Subject: Re: [PATCH 05/10] Containers(V10): Add container_clone() interface
Posted by Andrew Morton on Wed, 30 May 2007 07:16:10 GMT
View Forum Message <> Reply to Message

On Tue, 29 May 2007 06:01:09 -0700 menage@google.com wrote:

> This patch adds support for container_clone(), a speculative interface
> to creating new containers intended to be used for systems such as
> namespace unsharing.
>
> ...
>
> +
> +static atomic_t namecnt;
> +static void get_unused_name(char *buf)
> +{
> + sprintf(buf, "node%d", atomic_inc_return(&namecnt));
> +}

A stupid thing, but a sufficiently determined attacker could cause this to
wrap.

> +/**
> + * container_clone - duplicate the current container in the hierarchy
> + * that the given subsystem is attached to, and move this task into
> + * the new child
> + */
> +int container_clone(struct task_struct *tsk, struct container_subsys *subsys)
> +{
> + struct dentry *dentry;
> + int ret = 0;
> + char nodename[32];
> + struct container *parent, *child;
> + struct inode *inode;
> + struct css_group *cg;
> + struct containerfs_root *root;
> +
> + /* We shouldn't be called by an unregistered subsystem */
> + BUG_ON(!subsys->active);
> +
> + /* First figure out what hierarchy and container we're dealing
> + * with, and pin them so we can drop container_mutex */
> + mutex_lock(&container_mutex);
> + again:
> + root = subsys->root;
> + if (root == &rootnode) {
> + printk(KERN_INFO
> + "Not cloning container for unused subsystem %s\n",

```
> +         subsys->name);
> +  mutex_unlock(&container_mutex);
> +  return 0;
> + }
> + cg = &tsk->containers;
> + parent = task_container(tsk, subsys->subsys_id);
> + /* Pin the hierarchy */
> + atomic_inc(&parent->root->sb->s_active);
> +
> + mutex_unlock(&container_mutex);
> +
> + /* Now do the VFS work to create a container */
> + get_unused_name(nodename);
> + inode = parent->dentry->d_inode;
> +
> + /* Hold the parent directory mutex across this operation to
> +  * stop anyone else deleting the new container */
> + mutex_lock(&inode->i_mutex);
> + dentry = container_get_dentry(parent->dentry, nodename);
> + if (IS_ERR(dentry)) {
> +  printk(KERN_INFO
> +         "Couldn't allocate dentry for %s: %ld\n", nodename,
> +         PTR_ERR(dentry));
> +  ret = PTR_ERR(dentry);
> +  goto out_release;
```

Which I guess could cause a failure here.

Perhaps we could go back and rerun the get_unused_name() if EEXIST, if we're bothered.

I hope this is the biggest problem ;)

```
> + }
> +
> + /* Create the container directory, which also creates the container */
> + ret = vfs_mkdir(inode, dentry, S_IFDIR | 0755);
> + child = __d_cont(dentry);
> + dput(dentry);
> + if (ret) {
> +  printk(KERN_INFO
> +         "Failed to create container %s: %d\n", nodename,
> +         ret);
> +  goto out_release;
> + }
> +
> + if (!child) {
> +  printk(KERN_INFO
```

```
> +         "Couldn't find new container %s\n", nodename);
> +  ret = -ENOMEM;
> +  goto out_release;
> + }
> +
> + /* The container now exists. Retake container_mutex and check
> +  * that we're still in the same state that we thought we
> +  * were. */
> + mutex_lock(&container_mutex);
> + if ((root != subsys->root) ||
> +     (parent != task_container(tsk, subsys->subsys_id))) {
> +  /* Aargh, we raced ... */
> +  mutex_unlock(&inode->i_mutex);
> +
> +  deactivate_super(parent->root->sb);
> +  /* The container is still accessible in the VFS, but
> +   * we're not going to try to rmdir() it at this
> +   * point. */
> +  printk(KERN_INFO
> +         "Race in container_clone() - leaking container %s\n",
> +         nodename);
> +  goto again;
> + }
> +
> + /* All seems fine. Finish by moving the task into the new container */
> + ret = attach_task(child, tsk);
> + mutex_unlock(&container_mutex);
> +
> + out_release:
> + mutex_unlock(&inode->i_mutex);
> + deactivate_super(parent->root->sb);
> + return ret;
> +}
```