Subject: [PATCH] eCryptfs: Make key module subsystem respect namespaces
Posted by Michael Halcrow on Wed, 16 Apr 2008 19:24:17 GMT
View Forum Message <> Reply to Message

On Tue, Apr 15, 2008 at 04:34:02PM -0500, Serge E. Hallyn wrote:
> Quoting Andrew Morton (akpm@linux-foundation.org):
> > On Tue, 15 Apr 2008 15:23:13 -0500
> > Michael Halcrow <mhalcrow@us.ibm.com> wrote:
> >
> > > ...
> > > + rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid);
> > > + if (daemon->pid != current->pid) {
> > > + rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid);
> > > + BUG_ON(current->euid != daemon->euid);
> > > + BUG_ON(current->pid != daemon->pid);
> >
> > This code uses pids and uids all over the place. Will it operate
> > correctly in a containerised environment?
>
> Thanks Andrew.
>
> Mike, the pid_t definately needs to be replaced with a struct pid.
>
> As for the euid, it'd be best if you also compared the user_namespace *
> to make sure we support one ecryptfs deamon per user namespace.

Make eCryptfs key module subsystem respect namespaces.

Since I will be removing the netlink interface in a future patch, I
just made changes to the netlink.c code so that it will not break the
build. With my recent patches, the kernel module currently defaults to
the device handle interface rather than the netlink interface.

Signed-off-by: Michael Halcrow <mhalcrow@us.ibm.com>

```
---
 fs/ecryptfs/ecryptfs_kernel.h |   25 +++++++++-----
 fs/ecryptfs/messaging.c       |   71 ++++++++++++++++++++++++++++++-----------
 fs/ecryptfs/miscdev.c         |   68 +++++++++++++++++++++++++--------------
 fs/ecryptfs/netlink.c         |   25 +++++++++-----
 4 files changed, 126 insertions(+), 63 deletions(-)

diff --git a/fs/ecryptfs/ecryptfs_kernel.h b/fs/ecryptfs/ecryptfs_kernel.h
index 88b85f7..dc11431 100644
--- a/fs/ecryptfs/ecryptfs_kernel.h
+++ b/fs/ecryptfs/ecryptfs_kernel.h
@@ -34,6 +34,7 @@
 #include <linux/namei.h>
 #include <linux/scatterlist.h>
 #include <linux/hash.h>
+#include <linux/nsproxy.h>

 /* Version verification for shared data structures w/ userspace */
 #define ECRYPTFS_VERSION_MAJOR 0x00
@@ -410,8 +411,9 @@ struct ecryptfs_daemon {
 #define ECRYPTFS_DAEMON_MISCDEV_OPEN 0x00000008
  u32 flags;
  u32 num_queued_msg_ctx;
- pid_t pid;
+ struct pid *pid;
  uid_t euid;
+ struct user_namespace *user_ns;
  struct task_struct *task;
  struct mutex mux;
  struct list_head msg_ctx_out_queue;
@@ -610,10 +612,13 @@ int
 ecryptfs_setxattr(struct dentry *dentry, const char *name, const void *value,
     size_t size, int flags);
 int ecryptfs_read_xattr_region(char *page_virt, struct inode *ecryptfs_inode);
-int ecryptfs_process_helo(unsigned int transport, uid_t uid, pid_t pid);
-int ecryptfs_process_quit(uid_t uid, pid_t pid);
-int ecryptfs_process_response(struct ecryptfs_message *msg, uid_t uid,
-        pid_t pid, u32 seq);
+int ecryptfs_process_helo(unsigned int transport, uid_t euid,
+     struct user_namespace *user_ns, struct pid *pid);
+int ecryptfs_process_quit(uid_t euid, struct user_namespace *user_ns,
+     struct pid *pid);
+int ecryptfs_process_response(struct ecryptfs_message *msg, uid_t euid,
+        struct user_namespace *user_ns, struct pid *pid,
+        u32 seq);
 int ecryptfs_send_message(unsigned int transport, char *data, int data_len,
      struct ecryptfs_msg_ctx **msg_ctx);
 int ecryptfs_wait_for_response(struct ecryptfs_msg_ctx *msg_ctx,
@@ -623,13 +628,13 @@ void ecryptfs_release_messaging(unsigned int transport);

 int ecryptfs_send_netlink(char *data, int data_len,
      struct ecryptfs_msg_ctx *msg_ctx, u8 msg_type,
-     u16 msg_flags, pid_t daemon_pid);
+     u16 msg_flags, struct pid *daemon_pid);
 int ecryptfs_init_netlink(void);
 void ecryptfs_release_netlink(void);

 int ecryptfs_send_connector(char *data, int data_len,
       struct ecryptfs_msg_ctx *msg_ctx, u8 msg_type,
-      u16 msg_flags, pid_t daemon_pid);
+      u16 msg_flags, struct pid *daemon_pid);
 int ecryptfs_init_connector(void);
 void ecryptfs_release_connector(void);
 void
@@ -672,7 +677,8 @@ int ecryptfs_read_lower_page_segment(struct page *page_for_ecryptfs,
         struct inode *ecryptfs_inode);
 struct page *ecryptfs_get_locked_page(struct file *file, loff_t index);
 int ecryptfs_exorcise_daemon(struct ecryptfs_daemon *daemon);
-int ecryptfs_find_daemon_by_euid(struct ecryptfs_daemon **daemon, uid_t euid);
+int ecryptfs_find_daemon_by_euid(struct ecryptfs_daemon **daemon, uid_t euid,
+     struct user_namespace *user_ns);
 int ecryptfs_parse_packet_length(unsigned char *data, size_t *size,
      size_t *length_size);
 int ecryptfs_write_packet_length(char *dest, size_t size,
@@ -684,6 +690,7 @@ int ecryptfs_send_miscdev(char *data, size_t data_size,
      u16 msg_flags, struct ecryptfs_daemon *daemon);
 void ecryptfs_msg_ctx_alloc_to_free(struct ecryptfs_msg_ctx *msg_ctx);
 int
-ecryptfs_spawn_daemon(struct ecryptfs_daemon **daemon, uid_t euid, pid_t pid);
+ecryptfs_spawn_daemon(struct ecryptfs_daemon **daemon, uid_t euid,
+        struct user_namespace *user_ns, struct pid *pid);

 #endif /* #ifndef ECRYPTFS_KERNEL_H */
diff --git a/fs/ecryptfs/messaging.c b/fs/ecryptfs/messaging.c
index e3f2e97..fad161b 100644
--- a/fs/ecryptfs/messaging.c
+++ b/fs/ecryptfs/messaging.c
@@ -103,6 +103,7 @@ void ecryptfs_msg_ctx_alloc_to_free(struct ecryptfs_msg_ctx *msg_ctx)
 /**
  * ecryptfs_find_daemon_by_euid
  * @euid: The effective user id which maps to the desired daemon id
+ * @user_ns: The namespace in which @euid applies
  * @daemon: If return value is zero, points to the desired daemon pointer
  *
  * Must be called with ecryptfs_daemon_hash_mux held.
@@ -111,7 +112,8 @@ void ecryptfs_msg_ctx_alloc_to_free(struct ecryptfs_msg_ctx *msg_ctx)
  *
  * Returns zero if the user id exists in the list; non-zero otherwise.
  */
-int ecryptfs_find_daemon_by_euid(struct ecryptfs_daemon **daemon, uid_t euid)
+int ecryptfs_find_daemon_by_euid(struct ecryptfs_daemon **daemon, uid_t euid,
+     struct user_namespace *user_ns)
 {
  struct hlist_node *elem;
  int rc;
@@ -119,7 +121,7 @@ int ecryptfs_find_daemon_by_euid(struct ecryptfs_daemon **daemon,
uid_t euid)
  hlist_for_each_entry(*daemon, elem,
        &ecryptfs_daemon_hash[ecryptfs_uid_hash(euid)],
        euid_chain) {
```

```
-  if ((*daemon)->euid == euid) {
+  if ((*daemon)->euid == euid && (*daemon)->user_ns == user_ns) {
    rc = 0;
    goto out;
   }
@@ -186,6 +188,7 @@ out:
  * ecryptfs_spawn_daemon - Create and initialize a new daemon struct
  * @daemon: Pointer to set to newly allocated daemon struct
  * @euid: Effective user id for the daemon
+ * @user_ns: The namespace in which @euid applies
  * @pid: Process id for the daemon
  *
  * Must be called ceremoniously while in possession of
@@ -194,7 +197,8 @@ out:
  * Returns zero on success; non-zero otherwise
  */
 int
-ecryptfs_spawn_daemon(struct ecryptfs_daemon **daemon, uid_t euid, pid_t pid)
+ecryptfs_spawn_daemon(struct ecryptfs_daemon **daemon, uid_t euid,
+       struct user_namespace *user_ns, struct pid *pid)
 {
  int rc = 0;

@@ -206,6 +210,7 @@ ecryptfs_spawn_daemon(struct ecryptfs_daemon **daemon, uid_t euid,
pid_t pid)
   goto out;
  }
  (*daemon)->euid = euid;
+ (*daemon)->user_ns = user_ns;
  (*daemon)->pid = pid;
  (*daemon)->task = current;
  mutex_init(&(*daemon)->mux);
@@ -222,6 +227,7 @@ out:
  * ecryptfs_process_helo
  * @transport: The underlying transport (netlink, etc.)
  * @euid: The user ID owner of the message
+ * @user_ns: The namespace in which @euid applies
  * @pid: The process ID for the userspace program that sent the
  *      message
  *
@@ -231,32 +237,33 @@ out:
  * Returns zero after adding a new daemon to the hash list;
  * non-zero otherwise.
  */
-int ecryptfs_process_helo(unsigned int transport, uid_t euid, pid_t pid)
+int ecryptfs_process_helo(unsigned int transport, uid_t euid,
+     struct user_namespace *user_ns, struct pid *pid)
 {
  struct ecryptfs_daemon *new_daemon;
  struct ecryptfs_daemon *old_daemon;
  int rc;

  mutex_lock(&ecryptfs_daemon_hash_mux);
- rc = ecryptfs_find_daemon_by_euid(&old_daemon, euid);
+ rc = ecryptfs_find_daemon_by_euid(&old_daemon, euid, user_ns);
  if (rc != 0) {
   printk(KERN_WARNING "Received request from user [%d] "
-        "to register daemon [%d]; unregistering daemon "
-        "[%d]\n", euid, pid, old_daemon->pid);
+        "to register daemon [0x%p]; unregistering daemon "
+        "[0x%p]\n", euid, pid, old_daemon->pid);
   rc = ecryptfs_send_raw_message(transport, ECRYPTFS_MSG_QUIT,
             old_daemon);
   if (rc)
    printk(KERN_WARNING "Failed to send QUIT "
-         "message to daemon [%d]; rc = [%d]\n",
+         "message to daemon [0x%p]; rc = [%d]\n",
          old_daemon->pid, rc);
   hlist_del(&old_daemon->euid_chain);
   kfree(old_daemon);
  }
- rc = ecryptfs_spawn_daemon(&new_daemon, euid, pid);
+ rc = ecryptfs_spawn_daemon(&new_daemon, euid, user_ns, pid);
  if (rc)
   printk(KERN_ERR "%s: The gods are displeased with this attempt "
-        "to create a new daemon object for euid [%d]; pid [%d]; "
-        "rc = [%d]\n", __func__, euid, pid, rc);
+        "to create a new daemon object for euid [%d]; pid "
+        "[0x%p]; rc = [%d]\n", __func__, euid, pid, rc);
  mutex_unlock(&ecryptfs_daemon_hash_mux);
  return rc;
 }
@@ -277,7 +284,7 @@ int ecryptfs_exorcise_daemon(struct ecryptfs_daemon *daemon)
     || (daemon->flags & ECRYPTFS_DAEMON_IN_POLL)) {
   rc = -EBUSY;
   printk(KERN_WARNING "%s: Attempt to destroy daemon with pid "
-        "[%d], but it is in the midst of a read or a poll\n",
+        "[0x%p], but it is in the midst of a read or a poll\n",
         __func__, daemon->pid);
   mutex_unlock(&daemon->mux);
   goto out;
@@ -303,6 +310,7 @@ out:
 /**
  * ecryptfs_process_quit
  * @euid: The user ID owner of the message
+ * @user_ns: The namespace in which @euid applies
```

```
  * @pid: The process ID for the userspace program that sent the
  *       message
  *
@@ -310,17 +318,18 @@ out:
  * it is the registered that is requesting the deletion. Returns zero
  * after deleting the desired daemon; non-zero otherwise.
  */
-int ecryptfs_process_quit(uid_t euid, pid_t pid)
+int ecryptfs_process_quit(uid_t euid, struct user_namespace *user_ns,
+     struct pid *pid)
 {
  struct ecryptfs_daemon *daemon;
  int rc;

  mutex_lock(&ecryptfs_daemon_hash_mux);
- rc = ecryptfs_find_daemon_by_euid(&daemon, euid);
+ rc = ecryptfs_find_daemon_by_euid(&daemon, euid, user_ns);
  if (rc || !daemon) {
   rc = -EINVAL;
   printk(KERN_ERR "Received request from user [%d] to "
-        "unregister unrecognized daemon [%d]\n", euid, pid);
+         "unregister unrecognized daemon [0x%p]\n", euid, pid);
   goto out_unlock;
  }
  rc = ecryptfs_exorcise_daemon(daemon);
@@ -354,11 +363,13 @@ out_unlock:
  * Returns zero on success; non-zero otherwise
  */
 int ecryptfs_process_response(struct ecryptfs_message *msg, uid_t euid,
-        pid_t pid, u32 seq)
+         struct user_namespace *user_ns, struct pid *pid,
+         u32 seq)
 {
  struct ecryptfs_daemon *daemon;
  struct ecryptfs_msg_ctx *msg_ctx;
  size_t msg_size;
+ struct nsproxy *nsproxy;
  int rc;

  if (msg->index >= ecryptfs_message_buf_len) {
@@ -372,12 +383,24 @@ int ecryptfs_process_response(struct ecryptfs_message *msg, uid_t
euid,
  msg_ctx = &ecryptfs_msg_ctx_arr[msg->index];
  mutex_lock(&msg_ctx->mux);
  mutex_lock(&ecryptfs_daemon_hash_mux);
- rc = ecryptfs_find_daemon_by_euid(&daemon, msg_ctx->task->euid);
+ rcu_read_lock();
+ nsproxy = task_nsproxy(msg_ctx->task);
+ if (nsproxy == NULL) {
+  rc = -EBADMSG;
+  printk(KERN_ERR "%s: Receiving process is a zombie. Dropping "
+         "message.\n", __func__);
+  rcu_read_unlock();
+  mutex_unlock(&ecryptfs_daemon_hash_mux);
+  goto wake_up;
+ }
+ rc = ecryptfs_find_daemon_by_euid(&daemon, msg_ctx->task->euid,
+       nsproxy->user_ns);
+ rcu_read_unlock();
  mutex_unlock(&ecryptfs_daemon_hash_mux);
  if (rc) {
   rc = -EBADMSG;
   printk(KERN_WARNING "%s: User [%d] received a "
-         "message response from process [%d] but does "
+         "message response from process [0x%p] but does "
         "not have a registered daemon\n", __func__,
         msg_ctx->task->euid, pid);
   goto wake_up;
@@ -389,10 +412,17 @@ int ecryptfs_process_response(struct ecryptfs_message *msg, uid_t
euid,
         euid, msg_ctx->task->euid);
   goto unlock;
  }
+ if (nsproxy->user_ns != user_ns) {
+  rc = -EBADMSG;
+  printk(KERN_WARNING "%s: Received message from user_ns "
+         "[0x%p]; expected message from user_ns [0x%p]\n",
+         __func__, user_ns, nsproxy->user_ns);
+  goto unlock;
+ }
  if (daemon->pid != pid) {
   rc = -EBADMSG;
   printk(KERN_ERR "%s: User [%d] sent a message response "
-         "from an unrecognized process [%d]\n",
+         "from an unrecognized process [0x%p]\n",
         __func__, msg_ctx->task->euid, pid);
   goto unlock;
  }
@@ -446,7 +476,8 @@ ecryptfs_send_message_locked(unsigned int transport, char *data, int
data_len,
  struct ecryptfs_daemon *daemon;
  int rc;

- rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid);
+ rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid,
+       current->nsproxy->user_ns);
```

```
  if (rc || !daemon) {
   rc = -ENOTCONN;
   printk(KERN_ERR "%s: User [%d] does not have a daemon "
diff --git a/fs/ecryptfs/miscdev.c b/fs/ecryptfs/miscdev.c
index f7f2d90..8d39a04 100644
--- a/fs/ecryptfs/miscdev.c
+++ b/fs/ecryptfs/miscdev.c
@@ -46,7 +46,8 @@ ecryptfs_miscdev_poll(struct file *file, poll_table *pt)

  mutex_lock(&ecryptfs_daemon_hash_mux);
  /* TODO: Just use file->private_data? */
- rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid);
+ rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid,
+      current->nsproxy->user_ns);
  BUG_ON(rc || !daemon);
  mutex_lock(&daemon->mux);
  mutex_unlock(&ecryptfs_daemon_hash_mux);
@@ -92,10 +93,12 @@ ecryptfs_miscdev_open(struct inode *inode, struct file *file)
         "count; rc = [%d]\n", __func__, rc);
   goto out_unlock_daemon_list;
  }
- rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid);
+ rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid,
+      current->nsproxy->user_ns);
  if (rc || !daemon) {
   rc = ecryptfs_spawn_daemon(&daemon, current->euid,
-       current->pid);
+        current->nsproxy->user_ns,
+        task_pid(current));
   if (rc) {
    printk(KERN_ERR "%s: Error attempting to spawn daemon; "
          "rc = [%d]\n", __func__, rc);
@@ -103,18 +106,18 @@ ecryptfs_miscdev_open(struct inode *inode, struct file *file)
   }
  }
  mutex_lock(&daemon->mux);
- if (daemon->pid != current->pid) {
+ if (daemon->pid != task_pid(current)) {
   rc = -EINVAL;
-  printk(KERN_ERR "%s: pid [%d] has registered with euid [%d], "
-        "but pid [%d] has attempted to open the handle "
+  printk(KERN_ERR "%s: pid [0x%p] has registered with euid [%d], "
+         "but pid [0x%p] has attempted to open the handle "
         "instead\n", __func__, daemon->pid, daemon->euid,
-        current->pid);
+         task_pid(current));
   goto out_unlock_daemon;
  }
  if (daemon->flags & ECRYPTFS_DAEMON_MISCDEV_OPEN) {
   rc = -EBUSY;
   printk(KERN_ERR "%s: Miscellaneous device handle may only be "
-         "opened once per daemon; pid [%d] already has this "
+          "opened once per daemon; pid [0x%p] already has this "
          "handle open\n", __func__, daemon->pid);
   goto out_unlock_daemon;
  }
@@ -147,10 +150,11 @@ ecryptfs_miscdev_release(struct inode *inode, struct file *file)
  int rc;

  mutex_lock(&ecryptfs_daemon_hash_mux);
- rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid);
+ rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid,
+       current->nsproxy->user_ns);
  BUG_ON(rc || !daemon);
  mutex_lock(&daemon->mux);
- BUG_ON(daemon->pid != current->pid);
+ BUG_ON(daemon->pid != task_pid(current));
  BUG_ON(!(daemon->flags & ECRYPTFS_DAEMON_MISCDEV_OPEN));
  daemon->flags &= ~ECRYPTFS_DAEMON_MISCDEV_OPEN;
  atomic_dec(&ecryptfs_num_miscdev_opens);
@@ -247,7 +251,8 @@ ecryptfs_miscdev_read(struct file *file, char __user *buf, size_t count,

  mutex_lock(&ecryptfs_daemon_hash_mux);
  /* TODO: Just use file->private_data? */
- rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid);
+ rc = ecryptfs_find_daemon_by_euid(&daemon, current->euid,
+       current->nsproxy->user_ns);
  BUG_ON(rc || !daemon);
  mutex_lock(&daemon->mux);
  if (daemon->flags & ECRYPTFS_DAEMON_ZOMBIE) {
@@ -285,7 +290,8 @@ check_list:
   goto check_list;
  }
  BUG_ON(current->euid != daemon->euid);
- BUG_ON(current->pid != daemon->pid);
+ BUG_ON(current->nsproxy->user_ns != daemon->user_ns);
+ BUG_ON(task_pid(current) != daemon->pid);
  msg_ctx = list_first_entry(&daemon->msg_ctx_out_queue,
        struct ecryptfs_msg_ctx, daemon_out_list);
  BUG_ON(!msg_ctx);
@@ -355,15 +361,18 @@ out_unlock_daemon:
 /**
  * ecryptfs_miscdev_helo
  * @euid: effective user id of miscdevess sending helo packet
+ * @user_ns: The namespace in which @euid applies
  * @pid: miscdevess id of miscdevess sending helo packet
  *
  * Returns zero on success; non-zero otherwise
  */
-static int ecryptfs_miscdev_helo(uid_t uid, pid_t pid)
+static int ecryptfs_miscdev_helo(uid_t euid, struct user_namespace *user_ns,
+     struct pid *pid)
 {
  int rc;

- rc = ecryptfs_process_helo(ECRYPTFS_TRANSPORT_MISCDEV, uid, pid);
+ rc = ecryptfs_process_helo(ECRYPTFS_TRANSPORT_MISCDEV, euid, user_ns,
+       pid);
  if (rc)
   printk(KERN_WARNING "Error processing HELO; rc = [%d]\n", rc);
  return rc;
@@ -372,15 +381,17 @@ static int ecryptfs_miscdev_helo(uid_t uid, pid_t pid)
 /**
  * ecryptfs_miscdev_quit
  * @euid: effective user id of miscdevess sending quit packet
+ * @user_ns: The namespace in which @euid applies
  * @pid: miscdevess id of miscdevess sending quit packet
  *
  * Returns zero on success; non-zero otherwise
  */
-static int ecryptfs_miscdev_quit(uid_t euid, pid_t pid)
+static int ecryptfs_miscdev_quit(uid_t euid, struct user_namespace *user_ns,
+     struct pid *pid)
 {
  int rc;

- rc = ecryptfs_process_quit(euid, pid);
+ rc = ecryptfs_process_quit(euid, user_ns, pid);
  if (rc)
   printk(KERN_WARNING
          "Error processing QUIT message; rc = [%d]\n", rc);
@@ -392,13 +403,15 @@ static int ecryptfs_miscdev_quit(uid_t euid, pid_t pid)
  * @data: Bytes comprising struct ecryptfs_message
  * @data_size: sizeof(struct ecryptfs_message) + data len
  * @euid: Effective user id of miscdevess sending the miscdev response
+ * @user_ns: The namespace in which @euid applies
  * @pid: Miscdevess id of miscdevess sending the miscdev response
  * @seq: Sequence number for miscdev response packet
  *
  * Returns zero on success; non-zero otherwise
  */
 static int ecryptfs_miscdev_response(char *data, size_t data_size,
-       uid_t euid, pid_t pid, u32 seq)
+         uid_t euid, struct user_namespace *user_ns,
+        struct pid *pid, u32 seq)
 {
  struct ecryptfs_message *msg = (struct ecryptfs_message *)data;
  int rc;
@@ -410,7 +423,7 @@ static int ecryptfs_miscdev_response(char *data, size_t data_size,
   rc = -EINVAL;
   goto out;
  }
- rc = ecryptfs_process_response(msg, euid, pid, seq);
+ rc = ecryptfs_process_response(msg, euid, user_ns, pid, seq);
  if (rc)
   printk(KERN_ERR
          "Error processing response message; rc = [%d]\n", rc);
@@ -491,27 +504,32 @@ ecryptfs_miscdev_write(struct file *file, const char __user *buf,
   }
   rc = ecryptfs_miscdev_response(&data[i], packet_size,
             current->euid,
-           current->pid, seq);
+            current->nsproxy->user_ns,
+            task_pid(current), seq);
   if (rc)
    printk(KERN_WARNING "%s: Failed to deliver miscdev "
           "response to requesting operation; rc = [%d]\n",
           __func__, rc);
   break;
  case ECRYPTFS_MSG_HELO:
-  rc = ecryptfs_miscdev_helo(current->euid, current->pid);
+  rc = ecryptfs_miscdev_helo(current->euid,
+       current->nsproxy->user_ns,
+       task_pid(current));
   if (rc) {
    printk(KERN_ERR "%s: Error attempting to process "
-         "helo from pid [%d]; rc = [%d]\n", __func__,
-         current->pid, rc);
+          "helo from pid [0x%p]; rc = [%d]\n", __func__,
+          task_pid(current), rc);
    goto out_free;
   }
   break;
  case ECRYPTFS_MSG_QUIT:
-  rc = ecryptfs_miscdev_quit(current->euid, current->pid);
+  rc = ecryptfs_miscdev_quit(current->euid,
+       current->nsproxy->user_ns,
+       task_pid(current));
   if (rc) {
    printk(KERN_ERR "%s: Error attempting to process "
-         "quit from pid [%d]; rc = [%d]\n", __func__,
-         current->pid, rc);
```

```
+         "quit from pid [0x%p]; rc = [%d]\n", __func__,
+         task_pid(current), rc);
    goto out_free;
   }
   break;
diff --git a/fs/ecryptfs/netlink.c b/fs/ecryptfs/netlink.c
index eb70f69..e0abad6 100644
--- a/fs/ecryptfs/netlink.c
+++ b/fs/ecryptfs/netlink.c
@@ -45,7 +45,7 @@ static struct sock *ecryptfs_nl_sock;
  */
 int ecryptfs_send_netlink(char *data, int data_len,
      struct ecryptfs_msg_ctx *msg_ctx, u8 msg_type,
-     u16 msg_flags, pid_t daemon_pid)
+     u16 msg_flags, struct pid *daemon_pid)
 {
  struct sk_buff *skb;
  struct nlmsghdr *nlh;
@@ -60,7 +60,7 @@ int ecryptfs_send_netlink(char *data, int data_len,
   ecryptfs_printk(KERN_ERR, "Failed to allocate socket buffer\n");
   goto out;
  }
- nlh = NLMSG_PUT(skb, daemon_pid, msg_ctx ? msg_ctx->counter : 0,
+ nlh = NLMSG_PUT(skb, pid_nr(daemon_pid), msg_ctx ? msg_ctx->counter : 0,
    msg_type, payload_len);
  nlh->nlmsg_flags = msg_flags;
  if (msg_ctx && payload_len) {
@@ -69,7 +69,7 @@ int ecryptfs_send_netlink(char *data, int data_len,
   msg->data_len = data_len;
   memcpy(msg->data, data, data_len);
  }
- rc = netlink_unicast(ecryptfs_nl_sock, skb, daemon_pid, 0);
+ rc = netlink_unicast(ecryptfs_nl_sock, skb, pid_nr(daemon_pid), 0);
  if (rc < 0) {
   ecryptfs_printk(KERN_ERR, "Failed to send eCryptfs netlink "
     "message; rc = [%d]\n", rc);
@@ -99,6 +99,7 @@ static int ecryptfs_process_nl_response(struct sk_buff *skb)
 {
  struct nlmsghdr *nlh = nlmsg_hdr(skb);
  struct ecryptfs_message *msg = NLMSG_DATA(nlh);
+ struct pid *pid;
  int rc;

  if (skb->len - NLMSG_HDRLEN - sizeof(*msg) != msg->data_len) {
@@ -107,8 +108,10 @@ static int ecryptfs_process_nl_response(struct sk_buff *skb)
     "incorrectly specified data length\n");
   goto out;
  }
```

```
- rc = ecryptfs_process_response(msg, NETLINK_CREDS(skb)->uid,
-          NETLINK_CREDS(skb)->pid, nlh->nlmsg_seq);
+ pid = find_get_pid(NETLINK_CREDS(skb)->pid);
+ rc = ecryptfs_process_response(msg, NETLINK_CREDS(skb)->uid, NULL,
+          pid, nlh->nlmsg_seq);
+ put_pid(pid);
  if (rc)
   printk(KERN_ERR
         "Error processing response message; rc = [%d]\n", rc);
@@ -126,11 +129,13 @@ out:
  */
 static int ecryptfs_process_nl_helo(struct sk_buff *skb)
 {
+ struct pid *pid;
  int rc;

+ pid = find_get_pid(NETLINK_CREDS(skb)->pid);
  rc = ecryptfs_process_helo(ECRYPTFS_TRANSPORT_NETLINK,
-       NETLINK_CREDS(skb)->uid,
-       NETLINK_CREDS(skb)->pid);
+        NETLINK_CREDS(skb)->uid, NULL, pid);
+ put_pid(pid);
  if (rc)
   printk(KERN_WARNING "Error processing HELO; rc = [%d]\n", rc);
  return rc;
@@ -147,10 +152,12 @@ static int ecryptfs_process_nl_helo(struct sk_buff *skb)
  */
 static int ecryptfs_process_nl_quit(struct sk_buff *skb)
 {
+ struct pid *pid;
  int rc;

- rc = ecryptfs_process_quit(NETLINK_CREDS(skb)->uid,
-       NETLINK_CREDS(skb)->pid);
+ pid = find_get_pid(NETLINK_CREDS(skb)->pid);
+ rc = ecryptfs_process_quit(NETLINK_CREDS(skb)->uid, NULL, pid);
+ put_pid(pid);
  if (rc)
   printk(KERN_WARNING
         "Error processing QUIT message; rc = [%d]\n", rc);
--
1.5.3.6
```

____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers