

Solaris 10 and Rights Management

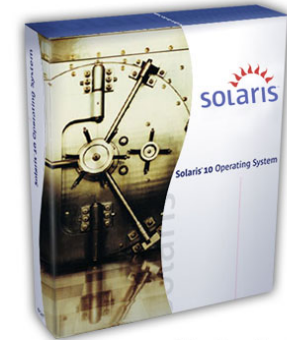
Scott Dickson

Solaris Ambassador

Sun Microsystems, Inc.

Rights Management

- User Rights Management (RBAC)
 - > Allows narrowed scope of root access
 - > Create roles and assign roles to individuals
 - > Manage roles in LDAP directory
- Process Rights Management
 - > Narrow scope of privileged system calls to just the set needed for application execution
 - > Removes the need for setuid applications
 - > Often removes the need to run as root



Unparalleled Security

Role-Based Access Control is ...

- A framework for delegating (system) administration tasks to users
 - > No need to share the root password
 - > Allow mundane users to run a few commands with extra privileges without giving them the “*keys to the kingdom*”
 - > Based on the ideas of roles, profiles, authorizations

Terminology – A Role is ...

- An account on the system, with its own UID, home directory, password, ...
- ... but it is not possible to log in as a role.
 - > A role is assumed by using su (1M) by a logged-in user who is authorized to assume that role.
 - > Login restriction is enforced by the pam_roles (5) module
 - > Usually intended for a specific function
 - > e.g. database administrator, operator, ...

Terminology – Rights Profile

- Often called just a *profile* or in SMC just *rights*
- Is a container that bundles RBAC attributes
 - > Includes authorizations
 - > Includes execution profiles
 - > Ordering matters. The first match is used
 - > Is mean to include all attributes in `user_attr(4)`
 - > Can contain other profiles

Terminology - Privilege

- An attribute of a process – part of process credentials
- Checked for by the kernel
- Usually given to programs
 - > Somewhat confusing because it is possible to set *limitprivs* and *defaultprivs* for a user entry in `user_attr` (4)
- More on this later

Terminology – Authorization

- Sort of like a movie ticket
 - > Application checks to see that you have the right ticket
- Assigned to users directly or via profiles
- Hierarchical
 - > Having the parent implies having all of the children
 - > Special “grant” authorization, so having an authorization does not automatically allow passing it to others
- Checked by applications
 - > In contrast to privileges (checked by the kernel)
 - > Allows applications to implement finer-grain controls
 - > Not used by kernel

Using Authorizations

- When do I use an authorization?
 - > When you are running with a privilege but not all users should benefit from all of your privilege
- Simple example: `cdrw(1)`
 - > Forced privilege via `setuid`, checks authorization
`solaris.device.cdrw`
- Complex example: `svc.{startd,configd}`
 - > System daemons check privileges of door caller

Terminology – Execution Profile

- Defined in `exec_attr(4)`
- Associates a rights profile with a list of commands
 - > Can optionally specify attributes for each command
 - > RUID, GUID, EUID, EGID
 - > Privileges, limitprivs
 - > Only commands in the rights profile will be executed by the profile shell
- One `exec_attr` file or table can be used across Solaris 8, Solaris 9, Solaris 10, Nevada, TSOL
 - > `suser` (S8, S9), `tsol` (TSOL8), `solaris` (S10, S10+)

Managing RBAC Configuration

- Location for each database controlled by nsswitch.conf
- When RBAC data is stored in local files only
 - > Edit files in /etc and /etc/security
 - > Use usermod(1M) and friends
 - > But note that these cannot do everything
- When RBAC configuration is stored in nameservice
 - > Use Solaris Management Console (SMC)
 - > GUI and CLI interfaces
- Can also use Webmin, Sun Identity Manager

When do I use a role?

- A role is useful if ...
 - > There is a need for a shared account
 - > There is a need for secondary authentication
 - > Often used for a well-defined business function
 - > Roles can be used for tasks which require no extra privileges, though most commonly they are used to grant extra privileges.

When do I create profiles?

- Profiles are the basic building blocks of Solaris RBAC
- Profiles ...
 - > Isolate implementation details
 - > e.g. Execution with privileges vs. setuid vs. privilege aware command
 - > Should exist for most Solaris functionality

Special Profiles

- All
 - > Allows all commands to be run if the user has the appropriate file permissions
 - > Does not run commands with privileges
 - > Do not modify this profile – will cause havoc with upgrades and patches
- Basic Solaris User
 - > Assigned to all users by default, in policy.conf(4)
 - > Grants “All” to all users
 - > Do not modify this profile – will cause havoc with upgrades and patches

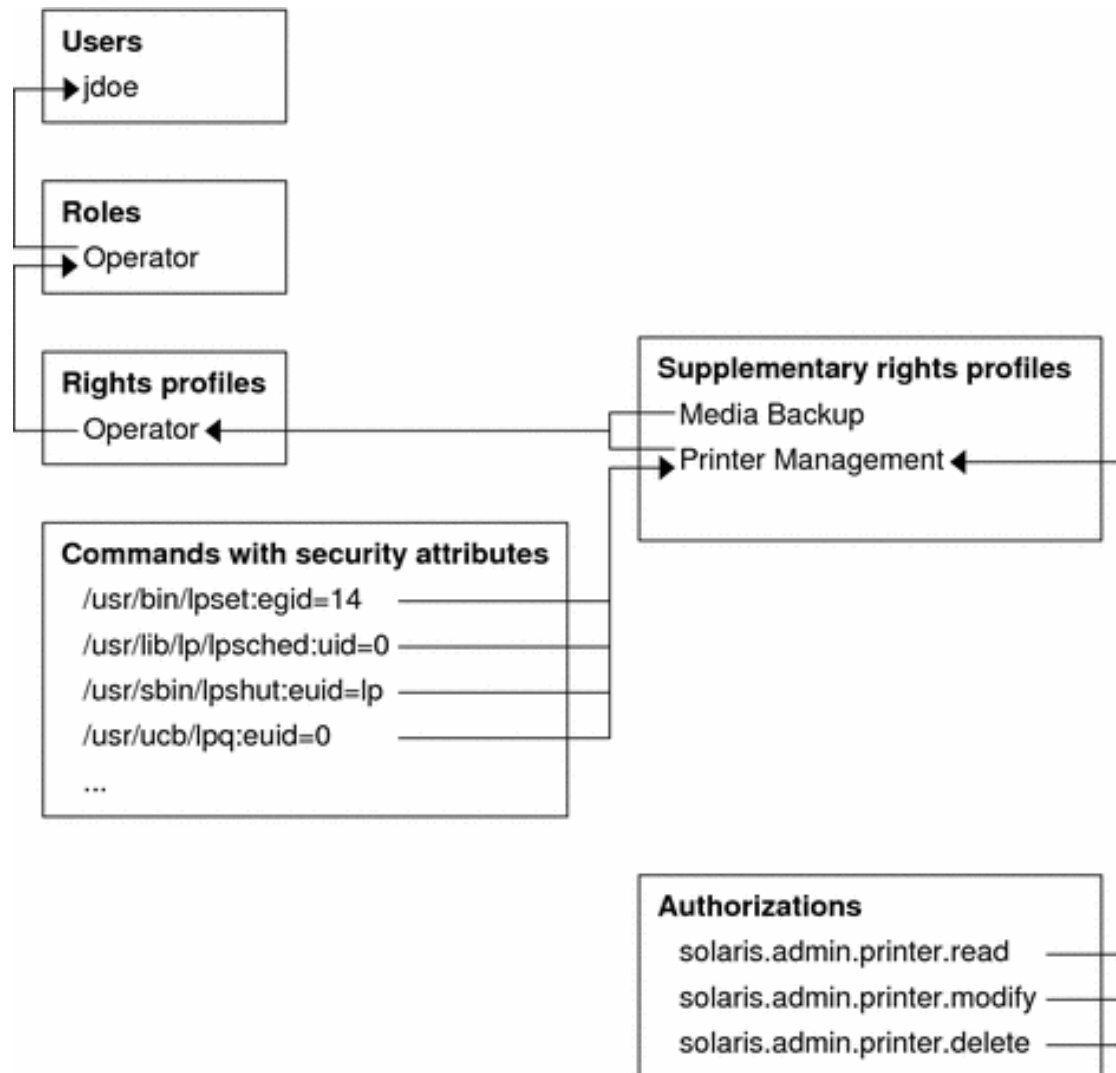
How does a profile get used?

- Profile shells
 - > /usr/bin/pfsh, /usr/bin/pfcsh, /usr/bin/pfksh
 - > Normal shells that check for profile entries.
 - > Invoke /usr/bin/pfexec to do the hard work
 - > /usr/bin/pfexec
 - > Small, setuid, checks rights profiles and starts commands with specified attributes.
 - > Can be invoked directly
 - > Is the closest RBAC equivalent to *sudo*

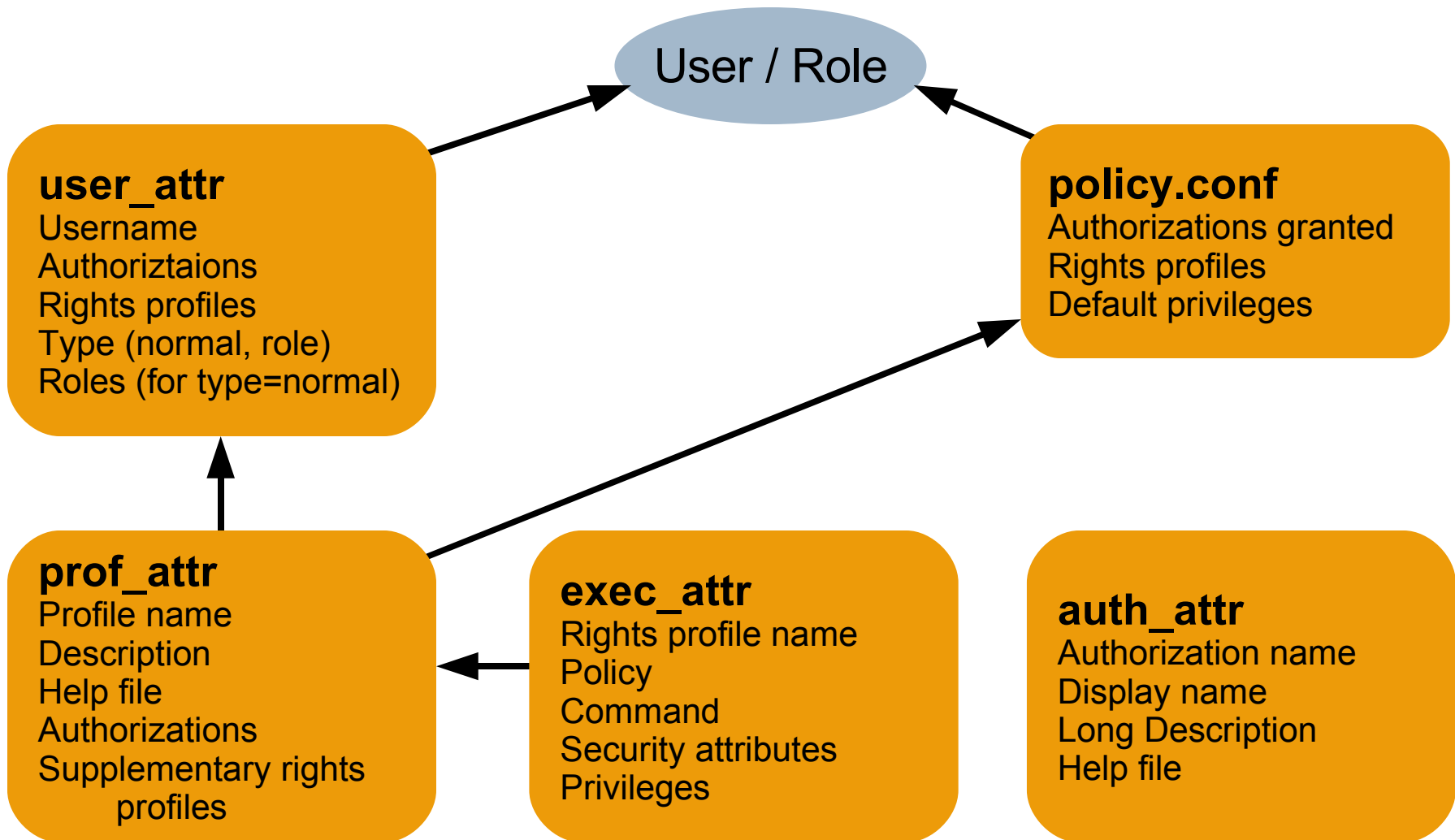
When to use a profile shell?

- Roles use profile login shells
- Users have regular shells
 - > But can be given a profile shell to ...
 - > Allow commands to be run directly with privileges
 - > Restrict which commands can be run (remove “All” or “Basic Solaris User” from their profiles)
 - > Regular shells (bash, ksh) can be used for running commands with privileges
 - > But users would need to invoke commands via pfexec manually (or use creative aliasing)

User Rights Management (Rights)



RBAC Database Relationships



Database Formats – user_attr

- Extended User Attributes database
- `username:res1:res2:res3:attr`
 - > `attr` is the list of attributes for the user / role.
 - > semicolon-separated list of
 - > `auths` – authorizations
 - > `profiles` – execution profiles
 - > `roles` – roles that the user may assume
 - > `type` – normal or role
 - > `project` – project to run under
 - > `defaultpriv` – default privileges for execution
 - > `limitpriv` – maximum privileges
 - > `lock_after_retries` – number of failed logins before lock

Database Formats – auth_attr

- Authorization Description database
- `name:res1:res2:shortdesc:longdesc:attr`
 - > name is the full name or prefix of authorization
 - > attr is a semicolon-separated list of key-value pairs
 - > Mostly used to identify the help file used by SMC for this authorization.

Database Formats – prof_attr

- Profile Description database
- `name:res1:res2:desc:attr`
 - > name is the name of the execution profile
 - > attr is a semicolon-separated list of key-value pairs
 - > Mostly used to identify the help file used by SMC for this authorization.

Database Formats – exec_attr

- Execution Profiles database
- `name:policy:res1:res2:id:attr`
 - > `policy` – security policy to be used: `suser` (standard Solaris superuser) or `solaris` (privilege-checking)
 - > `id` is the full path to the command to be executed
 - > `attr` is a semicolon-separated list of key-value pairs
 - > `uid, euid, gid, egid` – user and group for execution
 - > `privs, limitprivs` – privileges for execution

Database Formats – policy.conf

- Security policy configuration file
- Key-value pairs to define policy
 - > AUTHS_GRANTED – default set of authorizations
 - > AUTHS_GRANTED=solaris.device.cdrw
 - > PROFS_GRANTED – default set of profiles
 - > PROFS_GRANTED=Basic Solaris User
 - > PRIV_DEFAULT, PRIV_LIMIT – default privileges
 - > LOCK_AFTER_RETRIES=(YES|NO)
 - > CRYPT_ALGORITHMS_ALLOW,
CRYPT_ALGORITHMS_DEPRECATED,
CRYPT_ALGORITHMS_DEFAULT

Assigning Profiles

- Execution profiles can be directly assigned to users
`usermod -P "Network Management" plain`
- Execution profiles can be assigned to a role and users allowed to assume a role
`roleadd -P "File System Management" fsop`
`usermod -R fsop plain`
- Great basis for a religious war
 - > Second method is probably better practice
 - > First is a quick and easy hack

User Rights Management Example #1

```
$ profiles -l
```

```
Object Access Management:
```

```
    /usr/bin/chgrp      privs=file_chown
    /usr/bin/chmod     privs=file_owner
    [...]
```

```
[...]
```

```
$ ls -ld mnt
```

```
drwxr-xr-x  2 gbrunett gbrunett      512 Nov  7 12:54 mnt
```

```
$ chown bin:bin mnt
```

```
chown: mnt: Not owner
```

```
$ pfexec chown bin:bin mnt
```

```
$ ls -ld mnt
```

```
drwxr-xr-x  2 bin      bin      512 Nov  7 12:54 mnt
```

User Rights Management Example #2

```
# svcprop -p httpd -p general apache2
general/enabled boolean false
general/action_authorization astring sunw.apache.oper
general/entity_stability astring Evolving
httpd/ssl boolean false
httpd/stability astring Evolving
```

```
# auths weboper
sunw.apache.oper
```

```
# profiles -l weboper
```

```
    Apache Operator:
        /usr/sbin/svcadm
        /usr/bin/svcs
```

User Rights Management Example #2

```
$ svcs -o state,ctid,fmri apache2  
STATE          CTID    FMRI  
online         91050   svc:/network/http:apache2
```

```
$ svcadm restart apache2
```

```
$ svcs -o state,ctid,fmri apache2  
STATE          CTID    FMRI  
online         91064   svc:/network/http:apache2
```

```
$ ls  
ls: not found
```

```
$ echo *  
local.cshrc local.login local.profile
```

Lab 1 – Assign Profiles to a User

- Create a non-root user
- Assign an execution profile that will allow the user to execute `ifconfig` as if they were root
- Become the non-root user
- See what roles, profiles, and authorizations this user has assigned
- Run “`ifconfig -a`” using and not using the execution profile and compare the difference

Lab 1 – Assign Profiles to a User

- Create a non-root user

```
# useradd -d /export/home/plain -m \  
          -s /bin/bash plain  
# passwd plain
```

- Assign an execution profile that will allow the user to execute `ifconfig` as if they were root

```
# egrep ifconfig /etc/security/exec_attr  
Network Management:solaris:cmd:::/sbin/ifconfig:uid=0
```

-
- ```
usermod -P "Network Management" plain
```

```
egrep plain /etc/user_attr
plain::::type=normal;profiles=Network Management
```

# Lab 1 – Assign Profiles to a User

- Become the non-root user. See what roles, profiles, and authorizations this user has assigned

```
su - plain
$ roles
No roles
$ auths
<... lots of authorizations here ...>
$ profiles
Network Management
Network Wifi Management
Basic Solaris User
All
```

# Lab 1 – Assign Profiles to a User

- Run “ifconfig -a” using and not using the execution profile and compare the difference

```
$ /usr/sbin/ifconfig -a
```

```
bcmndis0:
```

```
flags=201004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP
,IPv4,CoS> mtu 1500 index 6
 inet 172.31.34.87 netmask ffffffff00
```

```
broadcast 172.31.34.255
```

```
$ pfexec /usr/sbin/ifconfig -a
```

```
bcmndis0:
```

```
flags=201004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP
,IPv4,CoS> mtu 1500 index 6
 inet 172.31.34.87 netmask ffffffff00
```

```
broadcast 172.31.34.255
```

```
 ether 0:b:6b:4b:f0:cd
```

## Lab 2 – Create and Use Roles

- Find which execution profiles are required to create new users
  - > (Hint: don't try to create the home directory – requires another execution profiles. Just create the user in /etc/passwd and set the password)
- Create a role that can use these execution profiles
- Assign this role to your non-root user
- Try to add a user
- Assume the role you created
- Try again to add a user

# Lab 2 – Create and Use Roles

- Find which execution profiles are required to create new users
  - > User Security – allows passwd
  - > User Management – allows useradd
- Create a role that can use these execution profiles
- Assign this to your non-root user
 

```
roleadd -P "User Security,User Management" ua
passwd ua
usermod -R ua plain
tail /etc/user_attr
plain::::type=normal;roles=ua;profiles=Network
Management
ua::::type=role;profiles=User Security,User
Management
```

# Lab 2 – Create and Use Roles

- Try to add a user

```
$ su - plain
```

```
$ /usr/sbin/useradd foo
```

```
UX: /usr/sbin/useradd: ERROR: Permission denied.
```

- Assume the role you created

- Try again to add a user

```
$ su ua
```

```
Password:
```

```
$ /usr/sbin/useradd foo
```

```
$
```

# Why do we need Privileges?

- Lots of software running as root
  - > Root software has power to do anything
- Customer concern about why software needs to run as root
- Any *root* software breach is catastrophic
- Most *root* software only need 1-2 special powers

# Process Privileges

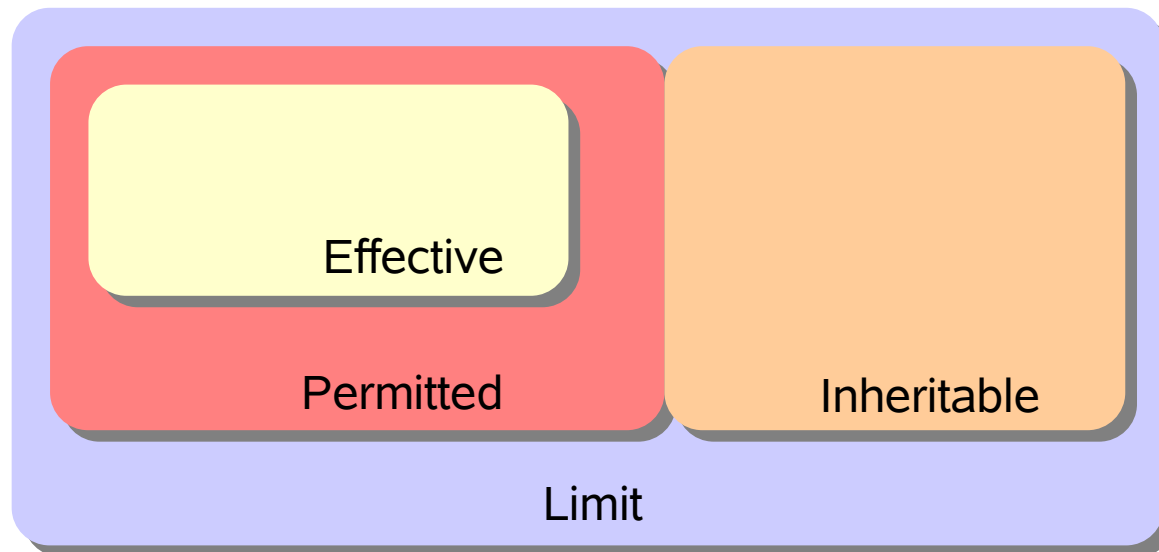
- Solaris kernel checks for privileges and not just `UID == 0`!
  - > Division of `root` authority into discrete privileges (67 and counting)
  - > Privileges can be granted to processes based on need.
  - > Privileges can be disabled or dropped when not needed.
  - > Child processes can have different (fewer) privileges than the parent.
- Completely backward compatible and extensible.
  - > No changes required to use existing code.
- Privilege bracketing helps to mitigate effects of future flaws.
  - > e.g., `proc_fork` and `proc_exec`
  - > e.g., `proc_info`

# Privilege Sets

- Currently 67 fine-grained privileges instead of UID=0
  - > `ppriv -lv` shows privileges and what they protect
- Each process has 4 privilege sets in its process credential
  - > Inheritable set (I) – set of privileges child processes get on exec
  - > Permitted set (P) – maximum set of privileges
  - > Effective set (E) – subset of P currently asserted as needed
  - > Limit set (L) – upper bound a process and children can obtain

# Process Privilege Sets

- E - Effective
  - > Privileges in effect
- P - Permitted set
  - > Upper bound of E
- I - Inheritable set
  - > Privileges of executed programs
- L - Limit set
  - > Upper bound for the process and all its descendants



# Basic Privileges

- Basic privileges are things normal users can normally do.
  - > `proc_fork, proc_exec, proc_session, proc_info, file_link_any`
- This set will expand
- Dropping `proc_fork` and `proc_exec` from system daemons that should never fork or exec gives extra protection from buffer overflow exploits.
- Dropping `proc_info` means you can't see other users' processes.

# Solaris 10's Privilege Names

|                      |                                         |
|----------------------|-----------------------------------------|
| "contract_event"     | Request reliable delivery of events     |
| "contract_observer"  | Observe contract events for other users |
| "cpc_cpu"            | Access to per-CPU perf counters         |
| "dtrace_kernel"      | DTrace kernel tracing                   |
| "dtrace_proc"        | DTrace process-level tracing            |
| "dtrace_user"        | DTrace user-level tracing               |
| "file_chown"         | Change file's owner/group IDs           |
| "file_chown_self"    | Give away (chown) files                 |
| "file_dac_execute"   | Override file's execute perms           |
| "file_dac_read"      | Override file's read perms              |
| "file_dac_search"    | Override dir's search perms             |
| "file_dac_write"     | Override (non-root) file's write perms  |
| "file_link_any"      | Create hard links to diff uid files     |
| "file_link_any"      | Create hard links to diff uid files     |
| "file_owner"         | Non-owner can do misc owner ops         |
| "file_setid"         | Set uid/gid (non-root) to diff id       |
| "ipc_dac_read"       | Override read on IPC/Shared Mem perms   |
| "ipc_dac_write"      | Override write on IPC/Shared Mem perms  |
| "ipc_owner"          | Override set perms/owner on IPC         |
| "net_icmpaccess"     | Send/Receive ICMP packets               |
| "net_privaddr"       | Bind to privilege port (<1023+extras)   |
| "net_rawaccess"      | Raw access to IP                        |
| "proc_audit"         | Generate audit records                  |
| "proc_chroot"        | Change root (chroot)                    |
| "proc_clock_highres" | Allow use of hi-res timers              |
| "proc_exec"          | Allow use of execve()                   |

|                    |                                       |
|--------------------|---------------------------------------|
| "proc_fork"        | Allow use of fork*() calls            |
| "proc_info"        | Examine /proc of other processes      |
| "proc_lock_memory" | Lock pages in physical memory         |
| "proc_owner"       | See/modify other process states       |
| "proc_priocntl"    | Increase priority/sched class         |
| "proc_session"     | Signal/trace other session process    |
| "proc_setid"       | Set process UID                       |
| "proc_taskid"      | Assign new task ID                    |
| "proc_zone"        | Signal/trace processes in other zones |
| "sys_acct"         | Manage accounting system (acct)       |
| "sys_admin"        | System admin tasks (e.g. domain name) |
| "sys_audit"        | Control audit system                  |
| "sys_config"       | Manage swap                           |
| "sys_devices"      | Override device restricts (exclusive) |
| "sys_ipc_config"   | Increase IPC queue                    |
| "sys_linkdir"      | Link/unlink directories               |
| "sys_mount"        | Filesystem admin (mount,quota)        |
| "sys_net_config"   | Config net interfaces,routes,stack    |
| "sys_nfs"          | Bind NFS ports and use syscalls       |
| "sys_res_config"   | Admin processor sets, res pools       |
| "sys_resource"     | Modify res limits (rlimit)            |
| "sys_suser_compat" | 3rd party modules use of suser        |
| "sys_time"         | Change system time                    |

Interesting  
Basic  
Zones

Some interesting privileges  
Non-root privileges  
Privileges removed from Zones

# Process Privilege Inheritance

- Limit (L) is unchanged
- L is used to bound privileges in Inheritable (I)
  - >  $I' = I \cap L$
- Child's Permitted (P') & Effective (E') are:
  - >  $P' = E' = I'$
- Typical process
  - >  $P = E = I = \{\text{basic}\}$
  - >  $L = \{\text{all privileges}\}$
  - > Since  $P = E = I$ , children run with same privileges

# Root Account Still Special

- *root* owns all configuration/system files
  - > UID 0 is therefore still very powerful
- Privilege escalation prevention
  - > Require ALL privileges to modify objects owned by *root* when euid  $\neq$  0
  - > Fine tuning in certain policy routines
    - > Not all privileges, only *nosuid* mounts
- Prefer services be non-UID 0 + privileges
  - > Additive approach is safer than UID 0 – privileges

# Using Process Privileges

- ppriv(1)

```
ppriv -e -D -s -proc_fork,-proc_exec /bin/sh -c finger
sh[387]: missing privilege "proc_fork" (euid = 0, syscall = 143)
needed at cfork+0x18
/bin/sh: permission denied
```

- User Rights Management (RBAC)

```
grep "Network Management" /etc/security/exec_attr
Network Management:solaris:cmd:::/sbin/ifconfig:privs=sys_net_config
Network Management:solaris:cmd:::/sbin/route:privs=sys_net_config
```

- Service Management Framework (SMF)

```
svcprop -p start rpc/bind | grep privileges
start/privileges astring
basic,file_chown,file_chown_self,file_owner,net_privaddr,
proc_setid,sys_nfs,net_bindmlp
stop/limit_privileges astring :default
```

- Privilege Aware Commands / Services

e.g., *ping*, *rmformat*, *quota*, *rpcbind*, *nfsd*, *mountd*

# Process Privileges Example #1

```
$ ppriv $$
```

```
28983: bash
flags = <none>
 E: basic
 I: basic
 P: basic
 L: all
```

```
$ ppriv -l basic
```

```
file_link_any
proc_exec
proc_fork
proc_info
proc_session
```

```
$ ppriv -De cat /etc/shadow
```

```
cat[3988]: missing privilege "file_dac_read" (euid =
101, syscall = 225) needed at ufs_iaccess+0xc9
cat: cannot open /etc/shadow
```

```
$ ppriv -s -proc_fork,-proc_exec -De /bin/vi
[attempt to run a command/escape to a shell]
```

```
vi[4180]: missing privilege "proc_fork" (euid = 101,
syscall = 143) needed at cfork+0x3b
```

# Process Privileges Example #2

```
ppriv -S `pgrep rpcbind`
933: /usr/sbin/rpcbind
flags = PRIV_AWARE
 E: net_bindmlp,net_privaddr,proc_fork,sys_nfs
 I: none
 P: net_bindmlp,net_privaddr,proc_fork,sys_nfs
 L: none
```

```
ppriv -S `pgrep statd`
5139: /usr/lib/nfs/statd
flags = PRIV_AWARE
 E: net_bindmlp,proc_fork
 I: none
 P: net_bindmlp,proc_fork
 L: none
```

# Process Privileges Example #3

## Solaris 9 Network Management Rights Profile

```
grep "Network Management" /etc/security/exec_attr
Network Management:suser:cmd:::/usr/sbin/ifconfig:uid=0
Network Management:suser:cmd:::/usr/sbin/route:uid=0
[...]
```

## Solaris 10 Network Management Rights Profile

```
grep "Network Management" /etc/security/exec_attr
Network Management:solaris:cmd:::/sbin/ifconfig:privs=sys_net_config
Network Management:solaris:cmd:::/sbin/route:privs=sys_net_config
[...]
```

## Assign a user's default privileges

```
grep defaultpriv /etc/user_attr
dickson::::type=normal;defaultpriv=basic,dtrace_proc,dtrace_user,dtra
ce_kernel
[...]
```

# Process Privileges Example #4

SMF Integration of privileges for a method.  
From /var/svc/manifest/system/sar.xml

```
<exec_method
 type='method'
 name='start'
 exec='/lib/svc/method/svc-sar %m'
 timeout_seconds='60'>
 <method_context>
 <method_credential
 user='sys'
 group='sys'
 privileges='basic,file_dac_write' />
 </method_context>
 </exec_method>
```

# Process Privilege Debugging

```
web_svc zone: # svcadm disable apache2
```

```
global zone: # privdebug -v -f -n httpd
```

```
web_svc zone: # svcadm enable apache2
```

```
global zone: [output of privdebug command]
```

| <u>STAT</u> | <u>TIMESTAMP</u> | <u>PPID</u> | <u>PID</u> | <u>PRIV</u>  | <u>CMD</u> |
|-------------|------------------|-------------|------------|--------------|------------|
| USED        | 273414882013890  | 4642        | 4647       | net_privaddr | httpd      |
| USED        | 273415726182812  | 4642        | 4647       | proc_fork    | httpd      |
| USED        | 273416683669622  | 1           | 4648       | proc_fork    | httpd      |
| USED        | 273416689205882  | 1           | 4648       | proc_fork    | httpd      |
| USED        | 273416694002223  | 1           | 4648       | proc_fork    | httpd      |
| USED        | 273416698814788  | 1           | 4648       | proc_fork    | httpd      |
| USED        | 273416703377226  | 1           | 4648       | proc_fork    | httpd      |

**privdebug is available from the OpenSolaris Security Community:**  
<http://opensolaris.org/os/community/security/projects/privdebug/>

# Configurable Privileges

- Like the Java sandbox of 1995, the zones sandbox model is being expanded
- Some privileges, previously excluded from a zone, may now be granted by the platform administrator to allow greater function within the zone, e.g. DTrace
  - > DTrace uses 3 privileges: `dtrace_user`, `dtrace_proc`, `dtrace_kernel`.
  - > `dtrace_user` and `dtrace_proc` may be granted via `zonecfg` to a zone, but not `dtrace_kernel`

# Lab 3 – View Privileges

- See what privileges your non-root shell has
- See what privileges your root shell has
- Get a list of all of the privileges defined on the system
- See what specific privileges make up the basic privilege

# Lab 3 – View Privileges

- See what privileges your non-root shell has  
`$ ppriv $$`
- See what privileges your root shell has  
`# ppriv $$`
- Get a list of all of the privileges defined on the system  
`# ppriv -lv`
- See what specific privileges make up the basic privilege  
`# ppriv -l basic`

## Lab 4 – Privileges (Nevada)

- Use `ppriv` to determine what privileges a non-root users needs to run `ifconfig -a`
- Add the needed privileges to that user
- Run the command again to see the privileges in use
- How is this different from the way a profile might be invoked

# Lab 4 – Privileges (Nevada)

- Use ppriv to determine what privileges a non-root users needs to run ifconfig -a

```
$ ppriv -eD ifconfig -a
```

```
bge0:
```

```
flags=201004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IP
v4,CoS> mtu 1500 index 2
 inet 10.192.130.116 netmask ffffffff0 broadcast
10.192.130.127
```

```
ifconfig[1006]: missing privilege
```

```
"net_rawaccess" (euid = 27591, syscall = 5) for
"devpolicy" needed at spec_open+0xc9
```

- Add the needed privileges to that user

In /etc/user\_attr

```
plain::::type=normal;roles=ua;profiles=Network \
Management;defaultpriv=basic,net_rawaccess
```

# Lab 4 – Privileges (Nevada)

- Run the command again to see the privileges in use

```
$ /usr/sbin/ifconfig -a
bge0:
flags=201004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IP
v4,CoS> mtu 1500 index 2
 inet 10.192.130.116 netmask fffffffc0 broadcast
10.192.130.127
 ether 0:c0:9f:5b:43:4f
```
- How is this different from the way a profile might be invoked

## Lab 4 – Privileges (Solaris 10)

- Use ppriv to determine what privileges a non-root users needs to read /etc/shadow
- Add the needed privileges to that user
- Verify that this works
- Is this a good idea? Or should this be part of a rights profile?

# Lab 4 – Privileges (Solaris 10)

- Use ppriv to determine what privileges a non-root users needs to read /etc/shadow

```
$ ppriv -eD cat /etc/shadow
```

```
cat[957]: missing privilege "file_dac_read" (euid =
27589, syscall = 225) needed at ufs_iaccess+0xe1
cat: cannot open /etc/shadow
```

- Add the needed privileges to that user  
In /etc/user\_attr

```
plain::::type=normal;defaultpriv=basic,file_dac_read
```

# Lab 4 – Privileges (Solaris 10)

- Run the command again to see the privileges in use  

```
$ tail -1 /etc/shadow
ua:OnwXhoW5GSajI:13629::::::
```
- Is this a good idea? Or should this be part of a rights profile?

# Solaris 10 and Rights Management

**Scott Dickson**

scott.dickson@sun.com