NFS ID Mapper

Id mapper is used by NFS to translate user and group ids into names, and to translate user and group names into ids. Part of this translation involves performing an upcall to userspace to request the information. There are two ways NFS could obtain this information: placing a call to /sbin/request-key or by placing a call to the rpc.idmap daemon.

NFS will attempt to call /sbin/request-key first. If this succeeds, the result will be cached using the generic request-key cache. This call should only fail if /etc/request-key.conf is not configured for the id_resolver key type, see the "Configuring" section below if you wish to use the request-key method.

If the call to /sbin/request-key fails (if /etc/request-key.conf is not configured with the id_resolver key type), then the idmapper will ask the legacy rpc.idmap daemon for the id mapping. This result will be stored in a custom NFS idmap cache.

Configuring

The file /etc/request-key.conf will need to be modified so /sbin/request-key can direct the upcall. The following line should be added:

```
#OP   TYPE    DESCRIPTION    CALLOUT INFO    PROGRAM ARG1 ARG2 ARG3 ... #======        =======
=============== =============== =============================== create        id_resolver     *
*              /usr/sbin/nfs.idmap %k %d 600
```

This will direct all id_resolver requests to the program /usr/sbin/nfs.idmap. The last parameter, 600, defines how many seconds into the future the key will expire. This parameter is optional for /usr/sbin/nfs.idmap. When the timeout is not specified, nfs.idmap will default to 600 seconds.

id mapper uses for key descriptions:

```
     uid:  Find the UID for the given user
     gid:  Find the GID for the given group
    user:  Find the user  name for the given UID
   group:  Find the group name for the given GID
```

You can handle any of these individually, rather than using the generic upcall program. If you would like to use your own program for a uid lookup then you would edit your request-key.conf so it look similar to this:

```
#OP   TYPE    DESCRIPTION    CALLOUT INFO    PROGRAM ARG1 ARG2 ARG3 ... #======        =======
=============== =============== =============================== create        id_resolver     uid:*
*              /some/other/program %k %d 600 create        id_resolver     *        *
/usr/sbin/nfs.idmap %k %d 600
```

Notice that the new line was added above the line for the generic program. request-key will find the first matching line and corresponding program. In this case, /some/other/program will handle all uid lookups and /usr/sbin/nfs.idmap will handle gid, user, and group lookups.

See Documentation/security/keys/request-key.rst for more information about the request-key function.

nfs.idmap

nfs.idmap is designed to be called by request-key, and should not be run "by hand". This program takes two arguments, a serialized key and a key description. The serialized key is first converted into a key_serial_t, and then passed as an argument to keyctl_instantiate (both are part of keyutils.h).

The actual lookups are performed by functions found in nfsidmap.h. nfs.idmap determines the correct function to call by looking at the first part of the description string. For example, a uid lookup description will appear as "uid:user@domain".

nfs.idmap will return 0 if the key was instantiated, and non-zero otherwise.