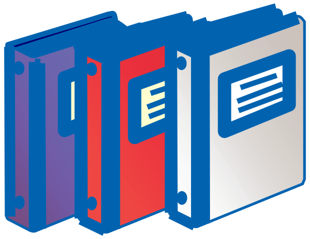
RACF and z/OS UNIX: Integrated more than you may know

Southern California RACF User's Group

October 2005



Bruce R. Wells
RACF Development, IBM
845-435-7498
brwells@us.ibm.com

Disclaimer

The information contained in this document is distributed on as "as is" basis, without any warranty either express or implied. The customer is responsible for use of this information and/or implementation of any techniques mentioned. IBM has reviewed the information for accuracy, but there is no guarantee that a customer using the information or techniques will obtain the same or similar results in its own operational environment.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used. Functionally equivalent programs that do not infringe IBM's intellectual property rights may be used instead. Any performance data contained in this document was determined in a controlled environment and therefore, the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM Products, programming or services in your country.

IBM retains the title to the copyright in this paper as well as title to the copyright in all underlying works. IBM retains the right to make derivative works and to republish and distribute this paper to whomever it chooses.

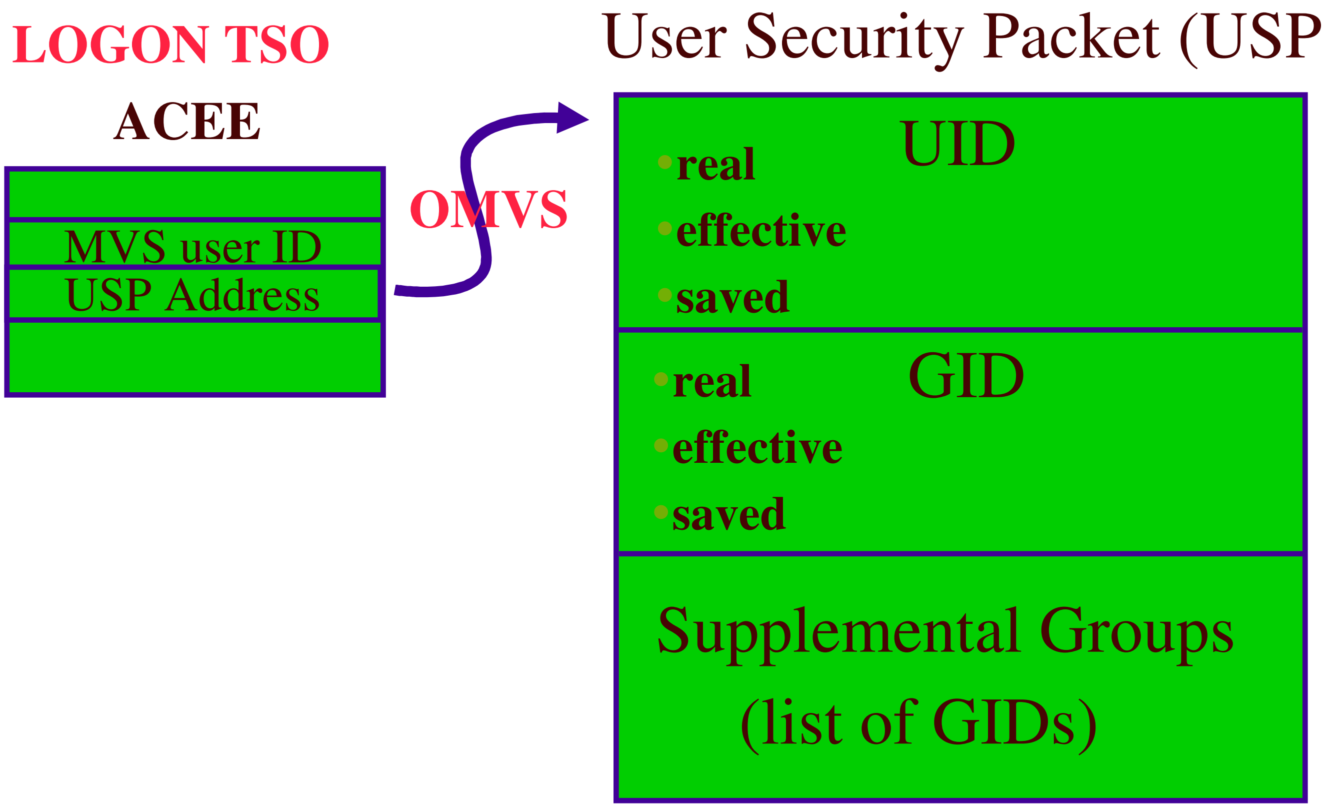
Trademarks

- **The following are trademarks or registered trademarks of the International Business Machines Corporation:**
  - z/OS
  - RACF
- **UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.**

Agenda

- Part 1 – UNIX identities
  - UNIX vs. MVS identity, user/group registry
  - Superusers: What they can do, and where you can find some Kryptonite
  - Sharing UIDs: unintentional identity theft
  - Automatic UIDs: let RACF figure it out
- Part 2 – file security
  - Those wacky UNIX permission bits
  - ACLs: not just in kneecaps anymore
  - Auditing: more like RACF than you think

UNIX identity (not drawn to scale)



LOGON TSO

ACEE

MVS user ID

USP Address

OMVS

User Security Packet (USP)

UID

- real
- effective
- saved

GID

- real
- effective
- saved

Supplemental Groups (list of GIDs)

From user's OMVS segment or from BPX.DEFAULT.USER

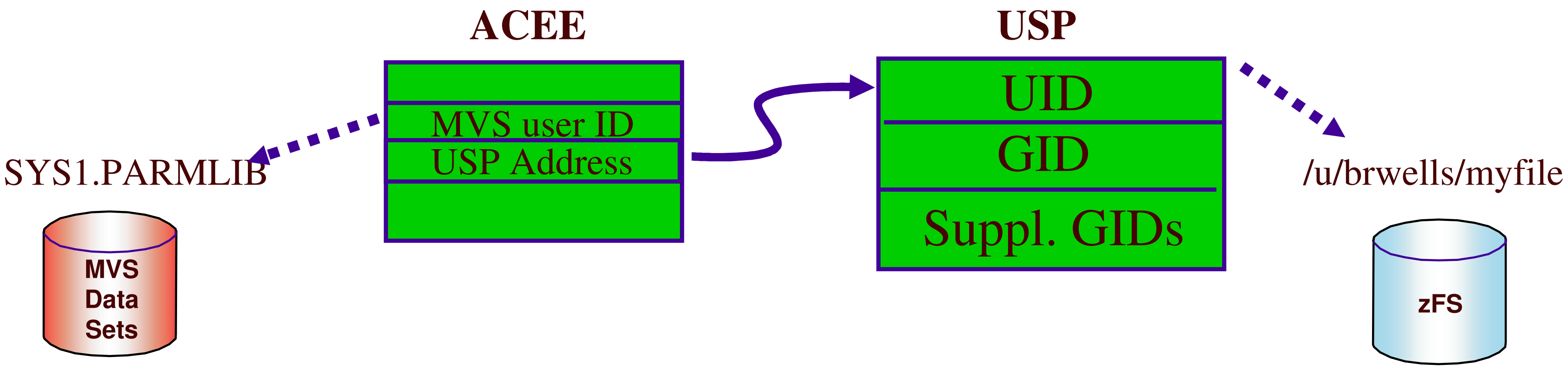
From OMVS segment of user's default group or from BPX.DEFAULT.USER

From OMVS segments of user's list of groups

- **USP created when first UNIX service is invoked**
- **use the id command to show user's UNIX identity**

```
# id mccartny
uid=64(MCCARTNY) gid=4(BEATLES) groups=61(QMEN),71(WINGS)
```

UNIX identity



ACEE

MVS user ID

USP Address

USP

UID

GID

Suppl. GIDs

SYS1.PARMLIB

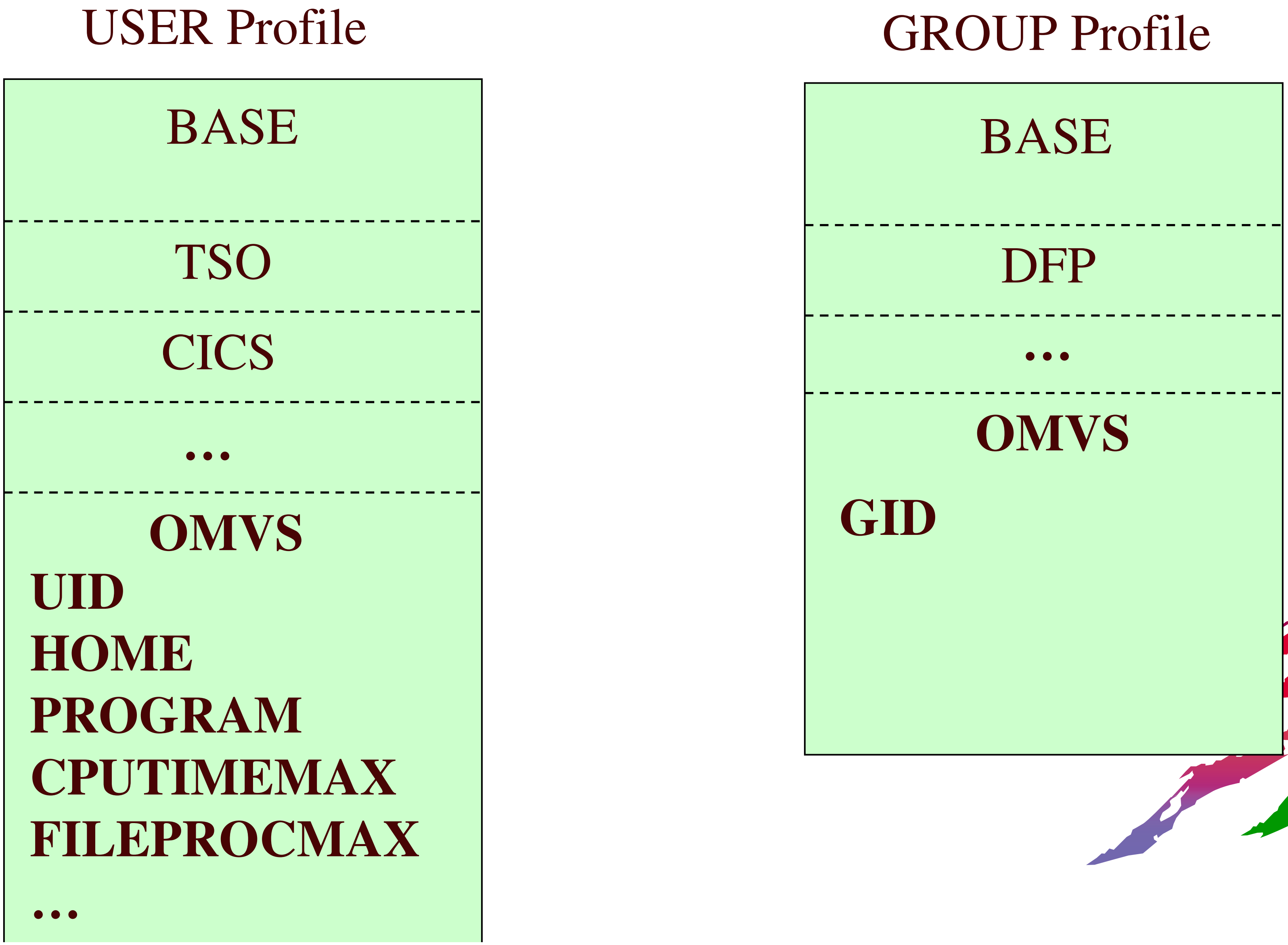
MVS Data Sets

/u/brwells/myfile

zFS

- When accessing MVS data sets and other RACF-protected resources:
  - 8-character MVS user ID (and group names) is checked against RACF profile
- When accessing UNIX files and directories:
  - Numeric UID and GIDs are checked against file owner and permissions

UNIX User and Group Registry: AKA RACF!



USER Profile

BASE

TSO

CICS

…

OMVS

UID

HOME

PROGRAM

CPUTIMEMAX

FILEPROCMAX

…

GROUP Profile

BASE

DFP

…

OMVS

GID

UNIX User and Group Registry: OMVS Segments

- User profiles need OMVS segments
  - UID - 0 to 2147483647 user identifier
  - HOME - current working directory
  - PROGRAM - initial program to execute
  - Other fields contain various resource limits
- Group profiles need OMVS segments
  - GID - 0 to 2147483647 group identifier
  - User's current connect group *and default group* need GID
- UIDs and GIDs should be unique

User Definition ... SUPERUSER!

- A superuser is defined as
  - UID 0, any GID
  - Trusted or privileged, any UID, any GID
- A superuser can:
  - Pass all z/OS UNIX security checks
  - Affect any UNIX process on the system
  - Change his identity
  - Use setrlimit to increase system limits

User Definition ... SUPERUSER!

- A superuser essentially has **SPECIAL** and **OPERATIONS**!!!!
- To the best of your ability, you should avoid assigning UID(0) to carbon-based life forms
  - use UNIXPRIV class or BPX.SUPERUSER (more later ...)
- UID(0) for started task users, and UNIX servers and daemons, is generally OK
  - use the NOPASSWORD attribute to prevent these from being logged onto

SUPERUSER Granularity: UNIXPRIV Class (Kryptonite)

- Used to assign subset of SUPERUSER authority to a user
- Enforces principle of least privilege
- Partial list of functions you can grant:
  - ability to read or write any HFS file
  - ability to change file ownership
  - ability to change file permissions/ACLs
  - ability to send signals to any process
  - ability to mount/unmount file systems

UNIXPRIV Resource Names

Example: File and Directory Access

Resource Name	Privilege	Access Req'd
SUPERUSER.FILESYS	**read any HFS file; read/search any HFS directory**	**READ**
SUPERUSER.FILESYS	**write any HFS file; also privileges of READ access**	**UPDATE**
SUPERUSER.FILESYS	**write any HFS directory; also privileges of UPDATE access**	**CONTROL**

See z/OS UNIX System Services Planning for complete list of UNIXPRIV resources

UNIXPRIV File related capabilities

Resource name	Ability it controls
SUPERUSER.FILESYS.	
CHOWN	change file ownership
CHANGEPERMS	change permission bits and ACLs
MOUNT	Manage the file system hierarchy
QUIESCE	quiesce a file system
PFSCTL	Use the pfsctl() service
VREGISTER	Use the vreg() service

UNIXPRIV other capabilities

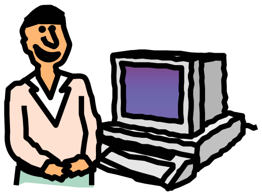
Resource name	Ability it controls
SUPERUSER.PROCESS.	
GETPSENT	Receive data (including ps output) for any process
KILL	Send signals to any process
PTRACE	Trace any process
SUPERUSER.	
SETPRIORITY	Increase your own priority
IPC.RMID	Release IPC resources

BPX.SUPERUSER

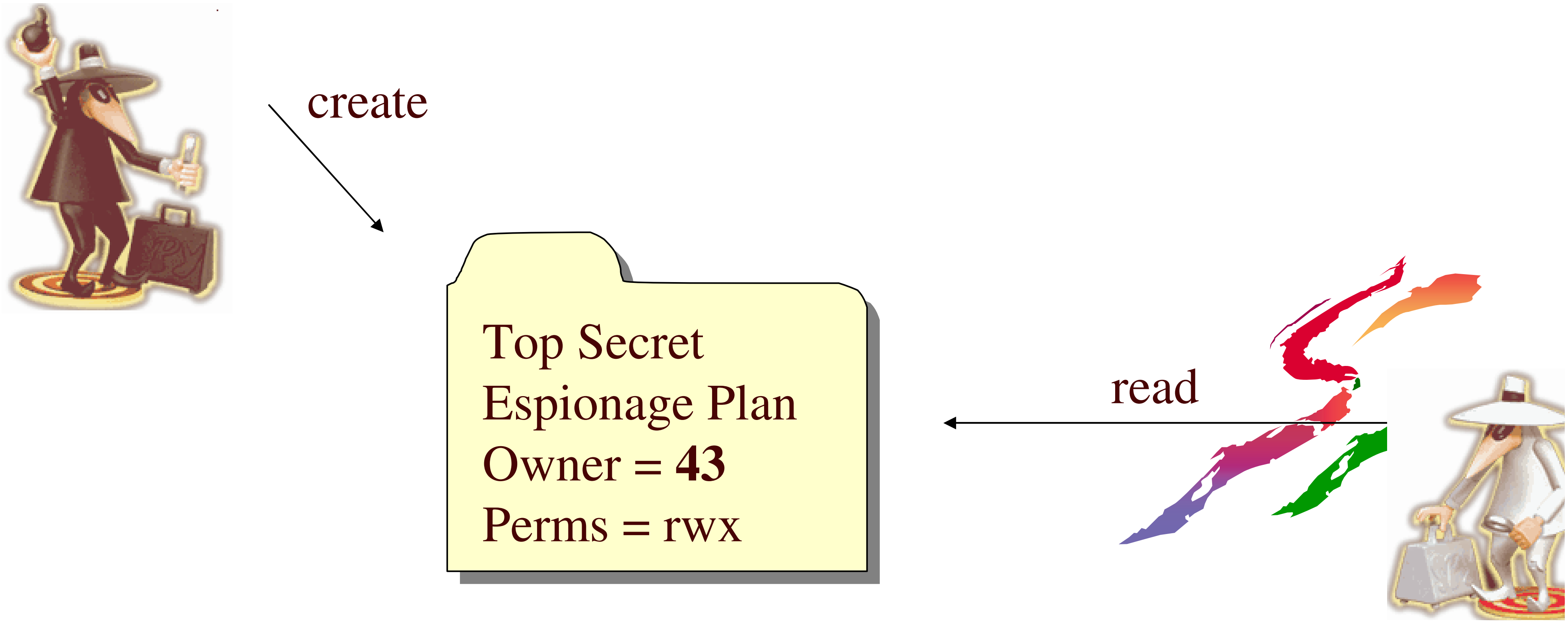
- FACILITY class resource which is yet another way to become superuser
- Controls who can issue su shell command to obtain effective UID 0
- Does not scope power at all, but at least you can audit when users switch into superuser mode
- User gets different shell prompt, and thus a visual clue that they are in superuser mode
- Recommend UNIXPRIV instead, but BPX.SUPERUSER is better than giving UID 0

Keep UIDs/GIDs unique – Why?

DDUSER BSPY OMVS(UID(**43**))



ALTUSER WSPY OMVS(UID(**43**))

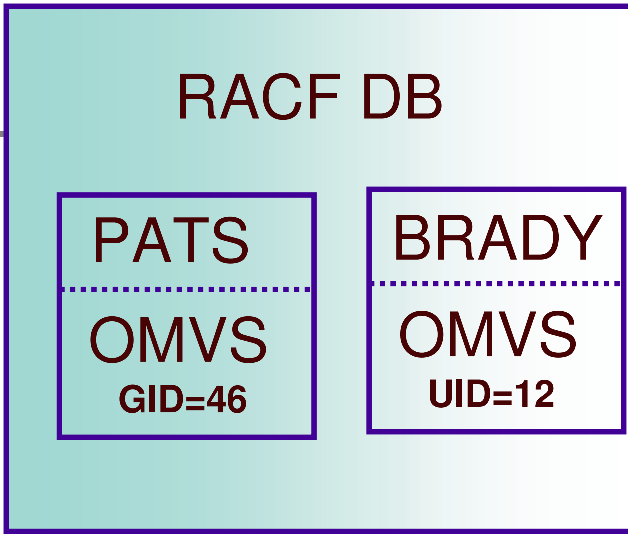


create

Top Secret
Espionage Plan
Owner = **43**
Perms = rwx

read

Prevention of shared IDs ... SHARED.IDS



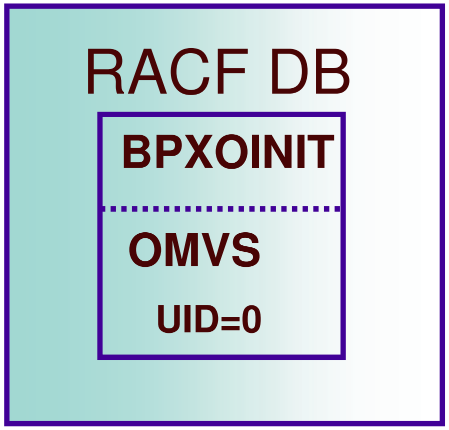
- RDEFINE UNIXPRIV SHARED.IDS UACC(NONE)
- SETROPTS RACLIST(UNIXPRIV) REFRESH
- ADDUSER MARCY OMVS(UID(12))

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

- ADDGROUP ADK OMVS(GID(46))

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

Prevention of shared IDs ...
Override using SHARED



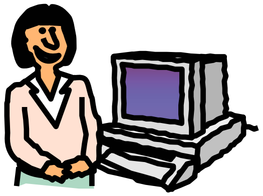
- PERMIT SHARED.IDS CLASS(UNIXPRIV) ID(UNIXGUY) ACCESS(READ)

- SETROPTS RACLIST(UNIXPRIV) REFRESH

UNIXGUY

AU OMVSKERN OMVS(UID(0) SHARED)

✓ OK!



MVSGAL

AU MYBUDDY OMVS(UID(0) SHARED)

IRR52175I You are not authorized to specify the SHARED keyword.

SEARCH enhancement to map UIDs and GIDs

- SEARCH CLASS(USER) UID(0)

```
OMVSKERN
BPXOINIT
SUPERGUY
```

- SEARCH CLASS(GROUP) GID(99)

```
RACFDEV
```

- SEARCH CLASS(USER) UID(1234567)

```
ICH31005I NO ENTRIES MEET SEARCH CRITERIA
```

Automatic UID/GID Assignment

- AUTOUID keyword in the OMVS segment of the ADDUSER and ALTUSER commands
- AUTOGID keyword in the OMVS segment of the ADDGROUP and ALTGROUP commands
- Derived values are guaranteed to be unique

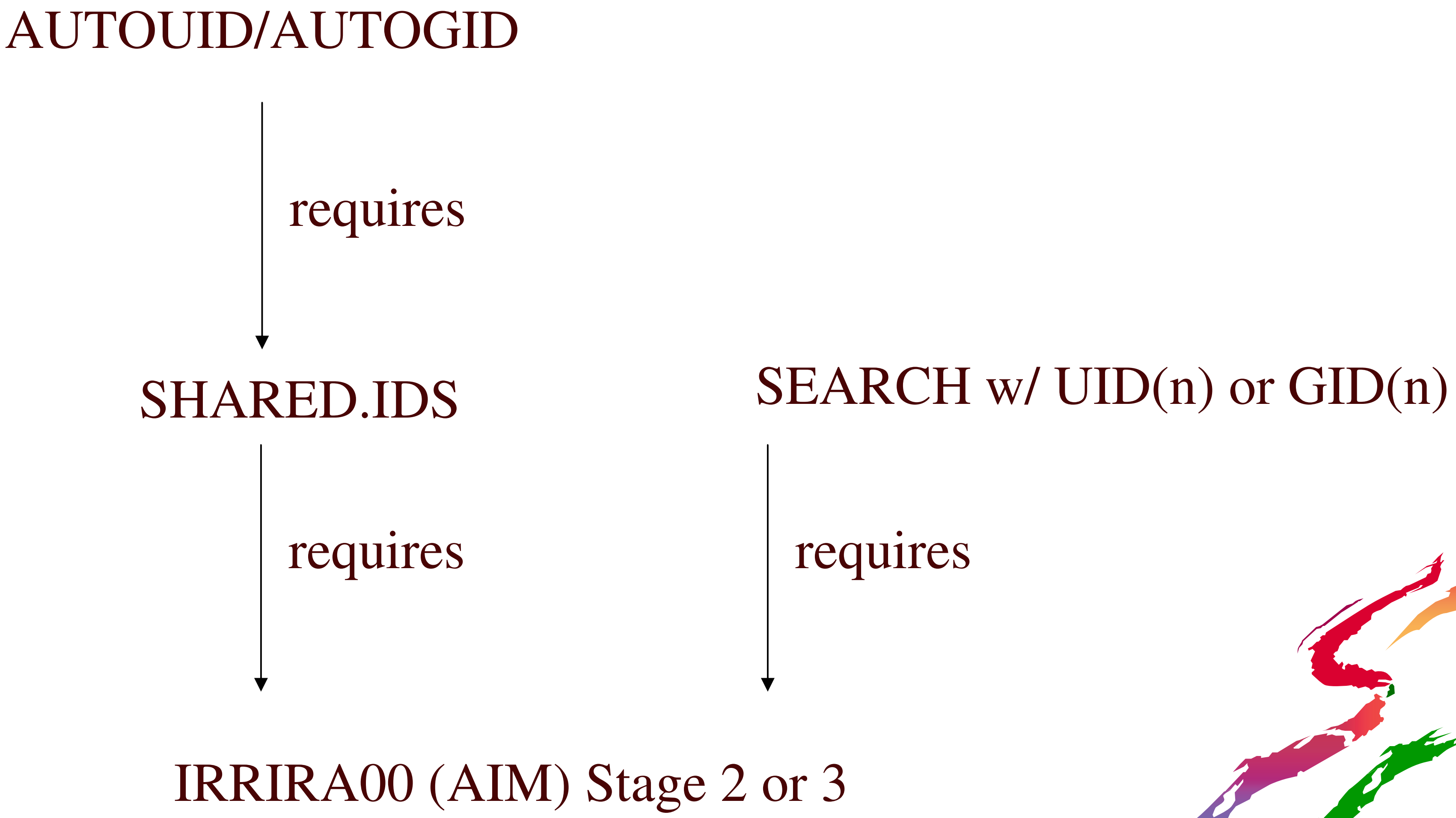
```
  ADDUSER MELVILLE OMVS(HOME(/u/melville) AUTOUID)

IRR52177I User MELVILLE was assigned an OMVS UID value of 4646

  ADDGROUP WHALES OMVS(AUTOGID)

IRR52177I Group WHALES was assigned an OMVS GID value of 105.
```

Automatic UID/GID Assignment ... BPX.NEXT.USER

- Uses APPLDATA of new **BPX.NEXT.USER** profile in the FACILITY class to derive candidate UID/GID values
- APPLDATA consists of 2 qualifiers separated by a forward slash ('/')
    - left qualifier specifies starting UID value, or range
    - right qualifier specifies starting GID value, or range
    - qualifiers can be null, or specified as 'NOAUTO', to prevent automatic assignment of UIDs or GIDs

RDEFINE FACILITY BPX.NEXT.USER APPLDATA(‘*10000-100000/500-50000*')

Functional Dependencies



AUTOUID/AUTOGID

requires

SHARED.IDS

SEARCH w/ UID(n) or GID(n)

requires

requires

IRRIRA00 (AIM) Stage 2 or 3

File Access Control with Permission Bits

File Owner

User (UID)	Group (GID)

Permission Bits

OWNER	GROUP	OTHER
rwx	r-x	-----

oedit /etc/profile

User

- effective UID
- effective GID
- Supplemental Groups

As per the UNIXPRIV profile RESTRICTED.FILESYS.ACCES

IF no access, check SUPERUSER.FILESYS in UNIXPRIV class

RACF AUDITOR can read and search any directory

Access Control Lists (ACLs)

- Each entry specifies a user (UID) or group (GID) and its allowable permissions
- Can contain a maximum of 1024 entries
- Support inheritance
- Activated with SETROPTS CLASSACT(FSSEC)

Top Secret
Superbowl Pool

User	BOB	rw-
User	BOSS	---
Group	DEPT1A	r--
Group	DEPT1B	r--
Group	DEPT1C	r--

File Access Control with Permission Bits and ACLs

Permission Bits

OWNER rwx	GROUP rwx	OTHER rwx
User1 rwx	Group1 rwx	
User2 rwx	Group2 rwx	
Usern rwx	Groupn rwx	

Access **C**ontrol **L**ist

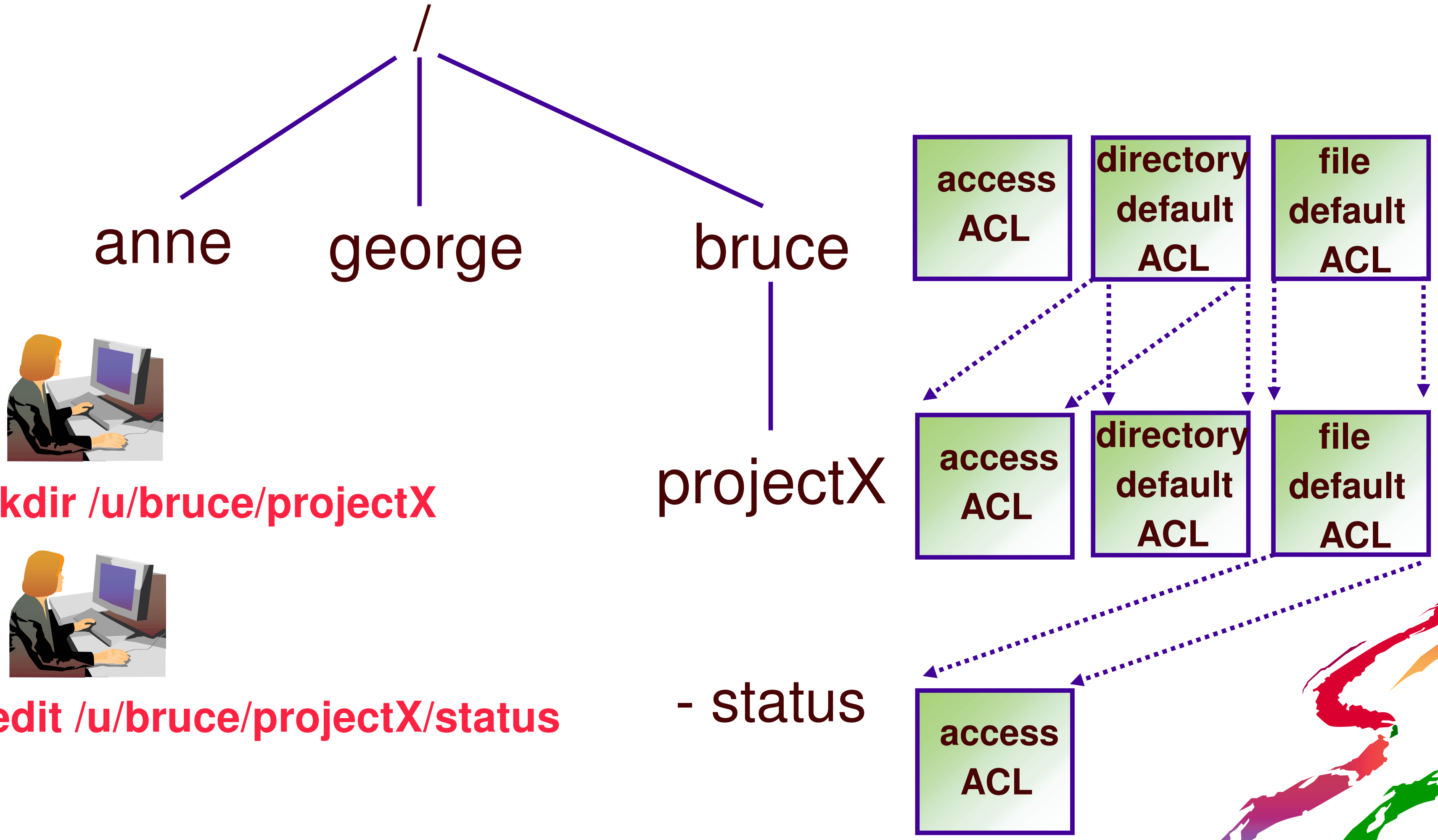
As per UNIXPRIV profile RESTRICTED.FILESYS.ACCESS

IF no access, check SUPERUSER.FILESYS or SUPERUSER.FILESYS.ACLOVERRID

IF FSSEC class active

See z/OS RACF Security Administrator's Guide Appendix F for detailed list of steps

ACL Inheritance



```
mkdir /u/bruce/projectX
```

```
oedit /u/bruce/projectX/status
```

Programs in the File System

- Can designate program as APF
  - extattr +a myprogram
  - requires READ to FACILITY profile BPX.FILEATTR.APF
  - find / -attr a
- Can designate a program as RACF program-controlled
  - extattr +p myprogram
  - requires READ to FACILITY profile BPX.FILEATTR.PROGCTL
  - find / -attr p

Programs in the File System ...

- Can indicate that a file system executable is to be obtained from traditional MVS search order (LPA and LINKLIB) by turning on the ‘sticky’ bit
  - chmod +t myprog
  - program name must adhere to MVS conventions (8 characters)
  - Traditional APF and Program-controlled libraries (data sets) apply

UNIX File Auditing

- Controlled by audit classes
    - SETROPTS LOGOPTIONS, SETROPTS AUDIT
        - DIRSRCH, DIRACC, FSOBJ, FSSEC
- And by file-level audit options
    - Similar to RALTER AUDIT() and GLOBALAUDIT()
    - Set with chaudit, not ALTDSD or RALT
- RACF UAUDIT attribute honored
- Always:

  SETROPTS LOGOPTIONS(ALWAYS(FSSEC)) !!!

Auditing UNIX Files: compared with data sets

<u>DATASET auditing</u>	<u>UNIX file auditing</u>
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)

Auditing UNIX Files: compared with data sets ...

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

ICH408I Violation Messages

```
ICH408I USER(REDTAIL ) GROUP(RAPTORS  ) NAME(PALE MALE)
 /u/bruce/work/projectX/secret/documents/Forecast
 CL(DIRSRCH ) FID(01C7D5D9D3F1F2001E04000004530000)
 INSUFFICIENT AUTHORITY TO OPEN
 ACCESS INTENT(--X)  ACCESS ALLOWED(OTHER      ---)
 EFFECTIVE UID(0000000295)  EFFECTIVE GID(0000000521)
```

```
ICH408I USER(TSOUSR1 ) GROUP(EMPLOYEE) NAME(BUBBA)
 CL(PROCESS )
 OMVS SEGMENT INCOMPLETELY DEFINED
```

```
ICH408I USER(TSOUSR1 ) GROUP(EMPLOYEE) NAME(BUBBA)
/bin     CL(FSSEC   )       FID(01C8D9E9F1F8F00001040000001D0000)
INSUFFICIENT AUTHORITY TO CHMOD
EFFECTIVE UID(0000000011)  EFFECTIVE GID(0000000500)
```

File System Security Reporting - HFS Unload!!!

- irrhfsu command available on **http://www-1.ibm.com/servers/eserver/zseries/zos/racf/goodies.html**
- Reports on HFS security data like IRRDBU00 reports on RACF profile data

0900	file name	i-node	uid	user id	gid	group name	set uid	set gid	sticky bit	owner read	owner write	owner execute	group read	etc ...

- Can be issued as a UNIX command, or batch
- Can run it against the whole file system, or against any number of sub-trees
- Output to screen, file, or data set

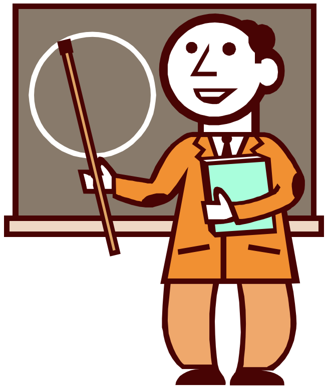
References

- UNIX System Services Planning
- UNIX System Services Command Reference
  - chmod, chown, chaudit, getfacl, setfacl, ls, find, umask
- UNIX tools and toys page (auditid)
- RACF Security Administrator's Guide
- RACF Auditor's Guide
- RACF downloads page (irrhfsu)

Recap - Integration Points

- User registry
- AUDITOR, UAUDIT, TRUSTED, PRIVILEGED, RESTRICTED attributes
- UNIX capabilities granted via RACF profiles
- ACL behavior, MultiLevel Security
- RACF auditing classes, LOGOPTIONS, file audit settings
- SMF, HFS Unload
- ISHELL

Appendix: Some command examples



Output of ls (list files) Command

file type and permissions

user and group owner

file name

```
# ls -E
total 192
 -rw-r--r--+    --s-    1 BPXROOT   2001          700  Mar   20  16:45 Odyssey
 --wx--S---     --s-    1 ACE       SYS1           30    Aug 23   2000 Program2
 -r-srwxrwx     --s-    1 BPXROOT   KNIGHTS      8240    Aug 23   2000 SetuidPgm
 drwxr-xr-x             2 BPXROOT   SYS1         8192    Mar 20  16:38 TestDirectory
 -rwxr----t     --s-    1 ACE       JESTERS      8240    Aug 11   2000 prog1
 -rwxr-x--x     ----    2 BPXROOT   SYS1         8240    Aug 11   2000 rac
 lrwxrwxrwx             1 BPXROOT   SYS1            3    Aug 20  16:43 racSymlink -> rac
 -rwxr-x--x     ----    2 BPXROOT   SYS1         8240    Mar 11   2000 raclink
 -rwxr-x---     aps-    1 BPXROOT   SYS1         8240    Aug 20  16:39 racp
 -rw-r--r--     --s-    1 1969      SYS1           99    Mar 20  16:46 woodstock
```

extended attributes

number of links

chmod Command - Change File Mode (permissions)

- change permissions of a file

  -
    ```
    chmod u=rwx,g=rwx,o=rx  a-file
    ```
- change permissions of a file with octal notation

  -
    ```
    chmod 775  a-file
    ```
- Set all read bits on for all files in a directory and its subdirectories using relative perms

  -
    ```
    chmod -R a+r MyDirectory
    ```

getfacl command

- getfacl Myfile
  - **Displays file name, user owner, and group owner**
  - **Displays base POSIX permissions in "acl format"**
  - **These can be suppressed**

```
#file:  MyFile
#owner: BPXROOT
#group: SYS1
user::rw-
group::r--
other::r--
```

setfacl command

- Create an access ACL with an entry for user bruce and group racf
  - **setfacl -m user:bruce:rwx,group:racf:r-x MyFile**
  - **getfacl MyFile**

```
#file:  MyFile
#owner: BPXROOT
#group: SYS1
user::rw-
group::r-x
other::r--
user:BRUCE:rwx
group:RACF:r-x
```

says modify acl entry, or add it if it does not exist

getfacl and setfacl with directory default acl

- Create a directory default ACL
  - setfacl -m default:user:bruce:rwx MyDir
  - or: setfacl -m d:u:bruce:rwx MyDir
  - getfacl -d MyDir

additional qualifier for directory default

```
#file:  MyDir
#owner: BPXROOT
#group: SYS1
default:user:BRUCE:rwx
```

chaudit Command: Setting File-level Auditing Options

- Audit successful write access to a file
  - chaudit w+s myfile
- Audit all access to a file
  - chaudit +sf myfile
- Set auditor audit bits to audit all attempts to execute a program
  - chaudit -a x+sf   myprog
- Audit all write and execute accesses to set-uid files
  - **chaudit x+sf,w+sf $(find / -perm -4000)**