Specifying the credentials for a spawned process#

Reference#

The following link serves as a reference for the requested functionality. CRHB0134

Background#

A running process has 2 sets of credentials of importance during its execution, the **real** user and group ids (*ruid* and *rgid* respectively) and the **effective** user and group ids (*euid* and *egid* respectively) . When a new (child) process is spawned, normally both sets of credentials (real and effective) are taken from the spawning (parent) process. The exception to this is if the executable file being spawned has either its *setuid* and/or *setgid* mode bits set in which case the *euid* will be set to the *uid* of the executable and/or the *egid* will be set to the *gid* of the executable. It is not possible to directly specify the *ruid* and *rgid* of a spawned process. Various "tricks" have been employed to attempt to work around this issue but none offer a well defined and secure approach.

In order to provide the requested flexibility of being able to set the **real** user and group ids of a spawned process, we have extended the functionality of *posix_spawn()* and specifically the *posix_spawnattr_xxx()* family of functions to allow for the explicit specification of a spawned (child) process' **ruid** and **rgid**.

Design/Implementation Review Meeting#

This section is concerned with the design decisions as they relate to the customer facing API as this is ultimately what the customer is exposed to and must interact with.

A meeting was held the week of Dec 1st, 2008 to discuss the requirements and to propose an implementation.

Attendees inlcuded:

- Alain Magloire - Manager/Customer prime
- Joe Mammone - software developer/customer prime
- Colin Burgess - software developer
- Mike Kisel - software developer

The basic requirements were discussed. It was indicated by Burgess/Kisel that it was not possible to modify the existing *spawn()* interface that the customer was using as it would break backwards compatibility. The reason for this is because the data structure which conveys the user request for a spawned process' attributes is not transparent to the user and therefore modifying this structure is not possible in an extensible and backwards compatible fashion. This exposed structure is also used internally within procnto, hence it is not possible to change without compatibility issues arising.

The proposed solution is to use the newly introduced *posix_spawn()* family of calls as this standard POSIX interface was designed to facilitate extensibility and such implementation specific extensions. By adding 2 new attribute manipulation functions, one to set and one to get the desired credentials, the desired functionality could be easily added to our existing *posix_spawnxxxx()* family of routines.

Because the *posix_spawn()* interface is different than the *spawn()* interface currently in use by the customer, contact with the customer was made in order to confirm that the required application changes were acceptable. Later that week we (via Joe Mammone) received notice (via email) that the customer had agreed to our proposed use of *posix_spawn()* as the change does not fundamentally alter any of the process spawning semantics.

The design requirements were finalized in favour of the use of *posix_spawn()*.

The next sections discuss some additional details on the design/implementation of the API as well as the internal additions required to support the added functionality.

Design#

The ability to support the establishment of credentials for a newly created process consists of 2 main areas of work. The first is the application API which allows the specification of the credentials. This API is implemented entirely within our 'C' library. The second is the ability to act upon such credentials within the kernel when the process is actually created.

'C' Library#

There is not much real design effort associated with this additional feature as the primary design is embodied in the *posix_spawnattr_xxx()* family of functions, which defines an extensible interface for setting and getting an attributes object and *posix_spawn()*, which accepts as one of its parameters, said attributes object from which to establish the conditions under which a newly spawned process is to execute. Since QNX introduced support for *posix_spawn()* in the previous 6.4.0 release, we are able to simply build upon the extensibility defined by POSIX in our own implementation.

procnto/kernel#

procnto is the system process which handles user requests (messages) and specifically requests to create new processes. "procnto" eventually makes use of a kernel service (*kerext_process_create()*) to effect the creation once most of the required creation attributes have been established and validated. One of the parameters to *kerext_process_create()* is a *proc_create_attr_t* which allows for the optional specification of various additional attributes to apply to process creation. This type has been extended to include the credentials to be established for the new process. "procnto" will receive the user message to create a new process (created with the *posix_spawn()* call) and translate the provided attributes into parameters and an environment required for the *kerext_process_create()* call. If the caller has provided a desired set of credentials for the new process, the optional field of the *proc_create_attr_t* related to credentials is filled in and will be applied during the process creation kernel call. If the caller does not provide a set of desired credentials, the credentials portion of the *proc_create_attr_t* type is left empty and the default inheritance will be used during the *kerext_process_create()* call.

The remainder of this document will then describe the implementation.

Implementation#

'C' Library#

One flag and two new attribute setting/getting functions have been added to the 'C' library, specifically in lib/c/1d.

- POSIX_SPAWN_SETCRED

This flag has been defined to cause the establishment of desired credentials specified in the *posix_spawnattr_t* attributes object to be acted upon. Without this flag, any *uid*/*gid* specified with *posix_spawnattr_setcred()* (below) will be ignored. This flag can be set with the *posix_spawnattr_setxflags()* call.

- *posix_spawnattr_setcred()*

allows the caller to specify both a user id (uid) and group id (gid) to assign to the *posix_spawnattr_t* attributes object. The caller must provide both parameters and can effectively leave one of them unchanged by passing the results of *getuid()* or *getgid()* respectively. In order for the *uid*/*gid* to take effect, the *posix_spawnattr_t* attributes object must be passed as a parameter to the *posix_spawn()* call, the **POSIX_SPAWN_SETCRED** flag must be set and the caller must have an **effective** user id of 0 (ie. root). Alternatively, if the caller provides a *uid* of *getuid()* and a *gid* of *getgid()*, the caller is not required to have *root* permissions since this is equivalent to the default inheritance behaviour.

- *posix_spawnattr_getcred()*

allows the caller to retrieve the *uid* and *gid* of the *posix_spawnattr_t* attributes object set in a previous *posix_spawnattr_setcred()* call

procnto/kernel#

The existing *proc_create_attr_t* type has been extended to include the real and effective user and group ids. The changes to procnto/kernel have been limited to 3 main files

- public/ker/objects.h - contains the definition of the *proc_create_attr_t* type. This type has been extended to include the *ruid*/*rgid* and *euid*/*egid* fields
- ker/kerext_process.c - *kerext_process_create()* has been modified to check for a non-NULL credentials field in the *proc_create_attr_t* parameter and if so use it to establish the credentials of the newly created process
- procmgr/procmgr_posix_spawn.c - contains the bulk of the changes and that consists primarily of one new function, some initializations and a variable name change. The function, *fill_struct_cred()*, will, if the POSIX_SPAWN_SETCRED flag is set, extract the specified credentials from the attributes object and fill in the *proc_create_attr_t* in preparation for the *kerext_process_create()* call.

CI and Submissions#

The CI posts related to the changes can be found at http://community.qnx.com/sf/discussion/do/listPosts/projects.core_os/discussion.osrev.topc6144
The submission for the 'C' Library changes can be found at http://websvn.ott.qnx.com/redir.cgi?repository=product&revision=213978
The submission for the procnto/kernel changes can be found at http://websvn.ott.qnx.com/redir.cgi?repository=product&revision=213977