE "db2obj" IN ('Y','G')                                       ",
 "  AND GRANTEE¬='PUBLIC'                                           ",
 "  WITH UR                                                         "

 Call Cursor
 address dsnrexx "execsql fetch c1 into  :gra, :acc"
 do while(sqlcode=Ø)
    row='PERMIT '||ssid||'.'||'DBADM '||,
        ' CLASS('||clas||') '||'ACC('||acc||') '||'ID('||gra||')'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :gra, :acc"
 end
 address dsnrexx "execsql close c1"
Return
```

```
/* End Database privileges                                            */

/* Function and stored procedure privileges                           */
Routine_Auth:
 SQLSTMT= "SELECT DISTINCT STRIP(SPECIFICNAME),                        ",
 "  STRIP(CASE ROUTINETYPE                                             ",
 "         WHEN 'P' THEN 'SP'                                          ",
 "         WHEN 'F' THEN 'UF'                                          ",
 "        END)                                                         ",
 "FROM SYSIBM.SYSROUTINEAUTH                                           ",
 "  WITH UR                                                            "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbitem, :uf"

 row = '/* '||db2c||' privilege */'
 address ispexec 'tbadd "rlist"'
 do while(sqlcode=Ø)
    clas='MDSN'||uf
    row='RDEF '||clas||' '||ssid||'.'||dbitem||'.'||db2c||,
        ' UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
    address ispexec 'tbadd "rlist"'
    row='PERMIT '||ssid||'.'||dbitem||'.'||db2c||,
        ' CLASS('||clas||') RESET'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbitem, :uf"
 end
 address dsnrexx "execsql close c1"
Return Ø

Routine_Ralt:
 SQLSTMT= "SELECT DISTINCT STRIP(SPECIFICNAME),                        ",
 "  STRIP(CASE ROUTINETYPE                                             ",
 "         WHEN 'P' THEN 'SP'                                          ",
 "         WHEN 'F' THEN 'UF'                                          ",
 "        END)                                                         ",
 "FROM SYSIBM.SYSROUTINEAUTH                                           ",
 "WHERE GRANTEE IN ('PUBLIC','PUBLIC*')                                ",
 "  WITH UR                                                            "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbitem, :uf"

 address ispexec 'tbadd "rlist"'
 do while(sqlcode=Ø)
    clas='MDSN'||uf
    row='RALT '||clas||' '||ssid||'.'||dbitem||'.'||db2c||,
        ' UACC(READ)'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbitem, :uf"
 end
 address dsnrexx "execsql close c1"
Return Ø

Routine_Opt:
 SQLSTMT= "SELECT DISTINCT STRIP(SPECIFICNAME), STRIP(GRANTEE), ",
 "CASE EXECUTEAUTH                                                  ",
 "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                               ",
 "END,                                                              ",
 "CASE ROUTINETYPE                                                  ",
 "  WHEN 'P' THEN 'SP'                                              ",
 "  WHEN 'F' THEN 'UF'                                              ",
 "END                                                               ",
 "FROM SYSIBM.SYSROUTINEAUTH                                        ",
 "WHERE GRANTEE NOT IN ('PUBLIC','PUBLIC*')                         ",
 "  AND EXECUTEAUTH IN ('Y','G')                                    ",
 "  WITH UR                                                         "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc, :uf"

 do while(sqlcode=Ø)
    clas='MDSN'||uf
    row='PERMIT '||ssid||'.'||dbn||'.'||db2c||,
        ' CLASS('||clas||') ACC('||ucc||') ID('||gra||')'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc, :uf"
 end
 address dsnrexx "execsql close c1"
Return Ø
/* End Function and stored procedure privileges                    */

/* Package privileges                                              */
Package_Privileges:
 SQLSTMT= "SELECT NAME, REPLACE(NAME,'AUTH','')                     ",
 "FROM SYSIBM.SYSCOLUMNS                                            ",
 "WHERE TBCREATOR='SYSIBM'                                          ",
 "  AND TBNAME IN ('SYSRESAUTH','SYSPACKAUTH')                      ",
 "  AND NAME LIKE '%AUTH%'                                          ",
 "  AND COLNO > 9                                                   ",
 "  WITH UR                                                         "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
 Call Init

 do while(sqlcode=Ø)
    address ispexec 'tbadd "slist"'
    address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
 end
 address dsnrexx "execsql close c1"
 address ispexec 'tbtop "slist"'
 address ispexec 'tbdispl "slist" panel(db2rm1)'
 if rc=8 then do
    address ispexec 'tbend "slist"'
    address ispexec 'tbend "rlist"'
    address ispexec "tbend "messdb""
    call create_messg
    return 99
 end
 if rc<8 & cmd='S' then do
    do while ztdsels > Ø
       if db2c='USE' then do
          Clas='DSNAMD'
          Call Packadm
          Call Pack_Det
       end
       else do
          Clas='MDSNPK'
          Call Pack_Col
          Call Pack_Ralt
          Call Pack_Prof
          Call Pack_Auth
       end
       if ztdsels = 1 then ztdsels=Ø
       if ztdsels > Ø then address ispexec 'tbdispl "slist" '
    end
    address ispexec 'tbtop "rlist"'
 end
Return Ø

Packadm:
 SQLSTMT= "SELECT DISTINCT STRIP(NAME)                               ",
 "FROM SYSIBM.SYSRESAUTH                                             ",
 "  WITH UR                                                          "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbitem"

 row = '/* '||db2c||' Package privilege */'
 address ispexec 'tbadd "rlist"'
 do while(sqlcode=Ø)
    row='RDEF '||clas||' '||ssid||'.'||dbitem||'.PACKADM'||,
        ' UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
    address ispexec 'tbadd "rlist"'
    row='PERMIT '||ssid||'.'||dbitem||'.PACKADM'||,
        ' CLASS('||clas||') RESET'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbitem"
 end
 address dsnrexx "execsql close c1"
Return Ø

Pack_Det:
 SQLSTMT= "SELECT DISTINCT STRIP(NAME)                                  ",
 "FROM SYSIBM.SYSRESAUTH                                               ",
 "WHERE GRANTEE = 'PUBLIC'                                             ",
 "  AND QUALIFIER='PACKADM'                                            ",
 "  AND OBTYPE='C'                                                     ",
 "  AND USEAUTH¬=' '                                                   ",
 "  WITH UR                                                            "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbitem"

 do while(sqlcode=Ø)
    row='RALT '||clas||' '||ssid||'.'||dbitem||'.PACKADM UACC(READ)'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbitem"
 end
 address dsnrexx "execsql close c1"

 SQLSTMT= "SELECT DISTINCT STRIP(NAME), STRIP(GRANTEE),                 ",
 "CASE(USEAUTH)                                                        ",
 "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                                  ",
 "END                                                                  ",
 "FROM SYSIBM.SYSRESAUTH                                               ",
 "WHERE GRANTEE¬= 'PUBLIC'                                             ",
 "  AND QUALIFIER='PACKADM'                                            ",
 "  AND OBTYPE='C'                                                     ",
 "  AND USEAUTH IN ('Y','G')                                           ",
 "  WITH UR                                                            "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"

 do while(sqlcode=Ø)
    row='PERMIT '||ssid||'.'||dbn||'.PACKADM '||,
        'CLASS('||clas||') ACC('||ucc||') ID('||gra||')'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"
 end
 address dsnrexx "execsql close c1"
Return Ø

Pack_Col:
 SQLSTMT= "SELECT DISTINCT STRIP(COLLID), STRIP(NAME)                   ",
 "FROM SYSIBM.SYSPACKAUTH                                              ",
 "  WITH UR                                                            "
 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbitem, :dbn"

 row = '/* '||db2c||' Package privilege */'
 address ispexec 'tbadd "rlist"'
 do while(sqlcode=Ø)
    row='RDEF '||clas||' '||ssid||'.'||dbitem||'.'||dbn||'.'||db2c||,
        ' UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
    address ispexec 'tbadd "rlist"'
    row='PERMIT '||ssid||'.'||dbitem||'.'||dbn||'.'||db2c||,
        ' CLASS('||clas||') RESET'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbitem, :dbn"
 end
 address dsnrexx "execsql close c1"
Return Ø

Pack_Ralt:
 SQLSTMT= "SELECT DISTINCT STRIP(COLLID), STRIP(NAME)              ",
 "FROM SYSIBM.SYSPACKAUTH                                          ",
 "WHERE GRANTEE IN ('PUBLIC','PUBLIC*')                            ",
 "  AND "db2obj" ¬=' '                                             ",
 "  AND GRANTEETYPE=' '                                            ",
 "  WITH UR                                                        "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbitem, :dbn"

 do while(sqlcode=Ø)
    row='RALT '||clas||' '||ssid||'.'||dbitem||'.'||dbn||' UACC(READ)'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbitem, :dbn"
 end
 address dsnrexx "execsql close c1"

 SQLSTMT= "SELECT DISTINCT STRIP(COLLID), STRIP(GRANTEE),          ",
 "CASE("db2obj")                                                   ",
 "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                              ",
 "END                                                              ",
 "FROM SYSIBM.SYSPACKAUTH                                          ",
 "WHERE GRANTEE NOT IN ('PUBLIC','PUBLIC*')                        ",
 "  AND "db2obj" IN ('Y','G')                                      ",
 "  AND NAME='*'                                                   ",
 "  AND GRANTEETYPE=' '                                            ",
 "  WITH UR                                                        "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"

 do while(sqlcode=Ø)
```

```
     row='PERMIT '||ssid||'.'||dbn||'.'||db2c||,
         ' CLASS('||clas||') ACC('||ucc||') ID('||gra||')'
     address ispexec 'tbadd "rlist"'
     address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"
  end
  address dsnrexx "execsql close c1"
 Return Ø

 Pack_Prof:
  SQLSTMT= "SELECT DISTINCT STRIP(COLLID), STRIP(NAME)              ",
  "FROM SYSIBM.SYSPACKAUTH X                                        ",
  "WHERE NAME¬='*' AND GRANTEETYPE=' '                              ",
  "  AND EXISTS(SELECT 1 FROM SYSIBM.SYSPACKAUTH                    ",
  "   WHERE NAME='*' AND COLLID=X.COLLID                            ",
  "     AND GRANTEETYPE=' ')                                        ",
  "  WITH UR                                                        "

  Call Cursor
  address dsnrexx "execsql fetch c1 into :dbn, :gra"

  do while(sqlcode=Ø)
     row='PERMIT '||ssid||'.'||dbn||'.'||gra||'.'||db2c||,
         ' CLASS('||clas||') FROM('||ssid||'.'||dbn||'.*.'||db2c||')'
     address ispexec 'tbadd "rlist"'
     address dsnrexx "execsql fetch c1 into :dbn, :gra"
  end
  address dsnrexx "execsql close c1"
 Return Ø

 Pack_Auth:
  SQLSTMT= "SELECT DISTINCT STRIP(COLLID), STRIP(NAME),             ",
  "STRIP(GRANTEE),                                                  ",
  "CASE("db2obj")                                                   ",
  "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                              ",
  "END                                                              ",
  "FROM SYSIBM.SYSPACKAUTH                                          ",
  "WHERE GRANTEE NOT IN ('PUBLIC','PUBLIC*')                        ",
  "  AND "db2obj" IN ('Y','G')                                      ",
  "  AND GRANTEETYPE=' '                                            ",
  "  AND NAME¬='*'                                                  ",
  "  WITH UR                                                        "

  Call Cursor
  address dsnrexx "execsql fetch c1 into :dbn, :nam, :gra, :ucc"

  do while(sqlcode=Ø)
     row='PERMIT '||ssid||'.'||dbn||'.'||nam||'.'||db2c||,
         ' CLASS('||clas||') ACC('||ucc||') ID('||gra||')'
     address ispexec 'tbadd "rlist"'
     address dsnrexx "execsql fetch c1 into :dbn, :nam, :gra, :ucc"
  end
  address dsnrexx "execsql close c1"
 Return Ø

 /* End Package privileges                                             */

 /* Plan privileges                                                    */
 Dba_Plan:
  SQLSTMT= "SELECT NAME, REPLACE(NAME,'AUTH','')                        ",
  "FROM SYSIBM.SYSCOLUMNS                                               ",
  "WHERE TBCREATOR='SYSIBM'                                             ",
  "  AND TBNAME='SYSPLANAUTH'                                           ",
  "  AND NAME LIKE '%AUTH%'                                             ",
  "  AND COLNO > 8                                                      ",
  "  ORDER BY NAME                                                      ",
  "  WITH UR                                                            "

  Call Cursor
  address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
  Call Init

  do while(sqlcode=Ø)
     address ispexec 'tbadd "slist"'
     address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
  end
  address dsnrexx "execsql close c1"
  address ispexec 'tbtop "slist"'
  address ispexec 'tbdispl "slist" panel(db2rm1)'
  if rc=8 then do
     address ispexec 'tbend "slist"'
     address ispexec 'tbend "rlist"'
     address ispexec "tbend "messdb""
     call create_messg
     return 99
  end
  if rc<8 & cmd='S' then do
     do while ztdsels > Ø
        Clas='MDSNPL'
        Call Plan_Auth
        Call Plan_Ralt
        Call Plan_Opt
        if ztdsels = 1 then ztdsels=Ø
        if ztdsels > Ø then address ispexec 'tbdispl "slist" '
     end
     address ispexec 'tbtop "rlist"'
  end
 Return Ø

 Plan_Auth:
  SQLSTMT= "SELECT DISTINCT STRIP(NAME)                                 ",
  "FROM SYSIBM.SYSPLANAUTH                                            ",
  "  WITH UR                                                          "

  Call Cursor
  address dsnrexx "execsql fetch c1 into :dbitem"

  row = '/* '||db2c||' privilege */'
  address ispexec 'tbadd "rlist"'
  do while(sqlcode=Ø)
     row='RDEF '||clas||' '||ssid||'.'||dbitem||'.'||db2c||,
         ' UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
     address ispexec 'tbadd "rlist"'
     row='PERMIT '||ssid||'.'||dbitem||'.'||db2c||,
         ' CLASS('||clas||') RESET'
     address ispexec 'tbadd "rlist"'
     address dsnrexx "execsql fetch c1 into :dbitem"
  end
  address dsnrexx "execsql close c1"
 Return Ø

 Plan_Ralt:
  SQLSTMT= "SELECT DISTINCT STRIP(NAME)                                        ",
  "FROM SYSIBM.SYSPLANAUTH                                            ",
  "WHERE GRANTEE IN ('PUBLIC','PUBLIC*')                              ",
  "   AND "db2obj" ¬=' '                                              ",
  "  WITH UR                                                          "

  Call Cursor
  address dsnrexx "execsql fetch c1 into :dbitem"

  address ispexec 'tbadd "rlist"'
  do while(sqlcode=Ø)
     row='RALT '||clas||' '||ssid||'.'||dbitem||'.'||db2c||,
         ' UACC(READ)'
     address ispexec 'tbadd "rlist"'
     address dsnrexx "execsql fetch c1 into :dbitem"
  end
  address dsnrexx "execsql close c1"
 Return Ø

 Plan_Opt:
  SQLSTMT= "SELECT DISTINCT STRIP(NAME), STRIP(GRANTEE),                      ",
  "CASE("db2obj")                                                     ",
  "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                                ",
  "END                                                                ",
  "FROM SYSIBM.SYSPLANAUTH                                            ",
  "WHERE GRANTEE NOT IN ('PUBLIC','PUBLIC*')                          ",
  "  AND "db2obj" IN ('Y','G')                                        ",
  "  WITH UR                                                          "
  Call Cursor
  address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"

  do while(sqlcode=Ø)
     row='PERMIT '||ssid||'.'||dbn||'.'||db2c||,
         ' CLASS('||clas||') ACC('||ucc||') ID('||gra||')'
     address ispexec 'tbadd "rlist"'
     address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"
  end
  address dsnrexx "execsql close c1"
 Return Ø
 /* End Plan privileges                                          */

 /* Schema privileges                                            */
 Schema_privileges:
  SQLSTMT= "SELECT NAME, REPLACE(NAME,'AUTH','')                  ",
  "FROM SYSIBM.SYSCOLUMNS                                         ",
  "WHERE TBCREATOR='SYSIBM'                                       ",
  "  AND TBNAME='SYSSCHEMAAUTH'                                   ",
  "  AND NAME LIKE '%AUTH%'                                       ",
  "  AND COLNO > 4                                                ",
  "  ORDER BY NAME                                                ",
  "  WITH UR                                                      "

  Call Cursor
  address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
  Call Init

  do while(sqlcode=Ø)
     address ispexec 'tbadd "slist"'
     address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
  end
  address dsnrexx "execsql close c1"
  address ispexec 'tbtop "slist"'
  address ispexec 'tbdispl "slist" panel(db2rm1)'
  if rc=8 then do
     address ispexec 'tbend "slist"'
     address ispexec 'tbend "rlist"'
     address ispexec "tbend "messdb""
     call create_messg
     return 99
  end
  if rc<8 & cmd='S' then do
     do while ztdsels > Ø
        Clas='MDSNSC'
        Call Schema_Auth
        Call Schema_Ralt
        Call Schema_Opt
        if ztdsels = 1 then ztdsels=Ø
        if ztdsels > Ø then address ispexec 'tbdispl "slist" '
     end
     address ispexec 'tbtop "rlist"'
  end
 Return Ø

 Schema_Auth:
  SQLSTMT= "SELECT DISTINCT STRIP(SCHEMANAME)                         ",
  "FROM SYSIBM.SYSSCHEMAAUTH                                          ",
  "  WITH UR                                                          "

  Call Cursor
  address dsnrexx "execsql fetch c1 into :dbitem"

  row = '/* '||db2c||' privilege */'
  address ispexec 'tbadd "rlist"'
  do while(sqlcode=Ø)
     row='RDEF '||clas||' '||ssid||'.'||dbitem||'.'||db2c||,
         ' UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
     address ispexec 'tbadd "rlist"'
     row='PERMIT '||ssid||'.'||dbitem||'.'||db2c||,
         ' CLASS('||clas||') RESET'
     address ispexec 'tbadd "rlist"'
     address dsnrexx "execsql fetch c1 into :dbitem"
  end
  address dsnrexx "execsql close c1"
 Return Ø

 Schema_Ralt:
  SQLSTMT= "SELECT DISTINCT STRIP(SCHEMANAME)                         ",
  "FROM SYSIBM.SYSSCHEMAAUTH                                          ",
  "WHERE GRANTEE IN ('PUBLIC','PUBLIC*')                              ",
  "   AND "db2obj" ¬=' '                                              ",
  "  WITH UR                                                          "

  Call Cursor
  address dsnrexx "execsql fetch c1 into :dbitem"

  address ispexec 'tbadd "rlist"'
  do while(sqlcode=Ø)
     row='RALT '||clas||' '||ssid||'.'||dbitem||'.'||db2c||,
         ' UACC(READ)'
     address ispexec 'tbadd "rlist"'
     address dsnrexx "execsql fetch c1 into :dbitem"
  end
  address dsnrexx "execsql close c1"
 Return Ø

 Schema_Opt:
  SQLSTMT= "SELECT DISTINCT STRIP(SCHEMANAME), STRIP(GRANTEE),      ",
  "CASE("db2obj")                                                     ",
  "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                               ",
  "END                                                               ",
  "FROM SYSIBM.SYSSCHEMAAUTH                                         ",
  "WHERE GRANTEE NOT IN ('PUBLIC','PUBLIC*')                         ",
  "  AND "db2obj" IN ('Y','G')                                       ",
  "  WITH UR                                                         "

  Call Cursor
  address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"

  do while(sqlcode=Ø)
     row='PERMIT '||ssid||'.'||dbn||'.'||db2c||,
         ' CLASS('||clas||') ACC('||ucc||') ID('||gra||')'
     address ispexec 'tbadd "rlist"'
     address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"
  end
  address dsnrexx "execsql close c1"
 Return Ø
 /* End Schema privileges                                            */

 /* Resource privileges                                              */
 System_Privileges:
  SQLSTMT= "SELECT NAME, REPLACE(NAME,'AUTH','')                     ",
  "FROM SYSIBM.SYSCOLUMNS                                            ",
  "WHERE TBCREATOR='SYSIBM'                                          ",
  "  AND TBNAME='SYSUSERAUTH'                                        ",
  "  AND NAME LIKE '%AUTH%'                                          ",
  "  AND COLNO NOT IN (7,8,1Ø,26,28,29)                              ",
  "  ORDER BY NAME                                                   ",
  "  WITH UR                                                         "

  Call Cursor
  address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
  Call Init

  do while(sqlcode=Ø)
     address ispexec 'tbadd "slist"'
     address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
  end
  address dsnrexx "execsql close c1"
  address ispexec 'tbtop "slist"'
  address ispexec 'tbdispl "slist" panel(db2rm1)'
  if rc=8 then do
     address ispexec 'tbend "slist"'
     address ispexec 'tbend "rlist"'
     address ispexec "tbend "messdb""
     call create_messg
     return 99
  end
  if rc<8 & cmd='S' then do
```

```
    do while ztdsels > Ø
       clas='MDSNSM'
       if db2c='SYSADM' | db2c='SYSOPR' |,
          db2c='SYSCTRL' then Clas='DSNAMD'
       row = '/* '||db2c||' privilege */'
       address ispexec 'tbadd "rlist"'
       row='RDEF '||clas||' '||ssid||'.'||db2c||' UACC(NONE) '||,
           'OWNER('||sqlid||') AUDIT(ALL(READ))'
       address ispexec 'tbadd "rlist"'
       row='PERMIT '||ssid||'.'||db2c||' CLASS('||clas||') RESET'
       address ispexec 'tbadd "rlist"'
       Call Sys_detail
       if db2c='BINDADD' then Call Bagent
       if ztdsels = 1 then ztdsels=Ø
       if ztdsels > Ø then address ispexec 'tbdispl "slist" '
    end
    address ispexec 'tbtop "rlist"'
 end
Return Ø

Sys_Detail:
 /* Select system authority */
 SQLSTMT= "SELECT DISTINCT STRIP(GRANTEE),                          ",
 "CASE("db2obj")                                                    ",
 "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                               ",
 "END                                                               ",
 "FROM SYSIBM.SYSUSERAUTH                                           ",
 "WHERE "db2obj" IN ('Y','G')                                       ",
 "  AND GRANTEE¬='PUBLIC'                                           ",
 "  WITH UR                                                         "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :gus, :acc"
 do while(sqlcode=Ø)
    row='PERMIT '||ssid||'.'||db2c||' CLASS('||clas||') '||,
        'ACC('||acc||') '||'ID('||gus||')'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :gus, :acc"
 end
 address dsnrexx "execsql close c1"
Return

Bagent:
 /* Select BINDAGENT authority */
 row='/* BINDAGENT privilege */'
 clas='MDSNSM'
 address ispexec 'tbadd "rlist"'
 SQLSTMT= "SELECT DISTINCT STRIP(GRANTOR)                           ",
 "FROM SYSIBM.SYSUSERAUTH                                           ",
 "WHERE BINDAGENTAUTH¬=' '                                          ",
 "  WITH UR                                                            "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :gra"
 do while(sqlcode=0)
    row='RDEF '||clas||' '||ssid||'.'||gra||'.'||'BINDAGENT '||,
        'UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
    address ispexec 'tbadd "rlist"'
    row='PERMIT '||ssid||'.'||gra||'.'||'BINDAGENT '||,
        'CLASS('||clas||') RESET'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :gra"
 end
 address dsnrexx "execsql close c1"
 SQLSTMT= "SELECT DISTINCT STRIP(GRANTOR), STRIP(GRANTEE),                  ",
 "CASE(BINDAGENTAUTH)                                                   ",
 "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                                   ",
 "END                                                                   ",
 "FROM SYSIBM.SYSUSERAUTH                                               ",
 "WHERE BINDAGENTAUTH in ('Y','G') AND GRANTEE¬='PUBLIC'                ",
 "  WITH UR                                                            "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :gus, :gre, :acc"
 do while(sqlcode=0)
    row='PERMIT '||ssid||'.'||gus||'.BINDAGENT '||,
        'CLASS('||clas||') '||'ACC('||acc||') '||'ID('||gre||')'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :gus, :gre, :acc"
 end
 address dsnrexx "execsql close c1"
Return
/* End Resource privileges                                              */

/* Table, View, Trigger privileges                                      */
Tab_Privilege:
 SQLSTMT= "SELECT NAME, REPLACE(NAME,'AUTH','')                             ",
 "FROM SYSIBM.SYSCOLUMNS                                                ",
 "WHERE TBCREATOR='SYSIBM'                                              ",
 "  AND TBNAME='SYSTABAUTH'                                             ",
 "  AND NAME LIKE '%AUTH%'                                              ",
 "  AND COLNO NOT IN (9, 25)                                            ",
 "  ORDER BY NAME                                                       ",
 "  WITH UR                                                            "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
 Call Init

 do while(sqlcode=0)
```

```
      address ispexec 'tbadd "slist"'
      address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
   end
   address dsnrexx "execsql close c1"
   address ispexec 'tbtop "slist"'
   address ispexec 'tbdispl "slist" panel(db2rm1)'
   if rc=8 then do
      address ispexec 'tbend "slist"'
      address ispexec 'tbend "rlist"'
      address ispexec "tbend "messdb""
      call create_messg
      return 99
   end
   if rc<8 & cmd='S' then do
      do while ztdsels > Ø
         Clas='MDSNTB'
         /* Call Tab_Auth */
         /* Call Tab_Ralt */
         /* Call Tab_Opt */
         if db2c='UPDATE' | db2c='REFERENCES' then Call Col_Opt
         if ztdsels = 1 then ztdsels=Ø
         if ztdsels > Ø then address ispexec 'tbdispl "slist" '
      end
      address ispexec 'tbtop "rlist"'
   end
  Return Ø

  Tab_Auth:
   SQLSTMT= "SELECT DISTINCT STRIP(TCREATOR), STRIP(TTNAME)           ",
   "FROM SYSIBM.SYSTABAUTH                                            ",
   "WHERE GRANTEETYPE=' '                                             ",
   "  WITH UR                                                         "

   Call Cursor
   address dsnrexx "execsql fetch c1 into :cre, :tab"

   row = '/* '||db2c||' privilege */'
   address ispexec 'tbadd "rlist"'
   i=1
   do while(sqlcode=Ø)
      row='RDEF '||clas||' '||ssid||'.'||cre||'.'||tab||'.'||db2c||,
          ' UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
      address ispexec 'tbadd "rlist"'
      row='PERMIT '||ssid||'.'||cre||'.'||tab||'.'||db2c||,
          ' CLASS('||clas||') RESET'
      address ispexec 'tbadd "rlist"'
      address dsnrexx "execsql fetch c1 into :cre, :tab"
      i=i+1
      if i>1ØØ then Return Ø
   end
```

```
 address dsnrexx "execsql close c1"
Return Ø

Tab_Ralt:
 SQLSTMT= "SELECT DISTINCT STRIP(TCREATOR), STRiP(TTNAME)           ",
 "FROM SYSIBM.SYSTABAUTH                                           ",
 "WHERE GRANTEE IN ('PUBLIC','PUBLIC*')                            ",
 "   AND "db2obj"¬=' '                                             ",
 "   AND GRANTEETYPE=' '                                           ",
 "  WITH UR                                                        "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :cre, :tab"

 address ispexec 'tbadd "rlist"'
 do while(sqlcode=Ø)
    row='RALT '||clas||' '||ssid||'.'||cre||'.'||tab||'.'||db2c||,
        ' UACC(READ)'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :cre, :tab"
 end
 address dsnrexx "execsql close c1"
Return Ø

Tab_Opt:
 SQLSTMT= "SELECT DISTINCT STRIP(TCREATOR),                         ",
 "         STRIP(TTNAME), STRIP(GRANTEE),                          ",
 "CASE("db2obj")                                                   ",
 "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                              ",
 "END                                                              ",
 "FROM SYSIBM.SYSTABAUTH                                           ",
 "WHERE GRANTEE NOT IN ('PUBLIC','PUBLIC*')                        ",
 "  AND "db2obj" IN ('Y','G')                                      ",
 "  AND GRANTEETYPE=' '                                            ",
 "  WITH UR                                                        "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :cre, :tab, :gra, :ucc"

 i=1
 do while(sqlcode=Ø)
    row='PERMIT '||ssid||'.'||cre||'.'||tab||'.'||db2c||,
        ' CLASS('||clas||') ACC('||ucc||') ID('||gra||')'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :cre, :tab, :gra, :ucc"
   i=i+1
   if i > 1ØØ then Return Ø
 end
 address dsnrexx "execsql close c1"
Return Ø
```

```
Col_Opt:
 ucl=' '
 if db2c='UPDATE' then ucl='*'
 SQLSTMT= "SELECT DISTINCT STRIP(T.TCREATOR),                          ",
 "                STRIP(T.TTNAME), STRIP(C.COLNAME)                  ",
 "FROM SYSIBM.SYSTABAUTH T,                                          ",
 "      SYSIBM.SYSCOLAUTH C                                          ",
 "WHERE T.GRANTEETYPE=' '                                            ",
 "   AND T.DATEGRANTED=C.DATEGRANTED                                 ",
 "   AND T.TIMEGRANTED=C.TIMEGRANTED                                 ",
 "   AND T.TTNAME=C.TNAME                                            ",
 "   AND T.TCREATOR=C.CREATOR                                        ",
 "   AND T.UPDATECOLS='"ucl"'                                        ",
 "   AND "db2obj"¬=' '                                               ",
 "   WITH UR                                                         "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :cre, :tab, :col"

 do while(sqlcode=Ø)
    row='RDEF '||clas||' '||ssid||'.'||cre||'.'||tab||'.'||,
        col||'.'||db2c||,
        ' UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
    address ispexec 'tbadd "rlist"'
    row='PERMIT '||ssid||'.'||cre||'.'||tab||'.'||col||'.'||db2c||,
        ' CLASS('||clas||') RESET'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :cre, :tab, :col"
 end
 address dsnrexx "execsql close c1"

 SQLSTMT= "SELECT DISTINCT STRIP(T.TCREATOR),                          ",
 "STRIP(T.TTNAME), STRIP(C.COLNAME), STRIP(T.GRANTEE),               ",
 "CASE("db2obj")                                                     ",
 "   WHEN 'Y' THEN 'READ' ELSE 'ALTER'                               ",
 "END                                                                ",
 "FROM SYSIBM.SYSTABAUTH T,                                          ",
 "      SYSIBM.SYSCOLAUTH C                                          ",
 "WHERE T.GRANTEETYPE=' '                                            ",
 "   AND T.DATEGRANTED=C.DATEGRANTED                                 ",
 "   AND T.TIMEGRANTED=C.TIMEGRANTED                                 ",
 "   AND T.TTNAME=C.TNAME                                            ",
 "   AND T.TCREATOR=C.CREATOR                                        ",
 "   AND T.GRANTEE NOT IN ('PUBLIC','PUBLIC*')                       ",
 "   AND T.UPDATECOLS='"ucl"'                                        ",
 "   AND "db2obj" IN ('Y','G')                                       ",
 "   WITH UR                                                         "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :cre, :tab, :col, :gra, :ucc"
 do while(sqlcode=Ø)
    row='PERMIT '||ssid||'.'||cre||'.'||tab||'.'||col||'.'||db2c||,
        ' CLASS('||clas||') ACC('||ucc||') ID('||gra||')'
    address ispexec 'tbadd "rlist"'
  address dsnrexx "execsql fetch c1 into :cre, :tab, :col, :gra, :ucc"
 end
 address dsnrexx "execsql close c1"
Return Ø
/* End Table, View, Trigger privileges                               */

/* Bufferpool, Stogroup, Tablespace privileges                       */
Use_Privilege:
 SQLSTMT= "SELECT DISTINCT OBTYPE,                                    ",
 "         CASE OBTYPE                                                ",
 "          WHEN 'B' THEN 'BUFFERPOOL'                                ",
 "          WHEN 'R' THEN 'TABLESPACE'                                ",
 "          WHEN 'S' THEN 'STOGROUP'                                  ",
 "         END                                                        ",
 "FROM SYSIBM.SYSRESAUTH                                              ",
 "WHERE OBTYPE IN ('B','R','S')                                       ",
 "  WITH UR                                                           "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
 Call Init

 do while(sqlcode=Ø)
    address ispexec 'tbadd "slist"'
    address dsnrexx "execsql fetch c1 into :db2obj, :db2c"
 end
 address dsnrexx "execsql close c1"
 address ispexec 'tbtop "slist"'
 address ispexec 'tbdispl "slist" panel(db2rm1)'
 if rc=8 then do
    address ispexec 'tbend "slist"'
    address ispexec 'tbend "rlist"'
    address ispexec "tbend "messdb""
    call create_messg
    return 99
 end
 if rc<8 & cmd='S' then do
    do while ztdsels > Ø
       if db2obj='S' then Clas='MDSNSG'
       if db2obj='R' then Clas='MDSNTS'
       if db2obj='B' then Clas='MDSNBP'
       Call Use_Item
       Call Use_Ralt
       Call Use_Opt
       if ztdsels = 1 then ztdsels=Ø
       if ztdsels > Ø then address ispexec 'tbdispl "slist" '
    end
    address ispexec 'tbtop "rlist"'
 end
Return Ø

Use_Item:
 SQLSTMT= "SELECT DISTINCT STRIP(NAME)                              ",
 "FROM SYSIBM.SYSRESAUTH                                             ",
 "WHERE OBTYPE = '"db2obj"'                                          ",
 "  WITH UR                                                          "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbitem"

 row = '/* '||db2c||' privilege */'
 address ispexec 'tbadd "rlist"'
 do while(sqlcode=Ø)
    row='RDEF '||clas||' '||ssid||'.'||dbitem||'.USE'||,
        ' UACC(NONE) '||'OWNER('||sqlid||') AUDIT(ALL(READ))'
    address ispexec 'tbadd "rlist"'
    row='PERMIT '||ssid||'.'||dbitem||'.USE'||,
        ' CLASS('||clas||') RESET'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbitem"
 end
 address dsnrexx "execsql close c1"
Return Ø

Use_Ralt:
 SQLSTMT= "SELECT DISTINCT STRIP(NAME)                              ",
 "FROM SYSIBM.SYSRESAUTH                                             ",
 "WHERE GRANTEE IN ('PUBLIC','PUBLIC*')                              ",
 "   AND OBTYPE='"db2obj"'                                           ",
 "   AND USEAUTH¬=' '                                                ",
 "  WITH UR                                                          "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbitem"

 address ispexec 'tbadd "rlist"'
 do while(sqlcode=Ø)
    row='RALT '||clas||' '||ssid||'.'||dbitem||'.USE'||,
        ' UACC(READ)'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbitem"
 end
 address dsnrexx "execsql close c1"
Return Ø

Use_Opt:
 SQLSTMT= "SELECT DISTINCT STRIP(NAME), STRIP(GRANTEE),            ",
 "CASE(USEAUTH)                                                     ",
 "  WHEN 'Y' THEN 'READ' ELSE 'ALTER'                               ",
 "END                                                               ",
 "FROM SYSIBM.SYSRESAUTH                                            ",
 "WHERE GRANTEE NOT IN ('PUBLIC','PUBLIC*')                         ",
 "  AND USEAUTH IN ('Y','G')                                        ",
 "  AND OBTYPE = '"db2obj"'                                         ",
 "  WITH UR                                                         "

 Call Cursor
 address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"

 do while(sqlcode=Ø)
    row='PERMIT '||ssid||'.'||dbn||'.USE'||,
        ' CLASS('||clas||') ACC('||ucc||') ID('||gra||')'
    address ispexec 'tbadd "rlist"'
    address dsnrexx "execsql fetch c1 into :dbn, :gra, :ucc"
 end
 address dsnrexx "execsql close c1"
Return Ø
/* End Bufferpool, Stogroup, Tablespace privileges                 */

Connection:
 call create_messg
 messg = "accessing  db2 system "ssid""
 messg = time() || " " || messg
 call send_messg
 messg = 'select     syscolumns     information'
 messg = time() || " " || messg
 call send_messg

 /* Dsnrexx Language Support                             */
 ADDRESS TSO "SUBCOM DSNREXX"
 IF RC THEN
 S_RC = RXSUBCOM(ADD,DSNREXX,DSNREXX)

 ADDRESS DSNREXX "CONNECT" SSID
Return

Skeleton:
 messg = 'Building the RACF Statements'
 messg = time() || " " || messg
 Call Send_messg
 /* RACF Statements                   */
 tempfile=userid()||'.RACF.DDL'
 address tso
 "delete '"tempfile"'"
 "free dsname('"tempfile"')"
 "free ddname(ispfile)"
```

```
  "free attrlist(formfile)"
  "attrib formfile blksize(1000) lrecl(100) recfm(f b) dsorg(ps)"
  "alloc ddname(ispfile) dsname('"tempfile"')",
         "new using (formfile) unit(3390) space(4 4) cylinders"
  address ispexec
  "ftopen"
  "ftincl RACFS"
  "ftclose"
  zedsmsg = "RACF shown"
  zedlmsg = "RACF Definitions shown"
  "setmsg msg(isrz001)"
  "edit dataset('"tempfile"')"
  address ispexec 'tbend "slist"'
  address ispexec 'tbend "rlist"'
 Return
 Cursor:
  address dsnrexx "execsql declare c1 cursor for s1"
  address dsnrexx 'execsql prepare s1 from :sqlstmt'
  address dsnrexx "execsql open c1"
 Return
 Init:
  address ispexec 'tbcreate "slist" names(db2obj, db2c)'
  address ispexec 'tbcreate "rlist" names(db2obj, db2c, row)'
  row = '/* RACF and DB2 Utility */'
  address ispexec 'tbadd "rlist"'
  row = '/* User:'||user||' */'
  address ispexec 'tbadd "rlist"'
  row = '/* Date:'||date||' Time:'||time||' */'
  address ispexec 'tbadd "rlist"'
 Return
 Create_messg:
  messg = "s"||userid()
  address ispexec "tbcreate "messdb" names(messg) write replace"
 Return
 Send_messg:
  address ispexec "tbadd " messdb
  address ispexec "control display lock "
  address ispexec "addpop row(13) column(6)"
  address ispexec "tbdispl "messdb" panel(db2rm2)"
  address ispexec rempop
 Return
```

DB2RM0

```
)attr default(%+_)
   [ type (output) intens(low)  color(green) caps(off)
   # type (output) intens(low)  color(white) caps(off)
   _ type (input)  intens(low)  color(yellow) caps(off) pad('_')
   + type (text)   intens(low)  color(green)
   / type (text)   intens(low)  color(yellow)
   ~ type (text)   intens(high) color(turquoise)
   @ type (text)   intens(high) color(red)   caps(off) hilite(reverse)
)body window(78,24) expand ($$)
/.........................................................................
                        + @ DB2 and RACF Utility +
%Command ===>_zcmd                                                     +
/.........................................................................
+
 SubSytem Id:    _ssid+
+
            + _z+[field1                                           +
            + _z+[field2                                           +
            + _z+[field3                                           +
            +
            + _z+[field4                                           +
            + _z+[field5                                           +
            + _z+[field6                                           +
            +
            + _z+[field7                                           +
            + _z+[field8                                           +
            + _z+[field9                                           +
+
/.........................................................................
+
                 #msg                                              +
/PF3 - End +                                                ~Jan 2004,"ZB"
)init
  .ZVARS = '(f1 f2 f3 f4 f5 f6 f7 f8 f9)'
  &field1 = 'Colection privileges'
  &field2 = 'Database privileges'
  &field3 = 'Function and procedure privileges'
  &field4 = 'Package privileges'
  &field5 = 'Plan privileges'
  &field6 = 'Schema privileges'
  &field7 = 'Resource privileges'
  &field8 = 'Table, View, Trigger privileges'
  &field9 = 'Bufferpool, Stogroup, Tablespace privileges'
  &msg = 'Place cursor on choice and press <Enter>'
  IF (&kurs = F1,FIELD1)
      .attr (field1) = 'color (yellow) caps(on)'
  IF (&kurs = F2,FIELD2)
      .attr (field2) = 'color (yellow) caps(on)'
  IF (&kurs = F3,FIELD3)
      .attr (field3) = 'color (yellow) caps(on)'
  IF (&kurs = F4,FIELD4)
      .attr (field4) = 'color (yellow) caps(on)'
  IF (&kurs = F5,FIELD5)
      .attr (field5) = 'color (yellow) caps(on)'
  IF (&kurs = F6,FIELD6)
```

```
      .attr (field6) = 'color (yellow) caps(on)'
  IF (&kurs = F7,FIELD7)
      .attr (field7) = 'color (yellow) caps(on)'
  IF (&kurs = F8,FIELD8)
      .attr (field8) = 'color (yellow) caps(on)'
  IF (&kurs = F9,FIELD9)
      .attr (field9) = 'color (yellow) caps(on)'
)proc
  &kurs = .CURSOR
  VPUT (ssid) PROFILE
  if (.pfkey = pfØ3) &pf3 = exit
)end
```

DB2RM1

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   $ type(output) intens(high) caps(off) color(yellow)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   } type(text)   intens(high) caps(off) color(white)
   [ type( input) intens(high) caps(on ) just(left )
   ] type( input) intens(high) caps(on ) just(left ) pad('-')
   ¬ type(output) intens(low ) caps(off) just(asis ) color(green)
)Body  Expand(//)
%-/-/- ? Selection Result +%-/-/-
%Command ===>_zcmd                                 / /%Scroll ===>_amt +
+-------------------------------------------------------------------------
+Valid cmd:|S+Select one or more objects+
|PF3+Return
+-------------------------------------------------------------------------
#cmd+ #DB2 objects+
)Model
 ]z+  ¬z
+
)Init
  .ZVARS = '(cmd db2c)'
  &amt = PAGE
  &cmd = ''
)Reinit
)Proc
  IF (&ZTDSELS ¬= ØØØØ)
     VER (&cmd, LIST, S)
)End
```

DB2RM2

```
)ATTR DEFAULT(%+_)
```

```
| TYPE (TEXT)   INTENS(LOW)  COLOR(WHITE)
@ TYPE (TEXT)   INTENS(HIGH) COLOR(RED)   CAPS(OFF) HILITE(REVERSE)
| TYPE (INPUT)  INTENS(NON)  COLOR(GREEN) CAPS(ON)  JUST(LEFT)
# TYPE (OUTPUT) INTENS(LOW)  COLOR(GREEN) CAPS(OFF)
)BODY DEFAULT(%~\) WINDOW(6Ø,8)
|ZCMD +            @ Message display |AMT  |
|----------------------------------------------------------
)MODEL CLEAR(MESSG)
#Z                                        +
)INIT
     .ZVARS = '(MESSG)'
)REINIT
)PROC
  IF (.PFKEY = PFØ3) &PF3 = EXIT
  IF (&ZCMD=END)
     &COMMAND = CANCEL
)END
```

RACFS

```
)TBA 72
)CM -----------------------------------------------------------------
)CM Skeleton: RACF and DB2 Utility                                 --
)CM -----------------------------------------------------------------
)DOT "RLIST"
&row
)ENDDOT
```

Bernard Zver
DBA (Slovenia)

RACF news

Kobil Systems has announced the availability of the Smart Card Terminal KAAN Professional on select IBM eServer zSeries systems. The KAAN Professional works with IBM's Trusted Key Entry (TKE) workstation and is available as an optional feature. KAAN Professional can help protect security functions as the cryptographic keys are transferred (retrieved, stored, or saved) to or from the PIN protected Smart Card.

The IBM zSeries servers equipped with a TKE workstation and KAAN Professional are suitable for companies having to operate a secure key management system.

For further information contact:
KOBIL Systems, Pfortenring 11, D-67547 Worms, Germany.
Tel: +49 (6241) 3004 0.
URL: www.kobil.com/e/index.php?m=pressinfos.

***

IBM has announced Version 3.1 of CICS TS and Version 6.0 of CICS Transaction Gateway.

CICS Transaction Gateway Version 6.0 is designed "to use CICS applications in comprehensive and sophisticated J2EE and Web services solutions hosted on powerful application servers, such as WebSphere Application Server", said IBM.

Customers will increase the functionality of existing legacy systems with new support for the storage of SSL certificates in RACF SSL, hardware cryptography via JSSE, and control of SSL cipher suite.

For further information contact your local IBM representative.
URL: www-306.ibm.com/software/htp/cics/tserver/v31/.

***

VASCO has announced Digipass Plug-in for RACF, which makes token authentication both easy and inexpensive, the company claims. Large corporations with hundreds or thousands of users, who may be geographically widespread, can be sure that those accessing mainframe systems really are who they say they are.

VASCO's Digipass authentication products are natively integrated into RACF, and no additional authentication server is necessary. This gives users a strong authentication solution that is cheap and easy to administer (again, say the company). No client-side software or configurations are needed, which enhances the ease of deployment and the scalability, as well as lowering the TCO dramatically.

For further information contact:
Vasco, 1901 South Meyers Road, Suite 210, Oakbrook Terrace, IL 60181, USA.
Tel: (630) 932 8844.
URL: www.vasco.com/products/range.html.

***