Subject: Re: [RFC][PATCH 3/4]: Enable multiple mounts of /dev/pts
Posted by serue on Wed, 06 Feb 2008 16:43:28 GMT
View Forum Message <> Reply to Message

Quoting Pavel Emelyanov (xemul@openvz.org):
> Serge E. Hallyn wrote:
> > Quoting Pavel Emelyanov (xemul@openvz.org):
> >> sukadev@us.ibm.com wrote:
> >>> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> >>> Subject: [RFC][PATCH 3/4]: Enable multiple mounts of /dev/pts
> >>>
> >>> To support multiple PTY namespaces, we should be allow multiple mounts of
> >>> /dev/pts, once within each PTY namespace.
> >>>
> >>> This patch removes the get_sb_single() in devpts_get_sb() and uses test and
> >>> set sb interfaces to allow remounting /dev/pts. The patch also removes the
> >>> globals, 'devpts_root' and uses current_pts_mnt() to access 'devpts_mnt'
> >>>
> >>> Changelog:
> >>> - Version 0: Based on earlier versions from Serge Hallyn and
> >>> Matt Helsley.
> >>>
> >>> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> >>> ---
> >>> fs/devpts/inode.c | 120 ++---------
> >>> 1 file changed, 101 insertions(+), 19 deletions(-)
> >>>
> >>> Index: linux-2.6.24/fs/devpts/inode.c
> >>> ===
> >>> --- linux-2.6.24.orig/fs/devpts/inode.c 2008-02-05 17:30:52.000000000 -0800
> >>> +++ linux-2.6.24/fs/devpts/inode.c 2008-02-05 19:16:39.000000000 -0800
> >>> @@ -34,7 +34,10 @@ static inline struct idr *current_pts_ns
> >>> }
> >>>
> >>> static struct vfsmount *devpts_mnt;
> >>> -static struct dentry *devpts_root;
> >>> +static inline struct vfsmount *current_pts_ns_mnt(void)
> >>> +{
> >>> + return devpts_mnt;
> >>> +}
> >>>
> >>> static struct {
> >>> int setuid;
> >>> @@ -130,7 +133,7 @@ devpts_fill_super(struct super_block *s,
> >>> inode->i_fop = &simple_dir_operations;
> >>> inode->i_nlink = 2;
> >>>
> >>> - devpts_root = s->s_root = d_alloc_root(inode);

> >>> + s->s_root = d_alloc_root(inode);
> >>> if (s->s_root)
> >>> return 0;
> >>>
> >>> @@ -140,10 +143,53 @@ fail:
> >>> return -ENOMEM;
> >>> }
> >>>
> >>> +/*
> >>> + * We use test and set super-block operations to help determine whether we
> >>> + * need a new super-block for this namespace. get_sb() walks the list of
> >>> + * existing devpts supers, comparing them with the @data ptr. Since we
> >>> + * passed 'current's namespace as the @data pointer we can compare the
> >>> + * namespace pointer in the super-block's 's_fs_info'. If the test is
> >>> + * TRUE then get_sb() returns a new active reference to the super block.
> >>> + * Otherwise, it helps us build an active reference to a new one.
> >>> + */
> >>> +
> >>> +static int devpts_test_sb(struct super_block *sb, void *data)
> >>> +{
> >>> + return sb->s_fs_info == data;
> >>> +}
> >>> +
> >>> +static int devpts_set_sb(struct super_block *sb, void *data)
> >>> +{
> >>> + sb->s_fs_info = data;
> >>> + return set_anon_super(sb, NULL);
> >>> +}
> >>> +
> >>> static int devpts_get_sb(struct file_system_type *fs_type,
> >>> int flags, const char *dev_name, void *data, struct vfsmount *mnt)
> >>> {
> >>> - return get_sb_single(fs_type, flags, data, devpts_fill_super, mnt);
> >>> + struct super_block *sb;
> >>> + int err;
> >>> +
> >>> + /* hereafter we're very simlar to get_sb_nodev */
> >>> + sb = sget(fs_type, devpts_test_sb, devpts_set_sb, data);
> >>> + if (IS_ERR(sb))
> >>> + return PTR_ERR(sb);
> >>> +
> >>> + if (sb->s_root)
> >>> + return simple_set_mnt(mnt, sb);
> >>> +
> >>> + sb->s_flags = flags;
> >>> + err = devpts_fill_super(sb, data, flags & MS_SILENT ? 1 : 0);
> >>> + if (err) {
> >>> + up_write(&sb->s_umount);

> >>> + deactivate_super(sb);
> >>> + return err;
> >>> + }
> >>> +
> >> That stuff becomes very very similar to that in proc :)
> >> Makes sense to consolidate. Maybe...
> >
> > Yeah, and the mqns that Cedric sent too. I think Cedric said he'd
> > started an a patch implementing a helper. Cedric?
>
> Mmm. I wanted to send one small objection to Cedric's patches with mqns,
> but the thread was abandoned by the time I decided to do-it-right-now.
>
> So I can put it here: forcing the CLONE_NEWNS is not very good, since
> this makes impossible to push a bind mount inside a new namespace, which
> may operate in some chroot environment. But this ability is heavily

Which direction do you want to go? I'm wondering whether mounts
propagation can address it.

Though really, I think you're right - we shouldn't break the kernel
doing CLONE_NEWMQ or CLONE_NEWPTS without CLONE_NEWNS, so we shouldn't
force the combination.

> exploited in OpenVZ, so if we can somehow avoid forcing the NEWNS flag
> that would be very very good :) See my next comment about this issue.
>
> > Pavel, not long ago you said you were starting to look at tty and pty
> > stuff - did you have any different ideas on devpts virtualization, or
> > are you ok with this minus your comments thus far?
>
> I have a similar idea of how to implement this, but I didn't thought
> about the details. As far as this issue is concerned, I see no reasons
> why we need a kern_mount-ed devtpsfs instance. If we don't make such,
> we may safely hold the ptsns from the superblock and be happy. The
> same seems applicable to the mqns, no?

But the current->nsproxy->devpts->mnt is used in several functions in
patch 3.

> The reason I have the kern_mount-ed instance of proc for pid namespaces
> is that I need a vfsmount to flush task entries from, but allowing
> it to be NULL (i.e. no kern_mount, but optional user mounts) means
> handing all the possible races, which is too heavy. But do we actually
> need the vfsmount for devpts and mqns if no user-space mounts exist?
>
> Besides, I planned to include legacy ptys virtualization and console
> virtualizatin in this namespace, but it seems, that it is not present

> in this particular one.

I had been thinking the consoles would have their own ns, since there's
really nothing linking them, but there really is no good reason why
userspace should ever want them separate. So I'm fine with combining
them.

> >>> + sb->s_flags |= MS_ACTIVE;
> >>> + devpts_mnt = mnt;
> >>> +
> >>> + return simple_set_mnt(mnt, sb);
> >>> }
> >>>
> >>> static struct file_system_type devpts_fs_type = {
> >>> @@ -158,10 +204,9 @@ static struct file_system_type devpts_fs
> >>>   * to the System V naming convention
> >>>   */
> >>>
> >>> -static struct dentry *get_node(int num)
> >>> +static struct dentry *get_node(struct dentry *root, int num)
> >>> {
> >>>   char s[12];
> >>> - struct dentry *root = devpts_root;
> >>>   mutex_lock(&root->d_inode->i_mutex);
> >>>   return lookup_one_len(s, root, sprintf(s, "%d", num));
> >>> }
> >>> @@ -207,12 +252,28 @@ int devpts_pty_new(struct tty_struct *tt
> >>>   struct tty_driver *driver = tty->driver;
> >>>   dev_t device = MKDEV(driver->major, driver->minor_start+number);
> >>>   struct dentry *dentry;
> >>> - struct inode *inode = new_inode(devpts_mnt->mnt_sb);
> >>> + struct dentry *root;
> >>> + struct vfsmount *mnt;
> >>> + struct inode *inode;
> >>> +
> >>>
> >>>   /* We're supposed to be given the slave end of a pty */
> >>>   BUG_ON(driver->type != TTY_DRIVER_TYPE_PTY);
> >>>   BUG_ON(driver->subtype != PTY_TYPE_SLAVE);
> >>>
> >>> + mnt = current_pts_ns_mnt();
> >>> + if (!mnt)
> >>> + return -ENOSYS;
> >>> + root = mnt->mnt_root;
> >>> +
> >>> + mutex_lock(&root->d_inode->i_mutex);
> >>> + inode = idr_find(current_pts_ns_allocated_ptys(), number);
> >>> + mutex_unlock(&root->d_inode->i_mutex);

```
> >>> +
> >>> + if (inode && !IS_ERR(inode))
> >>> +  return -EEXIST;
> >>> +
> >>> + inode = new_inode(mnt->mnt_sb);
> >>>   if (!inode)
> >>>    return -ENOMEM;
> >>>
> >>> @@ -222,23 +283,31 @@ int devpts_pty_new(struct tty_struct *tt
> >>>   inode->i_mtime = inode->i_atime = inode->i_ctime = CURRENT_TIME;
> >>>   init_special_inode(inode, S_IFCHR|config.mode, device);
> >>>   inode->i_private = tty;
> >>> + idr_replace(current_pts_ns_allocated_ptys(), inode, number);
> >>>
> >>> - dentry = get_node(number);
> >>> + dentry = get_node(root, number);
> >>>   if (!IS_ERR(dentry) && !dentry->d_inode) {
> >>>    d_instantiate(dentry, inode);
> >>> -  fsnotify_create(devpts_root->d_inode, dentry);
> >>> +  fsnotify_create(root->d_inode, dentry);
> >>>   }
> >>>
> >>> - mutex_unlock(&devpts_root->d_inode->i_mutex);
> >>> + mutex_unlock(&root->d_inode->i_mutex);
> >>>
> >>>   return 0;
> >>>  }
> >>>
> >>>  struct tty_struct *devpts_get_tty(int number)
> >>>  {
> >>> - struct dentry *dentry = get_node(number);
> >>> + struct vfsmount *mnt;
> >>> + struct dentry *dentry;
> >>>   struct tty_struct *tty;
> >>>
> >>> + mnt = current_pts_ns_mnt();
> >>> + if (!mnt)
> >>> +  return NULL;
> >>> +
> >>> + dentry = get_node(mnt->mnt_root, number);
> >>> +
> >>>   tty = NULL;
> >>>   if (!IS_ERR(dentry)) {
> >>>    if (dentry->d_inode)
> >>> @@ -246,14 +315,21 @@ struct tty_struct *devpts_get_tty(int nu
> >>>    dput(dentry);
> >>>   }
> >>>
```

```
> >>> - mutex_unlock(&devpts_root->d_inode->i_mutex);
> >>> + mutex_unlock(&mnt->mnt_root->d_inode->i_mutex);
> >>>
> >>>   return tty;
> >>>  }
> >>>
> >>>  void devpts_pty_kill(int number)
> >>>  {
> >>> - struct dentry *dentry = get_node(number);
> >>> + struct dentry *dentry;
> >>> + struct dentry *root;
> >>> + struct vfsmount *mnt;
> >>> +
> >>> + mnt = current_pts_ns_mnt();
> >>> + root = mnt->mnt_root;
> >>> +
> >>> + dentry = get_node(root, number);
> >>>
> >>>   if (!IS_ERR(dentry)) {
> >>>    struct inode *inode = dentry->d_inode;
> >>> @@ -264,17 +340,23 @@ void devpts_pty_kill(int number)
> >>>    }
> >>>    dput(dentry);
> >>>   }
> >>> - mutex_unlock(&devpts_root->d_inode->i_mutex);
> >>> + mutex_unlock(&root->d_inode->i_mutex);
> >>>  }
> >>>
> >>>  static int __init init_devpts_fs(void)
> >>>  {
> >>> - int err = register_filesystem(&devpts_fs_type);
> >>> - if (!err) {
> >>> -  devpts_mnt = kern_mount(&devpts_fs_type);
> >>> -  if (IS_ERR(devpts_mnt))
> >>> -   err = PTR_ERR(devpts_mnt);
> >>> - }
> >>> + struct vfsmount *mnt;
> >>> + int err;
> >>> +
> >>> + err = register_filesystem(&devpts_fs_type);
> >>> + if (err)
> >>> +  return err;
> >>> +
> >>> + mnt = kern_mount_data(&devpts_fs_type, NULL);
> >>> + if (IS_ERR(mnt))
> >>> +  err = PTR_ERR(mnt);
> >>> + else
> >>> +  devpts_mnt = mnt;
```

> >>> return err;
> >>> }
> >>>
> >>> _____
> >>> Containers mailing list
> >>> Containers@lists.linux-foundation.org
> >>> https://lists.linux-foundation.org/mailman/listinfo/containers
> >>>
> >> _____
> >> Containers mailing list
> >> Containers@lists.linux-foundation.org
> >> https://lists.linux-foundation.org/mailman/listinfo/containers
> >

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