Introducing:

z/OS Unix Security needs fixing, where do I start ?

Paul Arnerich

May 2015

(minor update May 2016)

Contact:

Paul.Arnerich@tsdd.co.uk

Abstract and Legal Stuff

Abstract

- For many z/OS customers, z/OS Unix Security has been rising up the priority list due in part to audit findings showing need for improvement and in some cases, years of low priority for this complex area of security management leaving a long list of challenges that need resolving. For some customers, the complexity has meant that there is a lack of clarity as to where to begin the remedial work, this session will look at the common areas of concern and propose strategies for both tactical (quick fix) and strategic (long term) remedial activities.

Trademarks

- POSIX® is a registered trademark of the IEEE.
- IBM® is a registered trademark of International Business Machines Corporation.
- The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:
  - DB2®, IBM®, MVS™, RACF®, RMF™, WebSphere ®, System z ®, z/OS ®
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Other company, product and service names may be trademarks or service marks of others.

Agenda

- **Background to the session**
  - **Assumptions**
  - **Why me**
  - **Why z/OS Unix on z/OS**
- **State of Play for most customers**
  - **Status Check**
  - **Governance issues**
- **z/OS Unix and the Security Policy**
  - **You will need one of these**
    - **It's a stick not a carrot**
- **Remedial actions hit list**
  - **Pre-requisites, plans and resources**
  - **UID Scheme, UID(0), BPX.DEFAULT.USER**
  - **Files/directories – normal ownership and unowned**
  - **BPX.SUPERUSER, BPX.DAEMON and UNIXPRIV**
  - **ACLs**

Assumptions

- This is a technical topic
  - I assume you have a good grounding in z/OS Unix
- This is focused on z/OS Unix within the context of Security
  - I assume you have a good grounding in z/OS Security
- There are External Security Managers other than RACF
  - However, for the sake of simplicity, this presentation uses RACF terminology
  - Top Secret and ACF2 people, please translate
- I'm English and speak the "Queen's English"
  - Speakers of other forms of English will have to do the translation in terms of smelling and grammar
  - There may be some observations that may not be fully understood in certain geographies, apologies
- Some phrases
  - ESM – External Security Manager (RACF, Top Secret or ACF2)
  - AOC – Area of Concern
  - SMP – Somebody else's problem
  - NMP – Not my problem
  - z/OS Unix is the 'approved' short form of z/OS Unix System Services
    - Not USS !

Why me ?

- Because I was asked to by the UK Guide Share Europe team a while back
  - The community has this as a hot topic

- Because I'm old ➔
  - (Not a bad likeness in a certain light)

- Because if you cut me in half, you will find a z
  - I've been a 'doing' z for a long time. Seen o, m and v come and go
  - I've been 'doing' z/OS Unix for a long time
    - Rolled out Open Edition back in MVS/ESA v4.3 (1993)
    - Ran like a dog with three legs from memory

- Because I have some experience of z/OS Unix
  - Didn't take "one step back" at the right time
  - For my sins I have been heavily involved in most of the WebSphere Application Server implementations in EMEA
    - WebSphere Application Server is a big z/OS Unix exploiter
  - Contributed to the z/OS Unix related Security Policies at EMEA customers
    - Because of all the above

Why z/OS Unix

- **Original reasons – mid 90's**
  - FIPS 151-2 in 1991, stated all suppliers to US government must be POSIX compliant
  - Market Share
  - The need to be 'Open'
  - Long term, the vision of having the capability to port UNIX apps to z/OS
    - win back some server business
  - Rebranding 'MVS' as a Server
- **Harsh reality**
  - Fewer mainframe system and subsystem programmers around
  - Diminishing Assembler skills in IBM and Vendor labs
  - Increased 'C-like' skills in IBM and Vendor labs
    - The available talent pool know C and think in C system calls
  - Large percentages of sub-systems and operating system coded as 'C-like'
    - using z/OS Unix syscalls for access to resources for access to memory and data

z/OS Unix on z/OS today

- It's a pre-requisite for many z/OS products and services:
  - HTTP Serving
  - Domino
  - WebSphere Application Server, WebSphere everything else
  - TCP/IP
  - Java
  - SMP/E
  - DB2
  - XML Parsing
  - CICS/CTS, CTG
  - IMS
  - LDAP
  - Healthchecker
  - zOSMF
  - ....and many more
- z/OS and z/OS Unix are meshed together
  - Can't run z/OS without it
  - Can't start TCP/IP without it
  - Not much else will run without it

State of Play

Categories of z/OS Unix customers

- **Two basic types:**
  - **z/OS Unix Savvy and Not z/OS Unix savvy**

Proton Savvy

- **z/OS Unix Savvy customers**
  - **Run middleware layer that is dependent on z/OS Unix**
    - **WebSphere Application Server**
    - **SAP, Java exploiters, Domino**
    - **Web Serving, NFS**
  - **Attributes**
    - **Learnt the hard way that z/OS Unix security (and other bits) matter**
    - **Typically exploiter teams had the most knowledge**
      - **Implementation was generally suited to that team not necessarily the general population**
    - **May still have security issues to remedy**
      - **Residual effect of only treating the 'stuff that matters'**
      - **Often still have UID(0) issues**

- **Not z/OS Unix savvy**
  - **Often ignored z/OS Unix as not relevant**
  - **Have the steepest curve to remedy**
    - **but probably the fewest users/resources affected**
  - **Frequently do not have procedures and policies in place for z/OS Unix**

State of play

- **Sites failing audit (internal and external) due to z/OS Unix exposures**
  - **Auditors know more about Unix than they often do about z/OS**
    - **A generalisation, but often the case in my experience**
  - **Human nature dictates that they will put the effort into their comfort zone**

- **Increased publicity of break-ins on z/OS**
  - **Increasingly publicly connected via TCP/IP**
    - **Means increased opportunity**
  - **Typically exploiting z/OS Unix weaknesses**
    - **cf. Logica break-in widely documented on t'internet**
      - **Obs: Swedish investigators managed to obtain 30,000 userid/password combos in 2 days from the Logica RACF database using a single PC and JTR**

- **Increased reliance on z/OS Unix with each release of z/OS**
  - **New software stacks such as Healthchecker, zOSMF, etc..**
  - **Old software stacks expanding with more 'C' based code**
    - **CICS, DB2, IMS, DFSMS**
  - **Any software install/implementation has z/OS Unix element**
    - **UIDs needed**
    - **FACILITY and UNIXPRIV profiles**
    - **Files, especially config type files**

z/OS Unix Infrastructure

- Overall, "we" have not handled z/OS Unix security well
  - in some cases, not handled at all
- How good a job did we do ? Generally a lousy job
  - not sure "our mothers would be proud of us"
- How did we get here ?
  - Too busy, not enough focus, didn't understand the severity
  - We were busy with other 'stuff', had other priorities
  - Ignorance
    - "what is z/OS Unix anyway and do we actually use it ?"
  - Lack of ownership of z/OS Unix amongst the technical teams
    - Often fell/falls between the cracks
      - Typically somewhere between z/OS team and Middleware teams
    - Which means it is a classic SEP
      - Somebody Else's Problem
  - Often didn't see it coming
    - And when we did, the head went into the sand
  - blah blah
- Whatever, it is now time to "Man up"
- Embrace the pain
  - Its not perfect, but now we get to start to improve it

Key z/OS Unix Areas

- z/OS Unix has 4 major areas of concern
  - AOC1 – Security
  - AOC2 - Performance and tuning
    - NMP (Not My Problem) ?
    - Yes and no
      - Most secure system is the one that is powered off whilst the best performing system is a the one with no security
      - We need to strike a business focused balance between these two extremes
    - Key security decisions for z/OS resources were made decades ago
      - Cost of those decisions has been in the hardware software calculations
      - New security hardening decisions need to be budgeted for in terms of performance
  - AOC3 – File Systems and Data Storage
    - NMP ? Yes and no
      - Decisions about data location (zFS versus Dataset) have security implications
  - AOC4 – Automation/Scheduling
    - NMP ? Yes and no
      - Security related review/audit/monitor will need batch processes
- This is a Security session so focus will be on AOC1
- But AOC 2/3/4 had better be in the plan
  - The zEngineering teams will need to be involved:
    - to action/implement some of the remedial actions
    - Provide infrastructure to support security efforts (automation/tools etc..)

Governance Issues

- **In order to fix something, you have to know what is wrong**
  - **Most effective way of establishing this is to compare your System z Security Policy (zSP) to Actual**
  - **Requires you to have a zSP**
    - **One that specifically covers z/OS Unix as well as all the rest**
    - **Obs: should also cover TCPIP in depth, TCPIP and z/OS Unix entwined together in many ways**
    - **Obs: TCPIP often quite lightly handled in customers zSP, a bit of an SEP**
  - **Requires you to have software install/implementation guidance that covers z/OS Unix**
    - **Software installation template must have a z/OS Unix section**
      - **UID/GID requirements**
      - **File system level security requirements**
      - **File and directory level security requirements**
      - **Plus all the other non security related z/OS Unix requirements**
    - **Change Control template must have a z/OS Unix section**
    - **Security Change template must have a z/OS Unix section**
      - **Including authorisation requirements**
    - **New User request template must cater for z/OS Unix**
      - **OMVS Segment needed ?**
      - **GID requirement ?**
      - **BPX.???????? FACILITY class and UNIXPRIV class profiles ?**
  - **All this needs an integrated approach including the whole zEngineering community**
    - **You are going to have to actually speak to ….. 'them'**

USS and the Security Policy

z Security Policy

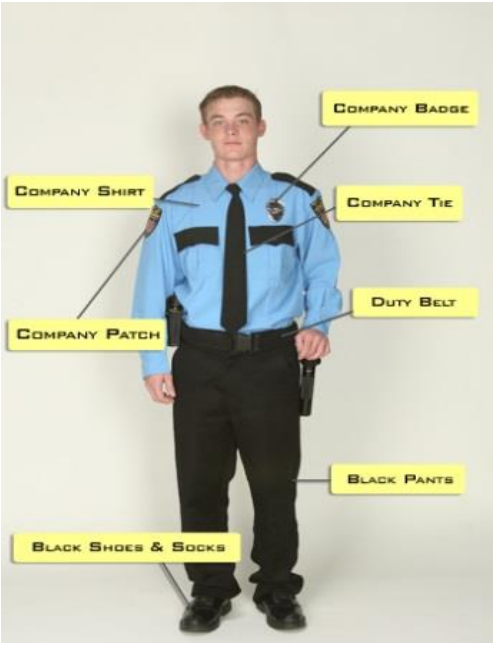
- **A refresh on the System z Security Policy**
  - **Without one you are in trouble**
- **Common Approach**
  - **Write a Policy based on requirements not based on actual**
  - **No 'off the shelf' solution, must be written to your requirements**
  - **Define all Resource Owners (RO)**
    - **Tough to do, can cheat a bit by declaring selected zEngineering teams as Proxy Authoriser (PA)**
    - **Authorisation requests can now be directed to the RO/PA**
      - **Responsibility (and accountability) lies with the RO/PA**
      - **Result is that Security team are not carrying the can, merely enacting the approved requirements of the logical owner of the resource**
  - **zSP must be signed off**
    - **Business units must collectively sign this, they will, this is now an SEP from their perspective, i.e. Your Problem**
  - **Compare zSP to Actual**
    - **Produces a work queue to check off**
    - **Expectation must be set that you will not match, you have only begun**
      - **Obs: The worse you compare at the beginning, the better your results looks**
      - **This is good, you can be measured as very successful very easily if things are very bad to start with**
  - **Categorise Policy non-compliant elements by:**
    - **Risk – some may be "within appetite"**
    - **Priority**
    - **Relative effort to remedy**

z Security Policy ….(cont.)

- **Next step is often to create a partner doc**
  - **zSID – z Security Implementation Document**
    - **Cut and paste the headings and policy line items from the zSP**
  - **As zSP items are ticked off, add the implementation actions and detail here**
    - **Keeps zSP clean and slim (ish)**
    - **Policy items marked as "implemented" or "not yet implemented"**
    - **Acts as a state of play – show it to the auditor**
- **Cons**
  - **Auditor knows exactly where you are not compliant**
- **Pros**
  - **Auditor knows exactly where you are not compliant**
- **Possible states:**
  - **Policy item = NO / Status = UNKNOWN**
    - **Sad face from Audit**
  - **Policy item = YES / Status = UNKNOWN**
    - **Sad face from Audit**
  - **Policy item = YES / Status = NOT YET IMPLEMENTED**
    - **Smiley face from Audit**
  - **Policy item = YES / Status = IMPLEMENTED**
    - **"bovvered" face from Audit - aka 'Whatever' or even 'meh'**

zSP Essentials

- **Minimum requirement for z/OS Unix related zSP items**
  - **Assignment of OMVS segment for users and groups**
  - **UID/GID assignment scheme/table**
  - **Human user assignment of UID(0)**
  - **Acquiring "appropriate privileges" for non-human users (e.g. Started Tasks)**
    - **Use of TRUSTED or PRIVILEGED in relation to this topic**
    - **Assignment of real UID(0), access to effective UID(0) – via BPX.SUPERUSER**
  - **Files/directories and File Systems**
    - **Ownership rules**
      - **How unowned files/directories are handled**
      - **Use of orphan USER and GROUP**
    - **Location of config files**
      - **File System or Dataset**
    - **/etc security**
      - **Especially the run commands (rc)**
    - **Use of ACLs**
      - **More likely – no use of ACLs**
    - **setuid and setgid bits**
  - **BPX.SAFFASTPATH**
  - **Activation (or not) of FSSEC and FSACCESS**
  - **BPXPRM security related values**
  - **FACILITY BPX.* profiles and UNIXPRIV class**
    - **e.g. BPX.DEFAULT.USER and NEXT**
  - **Use of su and sudo**



Next Steps

- **So, you have a policy, some governance and a list of non-compliant elements, what next ?**
  - **Triage, high priority first**
    - Within your budget, um.., budget ?
    - Yes, you will need a budget for the internal resource
  - **Consider the 80/20 rule**
    - Do the 'do-able' low hanging fruit
    - Make temporary exceptions for the tough stuff
  - **Form a committee**
    - You will need a virtual team of zEngineering resources, including:
      - z Sysprog
      - z Operations
      - z Security
      - z Information Management – DBA team
      - z Middleware/Transaction Management – WebSphere Application Server, CICS, IMS etc.
      - z Network
    - Better get a Project Manager
      - to manage the non-security resource requests
      - Run status meetings
      - Publicise success
      - Assist with change coordination
  - **Better get that budget**

Hit List

Remedial Actions (1)

- Need to test remedial actions where possible
  - Some changes will have to be 'gulp, buckle in'
- Need to roll from Testpit to Prod ("route to live")
  - Usual approach for z changes, not news, but be disciplined !
- Need mixture of skills
  - Security Engineering
  - Systems Programming, Storage Admin and more
  - z/OS Unix Script skills
    - Co-opt one of the traditional UNIX team
    - If none, get a young person, they probably know how
- Need to test rigorously
  - Obs: Curiously, rigorously and rigourousness are English, but not the converse
- Need backout plan(s)
- Some changes will be more comfortable if you have backups of:
  - External Security Manager DB
  - File Systems
- Need to carefully time some of the changes
  - Especially if you have to restore DB or File Systems
- Need to follow change process rigorously
  - As always, but even more so if that is possible

Remedial Actions (2)

- **Test all that you can**
  - Easier for some sites than others
  - Ensure your test platform contains the same Security DB and File Systems as your target system
    - No good proving your logic if the user that is tested has UID(0) on the Testpit
    - That would be no test at all
- **Need Guinea pigs**
  - Both as human users and non-human users
  - Human GPs
    - Yourselves first
    - Other zEngineers next, make them feel the pain too
      - Side benefit, they are likely the ones that have the most "wrong" access
  - Non-human GPs
    - Tricky
    - Be brave
    - Select trivial ones first
      - SMTP, Healthchecker, Security reporting etc.
    - Then the non-business critical but tough
      - Automation, Monitors
    - Then the 50/50 business critical tough ones
      - Scheduler, FTP, TCP/IP and friends
    - Finally the ones with data if you have any
      - Java based Apps, WebSphere Application Server, MQ etc.

UID policy

- Often, assignment has been 'organic'
  - Needs to be cleaned up
- You really need a published scheme for UID assignment
  - Better if it is enterprise wide (Utopia)
- Human users
  - Should be assigned according to a published scheme
  - Common practice - use an integer related to user
    - For example an employee number or other HR controlled string
  - But this is often not sustainable, integer design point was some other purpose
    - Better to just assign an integer within a range based on department
  - Prefer to separate human from non-human, so say 6 digit for humans
- Non human users
  - UIDs for system tasks should be assigned from reserved blocks
  - Again, a published scheme
  - Typically using the numeric range of 5 digits
  - UIDs for related tasks should be within the same number range
    - For example:
      - all WebSphere Application Server UIDs between 20,000 and 29,999
      - all TWS UIDs between 32,000 and 32,999
      - etc.
    - Larger sites may have to make these brackets wider
    - Larger sites may have to consider 6 digits, so consider humans as 7 digits

UID Actions (1)

- **ACTION: Define UID scheme**
  - High priority/Minimum effort/Low degree of difficulty
  - Policy presumably says this is required
  - Remedial action
    - None, documentation/argument action
- **ACTION: Realign human users to UID scheme**
  - Medium priority/Moderate effort/Low degree of difficulty
  - Policy presumably says you have to comply to scheme
  - If there are lots of challenges, may accept this for new users
    - Declare existing users misalignment as "within appetite"
  - Remedial action
    - Build list of files/directories owned by UID
    - Prepare script to 'chown' to new UID
    - Change UID to new UID
    - Run script to 'chown'
    - Logon on/activate or otherwise use
      - including FTP, HTTP, Batch submission etc.
    - Scan for old UID in files/directories

UID Actions (2)

- **ACTION: Realign non-human users to UID scheme**
  - Medium priority/Moderate effort/Medium to High degree of difficulty
  - Policy presumably says you have to comply to scheme
  - Some non-human users may be challenging
  - If there are lots of challenges, may accept this for new services only
    - Declare existing users misalignment as "within appetite"
  - Remedial action
    - Check this is "within appetite"
    - Build list of files/directories owned by UID
    - Prepare script to 'chown' to new UID
    - Stop related Services
    - Check the userid has NOPASSWORD (protected user)
    - Change UID to new UID
    - Run script to 'chown'
    - Restart related Services
    - Test every possible access point
    - Be mindful of 'off-platform' implications
      - For example, created files being accessed by HTTP or FTP
      - These may carry an 'off-platform' expectation of file UIDs
    - Scan for old UID in files/directories
    - Breath, take a well earned break

UID(0), Superuser et al

- Before looking at next action, a refresh on Superuser:
- Many names, myths and misconceptions surrounding this:
  - Superuser, Root Access, su
  - Real UID() versus Effective UID()
  - RACF Trusted and/or Privileged (or other ESM equivalent)
- Bottom line:
  - To invoke certain 'privileged' syscalls, a program must first acquire "appropriate privileges"
- So what does that mean ?
  - GA32-0884-00 - z/OS UNIX System Services Planning (v2.r1)
    - Phrase has 9 direct references but no explanation
  - SA23-2281-01 - z/OS UNIX System Services Programming: Assembler Callable Services Reference (v2.1)
    - Phrase has 161 direct references, all point to section on Authorisation which states:

      "Users authorized to perform special functions are defined as having *appropriate privileges*, and are called *superusers*. Users with appropriate privileges are also those with:

      - A user ID of zero
      - RACF-supported user privileges trusted and privileged, regardless of their user ID"
    - Well that's cleared that up then !

Some numbers

- To identify which syscall requires "appropriate privilege":
  - Read usage notes for each of:
    - the 1000+ C system calls
    - the 480+ Assembler z/OS Unix callable services
    - the 196 REXX z/OS Unix system calls
- Usage notes often state "may require"
  - For example, getpsent()
    - To issue this for your own UID, no additional authorisation is required
    - To issue this for all UIDs, you require "appropriate privilege"
      - UID(0) real or effective, TRUSTED/PRIVILGED (or equivalent)
- As at z/OS v1.13 (last time someone paid me to do this)
  - System calls = 32
    - Of which 11 have no SMF 80 event
  - Assembler Services = 46
    - Each of these services has two possible modules
  - REXX calls = 15
- Just for completeness
  - System calls and assembler services that require BPX.DAEMON = 14

Types of users with “appropriate privileges”

- Three ways of getting there:
  - Shell, batch job or STC with a real UID of zero
  - Shell, batch job or STC with a real UID of nonzero but an effective UID of zero
    - Shell acquires effective UID(0) through the use of the ‘su’ command
    - Others acquire effective UID(0) by issuing seteuid() to UID(0)
    - Either way, must have READ access to BPX.SUPERUSER
  - Task has RACF TRUSTED and/or PRIVILEGED (or other ESM equivalent)
    - Regardless of tasks real UID
    - TRUSTED – no security checking but auditable
    - PRIVILEGED - no security checking nor auditing
    - Coded in STARTED class profile
    - Can only apply to Started Tasks
    - Generally used for Resource Managers
      - JES2, HSM, DB2 etc.
- Some developers are not aware of these options
  - Challenge: “I need to issue getpsent() to list all processes, not just mine”
  - Path of least resistance, document that product needs UID(0)
    - “Take a rest, phew, that was hard work”
  - End of logic path
  - This is now an SEP
    - The “S” in this is you !

“Not really so super user” example

- “The user ID that is associated with the FTP server STARTED class must have UID 0.” – SC27-3650-03, page 741
  - z/OS v2r1 Communications Server: IP Configuration Guide
    - Similar in related Redbooks
- My experience is that nonzero is fine, but must have BPX.DAEMON
  - Does need BPX.SUPERUSER
    - This may depend on what FTP services you use
  - Last tested on z/OS v1.r13
- Caution
  - Real UID(0) implies many things
  - For example, number of running processes
  - Good coding can overcome this, or use of UNIXPRIV profiles
    - Not all coding is good, harsh but true
- In response to a PMR on FTP and UID(0), IBM state:
  - “Having the server setup to use SSL/TLS with TLSMECHANISM FTP specified requires that the process retain FTP's ID while performing data set and file accesses under the user's authority. Such context switches require SUPERUSER. TLSMECHANISM ATTLS does not require this.
  - Having the server setup for ANONYMOUS logins can also require SUPERUSER for the context switches (depending on the ANONYMOUSLEVEL specified). ”
    - By default TLSMECHANISM extension is not enabled
    - Have not had time to test if these statements prove to be ‘working as expected’

What about the rest of the TCP/IP services ?

- In my experience
- TCPIP itself really must be UID(0)
- Of the remaining 22 IP related services
  - inetd, rexecd, rlogind, rshd, sshd and syslogd really do require UID(0)
  - Requirement was 'removed' for pagent and iked at v1.r13
    - But it wasn't really needed from 1.7 onwards
  - 15 need BPX.SUPERUSER
  - 10 need BPX.DAEMON
- Each case lacks clear explanation as to why authorisation is required
- IBM should be petitioned to provide more clarity
  - As in the FTP example
  - That is, under what condition(s) is this elevated authority required ?
  - Good luck with that, feedback any results to me please, thanks.

UID Actions (3)

- **ACTION: Remove human user assignment of UID(0)**
  - **High priority/Minimum effort/Low degree of difficulty**
  - **Policy presumably says this is not compliant**
    - **To avoid repeat, assume this for ACTIONS**
  - **Remedial action**
    - **User must justify the requirement, why do they need it ?**
    - **Find matching UNIXPRIV profile to satisfy the requirement (unless it is specious)**
    - **If Resource Owner of UNIXPRIV profile approves it, provide access**
    - **Assign UID from scheme, using actions previously noted**
    - **Remove UID(0)**
    - **Test**
- **ACTION: Remove human user access to BPX.SUPERUSER**
  - **High priority/Minimum effort/Low degree of difficulty**
  - **No human requirement for access to BPX.SUPERUSER**
    - **No known IBM requirement for this today**
    - **But politics sometimes get in the way**
  - **Remedial action**
    - **If Resource Owner of BPX.SUPERUSER approves it, provide access**
      - **Preferably through emergency/break glass ID**
    - **Otherwise remove the access**

UID Actions (4)

- **ACTION: Remove non-human user assignment of UID(0)**
  - High priority/Maximum effort/High degree of difficulty
  - Remedial action
    - Every current non-human user with UID(0) must be tested for requirement
      - Evaluate product documentation, then challenge
        - As seen in earlier charts, this is not trivial
      - Test each service with UID(0) and without UID(0)
        - Same for BPX.SUPERUSER and BPX.DAEMON
    - Product owner must identify the requirement, why do they need it ?
    - Find matching UNIXPRIV profile to satisfy the requirement (unless it is specious)
    - If Resource Owner of UNIXPRIV profile approves it, provide access
    - Assign UID from scheme, using actions previously noted
    - Test, test, then test some more
- **ACTION: Remove non-human user access to BPX.SUPERUSER**
  - High priority/Maximum effort/High degree of difficulty
  - Many non-human users require access to BPX.SUPERUSER
    - But do not trust the documentation, test, test and test some more
  - Remedial action
    - If Resource Owner of BPX.SUPERUSER approves it, provide access
      - Preferably through emergency/break glass ID
    - Otherwise remove the access
      - Maybe schedule a week off just after the change

BPX.DEFAULT.USER Actions

- **z/OS V1.13 is the last release to support BPX.DEFAULT.USER**
  - Enough said. There are many papers available discussing this topic in detail
- **ACTION: Determine if BPX.DEFAULT.USER is in use**
  - High priority/Minimum effort/Low degree of difficulty
  - Remedial action
    - Quick Check - Run bpxcheck
      - Available from RACF downloads page
      - Will tell you if you have the profile and what state you are in for remedial steps
    - If you have it defined – loop through your SMF 80 records extended relocate sections
      - Look for two bytes of 317 (x'13D') "Indicates a default z/OS UNIX security environment is in effect"
        - Usually in the final section – (credit to Pat Loftus for pointing this out, thanks Pat)
      - Also look for Event codes 28-58, 60-65
      - SMF unload fields xxxx_DFLT_PROCESS
        - xxxx is the prefix to the SMF unload record, such as CMOD, COWN, FACC, IOEP
- **ACTION: If BPX.DEFAULT.USER is in use**
  - High priority/Maximum effort/Medium degree of difficulty
  - Remedial action 1
    - Assign a unique UID to each user and GID to each group
    - Chown all files/directories ownership to new UID/GID, some form of default owner
  - Remedial action 2
    - Use the BPX.UNIQUE.USER support to automatically assign a unique UID to each USS user and a unique GID for their group
    - May be 'frying pan into the fire' option

Files and directory ownership

- **Ownership rules**
  - Files/directories UID/GID ownership must be set according to policy
  - Many key system files/directories are logically owned by the zEngineering team
    - The GID is easy, use a Group that is defined as a resource owning entity
    - The UID is not so easy
      - Must be a userid that has a UID and is non-human
      - Cannot be a human because if they move on, and the userid is deleted, we have unowned files/directories
      - Policy should state which non-human userids should own files/directories
      - Generally a set of technical area related userids that are defined as PROTECTED
- **ACTION: Set correct files/directories ownership**
  - High priority/Moderate effort/Low degree of difficulty
  - Remedial action
    - Identify the files/directories that are incorrectly 'owned'
      - Use scripts to list all files/directories ownership and analyse
    - Define non-human userid(s) to own files/directories
      - Defined with NOPASSWORD (protected user)
    - Define Group(s) to own files/directories
    - Build and run script to 'chown' files/directories to new userid(s)/group(s)

Unowned files/directories

- **Unowned files/directories pose two problems**
  - **Performance and Security**
  - **In terms of security they pose a significant exposure**
  - **Consider the following:**
    - **A user has UID (45) and creates files**
    - **Files now have an FSP with a UID (45)**
    - **User is deleted, all files/directories are now unowned**
    - **New user is assigned UID (45)**
    - **New user now owns those files/directories**
  - **Typically caused by:**
    - **Removal of a user or group from ESM DB**
    - **Or when user has a UID re-assigned**
- **ACTION: Identify and amend unowned files/directories**
  - **High priority/Minimum effort/Low degree of difficulty**
  - **Remedial action**
    - **Define non-human userid(s) and group to act as 'orphan' file/directory owner**
      - **Defined with NOPASSWORD (protected user)**
    - **Scheduled BPXBATCH script to identify unowned files/directories**
      - **Use find with –nouser and –o –nogroup flags**
      - **Beware, this may run for a long time and perform a lot of I/O**
        - **Choose execution time carefully**
    - **Parse output to build script to 'chown' all unowned to ORPHAN userid/group**
    - **Post processing requires a manual task to identify correct owners**

z/OS Unix config files

- **Many z/OS Unix using services require configuration files**
  - **These may be able to be stored as PDS members rather than as files**
    - **E.g. most TCP/IP config files**
  - **Securing a PDS is generally better understood and therefore likely to be more secure than if in a File System file**
    - **Additionally, securing datasets using a profile readily caters for multiple levels of access and multiple users/groups with access**
- **ACTION: Identify config files that may be PDS member**
  - **High priority/Minimum effort/Low degree of difficulty**
  - **Remedial action**
    - **Resource Owner of Service to identify all config files that may be defined as PDS members**
    - **Co-ordinated change**
      - **Security Engineering to define correct DATASET profiles**
      - **Resource Owner of Service to move config to PDS members**
        - **Will generally require service stop/start**
        - **Documentation/change procedure updates**
        - **Must be tested**

Key file and directory permissions

- There are key z/OS Unix files and directories that must be secured
  - Policy should state the most important ones
  - Should dictate the correct file and directory mode bits
- Key directories:
  - / (root) – holds all mount points
  - /bin - core programs, many APF or Program Controlled
  - /dev - holds many files needed during IPL and Shell login
  - /etc - holds many key config files
  - /tmp – all users need write access
  - /var – many services need write access
- Key files:
  - /etc/rc – run commands executes at IPL time with UID(0) privilege
  - /etc/init.options – kernels control file
  - /etc/profile – default user profile settings/script
  - Automount master and maps
  - /etc/steplib – list of steplib datasets - location depends on BPXPRMxx
  - any cron files – presume cron not used so files should be secured
- ACTION: Ensure key files/directories have correct security
  - High priority/Minimum effort/Low degree of difficulty
  - Remedial action
    - Verify that all policy listed key files/directories are secured according to policy

BPXPRMxx security issues

- **Several keywords have significant security implications**
  - **MOUNT statements with**
    - **NOWRITEPROTECT**
      - **If coded, may result in data corruption in non-GRS environment**
      - **No checking for multiple File System access from multiple LPARs**
    - **NOSETUID**
      - **If coded, will result in loss of extended security functions**
      - **Any setuid() or setgid() bits are NOT honoured, for example APF and Program Control attributes**
    - **NOSECURITY**
      - **If coded, will result in unprotected files**
      - **Literally, no file or directory level security**
- **ACTION: Monitor mechanism to check for non-compliant BPXPRMxx definitions**
  - **High priority/Minimum effort/Low degree of difficulty**
  - **Remedial action**
    - **Implement regular scanning for these keywords in BPXPRMxx members and D OMVS,F output**
- **NOTE: Be cautious about using certain parms**
  - **STEPLIBLIST**
    - **Could override search order, spoofing of own program potentially**
  - **USERIDALIAS**
    - **May be prerequisite of software stack, avoid otherwise**
  - **BPXROOT override**

Access Control Lists

- Common feature in most UNIXs
  - z/OS Unix supports via RACF FSSEC class
- Purpose is to provide more granular security for file access
  - Allow multiple users/groups to be granted r/w/x access to file resources
  - Overcomes limitation of a single user/group setting for POSIX security control
- ACL is defined as an 'appendage' to a file or directory
  - The security data is resident in the inode of the File System containing the file or directory
- In order to use ACLs
  - FSSEC class to be active
  - An ACL has to be defined
  - File System has to be mounted with security enabled
- ACLs processed after the POSIX security bits
- Supports a default ACL definition
  - Both a default file and default directory ACL
  - Defined at the directory level
  - In addition to the directory's own possible ACL
- Any default will be applied to any new files/directories
  - But is not propagated to any files/directories already created

Authors view of ACLs

- **Put simply, ACLs should not be used**
  - With only the rarest of exceptions
  - Exceptions should be subject to the strictest scrutiny
  - Exceptions should require detailed business justification
    - that should be strongly challenged
- **Principle concerns with ACLs:**
  - Portability of files
  - Lack of POSIX compliance
  - Disk space overhead
  - Increased performance cost of I/O
  - Inheritance behaviour not consistent
  - Audit difficulties
  - Undercutting potential increase
  - Management difficulties
  - Lack of agility
  - Virus like qualities
    - A plague on your house !

Cure ACLs

- **ACTION: Remove ACLs**
  - **High priority/Maximum effort/Moderate degree of difficulty**
    - **Effort depends on quantity of ACLs already in place**
  - **Assume Policy dictates this**
  - **Remedial action**
    - **Part 1 - Identify ACLs in batch**
      - ⇨ **BPXBATCH using find / -acl**
    - **Part 2 – Convert Part 1 output to script to list ACL contents**
      - ⇨ **REXX followed by BPXBATCH getfacl**
    - **Part 3 – Analyse content to determine action**
      - ⇨ **Step can be skipped if Policy states no ACLs**
    - **Part 4 – remove ACLs if not justified**
    - **Part 5 – repeat for a 'number' of iterations**
      - ⇨ **More ACLs could have been created in elapsed remedial action time frame**
  - **Be warned this process may take many elapsed hours, one recent example:**
    - **Part 1 - Identify ACLs - elapsed time 5 hours**
      - ⇨ **Found 400,000**
    - **Part 2 – Convert and List contents – elapsed time 6 days**
    - **Part 3 – Analyse – elapsed time is "ongoing"**
  - **Avoid this advanced state of infection by ruling this OUT as an option**
    - **Localised and justified exceptions aside**
    - **At a minimum, do not allow inheritance (defaults) on any directory that has sub directories**
      - ⇨ **This will minimise the damage**

FSSEC Class

- **FSSEC has two purposes:**
  - Use of ACLs requires FSSEC class to be ACTIVE
  - FSSEC Class may be coded in LOGOPTIONS
  - These two purposes are not related, just happen to use the same Class name
  - Do not ACTIVATE FSSEC if you do not want ACLs
- **FSSEC in LOGOPTIONS**
  - A Quote:
  - "Avoid trouble: Enabling all auditing on classes that control access to objects in the UNIX System Services file system, such as RACF DIRACC, DIRSRCH, FSOBJ and FSSEC, or their equivalent in other SAF security managers, severely degrades performance."
    - WebSphere Application Server v8 Tuning Tips Infocenter
      - http://pic.dhe.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.multiplatform.doc%2Finfo%2Fae%2Fae%2Frprf_tunezsec.html
  - Classic trade off between performance and security
    - Most secure system is the one that is powered down
    - Best performing system is the one with no security
    - Performance overhead = CPU = increased software bills
      - This can be significant for high z/OS Unix usage customers

BPX.SAFFASTPATH

- Profile reduces RACF overhead
- Auditors don't like it
- When defined:
  - RACF is not called if z/OS Unix can determine that file access will be successful
    - for example a file that has permission bits of 777
  - Downside is that successful file accesses cannot be audited
  - Performance gain can be significant for high File System usage customers
    - E.g. WebSphere Application Server
- ACTION: Implement BPX.SAFFASTPATH if Policy caters for it
  - Medium priority/Minimum effort/Minimum degree of difficulty
  - Remedial action
    - Change must be loaded via an IPL or refresh of Kernel
      - SET OMVS=xx
- Alternative ACTION: Negotiate policy change for BPX.SAFFASTPATH
  - Medium priority/Moderate effort/Moderate degree of difficulty
  - Remedial action
    - Start negotiations with the auditor(s)

Summary

- z Security Policy covering z/OS Unix is key
- Resources will be needed to implement remedial actions
  - Budget and personnel
- UID(0)
  - for humans, remedy is straight forward and easy
  - for non-humans, remedy is bent and complex
  - BPX.SUPERUSER has to be removed from human users
- Files and Directories
  - Unowned files and directories are a serious security exposure
  - Identifying Resource Owners can be complex, consider a proxy owner/authoriser
  - Config files in a PDS member are preferred
  - Access Control Lists are a very bad idea
- Testing is essential
- Guinea Pigs are essential

Backup:
More on ACLs

Managing ACLs

- ACLs are listed with 'getfacl'
- ACLs created and modified with 'setfacl'
- Input to setfacl can be supplied in a file
- Authorisation:
  - Superuser
  - owner of the file
  - READ permission to the UNIXPRIV profile SUPERUSER.FILESYS.CHANGEPERMS

ACL POSIX compliance

- Use of ACLs is not POSIX compliant
  - IBMs implementation allows an ACL to both grant and restrict authority
  - This breaks the compliancy
- Ensure you have no contracts which require POSIX compliancy in terms of data processing and data holding
  - Typically a problem for American institutions
    - Due to their need to transact business with the US Defence Department
  - The US Defence Department's requirement for POSIX compliancy is due to their adoption of Federal Information Processing Standards FIPS 151-2
    - http://www.itl.nist.gov/fipspubs/fip151-2.htm
- IBM document their compliance statement in z/OS Unix manuals
  - "According to the X/Open UNIX® 95 specification, additional access control mechanisms may only restrict the access permissions that are defined by the file permission bits. They cannot grant additional access permissions. Because z/OS ACLs can grant and restrict access, the use of ACLs is not UNIX 95-compliant."
    - Reference is from z/OS V1R9.0 UNIX System Services Planning - GA22-7800-12

Best practice for ACLs

- **Put simply, ACLs should not be used**
  - **with only the rarest of exceptions to this policy**
- **Exceptions should be subject to the strictest scrutiny**
  - **Require detailed business justification that should be strongly challenged**
- **The reasons for this strong position are listed in the following charts.**

ACL concerns (1)

- **Portability of files**
  - **Backup/restore and file copy processing must be executed with correct commands and switches to ensure porting of the ACLs**
  - **Difficult to achieve as default setting for most copy/backup utilities is to ignore ACLs**
- **POSIX compliance**
  - **use of ACLs to grant access is not POSIX compliant.**
- **Space**
  - **ACLs adds a significant space overhead**
  - **A recent check at a large customer revealed 600,000 ACLs covering 900,000 files**
  - **The space overhead was significant**

ACL concerns (2)

- **Performance**
  - Path length of accessing files increases
    - Relative to the number of entries in each ACL
  - Read access to a file/directory that has POSIX value 777 must still be checked for each ACL
    - in case that ACL denies access
  - Performance problem exacerbated if the customer opts to implement file/directory defaults
    - Causes an ACL for every subsequent file and directory in the search path
- **Inheritance**
  - Inconsistency in how inherited ACLs will be created
    - Default ACLs for files/directories are not inherited across a mount point
- **Audit**
  - SETROPTS LOGOPTIONS for class FSSEC allows the auditing of the creation, modification and deletion of an ACL
  - But doesn't apply to default file/directory ACLs
  - Inconsistency of auditability

ACL concerns (3)

- **Undercutting**
  - ACLs make it possible to undercut the access granted in the POSIX settings at User or Group level
- **Management**
  - Extremely costly to manage if defaults are used
  - As files/directories created, new ACLs are created
  - Once ACL exists it is static, no longer tied to default
  - Makes administration of ACLs a manual effort
  - Affects every ACL in existence
  - Any attempt to change user/group structure will result in many hours if not days of ACL analysis just to identify the ACLs that require changing.
- **Agility**
  - By implementing ACLs, the data centre becomes tied to the depth of discrete definitions for file/directory access and has lost agility
- **AVOID LIKE THE PLAGUE !**