Subject: Re: [patch 09/10] unprivileged mounts: propagation: inherit owner from parent
Posted by serue on Mon, 21 Jan 2008 20:23:06 GMT
View Forum Message <> Reply to Message

Quoting Miklos Szeredi (miklos@szeredi.hu):
> From: Miklos Szeredi <mszeredi@suse.cz>
>
> On mount propagation, let the owner of the clone be inherited from the
> parent into which it has been propagated.
>
> If the parent has the "nosuid" flag, set this flag for the child as
> well. This is needed for the suid-less namespace (use case #2 in the
> first patch header), where all mounts are owned by the user and have
> the nosuid flag set. In this case the propagated mount needs to have
> nosuid, otherwise a suid executable may be misused by the user.
>
> Similar treatment is not needed for "nodev", because devices can't be
> abused this way: the user is not able to gain privileges to devices by
> rearranging the mount namespace.
>
> Signed-off-by: Miklos Szeredi <mszeredi@suse.cz>

As discussed many months ago this does seem like the most appropriate
behavior for propagation.

Acked-by: Serge Hallyn <serue@us.ibm.com>

> ---
>
> Index: linux/fs/namespace.c
> ===
> --- linux.orig/fs/namespace.c 2008-01-16 13:25:09.000000000 +0100
> +++ linux/fs/namespace.c 2008-01-16 13:25:11.000000000 +0100
> @@ -506,10 +506,10 @@ static int reserve_user_mount(void)
>   return err;
> }
>
> -static void __set_mnt_user(struct vfsmount *mnt)
> +static void __set_mnt_user(struct vfsmount *mnt, uid_t owner)
> {
>   WARN_ON(mnt->mnt_flags & MNT_USER);
> - mnt->mnt_uid = current->fsuid;
> + mnt->mnt_uid = owner;
>   mnt->mnt_flags |= MNT_USER;
>
>   if (!capable(CAP_SETUID))
> @@ -520,7 +520,7 @@ static void __set_mnt_user(struct vfsmou

```
>
>  static void set_mnt_user(struct vfsmount *mnt)
>  {
> - __set_mnt_user(mnt);
> + __set_mnt_user(mnt, current->fsuid);
>   spin_lock(&vfsmount_lock);
>   nr_user_mounts++;
>   spin_unlock(&vfsmount_lock);
> @@ -536,7 +536,7 @@ static void clear_mnt_user(struct vfsmou
>  }
>
>  static struct vfsmount *clone_mnt(struct vfsmount *old, struct dentry *root,
> -     int flag)
> +     int flag, uid_t owner)
>  {
>   struct super_block *sb = old->mnt_sb;
>   struct vfsmount *mnt;
> @@ -560,7 +560,10 @@ static struct vfsmount *clone_mnt(struct
>   /* don't copy the MNT_USER flag */
>   mnt->mnt_flags &= ~MNT_USER;
>   if (flag & CL_SETUSER)
> -  __set_mnt_user(mnt);
> +  __set_mnt_user(mnt, owner);
> +
> + if (flag & CL_NOSUID)
> +  mnt->mnt_flags |= MNT_NOSUID;
>
>   if (flag & CL_SLAVE) {
>    list_add(&mnt->mnt_slave, &old->mnt_slave_list);
> @@ -1066,7 +1069,7 @@ static int lives_below_in_same_fs(struct
>  }
>
>  struct vfsmount *copy_tree(struct vfsmount *mnt, struct dentry *dentry,
> -     int flag)
> +     int flag, uid_t owner)
>  {
>   struct vfsmount *res, *p, *q, *r, *s;
>   struct nameidata nd;
> @@ -1074,7 +1077,7 @@ struct vfsmount *copy_tree(struct vfsmou
>   if (!(flag & CL_COPY_ALL) && IS_MNT_UNBINDABLE(mnt))
>    return ERR_PTR(-EPERM);
>
> - res = q = clone_mnt(mnt, dentry, flag);
> + res = q = clone_mnt(mnt, dentry, flag, owner);
>   if (IS_ERR(q))
>    goto error;
>   q->mnt_mountpoint = mnt->mnt_mountpoint;
> @@ -1096,7 +1099,7 @@ struct vfsmount *copy_tree(struct vfsmou
```

```
>     p = s;
>     nd.path.mnt = q;
>     nd.path.dentry = p->mnt_mountpoint;
> -   q = clone_mnt(p, p->mnt_root, flag);
> +   q = clone_mnt(p, p->mnt_root, flag, owner);
>     if (IS_ERR(q))
>      goto error;
>     spin_lock(&vfsmount_lock);
> @@ -1121,7 +1124,7 @@ struct vfsmount *collect_mounts(struct v
>  {
>   struct vfsmount *tree;
>   down_read(&namespace_sem);
> - tree = copy_tree(mnt, dentry, CL_COPY_ALL | CL_PRIVATE);
> + tree = copy_tree(mnt, dentry, CL_COPY_ALL | CL_PRIVATE, 0);
>   up_read(&namespace_sem);
>   return tree;
>  }
> @@ -1292,7 +1295,8 @@ static int do_change_type(struct nameida
>   */
>  static int do_loopback(struct nameidata *nd, char *old_name, int flags)
>  {
> - int clone_fl;
> + int clone_fl = 0;
> + uid_t owner = 0;
>   struct nameidata old_nd;
>   struct vfsmount *mnt = NULL;
>   int err;
> @@ -1313,11 +1317,17 @@ static int do_loopback(struct nameidata
>   if (!check_mnt(nd->path.mnt) || !check_mnt(old_nd.path.mnt))
>    goto out;
>
> - clone_fl = (flags & MS_SETUSER) ? CL_SETUSER : 0;
> + if (flags & MS_SETUSER) {
> +  clone_fl |= CL_SETUSER;
> +  owner = current->fsuid;
> + }
> +
>   if (flags & MS_REC)
> -  mnt = copy_tree(old_nd.path.mnt, old_nd.path.dentry, clone_fl);
> +  mnt = copy_tree(old_nd.path.mnt, old_nd.path.dentry, clone_fl,
> +    owner);
>   else
> -  mnt = clone_mnt(old_nd.path.mnt, old_nd.path.dentry, clone_fl);
> +  mnt = clone_mnt(old_nd.path.mnt, old_nd.path.dentry, clone_fl,
> +    owner);
>
>   err = PTR_ERR(mnt);
>   if (IS_ERR(mnt))
```

```
> @@ -1541,7 +1551,7 @@ static int do_new_mount(struct nameidata
>   }
>
>   if (flags & MS_SETUSER)
> -  __set_mnt_user(mnt);
> +  __set_mnt_user(mnt, current->fsuid);
>
>   return do_add_mount(mnt, nd, mnt_flags, NULL);
>
> @@ -1937,7 +1947,7 @@ static struct mnt_namespace *dup_mnt_ns(
>   down_write(&namespace_sem);
>   /* First pass: copy the tree topology */
>   new_ns->root = copy_tree(mnt_ns->root, mnt_ns->root->mnt_root,
> -     CL_COPY_ALL | CL_EXPIRE);
> +     CL_COPY_ALL | CL_EXPIRE, 0);
>   if (IS_ERR(new_ns->root)) {
>    up_write(&namespace_sem);
>    kfree(new_ns);
> Index: linux/fs/pnode.c
> ===================================================================
> --- linux.orig/fs/pnode.c 2008-01-16 13:25:07.000000000 +0100
> +++ linux/fs/pnode.c 2008-01-16 13:25:11.000000000 +0100
> @@ -181,15 +181,28 @@ int propagate_mnt(struct vfsmount *dest_
>
>   for (m = propagation_next(dest_mnt, dest_mnt); m;
>     m = propagation_next(m, dest_mnt)) {
> -  int type;
> +  int clflags;
> +  uid_t owner = 0;
>    struct vfsmount *source;
>
>    if (IS_MNT_NEW(m))
>     continue;
>
> -  source =  get_source(m, prev_dest_mnt, prev_src_mnt, &type);
> +  source =  get_source(m, prev_dest_mnt, prev_src_mnt, &clflags);
>
> -  child = copy_tree(source, source->mnt_root, type);
> +  if (m->mnt_flags & MNT_USER) {
> +   clflags |= CL_SETUSER;
> +   owner = m->mnt_uid;
> +
> +   /*
> +    * If propagating into a user mount which doesn't
> +    * allow suid, then make sure, the child(ren) won't
> +    * allow suid either
> +    */
> +   if (m->mnt_flags & MNT_NOSUID)
```

```
> +    clflags |= CL_NOSUID;
> +  }
> +  child = copy_tree(source, source->mnt_root, clflags, owner);
>     if (IS_ERR(child)) {
>      ret = PTR_ERR(child);
>      list_splice(tree_list, tmp_list.prev);
> Index: linux/fs/pnode.h
> ===================================================================
> --- linux.orig/fs/pnode.h 2008-01-16 13:25:05.000000000 +0100
> +++ linux/fs/pnode.h 2008-01-16 13:25:11.000000000 +0100
> @@ -24,6 +24,7 @@
>   #define CL_PROPAGATION   0x10
>   #define CL_PRIVATE   0x20
>   #define CL_SETUSER  0x40
> +#define CL_NOSUID  0x80
>
>   static inline void set_mnt_shared(struct vfsmount *mnt)
>   {
> @@ -36,4 +37,6 @@ int propagate_mnt(struct vfsmount *, str
>     struct list_head *);
>   int propagate_umount(struct list_head *);
>   int propagate_mount_busy(struct vfsmount *, int);
> +struct vfsmount *copy_tree(struct vfsmount *, struct dentry *, int, uid_t);
> +
>   #endif /* _LINUX_PNODE_H */
> Index: linux/include/linux/fs.h
> ===================================================================
> --- linux.orig/include/linux/fs.h 2008-01-16 13:25:09.000000000 +0100
> +++ linux/include/linux/fs.h 2008-01-16 13:25:11.000000000 +0100
> @@ -1493,7 +1493,6 @@ extern int may_umount(struct vfsmount *)
>   extern void umount_tree(struct vfsmount *, int, struct list_head *);
>   extern void release_mounts(struct list_head *);
>   extern long do_mount(char *, char *, char *, unsigned long, void *);
> -extern struct vfsmount *copy_tree(struct vfsmount *, struct dentry *, int);
>   extern void mnt_set_mountpoint(struct vfsmount *, struct dentry *,
>         struct vfsmount *);
>   extern struct vfsmount *collect_mounts(struct vfsmount *, struct dentry *);
>
> --
> -
> To unsubscribe from this list: send the line "unsubscribe linux-fsdevel" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at  http://vger.kernel.org/majordomo-info.html
```