

open



USE



IMPROVE



EVANGELIZE

How to make your programs privilege aware

Wolfgang Ley
Technology Consultant
Sun Microsystems

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
:::
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை



Overview

- Why privileges?
- How are privileges managed/grouped
- Using privileges with your applications
 - Default configurations
 - Execution profiles
 - Daemons (via SMF)
 - Inside the application itself
 - Drivers
- Privilege debugging



Why privileges?

- Classical Unix credentials are userids and groupids (real, effective and saved)
- Privileges were mainly bound to userids
- Special userid 0 (root) with all rights
 - Once you're root (e.g. due to a security bug) you can do anything
- Split up the privileges into smaller pieces
 - Assign only those privileges which are needed
 - Give up privileges not needed
- Less setuid programs, less uid 0 processes



What are privileges?

- Individual privileges used inside the kernel
 - e.g. to open a privileged port, access a file etc.
- Access control based on available privilege (not on user-id)
- OpenSolaris already has 68+ different privileges (and the number will grow)
- Special set of “basic” privileges for everyone
 - file_link_any, proc_exec, proc_fork, proc_info, proc_session
 - This is not a static/stable set



How are privileges managed?

- Privileges are represented as strings
 - Use “ppriv -lv” to get a listing of available privileges
 - See manual page of privileges(5)
- Kernel internal they are stored as bitfields
- Associated with the process
 - As part of the credential structure (cred_t)
- Grouped into four different sets
 - Each of the set has a special meaning



Privilege sets

- **Effective set (E)**
 - Privileges currently active for the process
- **Inheritable set (I)**
 - Privileges which will be inherited on exec()
 - Childs will get this set
- **Permitted set (P)**
 - Maximum set of privileges available for this process
- **Limited set (L)**
 - Overall limit of available privileges
 - Takes effect on exec()



Privilege sets

- A process can turn privileges on and off
 - By adding or removing them from the E (effective) set
 - Can choose from the permitted set P
- A process can permanently give away a priv
 - By removing the privilege from the P set
 - Will be automatically removed from E, too
- Can restrict what child can do
 - By removing the privilege from the I set
- Or even what the children of the child can do
 - By removing the privilege from the L set



Privilege sets

- On `exec()` new privileges will be calculated
 - Intersection of inheritable (I) and limit (L) set
 - Will be new defaults (P, E and I) for the child
 - Child will inherit L from the parent
- Special case of `setuid 0`
 - Intersection will not be done between I and L
 - Intersection of all privileges with the limit (L) set



How to use privileges - defaults

- System defaults for all users
 - /etc/security/policy.conf
 - PRIV_DEFAULT=basic
 - PRIV_LIMIT=all
- Default for each user
 - In /etc/user_attr
 - Or in the nameservice (follows passwd search order)
- In real life not that useful though
 - Gives the user too much power
 - Or restricts the user too much



How to use privileges – execution profiles

- Use the `user_attr` source to assign a profile instead of the privileges
- Profiles are defined in `/etc/security/prof_attr`
 - More than just privileges (e.g. authorizations which are used inside applications for access control)
 - May contain list of privileged applications
- Privileges assigned to applications
 - In `/etc/security/exec_attr`
 - Can be `uid`, `euid`, `gid`, `egid` but also privileges
 - Keywords “`privs`” and “`limitprivs`”



How to use privileges – execution profiles

```
# grep IP\ Filter /etc/security/prof_attr
```

```
IP Filter Management:::IP Filter Administration:help=RtIPFilterMngmnt.html
```

```
# grep IP\ Filter /etc/security/exec_attr
```

```
IP Filter Management:solaris:cmd:::/usr/sbin/ipf:privs=sys_ip_config
```

```
IP Filter Management:solaris:cmd:::/usr/sbin/ipfs:privs=sys_ip_config
```

```
IP Filter Management:solaris:cmd:::/usr/sbin/ipfstat:privs=sys_ip_config;gid=sys
```

```
IP Filter Management:solaris:cmd:::/usr/sbin/ipmon:privs=sys_ip_config
```

```
IP Filter Management:solaris:cmd:::/usr/sbin/ipnat:privs=sys_ip_config;gid=sys
```

```
IP Filter Management:solaris:cmd:::/usr/sbin/ippool:privs=sys_ip_config;gid=sys
```

```
# grep Zone /etc/security/exec_attr
```

```
Zone Management:solaris:cmd:::/usr/sbin/zlogin:uid=0
```

```
Zone Management:solaris:cmd:::/usr/sbin/zoneadm:uid=0
```

```
Zone Management:solaris:cmd:::/usr/sbin/zonecfg:uid=0
```



How to use privileges – execution profiles

- Starting privileged programs directly
 - Via setuid wrapper: pfexec
- Let the shell handle the pfexec invocation
 - Use profile shells (pfsh, pfcsh, pfksh, ...)
 - Shell will start external application via pfexec
- Setup can be used to create restricted users
 - Remove default profile “Basic Solaris User”
 - Create special profile containing only the allowed cmds
 - Assign profiles to users
 - Give users profile shell to enforce pfexec usage



How to use privileges – daemons

- Let SMF (Service Management Facility) handle your service
- Create a service manifest (XML file)
- Specify privileges assigned to this service
 - Inside the start/stop method specification
 - Inside `<method_context>` is `<method_credential>`
 - Can assign userid, groupid and privileges
 - Keywords “privileges” and “limit_privileges”
- Service will automatically start with the least privileges (no need to run as user root)

How to use privileges – daemons

```
# more /var/svc/manifest/network/rpc/bind.xml
[...]
<exec_method
  type='method'
  name='start'
  exec='/lib/svc/method/rpc-bind %m'
  timeout_seconds='60'>
  <method_context>
    <method_credential
      user='root'
      group='root'
      privileges='basic,file_chown,file_chown_self,file_owner,
                net_privaddr,proc_setid,sys_nfs,net_bindmlp'
    />
  </method_context>
</exec_method>
```



How to use privileges by yourself

- Turn off privileges unless needed
 - Remove them from the effective set (E)
- Use privileges where needed
 - Add them to the effective set (E)
- Permanently drop a privilege if no longer needed
 - Remove the privilege from the permitted set (P)
- Decide whether you want to pass privileges to childs



How to use privileges by yourself

- You'll need to convert privilege strings to their internal representation
 - Type `priv_set_t` defined in `priv.h`
 - API documented in `priv_str_to_set(3C)`
- Functions to manipulate privilege sets
 - E.g. test for a privilege, calculate intersections etc.
 - API documented in `priv_addset(3C)`
- Finally set/unset privileges as needed
 - API documented in `getppriv(2)` and `priv_set(3C)`



How to use privileges by yourself

- Basic logic for setuid programs or daemons
 - Start with the “basic” set
 - Add privileges needed and remove unneeded ones
 - Compute the inverse and subtract this from P and L
 - Now you'll have E, P and L to the maximum set needed
 - Give up userid 0
 - Turn off privs from E and only use them where needed
 - Permanently give away privileges if no longer needed (by removing them from P and L).



How to use privileges by yourself

- Example program in paper
- Does only use official API
- Much easier API available inside ON
 - Interesting for OpenSolaris developers
 - Easy setup: `__init_suid_priv()` or `__init_daemon_priv()`
 - Easy use: `__priv_bracket()`
 - And give the privileges up: `__priv_relinquish()`
 - See references in paper



How to use privileges by drivers

- Privilege checks when accessing devices
- Already built in OpenSolaris
- Device policy can define which privileges are needed to open a device (for read or write)
 - Defined in `/etc/security/device_policy`
 - Managed by `add_drv` and `-p` option
- Drivers might also implement their own and new privileges
 - Defined in `/etc/security/extra_privs`
 - Managed by `add_drv` and `-P` option

How to use privileges by drivers

```

#
# Disk devices.
#
md:admin                write_priv_set=sys_config
fssnap:ctl  read_priv_set=sys_config  write_priv_set=sys_config
scsi_vhci:devctl        write_priv_set=sys_devices
#
# Other devices that require a privilege to open.
#
random                  write_priv_set=sys_devices
openeepr                write_priv_set=all
dld:ctl  read_priv_set=sys_net_config  write_priv_set=sys_net_config
aggr:ctl  read_priv_set=sys_net_config  write_priv_set=sys_net_config
#
# IP Filter
#
ipf      read_priv_set=sys_ip_config  write_priv_set=sys_ip_config

pcelx:* read_priv_set=net_rawaccess write_priv_set=net_rawaccess
    
```



Privilege Debugging

- Truss output will tell you why the call failed
 - `open64("/etc/shadow", O_RDONLY) Err#13 EACCES [file_dac_read]`
- Inspect privileges of processes
 - Using the `ppriv` command with a pid
- Run commands under privilege debugger
 - Start the via `ppriv -e`
 - Or enable privilege debugging for already running processes
- Globally enable privilege debugging
 - Via `priv_debug=1` in `/etc/system` or via `kmdb`



Privilege Debugging

```
schlepptop% truss cat /etc/shadow
execve("/usr/bin/cat", 0x08047CFC, 0x08047D08) argc = 2
[...]
open64("/etc/shadow", O_RDONLY)           Err#13 EACCES [file_dac_read]
```

```
schlepptop% ppriv -e -D cat /etc/shadow
cat[26423]: missing privilege "file_dac_read" (euid = 1001, syscall = 225)
           needed at ufs_iaccess+0xca
cat: cannot open /etc/shadow
```

```
schlepptop# ppriv -v `pgrep rpcbind`
224:  /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
```



Use privileges

- Use privileges in your own applications
- Authorizations might also be of interest
 - Not enforced by kernel but in userland
- Improve OpenSolaris by converting existing applications to use privileges
 - Use OpenSolaris bug database
 - Search for “needs to utilize privileges” in “State: 3 – Accepted”
 - File a new bug if there really isn't one (search in all bugs first)

open



USE



IMPROVE



EVANGELIZE

Thank you!

Wolfgang Ley
Technology Consultant
Wolfgang.Ley@sun.com

“open” artwork and icons by chandan:
<http://blogs.sun.com/chandan>

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
•••••
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை