z/OS UNIX Systems Services Security Best Practices

Vivian W Morabito

March 12, 2014

8:00AM – 9:00AM

Session Number 15119

morabito@us.ibm.com

Important to have a well designed security environment for UNIX System Services

Often this is not the case…

Sometimes due to:

- Misunderstanding
- Lack of awareness

Frequently due to:

- Little attention being given to thoroughly designing the security environment.

- UNIX System Services provides a full set of UNIX capabilities to z/OS
- z/OS UNIX System Services is fully POSIX compliant
- z/OS UNIX provides significant extra controls to restrict, limit, and divide authorities to protect against mis-use and prevent security and integrity breaches.

To assist in the overall z/OS UNIX Security Design

- z/OS UNIX Security Fundamentals REDP-4193
- This book may be obtained at:

  http://www.redbooks.ibm.com/abstracts/redp4193.html?Open

- A new Redbook is being written: Security on the IBM Mainframe: A Holistic Approach to reduce Risk and improve Security which will include information on z/OS UNIX Security.

z/OS UNIX Security: The view from 15000 feet

- The standard UNIX stuff:
  - File / Directory ownership
  - File permissions
  - UID 0
- z/OS UNIX Differentiators include:
  - UNIX related FACILITY & SURROGAT class profiles (BPX.*)
  - UNIXPRIV Classes
  - Access Control Lists
  - Security Auditing
  - Filesystem Access Prevention / Monitoring
  - Automatic Assignment of UIDs / GIDs
  - MLS / Security labels

Traditional & z/OS UNIX: file permissions

- z/OS UNIX, as a POSIX compliant system, provides all of the file permission support available in traditional UNIX implementations.
- In a nutshell, every file (directory) in the filesystem has permission bits giving it read, write, & execute (search) "rwx" permissions for the owning user, owning group, and others
- ls –l output shows:

```
-rw-r—r--    1 john1   staff    355 Jan 22 13:23 myfile
```

  - Owner (john1) can read and write myfile
  - Anyone in the group staff can read myfile, as can anyone else (others)

- Often expressed in their octal format (644 for above example)

z/OS UNIX: users and groups

- 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

- **RACF commands to define users and groups:**
  - ADDUSER (AU) and ADDGROUP (AG)
  - To modify - ALTUSER (ALU) and ALTGROUP (ALG)

- **To assign OMVS Segment to existing user:**
  - alu joe omvs(uid(4670) home('/u/joe') program('/bin/sh'))

z/OS UNIX

RECOMMENDATION:

Overall z/OS UNIX Security design should pay careful attention to the implementation of user and group IDs, and group membership, and the permissions granted to these users and groups.

Be aware of the umask setting when creating new files!!

- Traditional UNIX systems let superusers (UID 0) do any and all tasks requiring authorization
- Allowing all UID 0 grants too much unlimited authority for a z/OS system

z/OS UID 0 Superuser Authority

- Passes all z/OS UNIX Security checks
- Perform administrative activities
- Install products
- Can access and modify any files and directories
- Not Limited to only z/OS UNIX component
- Manages all z/OS UNIX processes
- Can run unlimited number of processes concurrently
- Propagates superuser privileges to forked child process
- Can change identity

z/OS: Defining full superuser privileges

- Assigning UID=0 in the RACF OMVS segment
- READ access to BPX.SUPERUSER profile in RACF FACILITY class

BPX.SUPERUSER

- profile in the FACILITY class
- Allows non-UID 0 to su to become UID 0
- **RECOMMENDATION:**
  - **Limit using BPX.SUPERUSER to those requiring it for installation purposes**
  - **Keep careful track of who this has been granted to – use a group to grant access rather than to individual users**
  - **Log successful accesses**

z/OS: Limiting users with superuser authority

- z/OS provides multiple facilities which:
  - Grant superuser privileges with a high degree of granularity to users who do not have superuser authority.
  - Allows the creation of users which have limited authority.
  - Minimize the number of superusers at an installation
  - **Reduce risk!!**

UNIX related FACILITY & SURROGATE class profiles

- All start with prefix BPX
- Define with UACC(NONE) and then PERMIT groups the minimum access needed.
- Generally, authorized users must have a least READ access to be able to use the specific UNIX function protected
- AVOID defining the generic profile BPX.**

BPX.DAEMON

- BPX.DAEMON controls which users are allowed to take on the identity of other users
  - Any superuser permitted to BPX.DAEMON profile has the daemon authority to change MVS identities via z/OS UNIX services without knowing the target uid's password
- Requires a Must Stay Clean Environment - all MVS and z/OS UNIX programs that are loaded or executed must be program controlled
- If BPX.DAEMON is not defined then all superusers have daemon authority!!

BPX.SERVER

- BPX.SERVER regulates the security environment of servers running on z/OS.
- BPX.SERVER restricts the use pthread_security_np() which creates or deletes the security environment for the caller's thread.
- Restricts the use of the BPX1ACK service, which determines access authority to z/OS resources
- Requires a Must Stay Clean Environment - all MVS and z/OS UNIX programs that are loaded or executed must be program controlled

RECOMMENDATION

- Always define the profiles BPX.DAEMON and BPX.SERVER in the FACILITY CLASS.
- Restrict access to these profiles to those with an absolute need.
- This is not expected to include human users, just userids for servers and daemons.

UNIXPRIV

- Profiles in UNIXPRIV grant RACF authorization for various z/OS UNIX privileges
- Always define with UACC(NONE) and then PERMIT users the minimum access needed.
- Allows superuser privileges to be granted with a high level of granularity
- Minimizes the number of users with superuser authority
- **REDUCES SECURITY RISK!!**

UNIXPRIV

CHOWN.UNRESTRICTED

Allows users to use the chown command to transfer ownership of their own files

With APAR OA41364 applied:

READ: change the user owner to a **non-UID 0** *-or-* change the group owner to the GID of a group to which they are not connected.

UPDATE: change the user owner to a **UID value of 0 (superuser)**

RECOMMENDATION:

- Not recommended
- If needed, only give READ or UPDATE access to system programmers/administrators.

UNIXPRIV

SUPERUSER.FILESYS.MOUNT

Allows users to mount a filesystem.

READ: mount a filesystem with the **nosetuid** option.

chmount to change mount attributes of a **nosetuid** mounted filesystem

UPDATE: mount a filesystem with the **setuid** option.

chmount to change mount attributes of a filesystem mounted with the setuid option.

RECOMMENDATION:

- If needed, grant access at lowest level of authority and limit the number of users with authority.

UNIXPRIV

SUPERUSER.FILESYS.USERMOUNT

Allows nonprivileged users to mount and unmount file systems with the nosetuid option.

- READ is the only access level
- User must also have:
  - Read-write-execute (rwx) access to the directory that the file system will be mounted on, which must be empty.
    - If sticky bit is on, on, the user must be the owner of that dir
  - Read-write-execute (rwx) access to the file system root.
- Provides convenience – moves management of user data away from the system administrator into the hands of the users that own the data.

UNIXPRIV

SHARED.IDS

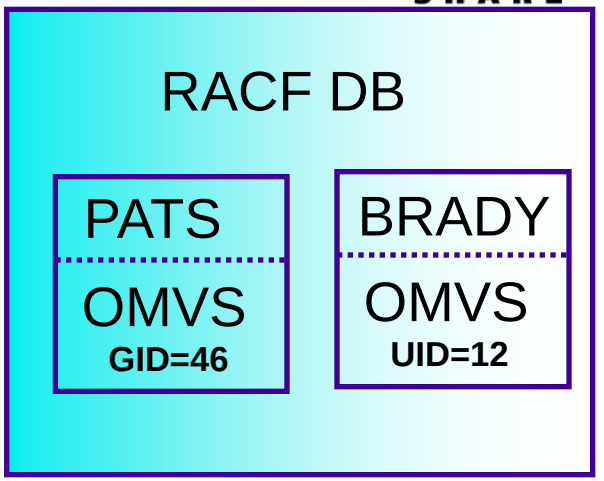
Enforces unique UNIX identifiers (UID & GID)

- 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

Prevention of shared IDs ... SHARED.IDS



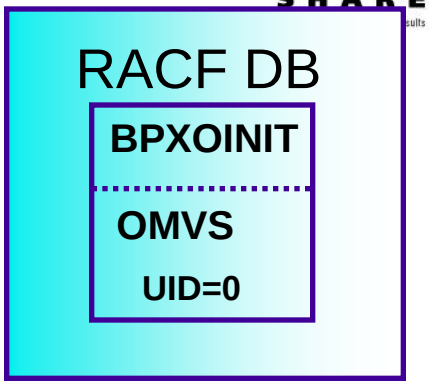
- RDEFINE UNIXPRIV SHARED.IDS UACC(NONE)
- SETROPTS RACLIST(UNIXPRIV) REFRESH
- ADDUSER MARCY OMVS(UID(12))

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



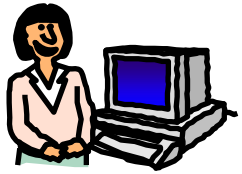
- PERMIT SHARED.IDS CLASS(UNIXPRIV) ID(UNIXGUY) ACCESS(READ)

- 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.

z/OS: Automatic Assignment of UIDs & GIDs

- A user must have a specific UID and GID to access the z/OS UNIX Environment
- RACF can automatically generate a unique ID value in the OMVS segment upon your request, using the BPX.NEXT.USER profile and the AUTOUID & AUTOGID operands of the add/altuser and add/altgroup commands
- Pre-requisites:
  - Define SHARED.IDS profile in the UNIXPRIV class, which will enforce unique UIDs & GIDs
  - Define BPX.NEXT.USER profile also in UNIXPRIV class with the APPLDATA field which specifies a starting value, or range of values, from which RACF will derive unused UID and GID values

BPX.UNIQUE.USER

- OMVS segment generation is enabled when the BPX.UNIQUE.USER profile is defined in the FACILITY class.
- An optional model user can be specified by the APPLDATA field for BPX.UNIQUE.USER.
  - The OMVS segment from the model user is used to initialize new OMVS segments for the user profile, this includes all attributes (HOME, PROGRAM and user limits) except the UID.
  - no passwords or pass-phrases on the model user!
- BPX.UNIQUE.USER uses BPX.NEXT.USER to derive un-used UIDs and GIDs
- with APAR OA42554 on R12/13, the home directory string can specified with &racuid to have RACF substitute the user's id when defining the home directory string in the OMVS segment.

FSACCESS Class Profile

- Acts as a “gatekeeper” to zFS filesystems
- After user granted access to the filesystem, subsequent decisions are based on permissions & ACLs
- Access check is based solely on the user’s z/OS userid – meaning that the superuser authority will not be used or influence the outcome of this access control check.
- Only zFS filesystems supported
- Root filesystem & zFS-es mounted NOSECURITY excluded

- . **RECOMMENDATION:**

FSACCESS may be of use for RACF administrator to prevent access to filesystem(s)

RECOMMENDATION

Define resources with UACC(NONE)

Keep the access lists of these resources to an absolute minimum.

Document all cases where these privileges have been granted, with the reasons.

filesystem security recommendations

- Public filesystems /tmp and /var should be mounted with NOSETUID in PARMLIB member BPXPRMxx
- Automount policy should explicitly set “setuid no”
- Automount policy files (e.g. /etc/u.map) should not specify <uc_name> or <asis_name> as first qualifier of filesytem datasets.
  - Note: filesystem create / delete may have to be done by an administrator.

THINGS TO BE AWARE OF

Some software packages for z/OS USS have been ported from other UNIX systems, where using UID 0 for authorization is the standard way of operating.

You only have a few options here:

- Run the package with UID 0 and accept the risk
- Find out what services it uses that require superuser authority, and if a profile exists for that service use it.
- Put pressure on the vendor to change the way he operates
- Reject the package in favor of one that meets your security requirements

THINGS TO BE AWARE OF

- Remove old versions of software products.
  - Many products install their latest release in a new directory and the old directory will still be there with the old executables that do not have the latest security fixes.

z/OS: Access Control Lists

- Access Control Lists (ACLs) are a common feature in most UNIX implementations, including z/OS UNIX
- Used in conjunction with permission bits
- Provide more granular security than POSIX permission bits
- ACLs exist in the filesystem associated with each file.

ACLs on z/OS UNIX

- Supported on zFS, HFS, and TFS filesystem
- ACLs are created and checked by RACF and not the kernel or filesystem.
- Must either be UID 0 or have READ access to SUPERUSER.FILESYS.CHANGEPERMS in UNIXPRIV to define ACLs, or be the file owner
- Must activate the FSSEC class before ACLs can be used in making access decisions
  - SETROPTS CLASSACT(FSSEC)

ACLs on z/OS….

- ACLs can both restrict & grant access to files & directories

- **3 types of ACLs:**
  - **Access**: used to provide protection for file system object
  - **File Default:** -- inherited by new files
  - **Directory Default:** – inherited by new sub-directories

- **2 types of ACL Entries**:
  - **Base ACL Entries**: same as POSIX permission bits!
    - Can be set / changed either via chmod or setfacl
  - **Extended ACL Entries**: are entries for individual users or groups.

z/OS Commands for ACLs

Shell Commands:

setfacl	*set, remove, modify ACL entries*
getfacl	*display owner, group, ACL entries*
ls	*+ indicates if extended ACL entries exist*
find	*-acl option to find objects with ACLs*
df	*determine if filesys supports ACLS*
getconf	*determine if filesys supports ACLS & max # of ACLs*
pax, tar	*store / restore ACL info in an archive*
cp –p, mv –Z	*preserve ACLs for files & directories*
filetest, test	*tests files & dirs for extended and / or default ACLs*

C / C++ Calls

Various C calls are available to add, delete, update, get, sort ACLs

REXX, ISHELL interfaces are available

ACL tool available

Tool available which will:

- set ACLs for a directory and everything underneath
- show ACLs for a file or directory

ftp://public.dhe.ibm.com/s390/zos/tools/wjsacl/wjsacl.txt

z/OS UNIX Security Auditing

- File level auditing
  - File-level Audit Attributes set via **chaudit** command
    - fileowner / superuser: non-auditor-requested audit attributes
    - Only users with auditor authority can change auditor-requested audit attributes
  - File-level Audit Attributes displayed via **ls –W**
    - displays both owner & auditor options
- Resource class level auditing
  - Controlled by 7 UNIX Audit Classes

z/OS UNIX 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

Auditing UNIX Files: compared with data sets

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)

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

z/OS: Security Auditing Recommendations

- Use file-level audit settings to log successful accesses to sensitive files, such as configuration files
- Do use SETROPTS LOGOPTIONS(ALWAYS(FSSEC))
  - Will record all attempts to change security info for files / dirs
- Do use SETROPTS LOGOPTIONS(FAILURES(PROCESS PROACT)
  - Logs failed attempts:
    - to change process UID/GID
    - inspect / affect processes other than their own
    - (these are not logged by default)
- Do use SETROPTS LOGOPTIONS(FAILURES(IPCOBJ))
  - Logs failed attempts to access in-memory IPC objects (these are not logged by default)

z/OS: Security Auditing Recommendations…

- Don’t use BPX.SAFFASTPATH if you ever need successful file access audit records created

- Don’t turn on successful directory search (DIRSRCH):
  - *too much information produced!!*
  - *Could be done on an exception basis using file level settings*

- Download available on the RACF web site which dumps security-relevant file system information into a flat file and allows the same query ability as the RACF IRRDBU00 (database unload) and IRRADU00 (SMF Unload) utilities

http://www-03.ibm.com/systems/z/os/zos/features/racf/downloads/irrhfsu.html

z/OS: Multilevel security (MLS)

- Multilevel security is a security policy that allows the classification of data and users based on a system of hierarchical security levels combined with a system of non-hierarchical security categories.

- In a multilevel security z/OS UNIX environment, authorization checks are performed for security labels in addition to POSIX permissions, to provide additional security.

See Planning for Multilevel Security and the Common Criteria GA22-7509-13

References

- z/OS V2R1.0 UNIX System Services Planning GA32-0884-00
- z/OS V2R1.0 UNIX System Services User’s Guide SA23-2279-00
- z/OS Security Server RACF Security Administrator's Guide SA23-2289-00
- z/OS Security Server RACF Auditor's Guide SA23-2290-00
- Planning for Multilevel Security and the Common Criteria GA22-7509-13

Connect with IBM System z on social media!

Subscribe to the new *IBM Mainframe Weekly* digital newsletter to get the latest updates on the IBM Mainframe!

LinkedIn

System z Advocates **
IBM Mainframe- Unofficial Group
IBM System z Events
Mainframe Experts Network
SHARE

facebook

IBM System z **
IBM Master the Mainframe Contest
IBM Destination z
SHARE Inc.

twitter

IBM System z **
IBM System z Events
Destination z
SHARE

System z SMEs and Executives:
Deon Newman - @deonnewm
Steven Dickens - @StevenDickens3
Michael Desens - @MikeDesens
Patrick Toole - @Pat_Toole_II
Kelly Ryan - @KellykmRyan
Richard Gamblin - @RichGx

Blogs

IBM Mainframe Insights **
Millennial Mainframer
#MainframeDebate blog
SHARE blog
IBM Destination z

YouTube

IBM System z **
Destination z

tumblr.

IBM Mainframe50

Include the hashtag #mainframe in your social media activity and #mainframe50 in 50th anniversary activity