Subject: Re: [patch 3/8] account user mounts
Posted by ebiederm on Sat, 21 Apr 2007 13:37:21 GMT
View Forum Message <> Reply to Message

Miklos Szeredi <miklos@szeredi.hu> writes:

> From: Miklos Szeredi <mszeredi@suse.cz>
>
> Add sysctl variables for accounting and limiting the number of user
> mounts.
>
> The maximum number of user mounts is set to 1024 by default. This
> won't in itself enable user mounts, setting a mount to be owned by a
> user is first needed

Since each mount has a user can we just make this a per user rlimit?

If we are going to implement a sysctl at this point I think it should
be a global limit that doesn't care if who you are. Even root can
have recursive mounts that attempt to get out of control.

Also currently you are not checking the max_users. It looks like
you do this in a later patch but still it is a little strange to
allow user own mounts and have accounting but to not check the
limit at this state.

Eric

>
> Signed-off-by: Miklos Szeredi <mszeredi@suse.cz>
> ---
>
> Index: linux/include/linux/sysctl.h
> ===
> --- linux.orig/include/linux/sysctl.h 2007-04-20 11:55:02.000000000 +0200
> +++ linux/include/linux/sysctl.h 2007-04-20 11:55:07.000000000 +0200
> @@ -818,6 +818,8 @@ enum
>   FS_AIO_NR=18, /* current system-wide number of aio requests */
>   FS_AIO_MAX_NR=19, /* system-wide maximum number of aio requests */
>   FS_INOTIFY=20, /* inotify submenu */
> + FS_NR_USER_MOUNTS=21, /* int:current number of user mounts */
> + FS_MAX_USER_MOUNTS=22, /* int:maximum number of user mounts */
>   FS_OCFS2=988, /* ocfs2 */
> };
>
> Index: linux/kernel/sysctl.c
> ===
> --- linux.orig/kernel/sysctl.c 2007-04-20 11:55:02.000000000 +0200

> +++ linux/kernel/sysctl.c 2007-04-20 11:55:07.000000000 +0200
> @@ -1063,6 +1063,22 @@ static ctl_table fs_table[] = {
> #endif
> #endif
>   {
> + .ctl_name = FS_NR_USER_MOUNTS,
> + .procname = "nr_user_mounts",
> + .data = &nr_user_mounts,
> + .maxlen = sizeof(int),
> + .mode = 0444,
> + .proc_handler = &proc_dointvec,
> + },
> + {
> + .ctl_name = FS_MAX_USER_MOUNTS,
> + .procname = "max_user_mounts",
> + .data = &max_user_mounts,
> + .maxlen = sizeof(int),
> + .mode = 0644,
> + .proc_handler = &proc_dointvec,
> + },
> + {
>   .ctl_name = KERN_SETUID_DUMPABLE,
>   .procname = "suid_dumpable",
>   .data = &suid_dumpable,
> Index: linux/Documentation/filesystems/proc.txt
> ===
> --- linux.orig/Documentation/filesystems/proc.txt 2007-04-20 11:55:02.000000000
> +0200
> +++ linux/Documentation/filesystems/proc.txt 2007-04-20 11:55:07.000000000 +0200
> @@ -923,6 +923,15 @@ reaches aio-max-nr then io_setup will fa
> raising aio-max-nr does not result in the pre-allocation or re-sizing
> of any kernel data structures.
>
> +nr_user_mounts and max_user_mounts
> +----------------------------------
> +
> +These represent the number of "user" mounts and the maximum number of
> +"user" mounts respectively. User mounts may be created by
> +unprivileged users. User mounts may also be created with sysadmin
> +privileges on behalf of a user, in which case nr_user_mounts may
> +exceed max_user_mounts.
> +
> 2.2 /proc/sys/fs/binfmt_misc - Miscellaneous binary formats
> --
>
> Index: linux/fs/namespace.c
> ===
> --- linux.orig/fs/namespace.c 2007-04-20 11:55:06.000000000 +0200

```
> +++ linux/fs/namespace.c 2007-04-20 11:55:07.000000000 +0200
> @@ -39,6 +39,9 @@ static int hash_mask __read_mostly, hash
>  static struct kmem_cache *mnt_cache __read_mostly;
>  static struct rw_semaphore namespace_sem;
>
> +int nr_user_mounts;
> +int max_user_mounts = 1024;
> +
>  /* /sys/fs */
>  decl_subsys(fs, NULL, NULL);
>  EXPORT_SYMBOL_GPL(fs_subsys);
> @@ -227,11 +230,30 @@ static struct vfsmount *skip_mnt_tree(st
>   return p;
>  }
>
> +static void dec_nr_user_mounts(void)
> +{
> + spin_lock(&vfsmount_lock);
> + nr_user_mounts--;
> + spin_unlock(&vfsmount_lock);
> +}
> +
>  static void set_mnt_user(struct vfsmount *mnt)
>  {
>   BUG_ON(mnt->mnt_flags & MNT_USER);
>   mnt->mnt_uid = current->uid;
>   mnt->mnt_flags |= MNT_USER;
> + spin_lock(&vfsmount_lock);
> + nr_user_mounts++;
> + spin_unlock(&vfsmount_lock);
> +}
> +
> +static void clear_mnt_user(struct vfsmount *mnt)
> +{
> + if (mnt->mnt_flags & MNT_USER) {
> +  mnt->mnt_uid = 0;
> +  mnt->mnt_flags &= ~MNT_USER;
> +  dec_nr_user_mounts();
> + }
>  }
>
>  static struct vfsmount *clone_mnt(struct vfsmount *old, struct dentry *root,
> @@ -283,6 +305,7 @@ static inline void __mntput(struct vfsmo
>  {
>   struct super_block *sb = mnt->mnt_sb;
>   dput(mnt->mnt_root);
> + clear_mnt_user(mnt);
>   free_vfsmnt(mnt);
```

> deactivate_super(sb);
> }
> @@ -1023,6 +1046,7 @@ static int do_remount(struct nameidata *
> down_write(&sb->s_umount);
> err = do_remount_sb(sb, flags, data, 0);
> if (!err) {
> + clear_mnt_user(nd->mnt);
> nd->mnt->mnt_flags = mnt_flags;
> if (flags & MS_SETUSER)
> set_mnt_user(nd->mnt);
> Index: linux/include/linux/fs.h
> ===
> --- linux.orig/include/linux/fs.h 2007-04-20 11:55:05.000000000 +0200
> +++ linux/include/linux/fs.h 2007-04-20 11:55:07.000000000 +0200
> @@ -50,6 +50,9 @@ extern struct inodes_stat_t inodes_stat;
>
> extern int leases_enable, lease_break_time;
>
> +extern int nr_user_mounts;
> +extern int max_user_mounts;
> +
> #ifdef CONFIG_DNOTIFY
> extern int dir_notify_enable;
> #endif
>
> --