IBM Education Assistant (IEA) for z/OS V2R3

RACF: Granular Security Administration

Agenda

- Trademarks
- Session Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Migration & Coexistence Considerations
- Installation
- Session Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - None

Session Objectives

- Improvements to Field Level Access Checking.
- Improvements to the GENERICOWNER function.

Overview

- Problem Statement / Need Addressed / User Stories
  - RACF Field Level Access Checking (FLAC) allows RACF admins the optional ability to delegate the administration of certain RACF database fields (those not in the BASE segment) to other low-level (not RACF SPECIAL) administrators.
    - This delegated authority allows a user to manipulate said fields for all profiles in the specified class.
    - **`PERMIT USER.OMVS.UID id(joe) access(update) class(FIELD)`** allows JOE to set any OMVS(UID(xx)) on the system.
  - RACF GROUP-SPECIAL support allows a different type of administrative delegation, where a low-level administrator is allowed to administer profiles which are owned by a specific RACF GROUP. This administrative ability is limited to the BASE segment data only.
    - **`Connect joe group(G1) special`**
      Gives joe the ability to manipulate most BASE segment fields in profiles owned by group G1.

Overview

- Problem Statement / Need Addressed / User Stories
  - A user with GROUP-SPECIAL authority may not manipulate fields outside of the BASE segment.
  - A user with FLAC access to some given fields is able to manipulate those fields for all profiles on the system.
  - There is no way to create an low-level (non-SPECIAL) administrator with the ability to fully administer profiles using GROUP-SPECIAL authority, without also giving them the ability to manipulate the non-BASE fields of profiles outside their group.
- Solution
  - Add an option to further scope FLAC by the rules governing BASE data manipulation.
  - This allows a user to manipulate non-BASE fields, but only for profiles in which they are also allowed to manipulate the BASE fields.
- Benefit
  - Added flexibility in administrative delegation.

Usage & Invocation

- The new functionality is activated by defining a new profile in the FIELD class
  - **`rdefine field FLAC.SKIP.BASECHECK`**
- If this profile does not exist, the administrative capabilities of all users are unchanged. Everything works like it did in z/OS 2.2.
- Otherwise, for a user to manipulate a non-BASE segment in a RACF profile, they must:
  - Have access to the FIELD class profile covering the field. (old)
  - Have BASE access to the profile, based on the rules enforced by the command processor via profile ownership, group-special or other. (new)
- Users with READ access to FLAC.SKIP.BASECHECK are not subject to the BASE access check.
- If the user can manipulate the installation-data field, they have BASE access.

Usage & Invocation

- To help understand this functionality, note that:
  - The only difference between a system with the FLAC.SKIP.BASECHECK profile defined, and one without is when the new profile exists, certain users may be denied access to non-BASE fields, where they would have access if the new profile was not defined.
  - Another way to think about this functionality is to assume that all users require BASE access in order to take advantage of their FIELD permissions, unless they have READ access to the new profile, or the profile does not exist.

Usage & Invocation

- Users with access of NONE to FLAC.SKIP.BASECHECK are still able to alter the fields in their user profiles that have UPDATE permission granted to &RACUID.
- RACF SPECIAL users have never been subject to FIELD class checking, and this is unchanged. The FLAC.SKIP.BASECHECK profile does not apply to RACF SPECIAL users.

Usage & Invocation

- The changes to Field Level Access Checking are **only** available via the alter and list RACF command processors, r_admin, and the RACF ISPF panels.
- Installation written programs using ICHEINTY or RACROUTE EXTRACT cannot make use of the new FLAC functionality.
  - Users with NONE access to FLAC.SKIP.BASECHECK will always fail to manipulate non-BASE fields. All others users will be successful, assuming they have appropriate access to the specified FIELD profiles.
  - This is due to the architecture of RACF and the way Field Level Access Checking was originally implemented.

Migration & Coexistence Considerations

- If the FLAC.SKIP.BASECHECK profile is not defined, nothing changes from z/OS 2.2.
- There are 2 ways to set up the FLAC.SKIP.BASECHECK profile, depending on the current use of FLAC on the system.
  - uacc(none) – recommended, especially if FLAC is not currently in use. Grant READ access to users with the need to manipulate profile fields system-wide.
  - uacc(read) – Allows existing administrators to retain their system-wide FLAC authority. New administrators should be given **`PERMIT FLAC.SKIP.BASECHECK ... access(NONE)`** if there is the need to scope their non-BASE updates to only profiles they own or administer via group-special.

Overview

- Problem Statement / Need Addressed / User Stories
  - The RACF GENERICOWNER (SETR GENERICOWNER) setting does not work properly with profiles in member and grouping classes.
- Solution
  - A new ENHANCEDGENERICOWNER option is added to SETROPTS which does support member and grouping classes.
- The GENERICOWNER functionality is usable in more classes.

Usage & Invocation

RACF member & grouping class overview

- Imagine a profile protecting a CICS transaction:
  - TCICSTRN:TRN1
    - uacc(none)
    - ACL: 50 users, 12 groups..
    - conditionalACL: 20 users with all sorts of conditions.
    - etc..
  - A sizable number of RACF commands to set all of this up.
- What if a second transaction, XHJE, is created which needs to be protected in exactly the same way?
  - Another bunch of commands.
- Now, 50 more transactions.
  - Yuck.

Usage & Invocation

RACF member & grouping class overview – continued.

- This can be made much easier if GROUPING profiles are used. Grouping profiles are created in a special grouping class which corresponds to a given member class.
- In thisr example, TCICSTRN is the member class and GCICSTRN is the grouping class.
- Profiles in the member class are 'normal' profiles and they are checked in the usual fashion.
- Profiles in the grouping class are not normal.
  - **`rdef GCICSTRN fakename addmem(TRN1 XHJE) ...`**
  - **`permit (all the users and groups as necessary)`**
  - **`setr raclist(gcicstrn) refresh`**

Usage & Invocation

RACF member & grouping class overview – continued.

- When the grouping class is raclisted, each profile is read, for every item in the member list, generate a profile in raclist cache containing all of the attributes of the 'fake' group profile.
  - raclist cache:
    TRN1 - full ACL from fakename
    XHJE – full ACL from fakename
- Updates are easy.
  - **`ralter gcicstrn fakename addmem(XXXX YYYY ...)`**
  - **`permit fakename id(guyWhoQuit) delete`**
  - **`setr raclist(GCICSTRN) refresh.`**
- Some member/grouping class pairs are special-cased and work very differently, these are out of scope for this discussion (racfvars, program, seclabel...)

Usage & Invocation

Existing GENERICOWNER support

- In order to define general resources to RACF a user must be SPECIAL, or have CLAUTH for that class defined in their user profile.
  - There are a few additional rules enforced by the RDEFINE command.
- GENERICOWNER is an additional control which restricts the ability of CLAUTH users to define profiles under certain circumstances.

Usage & Invocation

Existing GENERICOWNER support - continued

- If a generic profile exists which covers the profile being defined, the CLAUTH user must OWN the covering generic.
  - (issued by SPECIAL) **`rdef terminal A* uacc(none) owner(notme)`**
  - **`rdef terminal ABC`** ← NOT AUTH!!!
  - **`rdef terminal AA*`** ← NOT AUTH
  - **`rdef terminal DEF`** ← OK, (provided D* is owned by me.)
- GENERICOWNER functionality is for general resources only and does not work with DATASET profiles.

Usage & Invocation

Existing GENERICOWNER support - continued

- GENERICOWNER is documented to not work with grouping classes:
  - (**Security Administrators Guide**)
  - For example, when working with general resource grouping classes, assume that profile A* exists in the TERMINAL class and is owned by a group that user ELAINE does not have group-SPECIAL authority to. If the GENERICOWNER option is in effect, it will prevent user ELAINE from defining a more specific profile in the member class (for example, by using the command RDEF TERMINAL AA*). **However, having the GENERICOWNER option in effect will not prevent user ELAINE from defining a profile if specified on the ADDMEM operand for the grouping class profile (such as with the command `RDEF GTERMINL profile-name ADDMEM(AA*)`).**

Usage & Invocation

ENHANCEDGENERICOWNER

- The new ENHANCEDGENERICOWNER option of SETROPTS fixes this problem.
- **`SETROPTS [NOGENERICOWNER | GENERICIOWNER | ENHANCEDGENERICOWNER]`**
- When ENAHNCEDGENERICOWNER is active, and rdefine or ralter is used to add members to grouping profiles, the members are checked to insure they pass the GENERICOWNER test.

Usage & Invocation

ENHANCEDGENERICOWNER

- Given:
  - **rdef terminal A* uacc(none) owner(notme)**
- **SETR GENERICOWNER**
  - **rdef terminal ABC ← fail**
  - **rdef terminal AA* ← fail**
  - **rdef gterminl fake addmem(AA*) ← success**
- **SETR ENHANCEDGENERICOWNER**
  - **rdef terminal ABC ← fail**
  - **rdef terminal AA* ← fail**
  - **rdef gterminl fake2 addmem(AA*) ← fail**

Usage & Invocation

ENHANCEDGENERICOWNER support - continued

- The 'top' profile MUST be defined in the member class, or else [ENHANCED]GENERICOWNER does not take it into account.
  - **Good:**
    - **`rdef terminal A* uacc(none) owner(notme)`**
  - **Not Good:**
    - **`rdef gterminl fake uacc(none) addmem(A*)`**
- [ENHANCED]GENERICOWNER never has and still does not apply to RACF-SPECIAL users.
- Certain RACF classes are excluded, see documentation.

Interactions & Dependencies

- None

Installation

- None

Session Summary

- Field Level Access Checking enhancements.
- ENHANCEDGENERICOWNER option of SETROPTS

Appendix

- z/OS Security Server RACF Callable Services
  - Add ENHANCEDGENERICOWNER keyword to r_admin.
- z/OS Security Server RACF Command Language Reference
  - Add ENAHNCEDGENERICOWNER keyword to SETROPTS.
- z/OS Security Server RACF Macros and Interfaces
  - Add SMF record information related to new ENHANCEDGENERICOWNER keyword.
- z/OS Security Server RACF Security Administrator's Guide
  - Field Level Access Checking enhancements
  - ENHANCEDGENERICOWNER functionality.