UNIX Systems Services Security Setup (USS SS)

Julie Bergh

Review

- Overview
- BPXPRMxx
- BPXISEC1

Display OMVS Options

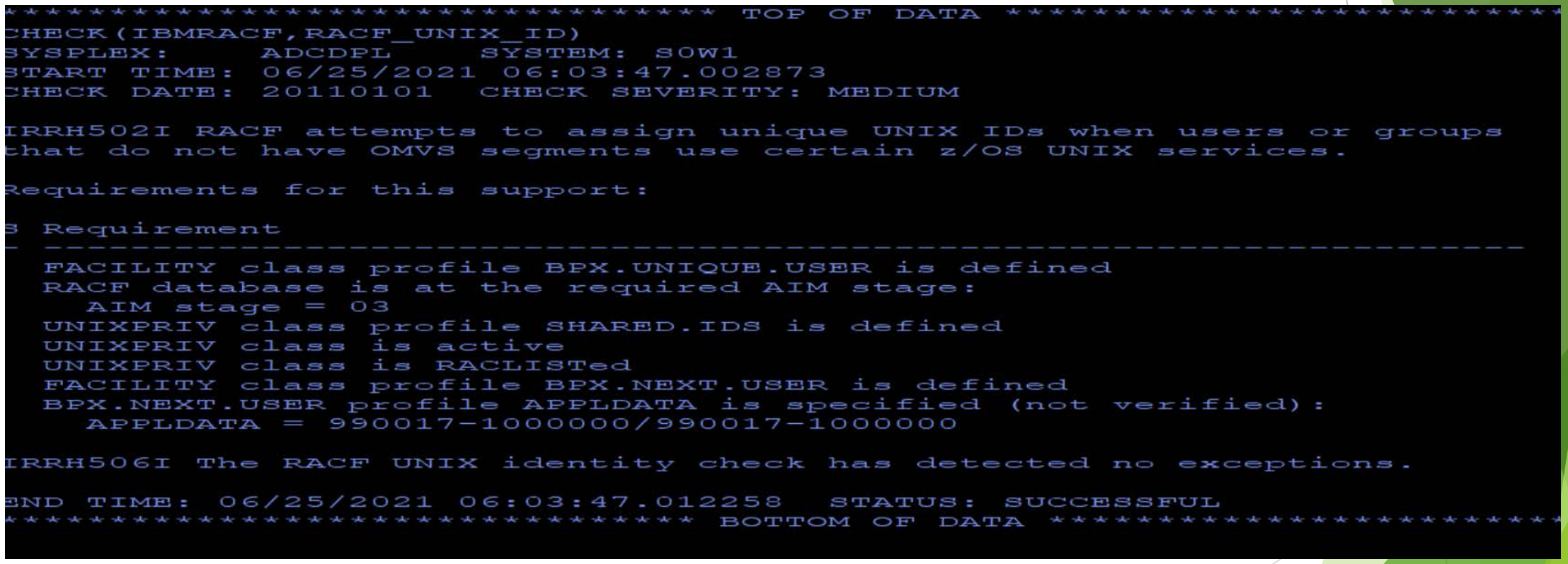
```
D OMVS,OPTIONS
```

```
BPXO043I 13.45.28 DISPLAY OMVS 783
OMVS       000F ACTIVE                    OMVS=(00,01,BP,IZ,RZ)
CURRENT UNIX CONFIGURATION SETTINGS:
MAXPROCSYS          =          1200      MAXPROCUSER         =          3
MAXFILEPROC         =         65535      MAXFILESIZE         = NOLIMI
MAXCPUTIME          =          7200      MAXUIDS             =
MAXPTYS             =           800      MAXIOBUFUSER        =
MAXMMAPAREA         =         40960      MAXASSIZE           = 214748
MAXTHREADS          =         20000      MAXTHREADTASKS      =          3
MAXCORESIZE         =       4194304      MAXSHAREPAGES       =      32768
IPCMSGQBYTES        =      20971520      IPCMSGQMNUM         =         10
IPCMSGNIDS          =           500      IPCSEMNIDS          =
IPCSEMNOPS          =            25      IPCSEMNSEMS         =
IPCSHMMPAGES        =         25600      IPCSHMNIDS          =
IPCSHMNSEGS         =           500      IPCSHMSPAGES        =        262
SUPERUSER           = OMVSKERN           ORKCOPY             = COPY
STEPLIBLIST         =
USERIDALIASTABLE=
PRIORITYPG VALUES:  NONE
PRIORITYGOAL VALUES:  NONE
MAXQUEUEDSIGS       =          1000      SHRLIBRGNSIZE       =     67108
SHRLIBMAXPAGES      =          4096      VERSION             = Z24A
SYSCALL COUNTS      = NO                 TTYGROUP            = TTY
SYSPLEX             = YES                BRLM SERVER         = N/A
LIMMSG              = NONE               AUTOCVT             = OFF
RESOLVER PROC       = RESOLVER           LOSTMSG             = ON
AUTHPGMLIST         = NONE
SWA                 = BELOW              NONEMPTYMOUNTPT     = NOWARN
SERV_LINKLIB        =
SERV_LPALIB         =
ALTROOT             =
MAXUSERMOUNTSYS     =             0      MAXUSERMOUNTUSER=
MAXPIPEUSER         =          8730      PWT                 = ENV
KERNELSTACKS        = ABOVE              UMASK               = NONE
SC_EXITTABLE        =
090       PATH=/
```

UNIX Superuser Health Check Entry

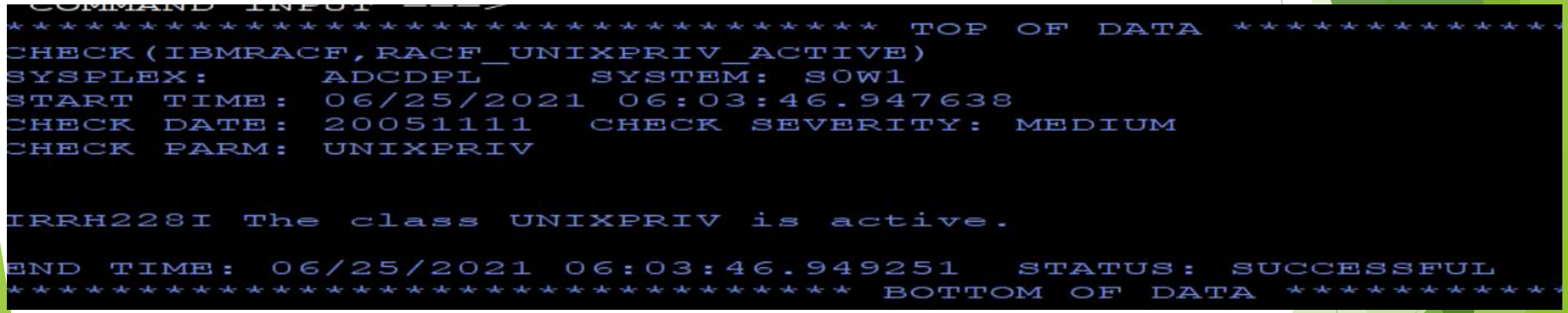
```
COMMAND INPUT ===>                                                  SCROLL ===>
********************************* TOP OF DATA **********************************
CHECK(IBMUSS,USS_SUPERUSER)
SYSPLEX:      ADCDPL      SYSTEM: SOW1
START TIME: 06/20/2021 06:03:46.681514
CHECK DATE: 20160331   CHECK SEVERITY: HIGH

BPXH003I z/OS UNIX System Services was initialized using
OMVS=(00,01,BP,IZ,RZ), where each 2-character item is a BPXPRMxx suffix.

BPXH086I No problems were found with userID OMVSKERN.

END TIME: 06/20/2021 06:03:46.694668   STATUS: SUCCESSFUL
******************************** BOTTOM OF DATA ********************************
```

RACF Health Check Entry



```
********************************** TOP OF DATA **********************************
CHECK(IBMRACF,RACF_UNIX_ID)
SYSPLEX:      ADCDPL      SYSTEM: S0W1
START TIME:  06/25/2021 06:03:47.002873
CHECK DATE:  20110101   CHECK SEVERITY: MEDIUM

IRRH502I RACF attempts to assign unique UNIX IDs when users or groups
that do not have OMVS segments use certain z/OS UNIX services.

Requirements for this support:

S Requirement
- -----------------------------------------------------------------------
  FACILITY class profile BPX.UNIQUE.USER is defined
  RACF database is at the required AIM stage:
    AIM stage = 03
  UNIXPRIV class profile SHARED.IDS is defined
  UNIXPRIV class is active
  UNIXPRIV class is RACLISTed
  FACILITY class profile BPX.NEXT.USER is defined
  BPX.NEXT.USER profile APPLDATA is specified (not verified):
    APPLDATA = 990017-1000000/990017-1000000

IRRH506I The RACF UNIX identity check has detected no exceptions.

END TIME:  06/25/2021 06:03:47.012258   STATUS: SUCCESSFUL
******************************** BOTTOM OF DATA *********************************
```

RACF Health Check Entry



OMVS Started Tasks / UMASK
SYS1.SAMPLIB(BPXISEC1)

```
ADDGROUP OMVSGRP OMVS(GID(x))  OWNER(some-group) SUPGROUP(some-group)

ADDUSER OMVSKERN DFLTGRP(OMVSGRP) OWNER(OMVSGRP) +
  OMVS(UID(0) HOME('/')    PROGRAM('/bin/sh')) NOPASSWORD

RDEFINE STARTED OMVS.*  OWNER(stc-grp) +
  STDATA(USER(OMVSKERN) GROUP(OMVSGRP) TRUSTED(YES))

RDEFINE STARTED BPXOINIT.* OWNER(stc-grp) +
  STDATA(USER(OMVSKERN) GROUP(OMVSGRP) TRUSTED(NO))
```

UMASK = NONE – Default

UID(0) – SUPERUSER

- Passes all z/OS UNIX Security checks
- Perform administrative activities
- Install products
- Can access and modify any files and directories
- Manages all z/OS UNIX processes
- Can run unlimited number of processes concurrently
- Propagates superuser privileges to forked child process
- Can change identity
- SUPERUSER – BPXROOT

```
ADDUSER BPXROOT DFLTGRP(OMVSGRP) OWNER(OMVSGRP)+  OMVS(UID(0)
HOME('/') PROGRAM('/bin/sh')) NOPASSWORD
```

UID(0) – SUPERUSER

- You can assign superuser authority in three ways:
  - Using resource profiles in the UNIXPRIV class (preferred method).
  - Using the BPX.SUPERUSER resources in the FACILITY class.
  - Assigning a UID of 0 (least desirable method).

NOTE: Started task running with TRUSTED or PRIVILIGED attribute are considered UNIX SUPERUSERs even when they have a UID of non zero in their OMVS segment

- **RECOMMENDATION:**
  - **Limit using BPX.SUPERUSER to those requiring it for installation purposes**
  - **Keep careful track of who this has been granted access to BPX.SUPERUSER**
  - **Log successful accesses**

UID(0) – SUPERUSER

- BPX Profiles in FACILITY class are used mainly to control UNIX functions and attributed used by UNIX programs (more on this later)

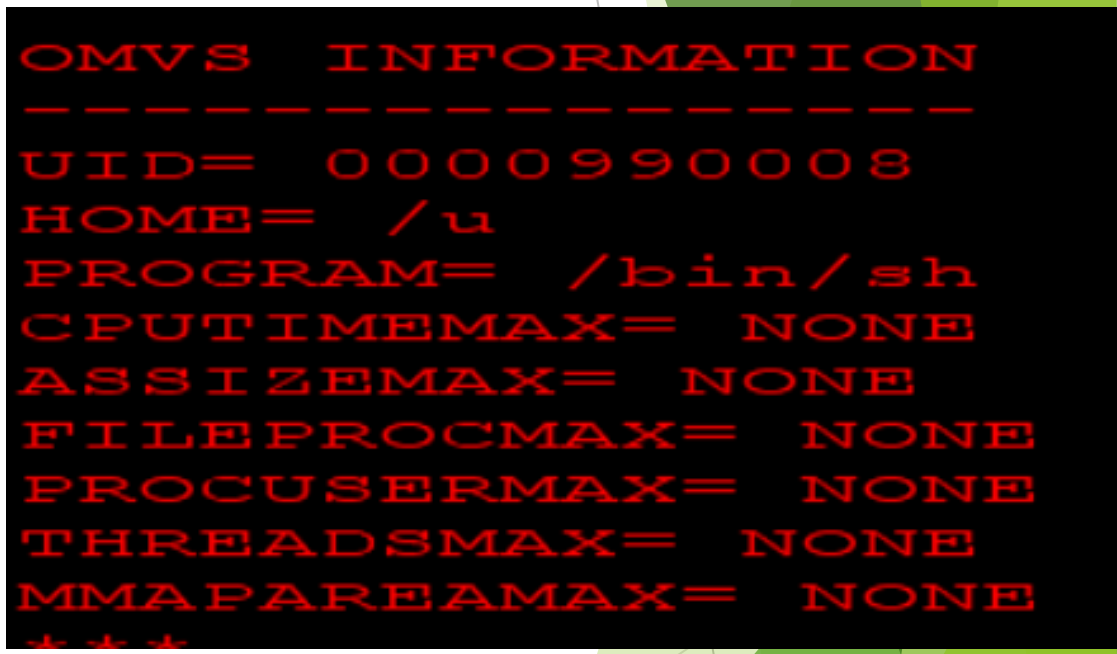
- READ access to BPX.SUPERUSER profile in RACF FACILITY class
  - Allows non-UID 0 to su to become UID 0

RDEFINE FACILITY BPX.SUPERUSER UACC(NONE) OWNER(some-grp)

PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(zzzzzzzz) ACCESS(READ)

UNIX User Segment Display

- All users and programs that need access to z/OS UNIX must have a **RACF user profile defined with an OMVS segment**, which contains 9 fields:
  - UID - 0 to 2147483647
  - HOME - directory in file system that becomes current when user goes into the shell
  - PROGRAM - Name of the program that will be started when user logon
  - CPUTIMEMAX, ASSIZEMAX, FILEPROCMAX, PROCUSERMAX, THREADSMAX, and MMAPAREAMAX



Automatic Assignment of UIDs & GIDs

- RACF can automatically generate a unique ID value, using the BPX.NEXT.USER profile and the AUTOUID & AUTOGID operands of the add/altuser and add/altgroup commands

- AG @ACCESS . . . . . OMVS(AUTOGID)

- AU USER1 . . . . . OMVS(AUTOUID)

- **NOTE:**
  - The FACILITY class does not have to be active for RACF to use BPX.NEXT.USER.
  - When creating the BPX.NEXT.USER profile, generic characters cannot be used in the name.
  - ALU and ALG commands, the AUTOUID and AUTOGID options cannot be used to change the ID value if one exists for the user.

Defining Automatic Values

- **BPX.NEXT.USER** in FACILITY class
  - APPLDATA is used contains the starting UID/GID value or range
  - The ‘/’ is the separator for the values in the APPLDATA
  - The starting value is the value RACF attempts to use in ID assignment, after determining that the ID is not in use. If it is in use, the value is incremented until an appropriate value is found.
  - UACC and Access List are not used
  - First value in APPLDATA represents UID
  - Second value in APPLDATA represents GID
  - RDEFINE FACILITY BPX.NEXT.USER OWNER(some-grp) APPLDATA(‘1234/5678’)
  - User of NOAUTO to not assign UID or GID
  - RDEFINE FACILITY BPX.NEXT.USER OWNER(some-grp) APPLDATA(‘NOAUTO/5678’)

BPX.UNIQUE.USER

Propagate common program, home, and other OMVS attributes to first-time UNIX users.

BPXMODEL

APPLDATA field of the BPX.UNIQUE.USER profile in the FACILITY class

RDEFINE FACILITY BPX.UNIQUE.USER APPLDATA('BPXMODEL')
Rule: Specify no generic characters in the BPX.UNIQUE.USER profile name.

&RACUID in BPXMODEL

ADDUSER BPXMODEL NAME('OMVS model user profile') NOPASSWORD RESTRICTED +
OMVS(HOME('/u/&racuid') PROGRAM('/bin/sh'))

Automatic Assignment of UIDs & GIDs

- Pre-requisites:
  - Define SHARED.IDS profile in the UNIXPRIV class, which will enforce unique UIDs & GIDs

Shared IDs

SHARED.IDS – in UNIXPRIV - Enforces unique UNIX identifiers (UID & GID)

- Prevents assignment of an ID which is already in use
- Does not affect pre-existing shared IDs
- SHARED operand allows identifiers to be shared (must have READ access)

RECOMMENDATION:

- Define SHARED.IDS profile UACC(NONE)
- Provide READ access to a very limited number of trusted Administrator

The AUTOUID and AUTOGID operands cannot be specified with the SHARED operand. Doing so results in command failure and message IRR52186I being issued.

Errors

RDEFINE UNIXPRIV SHARED.IDS OWNER($RESGRP)

ADDUSER USER1 OMVS(UID(12))

IRR52174I Incorrect UID 12. This value is already in use by USER99

ADDGROUP ABCD OMVS(GID(46))

IRR52174I Incorrect GID 46. This value is already in use by DEFG.

Override Shared IDs

PERMIT SHARED.IDS CLASS(UNIXPRIV) ID(user1) ACCESS(READ)

AU user1 OMVS(UID(0) SHARED) – OK

AU MYBUDDY OMVS(UID(0) SHARED)

IRR52175I You are not authorized to specify the SHARED keyword

- BPX.**

- RDEFINE FACILITY BPX.** OWNER(some-grp) UACC(READ)
- RDEFINE FACILITY BPX.** OWNER(some-grp) WARNING
- RDEFINE FACILITY BPX.** OWNER(some-grp) UACC(NONE)

UNIX related FACILITY & SURROGAT class profiles

Other BPX FACILTY Class profiles

BPX.CF - Controls access to the _cpl service. The __cpl callable service calculates coupling facility structure sizes that are required by the Coupling Facility Resource Manager (CFRM) policy through a web interface.

BPX.CONSOLE – Controls the _console() or _console2() services.

BPX.DAEMON – Controls which users are allowed to take on the identity of other users

BPX.DAEMON.HFSCTL - Controls which users with daemon authority are allowed to load uncontrolled programs from MVS libraries into their address space. BPX.DAEMON.HFSCTL does not allow generic profiles.

BPX.DEBUG - Controls which users can run restricted processes.

Other BPX FACILTY Class profiles

BPX.EXECMVSAPF.program_name – Allows unauthorized callers of the execmvs callable service to pass an argument that is greater than 100 characters to an authorized program

BPX.FILEATTR.APF - Controls which users are allowed to set the APF-authorized attribute in a z/OS UNIX file.

BPX.FILEATTR.PROGCTL - Controls which users are allowed to set the program control attribute..

BPX.FILEATTR.SHARELIB - Controls that extra privilege is required when setting the shared library extended attribute through the chattr() callable service.

BPX.JOBNAME – Controls which users are allowed to set their own job names by using the _BPX_JOBNAME environment variable or the inheritance structure on spawn().

Other BPX FACILTY Class profiles

BPX.MAINCHECK – Extends enhanced program security protection to your UNIX daemons and servers that do not use RACF execute-controlled programs.

BPX.MAP - Controls access to the _map and _map_init services.

BPX.POE - Controls access to the _poe service

BPX.SAFFASTPATH – Enables faster security checks for file system and IPC constructs. When the BPX.SAFFASTPATH FACILITY class profile is defined, the security product is not called if UNIX can quickly determine that file access will be successful

Other BPX FACILTY Class profiles

BPX.SERVER - Restricts the use of the pthread_security_np() service.

BPX.SMF – Checks whether the caller attempting to cut an SMF record is allowed to write an SMF record. It also tests whether an SMF type or subtype is being recorded.

BPX.SHUTDOWN - Controls access to the oe_env_np service to register and block for OMVS shutdown.

BPX.SRV.userid - Allows users to change their UID if they have access to BPX.SRV.*userid*, where *userid* is the MVS user ID that is associated with the target UID. BPX.SRV.userid is a RACF SURROGAT class profile.

Other BPX FACILTY Class profiles

BPX.STOR.SWAP - Controls which users can make address spaces non-swappable.

BPX.STICKYSUG.*program_name* - Enables the exec and spawn services to use the MVS program search order to locate the program to be run when the specified path name resolves to a file with the sticky attribute and either the set-user-id or set-group-id attributes.

BPX.UNLIMITED.OUTPUT - Allows users to use the _BPX_UNLIMITED_OUTPUT environment variable to override the default spooled output limits for processes

BPX.WLMSERVER - Controls access to the WLM server functions _server_init() and _server_pwu(). It also controls access to these C language WLM interfaces

UNIXPRIV Class

RDEF UNIXPRIV ** OWNER(some-grp) UACC(READ)

RDEF UNIXPRIV ** OWNER(some-grp) WARNING

RDEF UNIXPRIV ** OWNER(some-grp) UACC(NONE)

UNIXPRIV

CHOWN.UNRESTRICTED - Allows users to run the **chown** command to transfer ownership of their own files.

FILE.GROUPOWNER.SETGID - Specifies that a directory's set-gid bit is used to determine the group owner of any new objects that are created within the directory.

RESTRICTED.FILESYS.ACCESS - Specifies that RESTRICTED users cannot gain file access by virtue of the other permission bits. To override it for a specific user or group, the required minimum required access is READ.

SHARED.IDS – Discussed earlier

SUPERUSER.FILESYS.ACLOVERRIDE - Specifies that ACL contents override the access that was granted by SUPERUSER.FILESYS.

UNIXPRIV

SUPERUSER.FILESYS – Allows access to local files.

SUPERUSER.FILESYS.CHANGEPERMS- Allows users to use the chmod command to change the permission bits of any file and to use the setfacl command to manage access control lists for any file

SUPERUSER.FILESYS.CHOWN - Allows users to use the chown command to change ownership of any file

UNIXPRIV

SUPERUSER.FILESYS.DIRSRCH - Allows users to read and search any local directories.

SUPERUSER.FILESYS.MOUNT –

- **MOUNT** command or the **mount** shell command with the **nosetuid** option.
- **UNMOUNT** command or the **unmount** shell command with the **nosetuid** option.

SUPERUSER.FILESYS.QUIESCE - Allows user to run **quiesce** and **unquiesce** commands for a file system that is mounted with the **nosetuid** option.

SUPERUSER.FILESYS.PFSCTL - Allows user to use the (Physical File System) pfsctl() callable service

UNIXPRIV

SUPERUSER.FILESYS.USERMOUNT - Allows Nonprivileged users to mount and unmount file systems with the nosetuid option.

SUPERUSER.FILESYS.VREGISTER – Allows a server to use the vreg() callable service to register as a VFS file server

SUPERUSER.IPC.RMID - Allows user to run the **ipcrm** command to release IPC resources

UNIXPRIV

SUPERUSER.PROCESS.GETPSENT - ** Allows user to use the w_getpsent() and __getthent() callable services to receive data for any process. Allows users of the **ps command to output information about all processes.

SUPERUSER.PROCESS.KILL - Allows user to use the kill() callable service to send signals to any process. The minimum required access is READ.

**SUPERUSER.PROCESS.PTRACE - ** Allows user to use the ptrace() callable service through the dbx debugger to trace any process. (Authorization to the BPX.DEBUG resource also is required to trace processes that run with APF authority or BPX.SERVER authority.)

UNIXPRIV

SUPERUSER.SETPRIORITY - Allows user to increase own priority

SUPERUSER.SHMMCV.LIMIT – Allows users to create additional environment variables (up to 4,194,304)

RACF Health Check Entry

- RACF_SENSITIVE_RESOURCES check examines the security characteristics of several system-critical data sets and general resources other than data sets. UNIX related ones in FACILITY Class
  - BPX.DEBUG
  - BPX.WLMSERVER
  - BPX.FILEATTR.SHARELIB
  - BPX.JOBNAME
  - BPX.POE
  - BPX.SMF
  - BPX.STOR.SWAP
  - BPX.UNLIMITED.OUTPUT

RACF Health Check Entry

- RACF_SENSITIVE_RESOURCES check examines the security characteristics of several system-critical data sets and general resources other than data sets. UNIX related ones in UNIXPRIV and SURROGAT class
  - SUPERUSER.FILESYS.PFSCTL
  - SUPERUSER.FILESYS.QUIESCE
  - SUPERUSER.FILESYS.VREGISTER
  - SUPERUSER.IPC.RMID
  - SUPERUSER.SETPRIORITY
  - BPX.SRV.*userid*
  - SUPERUSER.PROCESS.GETPSENT
  - SUPERUSER.PROCESS.KILL
  - SUPERUSER.PROCESS.PTRACE

Auditing

- **DIRSRCH** - Controls auditing of directory searches
- **DIRACC** - Controls auditing for access checks for read/write access to directories
- **FSOBJ** - Controls auditing for all access checks for file system objects except directory searches
- **FSSEC** - Controls auditing for changes to the security data (FSP and ACL) for file system objects
- **IPCOBJ** - Specifies auditing options for IPC accesses.
- **PROCESS** - Controls auditing of changes to the UIDs and GIDs of processes.
- **PROCACT** - Controls auditing of functions that look at data from or effect other processes

Audit Classes

Class	SETROPTS AUDIT	SETROPTS LOGOPTIONS
FSOBJ	Creation and deletion of all file system objects	Access to files
DIRACC	N/A	Read/write access to directories
DIRSRCH	N/A	Search access to directories
FSSEC	N/A	Changes to security data of all file system objects
PROCESS	Dubbing and undubbing of processes	Changes to process identity (UID and GID)
PROCACT	N/A	Functions that inspect (e.g. getpsent) or update (e.g. kill, ptrace) other processes
IPCOBJ	Creation and deletion of InterProcess Communication objects	Access to IPC objects, and changes to permissions and ownership

Audit Classes - Comparison

DATASET auditing	UNIX file auditing
SETROPTS LOGOPTIONS for DATASET class controls access logging	SETROPTS LOGOPTIONS for FSOBJ, DIRACC, and DIRSRCH classes contols access logging
SETROPTS AUDIT(DATASET) audits profile creation/deletion	SETROPTS AUDIT(FSOBJ) audits file creation/deletion
SETROPTS AUDIT(DATASET) audits changes to RACF profiles	SETROPTS LOGOPTIONS for FSSEC audits changes to file owner, permission bits and audit settings
Profile-level auditing can be specified by profile OWNER (AUDIT option of ALTDSD)	File-level auditing can be specified by file owner (chaudit command)
Profile-level auditing can be specified by auditor (GLOBALAUDIT option of ALTDSD)	File-level auditing can be specified by auditor (chaudit command with -a option)

Audit Classes - Comparison...

DATASET auditing	UNIX file auditing
LOGOPTIONS with ALWAYS and NEVER overrides profile settings	same for file settings
LOGPTIONS with SUCCESSES or FAILURES merged with profile-level settings	same for file settings
LOGOPTIONS with DEFAULT uses the profile-level settings	same for file settings
Default profile setting is READ failures for owner options, and no settings for auditor options (implies UPDATE, CONTROL, and ALTER failures too)	Default is read, write, and execute failures for owner settings (note that UNIX permissions are not hierarchical - these are separate settings for each access type)
Display profile options with LISTDSD	Display file options with ls -W

Reporting

directory search record extension	check directory access record extension
check file access record extension	change audit record extension
change directory record extension	change file mode record extension
change file ownership record extension	clear SETID bits record extension
EXEC SETUID/SETGID record extension	GETPSENT record extension
initialize z/OS UNIX record extension	z/OS UNIX process completion record
KILL record extension	LINK record extension
MKDIR record extension	MKNOD record extension
mount file system record extension	OPENFILE record extension
PTRACE record extension	rename file record extension

Reporting

RMDIR record extension	SETEGID record extension
SETEUID record extension	SETGID record extension
SETUID record extension	SYMLINK record extension
UNLINK record extension	unmount file system record extension
check file owner record extension	check privilege record extension
open slave TTY record extension	IPCCHK record extension
IPCGET record extension	IPCCTL record extension
SETGROUP record extension	CKOWN record extension
access rights record extension	SETFACL record extension
DELFACL record extension	SETFSECL record extension

Troubleshooting Access – Brief Discussion

- High level UNIX access checking

References & Questions

- References
  - z/OS 2.4 UNIX Systems Services Planning
  - z/OS 2.4 Security Server RACF Auditor’s Guide
  - z/OS 2.4 Security Server RACF Administration Guide

z/OS UNIX Basic Security

Julie Bergh