//* IBM Z Day
//* open | on | secure

October 1, 2024

IBM TechXchange Virtual Event

How to Implement and Enforce Your Security Policy

Joel Tilton, CISSP

Director, Mainframe Security Engineering

Disclaimers

- All products, trademarks, and information mentioned are the property of the respective vendors.
- Mention of a product does not imply a recommendation.
- Always test new profiles on a non-production system.
- Only you can prevent IPLs!
- The views expressed are his own personal views, and are not endorsed or supported by, and do not necessarily express or reflect, the views, positions or strategies of his employer.

About Joel Tilton

Joel Tilton is a former employee of IBM, where he got his start with mainframes, who continues to champion mainframe security issues and solutions.

The majority of Joel's experience is focused on IBM mainframe systems, where he performs as a Technician, project manager and a Director. Joel's specialist subject is IT Security, in particular z/OS and associated subsystems (CICS, DB2, MQ, SERVAUTH IPs & Ports, & zSecure) security with RACF.

Joel has a true passion for security and the mainframe. Long live the mainframe!

Over 25+ years technical IT experience, the majority of which was gained in hands-on technical roles, performing a variety of duties in diverse and complex environments.

Joel also leads the NY / Tampa Bay / Raleigh / Dallas RACF Users group https://racfusers.com/

Session objectives

- What is zSecure Command Verifier?
- Design Command Verifier policies redefining what it means to have system SPECIAL
- Add another layer of security on SETROPTS commands
- Enforce Privilege Boundaries with =NOCHANGE policies
- Command Level Profiles for Increased Granularity
  - C4R.*command*.=SPECIAL / AUDITOR
- Automate Routine RACF commands
- Protect your audit remediation investment
  - Ensure remediated profiles stay remediated
- https://www.ibm.com/docs/en/szs/3.1.0?topic=command-verifier
- Most importantly, have fun

How to Carve the system SPECIAL turkey?

zSecure Command Verifier Tips

- Uses IRREVX01 Dynamic Exit Point
- Gets control both **before and after** RACF
    - Allows for insertion of RACF commands

- Does not apply to the following commands of course:
    - RVARY RACLINK RACDCERT RACPRIV RACMAP

- Uses XFACILIT by default
    - Longer resources names than FACILITY can provide are necessary ➔ 246

- Qualifiers with = & / can **not** be covered by a generic
    - = are mandatory policy profiles; think of them as overrides
    - / are default policy profiles; only provide a value if command issuer does not specify

- Can customize to use your own general resource class
    - Recommend setting default RC to 4 ➔ This is how the code really works
    - zSecure Access Monitor Simulations
- + = a single *
    - To build a profile to protect a backstop you would RDEFINE ➔ C4R.RACF.++

Validation of IRREVX01 – Close the Loop for Auditors

- zSecure Alert now has an alert to validate if IRREVX01 is disabled in v2.1.1 ➔ RACF control alert 1508
- Recommend Security
  - RDEFINE FACILITY CSVDYNEX.** uacc(NONE) audit(ALL(READ))

- CSV420I MODULE C4RMAIN HAS BEEN DELETED FROM EXIT IRREVX01

- Validate IRREVX01 is there by issuing:
- **D PROG,EXIT,EXITNAME=IRREVX01**

```
CSV461I 05.14.24 PROG,EXIT DISPLAY 296
 EXIT              MODULE   STATE MODULE   STATE MODULE   STATE
 IRREVX01          C4RMAIN    A
```

- Requires READ to CSVDYNEX.LIST in FACILITY
  - Recommended security RDEFINE FACILITY CSVDYNEX.LIST uacc(NONE) audit(failures(READ))

What Command Verifier *IS* and What it *IS NOT*

- Abstracts controls for RACF commands back into RACF itself
- Command Verifier is / can:
  - Provides tighter control of RACF Commands
    - Do system SPECIALs really need to have that much power every time at logon?
  - Uses dynamic exit point IRREVX01
  - Complements RACF with additional security; prevent security elevation attacks

- Command Verifier is not:
  - A policy rule editor; you need to be able to create the rules on your own
  - A replacement for a good security policy
  - A replacement for the RACF Systems Programmer / Security Engineer

- REMINDER: Will not work for the following commands
  - RVARY
  - RACLINK
  - RACDCERT
  - RACPRIV
  - RACMAP

Tightening SETROPTS Command Security

- Why? Because it is not access we need 24x7
- **C4R.RACF.**   UACC(READ)   AUDIT(FAILURES(READ))**
  - READ   = SETR RACLIST() REFRESH & SETR LIST
  - UPDATE   = All other SETR commands
- Permit ***tightly*** controlled group with UPDATE access
  - **C4R.CONNECT.ID.*owner.group_name***
- Use CONNECT REVOKE so using authority takes THOUGHT
  - **CONNECT JOEL GROUP(SETROPTS) OWNER() REVOKE**
  - Set up zSecure “Connect to an import group” Alert ID 1701
- Guard against accidents with SETROPTS KDFAES settings
- C4R.RACF.USER.PASSWORD.ALGORITHM
- C4R.RACF.USER.PASSWORD.SPECIALCHARS
  - Empty ACLs!

Tightening SETROPTS Command Security – Refreshes

- **C4R.RACF.*class*.GENERIC**
- **C4R.RACF.*class*.RACLIST**
- **C4R.RACF.DATASET.GENERIC   UACC(READ)AUDIT(FAIL(READ))**
  - Permit NONE for unauthorized Users
  - Only security engineering team should need
  - UPDATE for SETROPTS group controls SETR NOGENERIC(DATASET)
    - Can you imagine would happen if this command were issued?

A Word About SETROPTS LIST & C4R.RACF.LIST

- Could we secure SETROPTS LIST? Of course!
- `C4R.RACF.LIST    UACC(NONE)  AUDIT(FAILURES(READ))`

- What has really been achieved?

- Only locking it away from people who do not know how to write code
  - Which still does adds security value in my opinion

- The SETROPTS LIST information comes from the RCVT

- The RCVT can not live in fetch-protected storage due to many problem-state programs

Restrict Access to C4R.RACF.** Policies – =NOCHANGE

- **C4R.XFACILIT.=NOCHANGE.C4R.RACF.** UACC(NONE) AUDIT(FAILURES(READ)) APPLDATA(‘LEVEL=xx’)**
- Permit highly restricted group UPDATE
- We have now abstracted the ability to modify any field or delete any profile in the XFACILIT class starting with C4R.RACF
- If you are not on this ACL with UPDATE then your RACF command will fail
- Ensure only authorized users can administer sensitive RACF profiles

NOCHANGE Squared - Let’s have Some Fun Now!

- **C4R.XFACILIT.=NOCHANGE.C4R.XFACILIT.=NOCHANGE.C4R.RACF.** UACC(NONE) AUDIT(FAILURES(READ)) APPLDATA(‘LEVEL=xx’)**
- _**C4R.XFACILIT.=NOCHANGE.C4R.RACF.** UACC(NONE) AUDIT(FAILURES(READ)) APPLDATA(‘LEVEL=xx’)**_

- **C4R.XFACILIT.=NOCHANGE.C4R.SERVAUTH.=NOCHANGE.EZB.PORTACCESS.++ UACC(NONE) AUDIT(FAILURES(READ)) APPLDATA(‘LEVEL=xx’)**

- We have set up a NOCHANGE policy to protect administration of the NOCHANGE policy profile
- Permit highly restricted group UPDATE

- We have now abstracted the ability to modify any field or delete any profile in the XFACILIT class starting with C4R.XFACILIT.=NOCHANGE.C4R.RACF.**

- If you are not on this ACL with UPDATE then your RACF command will be failed!

- Ensure only authorized users can administer sensitive RACF profiles

Protect System and Group Authorities

- C4R.USER.ATTR.SPECIAL.**   UACC(NONE) AUDIT(FAILURES(READ))
- C4R.CONNECT.ATTR.SPECIAL.** UACC(NONE) AUDIT(FAILURES(READ))
- C4R.USER.ATTR.SPECIAL.*owner.UserID*
- READ = NOSPECIAL
- UPDATE = SPECIAL

- Permit highly restricted group UPDATE

- What type of attack vector might this protect?

- If you are not on this ACL with UPDATE then you will never issue ADDUSER / ALTUSER UserID SPECIAL ever again!

Allow Use of PERMIT Command to DATASET profiles

- Allow certain users to issue PERMITs to datasets all day long without need for SYSTEM or group SPECIAL

- In Three Simple Pieces:

- `C4R.PERMIT.=SPECIAL`
  - UPDATE access for users that need to issue PERMIT commands

- `C4R.DATASET.ACL.**` ➔ `C4R.class.ACL.userid.access.profile`
  - UPDATE to Users that need to administer dataset profiles

- `C4R.*.ACL.**` ➔ `C4R.class.ACL.userid.access.profile`
  - UPDATE to all system SPECIALs so they can still use PERMIT for general resources

Control Permits based on Group Naming Structure

- Allow PERMIT commands for certain group patterns; exclude PERMIT DELETE commands

- C4R.DATASET.ACL.*group*.DELETE.** UACC(NONE)
  - Tightly control removal of access
- C4R.DATASET.ACL.*group*.** UACC(UPDATE)
  - Allow native RACF authority to handling granting access

- Good idea to control self-authorization
- C4R.*class*.ACL.=RACUID.*access*.*profile*
  - Control permits to your UserID

- C4R.*class*.ACL.=RACGPID.*access*.*profile*
  - Control permits to groups that you are connected

Control Whom can Grant Access to the RACF DB

- C4R.DATASET.=NOCHANGE.*profile*
  - Must set ‘level=xx’ in appldata to match level setting of profile

-
```
RDEFINE C4R.DATASET.=NOCHANGE.SYS1.RACF*.**
appldata(‘level=0’) UACC(NONE) AUDIT(ALL(READ))
OWNER()
```

- UPDATE for authorized personnel; elevated privilege group
- =NOCHANGE can not be covered by generics

- Caveats:
  - Set a LEVEL value once and don’t change it.

Prevent Permits to IBMUSER & SYS1

- **C4R.DATASET.ACL.IBMUSER.** UACC(NONE) AUDIT(ALL)**
  - Empty ACL!
  - The whole world knows about this account. Do not use it. Do not grant access to it.
  - **ALU IBMUSER REVOKE RESTRICTED PROTECTED**

- **C4R.DATASET.ACL.SYS1.** UACC(NONE) AUDIT(ALL)**
  - Empty ACL!
  - Hopefully you are not using SYS1 to grant access either ☹

- Imagine the possibilities if you expand this to other sensitive groups/UserIDs/ACLs to ensure nobody can “go crazy” with the PERMIT command

Control the Powerful RESET keyword

- Set policies for using RESET since it can be extremely dangerous if used improperly

- `C4R.*.ACL.=RESET.** UACC(NONE) AUDIT(ALL(READ))`
  - Standard Access Control List
  - Empty ACL

- `C4R.*.CONDACL.=RESET.** UACC(NONE) AUDIT(ALL(READ))`
  - Conditional Access Control List
  - Empty ACL

- Example:
- PERMIT ‘CRITICAL.DATASET’ ID(batch01) access(UPDATE) RESET
- PERMIT ‘CRITICAL.DATASET’ ID(batch02) access(UPDATE) RESET
- PERMIT ‘CRITICAL.DATASET’ ID(batch03) access(UPDATE) RESET

Control CONNECT Commands to Isolate Privilege Boundaries

- **`C4R.CONNECT.ID.`*`group.UserID`***
  - UPDATE grants authority to issue CONNECT command
  - 42 policy profiles in total

- **`C4R.CONNECT.ID.`*`privilege_group_pattern.UserIDPatter*`***

- **`C4R.CONNECT.ID.everyday_group_pattern.UserIDPattern*`**

- **`C4R.CONNECT.ID.**`**
  - CONNECT command backstop
  - Yes I actually control the ability for anyone to issue a CONNECT command in addition to native RACF security

- Control CONNECT command to sensitive groups
  - Security engineers, admins, system programmers

Controlled Temporary Special – Isolate Commands for a Help Desk

- Allow a help desk to only reset or resume specific UserIDs

- **C4R.ALTUSER.=CTLSPEC**
  - UPDATE to Users that need to issue ALTUSER but with controls
  - So you have ALTUSER but if and only if you also have access to a policy profile for each and every keyword

- **C4R.USER.ATTR.RESUME.*group.UserID***
- **C4R.USER.ATTR.PASSWORD.*group.UserID***

- **C4R.USER.ATTR.PROTECTED.** UACC(NONE) AUDIT(FAILURES(READ)**
- **C4R.USER.PWEXP.** UACC(NONE) AUDIT(FAILURES(READ)**

- WARNING: Be mindful of UACCs on C4R.USER policy profiles!

Read Only Auditor – With Granularity

- Of course, with z/OS 2.2 ROAUDITOR is available at the UserID and Group level

- Define the following `UACC(NONE) AUDIT(NONE)`
  - `C4R.LISTDSD.=AUDITOR`
  - `C4R.LISTGRP.=AUDITOR`
  - `C4R.LISTUSER.=AUDITOR`
  - `C4R.RLIST.=AUDITOR`
  - `C4R.SEARCH.=AUDITOR`

- Note the third qualifier can **not** be covered by a generic!
  - Check the documentation for details like this
- UPDATE only valid access level

- More granularity than ROAUDITOR
- Will not include SETROPTS LIST access ☺

- Yes this can be done for SPECIAL too
- C4R.*command*.=SPECIAL
- `C4R.SETROPTS.=SPECIAL`

RACF Command Automation

- **C4R.CONNECT.=PSTCMD.GROUP.*group_name* *APPLDATA(‘ALU (&PROFILE) MFA(ACTIVE FACTOR() TAGS(REGSTATE:OPEN));ALU (&PROFILE) NOPASSWORD OWNER(group_name);ALU (&PROFILE) REVOKE ’)***
- Multiple RACF commands separate by semicolon ;
- Ensure we always set up certain UserIDs for MFA, change their owner, remove their password and revoke them
- C4R.ALTUSER.=PRECMD.SPECIAL
  - ALTUSER (&PROFILE) REVOKE NOPASSWORD OWNER(GROUP_NAME)
  - Goal is to lock up a highly privileged UserID until its needed

Protect Against Unauthorized Dynamic CDT Changes

- RALT CDT $$OCCAN CDTINFO(NORACLIST)

```
T0094020 00000281  ICH408I USER(        ) GROUP(        ) NAME(TILTON,JOEL         ) 950
     950 00000281    C4R.CDT.=NOCHANGE.$$OCCAN CL($C4RVFY )
     950 00000281    INSUFFICIENT ACCESS AUTHORITY
     950 00000281    FROM C4R.CDT.=NOCHANGE.** (G)
     950 00000281    ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

- **C4R.CDT.=NOCHANGE.****
  - Control modification, deletion and creation of existing profiles

- **C4R.CDT.CDTINFO.****
  - Control access to CDTINFO segment
  - READ = Browse
  - UPDATE = modify

- **C4R.CDT.ID.****
  - Control creation of existing profiles

Some Super Cool Things

- `C4R.DATASET.TYPE.DISCRETE.**  UACC(NONE)`
  - Empty ACL! Even system SPECIALs!!
  - Prevent discrete dataset profiles ➔ ICH408I

- C4R.`LISTDSD.TYPE.AUTO.**  UACC(READ)`
  - Change LISTDSD behavior so it always finds best fitting generic instead
  - Discrete search ignored!

- `C4R.*./OWNER.**  UACC(READ)`
  - Automatically assign OWNER() of your default group
  - Perhaps better than your UserID

Setting Up Command Audit Trail

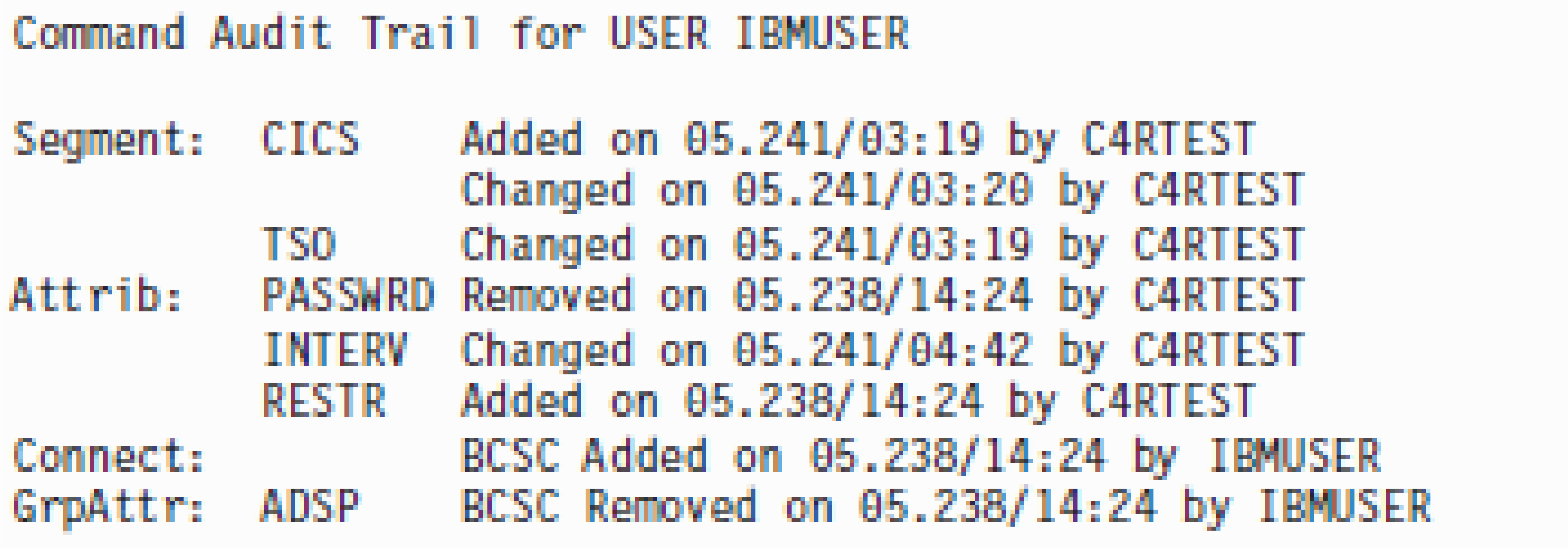
- The C4RMAIN module can collect data for these classes & attributes
- Stores in USRDATA fields; ensure you have space in your RACF DB

- `C4R.class.=CMDAUD.=ACL.** UACC(NONE)`
- `C4R.class.=CMDAUD.=ATTR.** UACC(NONE)`
- `C4R.class.=CMDAUD.=CONNECT.** UACC(NONE)`
- `C4R.class.=CMDAUD.=MEMBER.** UACC(NONE)`
- `C4R.class.=CMDAUD.=SEGMENT.** UACC(NONE)`
- `C4R.class.=CMDAUD.=SURROGATE.** UACC(NONE)`
  - `Records surrogate UserID instead of Execution UserID`
  - `GA in zSecure v2.5 Q3 2021`

- `C4R.class.=CMDAUD.=MAINT.** UACC(NONE)`
  - Controls ability to display and destroy
  - READ = automatically displayed when issuing any RACF list command
  - UPDATE = use C4RCATMN command to display
  - CONTROL = use C4RCATMN to delete audit trail data

Command Verifier Audit Trail

- Displays with RACF list commands at the very end
  - C4R.LISTUSER.=SPECIAL/AUDITOR
- No way to display with zSecure UI yet…
  - Idea ZSECURE-I-115
- Does not track SETROPTS changes yet…
  - Idea ZCMD-I-63



```
Command Audit Trail for USER IBMUSER

Segment:  CICS    Added on 05.241/03:19 by C4RTEST
                  Changed on 05.241/03:20 by C4RTEST
          TSO     Changed on 05.241/03:19 by C4RTEST
Attrib:   PASSWRD Removed on 05.238/14:24 by C4RTEST
          INTERV  Changed on 05.241/04:42 by C4RTEST
          RESTR   Added on 05.238/14:24 by C4RTEST
Connect:          BCSC Added on 05.238/14:24 by IBMUSER
GrpAttr:  ADSP    BCSC Removed on 05.238/14:24 by IBMUSER
```

And that’s how you carve up system SPECIAL!

Fuel Your AI at the ultimate IBM learning event

IBM TechXchange Conference
October 21-24, 2024
Mandalay Bay – Las Vegas

IBM Z® and IBM® LinuxONE clients, partners, user groups, & communities share tech experiences

Track Topics

- Enterprise AI
- AI-assisted app modernization
- Security
- IBM watsonx Code assistant for Z
- Sustainability
- DevOps
- Skills
- AIOps

Register Now! ibm.biz/ibm-techxchange

IBM TechXchange Conference 2024

IBM

Deep learning experience.
Specific. Hands-on. Real.

200+ IBM Z and IBM LinuxONE Deep tech sessions & instructor-led labs ... including IBM z/OS Academy

Engage demos & SMEs @ our sandbox

- The Plexi
- LEGO Brick Model
- Demos, SMEs, AMAs Games

SHARE
EDUCATE • NETWORK • INFLUENCE

IDUG

OPEN MAINFRAME PROJECT

Community, user groups, and customer advisory boards