Subject: [patch 3/9] unprivileged mounts: account user mounts
Posted by Miklos Szeredi on Tue, 08 Jan 2008 11:35:05 GMT
View Forum Message <> Reply to Message

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

[akpm]
- don't use enumerated sysctls

Signed-off-by: Miklos Szeredi <mszeredi@suse.cz>

Index: linux/Documentation/filesystems/proc.txt
===
--- linux.orig/Documentation/filesystems/proc.txt 2008-01-03 17:12:58.000000000 +0100
+++ linux/Documentation/filesystems/proc.txt 2008-01-03 21:15:35.000000000 +0100
@@ -1012,6 +1012,15 @@ 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 2008-01-03 21:14:16.000000000 +0100
+++ linux/fs/namespace.c 2008-01-03 21:15:35.000000000 +0100
@@ -44,6 +44,9 @@ static struct list_head *mount_hashtable
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 */
 struct kobject *fs_kobj;
 EXPORT_SYMBOL_GPL(fs_kobj);
@@ -477,11 +480,30 @@ 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->fsuid;
  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,
@@ -542,6 +564,7 @@ static inline void __mntput(struct vfsmo
   */
  WARN_ON(atomic_read(&mnt->__mnt_writers));
  dput(mnt->mnt_root);
+ clear_mnt_user(mnt);
  free_vfsmnt(mnt);
  deactivate_super(sb);
 }
@@ -1306,6 +1329,7 @@ static int do_remount(struct nameidata *
  else
   err = do_remount_sb(sb, flags, data, 0);
  if (!err) {
```

```
+  clear_mnt_user(nd->path.mnt);
   nd->path.mnt->mnt_flags = mnt_flags;
   if (flags & MS_SETUSER)
    set_mnt_user(nd->path.mnt);
Index: linux/include/linux/fs.h
===================================================================
--- linux.orig/include/linux/fs.h 2008-01-03 20:52:38.000000000 +0100
+++ linux/include/linux/fs.h 2008-01-03 21:15:35.000000000 +0100
@@ -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
Index: linux/kernel/sysctl.c
===================================================================
--- linux.orig/kernel/sysctl.c 2008-01-03 17:13:22.000000000 +0100
+++ linux/kernel/sysctl.c 2008-01-03 21:15:35.000000000 +0100
@@ -1288,6 +1288,22 @@ static struct ctl_table fs_table[] = {
 #endif
 #endif
  {
+  .ctl_name = CTL_UNNUMBERED,
+  .procname = "nr_user_mounts",
+  .data  = &nr_user_mounts,
+  .maxlen  = sizeof(int),
+  .mode  = 0444,
+  .proc_handler = &proc_dointvec,
+ },
+ {
+  .ctl_name = CTL_UNNUMBERED,
+  .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,

--
_____________________________________________________
```

Subject: Re: [patch 3/9] unprivileged mounts: account user mounts
Posted by Dave Hansen on Tue, 08 Jan 2008 18:18:47 GMT
View Forum Message <> Reply to Message

On Tue, 2008-01-08 at 12:35 +0100, Miklos Szeredi wrote:
> plain text document attachment
> (unprivileged-mounts-account-user-mounts.patch)
> From: Miklos Szeredi <mszeredi@suse.cz>
>
> Add sysctl variables for accounting and limiting the number of user
> mounts.
...
> +int nr_user_mounts;
> +int max_user_mounts = 1024;

Just from a containers point of view, I think this is something we'll
need to fix up in the near future if it stays in the current form.

Instead of having a global tracking, perhaps we could have a per-user
limit tracked in 'struct user'. The plans are to ensure that two
containers' users "dave" each have a different 'struct user', so that
seems to be a decent place to track it.

Also, is a read-only sysctl really the best way to get the number of
user mounts back out of the kernel? What would you use it for?

Do you need any special logic for setting 'max_user_mounts' in the case
where it gets set _below_ 'nr_user_mounts'?

> /* /sys/fs */
> struct kobject *fs_kobj;
> EXPORT_SYMBOL_GPL(fs_kobj);
> @@ -477,11 +480,30 @@ 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->fsuid;
>   mnt->mnt_flags |= MNT_USER;
> + spin_lock(&vfsmount_lock);
> + nr_user_mounts++;
> + spin_unlock(&vfsmount_lock);
> +}

One little nitpick on the patch layout: It's a wee bit difficult to
audit how the set function is used vs the clear one when its users don't
come until the later patches. It might be worth introducing the users
here, too.

> +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,
> @@ -542,6 +564,7 @@ static inline void __mntput(struct vfsmo
>   */
>   WARN_ON(atomic_read(&mnt->__mnt_writers));
>   dput(mnt->mnt_root);
> + clear_mnt_user(mnt);
>   free_vfsmnt(mnt);
>   deactivate_super(sb);
> }
> @@ -1306,6 +1329,7 @@ static int do_remount(struct nameidata *
>   else
>   err = do_remount_sb(sb, flags, data, 0);
>   if (!err) {
> + clear_mnt_user(nd->path.mnt);
>   nd->path.mnt->mnt_flags = mnt_flags;
>   if (flags & MS_SETUSER)
>   set_mnt_user(nd->path.mnt);
> Index: linux/include/linux/fs.h
> ===
> --- linux.orig/include/linux/fs.h 2008-01-03 20:52:38.000000000 +0100
> +++ linux/include/linux/fs.h 2008-01-03 21:15:35.000000000 +0100
> @@ -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
> Index: linux/kernel/sysctl.c
> ===
> --- linux.orig/kernel/sysctl.c 2008-01-03 17:13:22.000000000 +0100
> +++ linux/kernel/sysctl.c 2008-01-03 21:15:35.000000000 +0100
> @@ -1288,6 +1288,22 @@ static struct ctl_table fs_table[] = {
> #endif
> #endif
>   {
> + .ctl_name = CTL_UNNUMBERED,
> + .procname = "nr_user_mounts",
> + .data = &nr_user_mounts,
> + .maxlen = sizeof(int),
> + .mode = 0444,
> + .proc_handler = &proc_dointvec,
> + },
> + {
> + .ctl_name = CTL_UNNUMBERED,
> + .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,
>
> --
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers
-- Dave

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

Subject: Re: [patch 3/9] unprivileged mounts: account user mounts
Posted by Miklos Szeredi on Tue, 08 Jan 2008 19:18:38 GMT
View Forum Message <> Reply to Message

> On Tue, 2008-01-08 at 12:35 +0100, Miklos Szeredi wrote:
> > plain text document attachment
> > (unprivileged-mounts-account-user-mounts.patch)
> > From: Miklos Szeredi <mszeredi@suse.cz>
> >
> > Add sysctl variables for accounting and limiting the number of user
> > mounts.
> ...
> > +int nr_user_mounts;
> > +int max_user_mounts = 1024;
>
> Just from a containers point of view, I think this is something we'll
> need to fix up in the near future if it stays in the current form.
>
> Instead of having a global tracking, perhaps we could have a per-user
> limit tracked in 'struct user'. The plans are to ensure that two
> containers' users "dave" each have a different 'struct user', so that
> seems to be a decent place to track it.

At one time there was a per-user accounting patch, but it was dropped,
because it was deemed an unnecessary additional compexity.

max_user_mounts is analogue to files_stat.max_files (which is a sysctl
tunable also), and it's purpose is really to make sure that a user is
not able to create an insane number of mounts, and not to acurately
limit normal usage.

So I'm not sure a per-container or per-user count is really needed.

> Also, is a read-only sysctl really the best way to get the number of
> user mounts back out of the kernel? What would you use it for?

Just to check, why I got that (EPERM or whatever) error for the mount
command.

> Do you need any special logic for setting 'max_user_mounts' in the case
> where it gets set _below_ 'nr_user_mounts'?

No, I don't think such corner cases really matter in this case.

> > /* /sys/fs */
> > struct kobject *fs_kobj;
> > EXPORT_SYMBOL_GPL(fs_kobj);
> > @@ -477,11 +480,30 @@ 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->fsuid;
> > mnt->mnt_flags |= MNT_USER;
> > + spin_lock(&vfsmount_lock);
> > + nr_user_mounts++;
> > + spin_unlock(&vfsmount_lock);
> > +}
>
> One little nitpick on the patch layout: It's a wee bit difficult to
> audit how the set function is used vs the clear one when its users don't
> come until the later patches. It might be worth introducing the users
> here, too.

Yeah, maybe some of the patches should be folded together. If a
resubmit is necessary I'll look into that.

Miklos

Subject: Re: [patch 3/9] unprivileged mounts: account user mounts
Posted by serue on Mon, 14 Jan 2008 21:53:51 GMT
View Forum Message <> Reply to Message

Quoting Miklos Szeredi (miklos@szeredi.hu):
> 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
>

> [akpm]
> - don't use enumerated sysctls
>
> Signed-off-by: Miklos Szeredi <mszeredi@suse.cz>

Seems sane enough, given your responses to Dave.

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

> ---
>
> Index: linux/Documentation/filesystems/proc.txt
> ===
> --- linux.orig/Documentation/filesystems/proc.txt 2008-01-03 17:12:58.000000000 +0100
> +++ linux/Documentation/filesystems/proc.txt 2008-01-03 21:15:35.000000000 +0100
> @@ -1012,6 +1012,15 @@ 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 2008-01-03 21:14:16.000000000 +0100
> +++ linux/fs/namespace.c 2008-01-03 21:15:35.000000000 +0100
> @@ -44,6 +44,9 @@ static struct list_head *mount_hashtable
> 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 */
> struct kobject *fs_kobj;
> EXPORT_SYMBOL_GPL(fs_kobj);
> @@ -477,11 +480,30 @@ 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->fsuid;
>   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,
> @@ -542,6 +564,7 @@ static inline void __mntput(struct vfsmo
>    */
>   WARN_ON(atomic_read(&mnt->__mnt_writers));
>   dput(mnt->mnt_root);
> + clear_mnt_user(mnt);
>   free_vfsmnt(mnt);
>   deactivate_super(sb);
>  }
> @@ -1306,6 +1329,7 @@ static int do_remount(struct nameidata *
>   else
>    err = do_remount_sb(sb, flags, data, 0);
>   if (!err) {
> +  clear_mnt_user(nd->path.mnt);
>    nd->path.mnt->mnt_flags = mnt_flags;
>    if (flags & MS_SETUSER)
>     set_mnt_user(nd->path.mnt);
> Index: linux/include/linux/fs.h
> ===================================================================
> --- linux.orig/include/linux/fs.h 2008-01-03 20:52:38.000000000 +0100
> +++ linux/include/linux/fs.h 2008-01-03 21:15:35.000000000 +0100
```

```
> @@ -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
> Index: linux/kernel/sysctl.c
> ===================================================================
> --- linux.orig/kernel/sysctl.c 2008-01-03 17:13:22.000000000 +0100
> +++ linux/kernel/sysctl.c 2008-01-03 21:15:35.000000000 +0100
> @@ -1288,6 +1288,22 @@ static struct ctl_table fs_table[] = {
>  #endif
>  #endif
>   {
> +  .ctl_name = CTL_UNNUMBERED,
> +  .procname = "nr_user_mounts",
> +  .data  = &nr_user_mounts,
> +  .maxlen  = sizeof(int),
> +  .mode  = 0444,
> +  .proc_handler = &proc_dointvec,
> + },
> + {
> +  .ctl_name = CTL_UNNUMBERED,
> +  .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,
>
> --
```