Subject: Re: [PATCH 6/7]: Check for user-space mount of /dev/pts
Posted by serue on Tue, 25 Mar 2008 14:54:48 GMT
View Forum Message <> Reply to Message

Quoting sukadev@us.ibm.com (sukadev@us.ibm.com):
>
> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> Subject: [PATCH 6/7]: Check for user-space mount of /dev/pts
>
> When the pts namespace is cloned, the /dev/pts is not useful unless it
> is remounted from the user space.
>
> If user-space clones pts namespace but does not remount /dev/pts, it
> would end up using the /dev/pts mount from parent-pts-ns but allocate
> the pts indices from current pts ns.

So why not use the allocated_ptys from the parent ptsns? It's what
userspace asked for and it's safe to do.

> This patch (hack ?) prevents creation of PTYs in user space unless
> user-space mounts /dev/pts.
>
> (While this patch can be folded into others, keeping this separate
> for now for easier review (and to highlight the hack :-)
>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> ---
> fs/devpts/inode.c | 25 ++++++++++++++++++++++++--
> include/linux/devpts_fs.h | 20 +++++++++++++++++++-
> 2 files changed, 42 insertions(+), 3 deletions(-)
>
> Index: 2.6.25-rc5-mm1/include/linux/devpts_fs.h
> ==========
> --- 2.6.25-rc5-mm1.orig/include/linux/devpts_fs.h 2008-03-24 20:08:33.000000000 -0700
> +++ 2.6.25-rc5-mm1/include/linux/devpts_fs.h 2008-03-24 20:08:57.000000000 -0700
> @@ -23,6 +23,7 @@ struct pts_namespace {
> struct kref kref;
> struct idr allocated_ptys;
> struct vfsmount *mnt;
> + int user_mounted;
> };
>
> extern struct pts_namespace init_pts_ns;
> @@ -30,6 +31,8 @@ extern struct pts_namespace init_pts_ns;
> #define DEVPTS_SUPER_MAGIC 0x1cd1
> static inline struct pts_namespace *pts_ns_from_inode(struct inode *inode)
> {
> + struct pts_namespace *ns;

```
> +
>   /*
>    * Need this bug-on for now to catch any cases in tty_open()
>    * or release_dev() I may have missed.
> @@ -43,7 +46,22 @@ static inline struct pts_namespace *pts_
>    * should not need a lock here.
>    */
>
> - return (struct pts_namespace *)inode->i_sb->s_fs_info;
> + ns = (struct pts_namespace *)inode->i_sb->s_fs_info;
> +
> + /*
> +  * If user-space did not mount pts ns after cloning pts namespace,
> +  * the child process would end up accessing devpts mount of the
> +  * parent but use allocated_ptys from the cloned pts ns.
> +  *
> +  * This check prevents creating ptys unless user-space mounts
> +  * devpts  in the new pts namespace.
> +  *
> +  * Is there a cleaner way to prevent this ?
> +  */
> + if (!ns->user_mounted)
> +  return NULL;
> +
> + return ns;
>  }
>
>  static inline struct pts_namespace *current_pts_ns(void)
> Index: 2.6.25-rc5-mm1/fs/devpts/inode.c
> ===================================================================
> --- 2.6.25-rc5-mm1.orig/fs/devpts/inode.c 2008-03-24 20:08:33.000000000 -0700
> +++ 2.6.25-rc5-mm1/fs/devpts/inode.c 2008-03-24 20:08:57.000000000 -0700
> @@ -201,8 +201,11 @@ static int devpts_get_sb(struct file_sys
>   if (IS_ERR(sb))
>    return PTR_ERR(sb);
>
> - if (sb->s_root)
> + if (sb->s_root) {
> +  if (!(flags & MS_KERNMOUNT))
> +   ns->user_mounted = 1;
>    return simple_set_mnt(mnt, sb);
> + }
>
>   sb->s_flags = flags;
>   err = devpts_fill_super(sb, data, flags & MS_SILENT ? 1 : 0);
> @@ -248,6 +251,10 @@ int devpts_new_index(struct pts_namespac
>   int index;
>   int idr_ret;
```

```
>
> + if (!pts_ns || !pts_ns->user_mounted) {
> +  printk(KERN_ERR "devpts_new_index() without user_mount\n");
> +  return -ENOSYS;
> + }
>  retry:
>   if (!idr_pre_get(&pts_ns->allocated_ptys, GFP_KERNEL)) {
>    return -ENOMEM;
> @@ -273,7 +280,7 @@ retry:
>
>  void devpts_kill_index(struct pts_namespace *pts_ns, int idx)
>  {
> -
> + BUG_ON(!pts_ns->user_mounted);
>   down(&allocated_ptys_lock);
>   idr_remove(&pts_ns->allocated_ptys, idx);
>   up(&allocated_ptys_lock);
> @@ -293,6 +300,11 @@ int devpts_pty_new( struct pts_namespace
>   BUG_ON(driver->type != TTY_DRIVER_TYPE_PTY);
>   BUG_ON(driver->subtype != PTY_TYPE_SLAVE);
>
> + if (!pts_ns || !pts_ns->user_mounted) {
> +  printk(KERN_ERR "devpts_pty_new() without user_mount\n");
> +  return -ENOSYS;
> + }
> +
>   mnt = pts_ns->mnt;
>   root = mnt->mnt_root;
>
> @@ -332,6 +344,11 @@ struct tty_struct *devpts_get_tty(struct
>   struct dentry *dentry;
>   struct tty_struct *tty;
>
> + if (!pts_ns || !pts_ns->user_mounted) {
> +  printk(KERN_ERR "devpts_get_tty() without user_mount\n");
> +  return ERR_PTR(-ENOSYS);
> + }
> +
>   mnt = pts_ns->mnt;
>
>   dentry = get_node(mnt->mnt_root, number);
> @@ -353,6 +370,10 @@ void devpts_pty_kill(struct pts_namespac
>   struct dentry *dentry;
>   struct dentry *root;
>
> + if (!pts_ns || !pts_ns->user_mounted) {
> +  printk(KERN_ERR "devpts_pty_kill() without user_mount\n");
> +  BUG_ON(1);
```

> + }
> root = pts_ns->mnt->mnt_root;
>
> dentry = get_node(root, number);

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