Subject: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by serue on Tue, 07 Nov 2006 04:18:14 GMT
View Forum Message <> Reply to Message

Cedric has previously sent out a patchset
(http://lists.osdl.org/pipermail/containers/2006-August/000078.html)
impplementing the very basics of a user namespace. It ignores filesystem
access checks, so that uid 502 in one namespace could access files
belonging to uid 502 in another namespace, if the containers were so set
up.

This isn't necessarily bad, since proper container setup should prevent
problems. However there has been concern, so here is a patchset which
takes one course in addressing the concern.

It adds a user namespace pointer to every superblock, and to enhances
fsuid equivalence checks with a (inode->i_sb->s_uid_ns ==
current->nsproxy->uid_ns) comparison.

I've tested this as follows:

Created a bare-minimum loopback filesystem which has su, ps, touch, and sh
and requisites (like /etc/pam.d). Under that, created a user hallyn with
the same uid as user hallyn on the root filesystem. Under both /home/hallyn
and /mnt/0/home/hallyn (/home/hallyn on the loopbackfs) created a directory
'priv' with 0700 perms.

unsharens -U /bin/sh
su hallyn
ls /home/hallyn/priv
 (permission denied)
mount -o loop /usr/src/disk.img /mnt/0
mount -t proc none /mnt/0/proc
mount -t devpts none /mnt/0/dev/pts
chroot /mnt/0
su hallyn
ls /home/hallyn/priv
 ab

And, finally, of course

mount -o loop /usr/src/disk.img /mnt/0
mount -t proc none /mnt/0/proc
mount -t devpts none /mnt/0/dev/pts
unsharens -U /bin/sh
chroot /mnt/0
su hallyn
ls /home/hallyn/priv

(permission denied)

This is only a rough prototype to start some discussion. i.e. I ignore groups, so kernel/sys.c:in_group_p() for instance will need to be updated.

A few issues to be discussed:

1. I am not doing anything about root access. There are several ways we can address this.

a. implement CAP_NS_OVERRIDE, without which cross-ns access is not allowed
b. just don't allow any cross-ns access at all
c. a more complicated scheme where root process in parent and child namespaces can access each other until somehow the parent-ns cuts off the child's access.

2. This patch takes the easy route of adding user_ns pointers to the superblock. It would be very nice to add it to the vfsmount instead, so that admins could simply mount --bind into various namespaces, rather than having to use completely separate filesystems. However several fsuid equivalence checks happen with only an inode available. The hardest to address so far appear to be fs/namei.c:generic_permission as called from, say nfs, fs/generic_acl.c:generic_acl_set, and fs/attr.c:inode_change_ok called from jffs2.

Still, putting the user_ns in the superblock and forcing the use of separate filesystems (i.e. through a lightweight stackable read-only filesystem) isn't *so* bad, is it?

thanks,
-serge

Subject: [RFC] [PATCH 1/4] uid_ns: introduce inode uid check helper
Posted by serue on Tue, 07 Nov 2006 04:19:15 GMT
View Forum Message <> Reply to Message

Uid namespaces will require that when a tasks' permission to an inode is checked, not just the uid, but also the namespace is checked. Since this is a pervasive change, let's start by introducing a helper without making any semantic changes, so we can make the semantic change in one place.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

---

 fs/namei.c | 11 ++++++-----
 include/linux/fs.h | 5 +++++
 2 files changed, 11 insertions(+), 5 deletions(-)

f3afe1adf8ebb6d2e7975dada086df5e9ea36d3c
diff --git a/fs/namei.c b/fs/namei.c
index ede2761..522ec89 100644
--- a/fs/namei.c
+++ b/fs/namei.c
@@ -184,7 +184,7 @@ int generic_permission(struct inode *ino
 {
  umode_t   mode = inode->i_mode;

- if (current->fsuid == inode->i_uid)
+ if (inode_task_same_uid(inode, current))
   mode >>= 6;
  else {
   if (IS_POSIXACL(inode) && (mode & S_IRWXG) && check_acl) {
@@ -436,7 +436,7 @@ static int exec_permission_lite(struct i
  if (inode->i_op && inode->i_op->permission)
   return -EAGAIN;

- if (current->fsuid == inode->i_uid)
+ if (inode_task_same_uid(inode, current))
   mode >>= 6;
  else if (in_group_p(inode->i_gid))
   mode >>= 3;
@@ -1360,9 +1360,9 @@ static inline int check_sticky(struct in
 {
  if (!(dir->i_mode & S_ISVTX))
   return 0;
- if (inode->i_uid == current->fsuid)
+ if (inode_task_same_uid(inode, current))
   return 0;
- if (dir->i_uid == current->fsuid)
+ if (inode_task_same_uid(dir, current))
   return 0;
  return !capable(CAP_FOWNER);
 }
@@ -1572,7 +1572,8 @@ int may_open(struct nameidata *nd, int a

  /* O_NOATIME can only be set by the owner or superuser */
  if (flag & O_NOATIME)

```
-  if (current->fsuid != inode->i_uid && !capable(CAP_FOWNER))
+  if (!inode_task_same_uid(inode, current) &&
+      !capable(CAP_FOWNER))
    return -EPERM;

  /*
diff --git a/include/linux/fs.h b/include/linux/fs.h
index 4090d9d..699c7b5 100644
--- a/include/linux/fs.h
+++ b/include/linux/fs.h
@@ -2157,5 +2157,10 @@ static inline void free_secdata(void *se
 { }
 #endif /* CONFIG_SECURITY */

+static inline int inode_task_same_uid(struct inode *ino,
+  struct task_struct *tsk)
+{
+ return (ino->i_uid == tsk->fsuid);
+}
 #endif /* __KERNEL__ */
 #endif /* _LINUX_FS_H */
--
1.1.6
```

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: [RFC] [PATCH 2/4] uid_ns: replace inode->fsuid checks under fs/
Posted by serue on Tue, 07 Nov 2006 04:19:39 GMT
View Forum Message <> Reply to Message

Replace inode->fsuid in fs/*.c with inode_task_same_uid(), which
will later be used to compare uid namespaces.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

```
 fs/attr.c        |   10 ++++++----
 fs/fcntl.c       |    3 ++-
 fs/generic_acl.c |    2 +-
 fs/locks.c       |    4 ++--
 fs/posix_acl.c   |    2 +-
 fs/utimes.c      |    4 ++--
 6 files changed, 14 insertions(+), 11 deletions(-)
```

```
54f0e4ce61c74cc7419988fbbadd0a3c54e21893
diff --git a/fs/attr.c b/fs/attr.c
index 97de946..b913555 100644
--- a/fs/attr.c
+++ b/fs/attr.c
@@ -30,20 +30,21 @@ int inode_change_ok(struct inode *inode,

  /* Make sure a caller can chown. */
  if ((ia_valid & ATTR_UID) &&
-     (current->fsuid != inode->i_uid ||
+     (!inode_task_same_uid(inode, current) ||
       attr->ia_uid != inode->i_uid) && !capable(CAP_CHOWN))
   goto error;

  /* Make sure caller can chgrp. */
  if ((ia_valid & ATTR_GID) &&
-     (current->fsuid != inode->i_uid ||
+     (!inode_task_same_uid(inode, current) ||
      (!in_group_p(attr->ia_gid) && attr->ia_gid != inode->i_gid)) &&
      !capable(CAP_CHOWN))
   goto error;

  /* Make sure a caller can chmod. */
  if (ia_valid & ATTR_MODE) {
-  if ((current->fsuid != inode->i_uid) && !capable(CAP_FOWNER))
+  if (!inode_task_same_uid(inode, current) &&
+      !capable(CAP_FOWNER))
    goto error;
   /* Also check the setgid bit! */
   if (!in_group_p((ia_valid & ATTR_GID) ? attr->ia_gid :
@@ -53,7 +54,8 @@ int inode_change_ok(struct inode *inode,

  /* Check for setting the inode time. */
  if (ia_valid & (ATTR_MTIME_SET | ATTR_ATIME_SET)) {
-  if (current->fsuid != inode->i_uid && !capable(CAP_FOWNER))
+  if (!inode_task_same_uid(inode, current) &&
+      !capable(CAP_FOWNER))
    goto error;
  }
 fine:
diff --git a/fs/fcntl.c b/fs/fcntl.c
index 8ba82c9..b1ed443 100644
--- a/fs/fcntl.c
+++ b/fs/fcntl.c
@@ -215,7 +215,8 @@ static int setfl(int fd, struct file * f

  /* O_NOATIME can only be set by the owner or superuser */
  if ((arg & O_NOATIME) && !(filp->f_flags & O_NOATIME))
```

```
-  if (current->fsuid != inode->i_uid && !capable(CAP_FOWNER))
+  if (!inode_task_same_uid(inode, current) &&
+      !capable(CAP_FOWNER))
     return -EPERM;

   /* required for strict SunOS emulation */
diff --git a/fs/generic_acl.c b/fs/generic_acl.c
index 9ccb789..a6402a9 100644
--- a/fs/generic_acl.c
+++ b/fs/generic_acl.c
@@ -78,7 +78,7 @@ generic_acl_set(struct inode *inode, str

   if (S_ISLNK(inode->i_mode))
     return -EOPNOTSUPP;
- if (current->fsuid != inode->i_uid && !capable(CAP_FOWNER))
+ if (inode_task_same_uid(inode, current) && !capable(CAP_FOWNER))
     return -EPERM;
   if (value) {
     acl = posix_acl_from_xattr(value, size);
diff --git a/fs/locks.c b/fs/locks.c
index e0b6a80..f5c4787 100644
--- a/fs/locks.c
+++ b/fs/locks.c
@@ -1452,7 +1452,7 @@ int setlease(struct file *filp, long arg
   struct inode *inode = dentry->d_inode;
   int error;

- if ((current->fsuid != inode->i_uid) && !capable(CAP_LEASE))
+ if (inode_task_same_uid(inode, current) && !capable(CAP_LEASE))
     return -EACCES;
   if (!S_ISREG(inode->i_mode))
     return -EINVAL;
@@ -1486,7 +1486,7 @@ int fcntl_setlease(unsigned int fd, stru
   struct inode *inode = dentry->d_inode;
   int error;

- if ((current->fsuid != inode->i_uid) && !capable(CAP_LEASE))
+ if (inode_task_same_uid(inode, current) && !capable(CAP_LEASE))
     return -EACCES;
   if (!S_ISREG(inode->i_mode))
     return -EINVAL;
diff --git a/fs/posix_acl.c b/fs/posix_acl.c
index aec931e..dd34bee 100644
--- a/fs/posix_acl.c
+++ b/fs/posix_acl.c
@@ -217,7 +217,7 @@ posix_acl_permission(struct inode *inode
                switch(pa->e_tag) {
                        case ACL_USER_OBJ:
```

```
     /* (May have been checked already) */
-                            if (inode->i_uid == current->fsuid)
+    if (inode_task_same_uid(inode, current))
                                    goto check_perm;
                             break;
                     case ACL_USER:
diff --git a/fs/utimes.c b/fs/utimes.c
index 558f581..77ad8c9 100644
--- a/fs/utimes.c
+++ b/fs/utimes.c
@@ -61,7 +61,7 @@ asmlinkage long sys_utime(char __user *
             if (IS_IMMUTABLE(inode))
                 goto mnt_drop_write_and_out;

-  if (current->fsuid != inode->i_uid &&
+  if (!inode_task_same_uid(inode, current) &&
       (error = vfs_permission(&nd, MAY_WRITE)) != 0)
    goto mnt_drop_write_and_out;
  }
@@ -119,7 +119,7 @@ long do_utimes(int dfd, char __user *fil
             if (IS_IMMUTABLE(inode))
    goto mnt_drop_write_and_out;

-  if (current->fsuid != inode->i_uid &&
+  if (!inode_task_same_uid(inode, current) &&
       (error = vfs_permission(&nd, MAY_WRITE)) != 0)
    goto mnt_drop_write_and_out;
  }
--
1.1.6
```

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: [RFC] [PATCH 3/4] uid_ns: replace i_uid check in fs/namespace.c
Posted by serue on Tue, 07 Nov 2006 04:20:07 GMT
View Forum Message <> Reply to Message

Replace a inode->i_uid==current->fsuid check in fs/namespace.c with
the inode_task_same_uid helper, which will eventually be checking
uid namespaces.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

```
 fs/namespace.c |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

bccafb526d224ae0c82c3370c0056eef5686bb4a
diff --git a/fs/namespace.c b/fs/namespace.c
index ec1a255..e199769 100644
--- a/fs/namespace.c
+++ b/fs/namespace.c
@@ -697,7 +697,7 @@ static int mount_is_safe(struct nameidat
  if (S_ISLNK(nd->dentry->d_inode->i_mode))
   return -EPERM;
  if (nd->dentry->d_inode->i_mode & S_ISVTX) {
-  if (current->uid != nd->dentry->d_inode->i_uid)
+  if (!inode_task_same_uid(nd->dentry->d_inode, current))
    return -EPERM;
  }
  /*
--
1.1.6
```

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: [RFC] [PATCH 4/4] uid_ns: Add filesystem uid checks
Posted by serue on Tue, 07 Nov 2006 04:20:32 GMT
View Forum Message <> Reply to Message

With user namespaces, two users in different namespaces might
have the same uid, but should not have access to each others
files.

Add a user_namespace pointer to the superblock, and check that
whenever the uid is checked.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

```
 fs/super.c         |    4 ++++
 include/linux/fs.h |    6 +++++-
 2 files changed, 9 insertions(+), 1 deletions(-)

b47f5fefa827b613c885fe851a8ce2642c2cb135
diff --git a/fs/super.c b/fs/super.c
index 95214f0..33354a8 100644
--- a/fs/super.c
```

```
+++ b/fs/super.c
@@ -37,6 +37,7 @@
 #include <linux/idr.h>
 #include <linux/kobject.h>
 #include <linux/mutex.h>
+#include <linux/user.h>
 #include <asm/uaccess.h>


@@ -81,6 +82,8 @@ static struct super_block *alloc_super(s
     * lock ordering than usbfs:
     */
   lockdep_set_class(&s->s_lock, &type->s_lock_key);
+  s->s_user_ns = current->nsproxy->user_ns;
+  get_user_ns(current->nsproxy->user_ns);
   down_write(&s->s_umount);
   s->s_count = S_BIAS;
   atomic_set(&s->s_active, 1);
@@ -109,6 +112,7 @@ out:
  */
 static inline void destroy_super(struct super_block *s)
 {
+ put_user_ns(s->s_user_ns);
  security_sb_free(s);
  kfree(s);
 }
diff --git a/include/linux/fs.h b/include/linux/fs.h
index 699c7b5..6aac556 100644
--- a/include/linux/fs.h
+++ b/include/linux/fs.h
@@ -279,6 +279,7 @@ extern int dir_notify_enable;
 #include <linux/sched.h>
 #include <linux/mutex.h>
 #include <linux/kevent.h>
+#include <linux/nsproxy.h>

 #include <asm/atomic.h>
 #include <asm/semaphore.h>
@@ -294,6 +295,7 @@ struct kstatfs;
 struct vm_area_struct;
 struct vfsmount;
 struct pagevec;
+struct user_namespace;

 extern void __init inode_init(unsigned long);
 extern void __init inode_init_early(void);
@@ -982,6 +984,7 @@ struct super_block {
  unsigned char  s_blocksize_bits;
```

```
  unsigned char  s_dirt;
  unsigned long long s_maxbytes; /* Max file size */
+ struct user_namespace *s_user_ns;
  struct file_system_type *s_type;
  struct super_operations *s_op;
  struct dquot_operations *dq_op;
@@ -2160,7 +2163,8 @@ static inline void free_secdata(void *se
 static inline int inode_task_same_uid(struct inode *ino,
   struct task_struct *tsk)
 {
- return (ino->i_uid == tsk->fsuid);
+ return (ino->i_uid == tsk->fsuid &&
+  ino->i_sb->s_user_ns == current->nsproxy->user_ns);
 }
 #endif /* __KERNEL__ */
 #endif /* _LINUX_FS_H */
--
1.1.6
```

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by Herbert Poetzl on Wed, 08 Nov 2006 00:52:12 GMT
View Forum Message <> Reply to Message

On Mon, Nov 06, 2006 at 10:18:14PM -0600, Serge E. Hallyn wrote:
> Cedric has previously sent out a patchset
> (http://lists.osdl.org/pipermail/containers/2006-August/000078.html)
> impplementing the very basics of a user namespace. It ignores
> filesystem access checks, so that uid 502 in one namespace could
> access files belonging to uid 502 in another namespace, if the
> containers were so set up.
>
> This isn't necessarily bad, since proper container setup should
> prevent problems. However there has been concern, so here is a
> patchset which takes one course in addressing the concern.
>
> It adds a user namespace pointer to every superblock, and to
> enhances fsuid equivalence checks with a (inode->i_sb->s_uid_ns ==
> current->nsproxy->uid_ns) comparison.

I don't consider that a good idea as it means that a filesystem
(or to be precise, a superblock) can only belong to one specific
namespace, which is not very useful for shared setups

Linux-VServer provides a mechanism to do per inode (and per
nfs mount) tagging for similar 'security' and more important
for disk space accounting and limiting, which permits to have
different disk limits, quota and access on a shared partition

i.e. I do not like it

best,
Herbert

> I've tested this as follows:
>
> Created a bare-minimum loopback filesystem which has su, ps, touch,
> and sh and requisites (like /etc/pam.d). Under that, created a user
> hallyn with the same uid as user hallyn on the root filesystem.
> Under both /home/hallyn and /mnt/0/home/hallyn (/home/hallyn on the
> loopbackfs) created a directory 'priv' with 0700 perms.
>
> unsharens -U /bin/sh
> su hallyn
> ls /home/hallyn/priv
> (permission denied)
> mount -o loop /usr/src/disk.img /mnt/0
> mount -t proc none /mnt/0/proc
> mount -t devpts none /mnt/0/dev/pts
> chroot /mnt/0
> su hallyn
> ls /home/hallyn/priv
> ab
>
> And, finally, of course
>
> mount -o loop /usr/src/disk.img /mnt/0
> mount -t proc none /mnt/0/proc
> mount -t devpts none /mnt/0/dev/pts
> unsharens -U /bin/sh
> chroot /mnt/0
> su hallyn
> ls /home/hallyn/priv
> (permission denied)
>
> This is only a rough prototype to start some discussion. i.e. I
> ignore groups, so kernel/sys.c:in_group_p() for instance will need to be
> updated.
>
> A few issues to be discussed:
>
> 1. I am not doing anything about root access. There are several ways we

> can address this.
>
> a. implement CAP_NS_OVERRIDE, without which cross-ns access is
>   not allowed
> b. just don't allow any cross-ns access at all
> c. a more complicated scheme where root process in parent and child
>   namespaces can access each other until somehow the
>   parent-ns cuts off the child's access.
>
> 2. This patch takes the easy route of adding user_ns pointers to the
> superblock. It would be very nice to add it to the vfsmount instead, so
> that admins could simply mount --bind into various namespaces, rather
> than having to use completely separate filesystems. However several
> fsuid equivalence checks happen with only an inode available. The
> hardest to address so far appear to be fs/namei.c:generic_permission as
> called from, say nfs, fs/generic_acl.c:generic_acl_set, and
> fs/attr.c:inode_change_ok called from jffs2.
>
> Still, putting the user_ns in the superblock and forcing the use
> of separate filesystems (i.e. through a lightweight stackable
> read-only filesystem) isn't *so* bad, is it?
>
> thanks,
> -serge
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by Trond Myklebust on Wed, 08 Nov 2006 17:46:10 GMT
View Forum Message <> Reply to Message

On Wed, 2006-11-08 at 01:52 +0100, Herbert Poetzl wrote:
> On Mon, Nov 06, 2006 at 10:18:14PM -0600, Serge E. Hallyn wrote:
> > Cedric has previously sent out a patchset
> > (http://lists.osdl.org/pipermail/containers/2006-August/000078.html)
> > impplementing the very basics of a user namespace. It ignores
> > filesystem access checks, so that uid 502 in one namespace could
> > access files belonging to uid 502 in another namespace, if the
> > containers were so set up.
> >
> > This isn't necessarily bad, since proper container setup should

> > prevent problems. However there has been concern, so here is a
> > patchset which takes one course in addressing the concern.
> >
> > It adds a user namespace pointer to every superblock, and to
> > enhances fsuid equivalence checks with a (inode->i_sb->s_uid_ns ==
> > current->nsproxy->uid_ns) comparison.
>
> I don't consider that a good idea as it means that a filesystem
> (or to be precise, a superblock) can only belong to one specific
> namespace, which is not very useful for shared setups
>
> Linux-VServer provides a mechanism to do per inode (and per
> nfs mount) tagging for similar 'security' and more important
> for disk space accounting and limiting, which permits to have
> different disk limits, quota and access on a shared partition
>
> i.e. I do not like it

Indeed. I discussed this with Eric at the kernel summit this summer and explained my reservations. As far as I'm concerned, tagging superblocks with a container label is an unacceptable hack since it completely breaks NFS caching semantics.

Cheers,
  Trond

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by ebiederm on Wed, 08 Nov 2006 20:34:09 GMT
View Forum Message <> Reply to Message

Trond Myklebust <trond.myklebust@fys.uio.no> writes:

> On Wed, 2006-11-08 at 01:52 +0100, Herbert Poetzl wrote:
>> On Mon, Nov 06, 2006 at 10:18:14PM -0600, Serge E. Hallyn wrote:
>> > Cedric has previously sent out a patchset
>> > (http://lists.osdl.org/pipermail/containers/2006-August/000078.html)
>> > impplementing the very basics of a user namespace. It ignores
>> > filesystem access checks, so that uid 502 in one namespace could
>> > access files belonging to uid 502 in another namespace, if the
>> > containers were so set up.
>> >
>> > This isn't necessarily bad, since proper container setup should

>> > prevent problems. However there has been concern, so here is a
>> > patchset which takes one course in addressing the concern.
>> >
>> > It adds a user namespace pointer to every superblock, and to
>> > enhances fsuid equivalence checks with a (inode->i_sb->s_uid_ns ==
>> > current->nsproxy->uid_ns) comparison.
>>
>> I don't consider that a good idea as it means that a filesystem
>> (or to be precise, a superblock) can only belong to one specific
>> namespace, which is not very useful for shared setups
>>
>> Linux-VServer provides a mechanism to do per inode (and per
>> nfs mount) tagging for similar 'security' and more important
>> for disk space accounting and limiting, which permits to have
>> different disk limits, quota and access on a shared partition
>>
>> i.e. I do not like it
>
> Indeed. I discussed this with Eric at the kernel summit this summer and
> explained my reservations. As far as I'm concerned, tagging superblocks
> with a container label is an unacceptable hack since it completely
> breaks NFS caching semantics.

As I recall there are two basic issues.

Putting the default on the mount structure instead of the superblock
for filesystems that are not uid namespaces aware sounded reasonable,
and allowed certain classes of sharing between namespaces where they
agreed on a subset of the uids (especially for read-only data).

The other was to have a mechanism that allows a uid namespace aware
filesystem (like some of the distributed filesystems can be) to perform
the mapping on their own.

Some mostly this is a case of simply not going far enough in the uid
namespace direction.

Eric

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by Herbert Poetzl on Wed, 08 Nov 2006 21:27:01 GMT
View Forum Message <> Reply to Message

On Wed, Nov 08, 2006 at 01:34:09PM -0700, Eric W. Biederman wrote:
> Trond Myklebust <trond.myklebust@fys.uio.no> writes:
>
> > On Wed, 2006-11-08 at 01:52 +0100, Herbert Poetzl wrote:
> >> On Mon, Nov 06, 2006 at 10:18:14PM -0600, Serge E. Hallyn wrote:
> >> > Cedric has previously sent out a patchset
> >> > (http://lists.osdl.org/pipermail/containers/2006-August/000078.html)
> >> > impplementing the very basics of a user namespace. It ignores
> >> > filesystem access checks, so that uid 502 in one namespace could
> >> > access files belonging to uid 502 in another namespace, if the
> >> > containers were so set up.
> >> >
> >> > This isn't necessarily bad, since proper container setup should
> >> > prevent problems. However there has been concern, so here is a
> >> > patchset which takes one course in addressing the concern.
> >> >
> >> > It adds a user namespace pointer to every superblock, and to
> >> > enhances fsuid equivalence checks with a (inode->i_sb->s_uid_ns ==
> >> > current->nsproxy->uid_ns) comparison.
> >>
> >> I don't consider that a good idea as it means that a filesystem
> >> (or to be precise, a superblock) can only belong to one specific
> >> namespace, which is not very useful for shared setups
> >>
> >> Linux-VServer provides a mechanism to do per inode (and per
> >> nfs mount) tagging for similar 'security' and more important
> >> for disk space accounting and limiting, which permits to have
> >> different disk limits, quota and access on a shared partition
> >>
> >> i.e. I do not like it
> >
> > Indeed. I discussed this with Eric at the kernel summit this summer and
> > explained my reservations. As far as I'm concerned, tagging superblocks
> > with a container label is an unacceptable hack since it completely
> > breaks NFS caching semantics.
>
> As I recall there are two basic issues.
>
> Putting the default on the mount structure instead of the superblock
> for filesystems that are not uid namespaces aware sounded reasonable,
> and allowed certain classes of sharing between namespaces where they
> agreed on a subset of the uids (especially for read-only data).

yes, that is especially interesting for --bind mounts
when you 'know' that you will dedicate a certain
sub-tree to one context/guest

> The other was to have a mechanism that allows a uid namespace aware

> filesystem (like some of the distributed filesystems can be) to perform
> the mapping on their own.

Linux-VServer currently provides different 'tagging'
methods to make filesystems context aware, some of
them are based on reusing some (upper 8/16) bits of
uid and gid, others store the context id inside
(currently) unused places in the on disk inodes

those are currently working for ext2/3, jfs, xfs,
reiser and ocfs2 as well as nfs

HTH,
Herbert

> Some mostly this is a case of simply not going far enough in the uid
> namespace direction.
>
> Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by serue on Wed, 08 Nov 2006 21:54:49 GMT
View Forum Message <> Reply to Message

Quoting Herbert Poetzl (herbert@13thfloor.at):
> On Wed, Nov 08, 2006 at 01:34:09PM -0700, Eric W. Biederman wrote:
> > Trond Myklebust <trond.myklebust@fys.uio.no> writes:
> >
> > > On Wed, 2006-11-08 at 01:52 +0100, Herbert Poetzl wrote:
> > > > On Mon, Nov 06, 2006 at 10:18:14PM -0600, Serge E. Hallyn wrote:
> > > > > Cedric has previously sent out a patchset
> > > > > (http://lists.osdl.org/pipermail/containers/2006-August/000078.html)
> > > > > impplementing the very basics of a user namespace. It ignores
> > > > > filesystem access checks, so that uid 502 in one namespace could
> > > > > access files belonging to uid 502 in another namespace, if the
> > > > > containers were so set up.
> > > > >
> > > > > This isn't necessarily bad, since proper container setup should
> > > > > prevent problems. However there has been concern, so here is a
> > > > > patchset which takes one course in addressing the concern.
> > > > >
> > > > > It adds a user namespace pointer to every superblock, and to
> > > > > enhances fsuid equivalence checks with a (inode->i_sb->s_uid_ns ==

> > >> > current->nsproxy->uid_ns) comparison.
> > >>
> > >> I don't consider that a good idea as it means that a filesystem
> > >> (or to be precise, a superblock) can only belong to one specific
> > >> namespace, which is not very useful for shared setups
> > >>
> > >> Linux-VServer provides a mechanism to do per inode (and per
> > >> nfs mount) tagging for similar 'security' and more important
> > >> for disk space accounting and limiting, which permits to have
> > >> different disk limits, quota and access on a shared partition
> > >>
> > >> i.e. I do not like it
> > >
> > > Indeed. I discussed this with Eric at the kernel summit this summer and
> > > explained my reservations. As far as I'm concerned, tagging superblocks
> > > with a container label is an unacceptable hack since it completely
> > > breaks NFS caching semantics.

So from your pov the same objection would apply to tagging vfsmounts,
or not?

What is the scenario where the caching is broken? It can't be multiple
clients accessing the same NFS export from the same NFS service container,
since that would just be an erroneous setup, right?

> > As I recall there are two basic issues.
> >
> > Putting the default on the mount structure instead of the superblock
> > for filesystems that are not uid namespaces aware sounded reasonable,
> > and allowed certain classes of sharing between namespaces where they
> > agreed on a subset of the uids (especially for read-only data).
>
> yes, that is especially interesting for --bind mounts
> when you 'know' that you will dedicate a certain
> sub-tree to one context/guest

Ok, so you wouldn't object to a patch which tagged vfsmounts?

I guess a NULL vfsmnt->user_ns pointer would mean ignore user_ns and
only apply uid checks (useful for ro bind mount of /usr into multiple
containers).

That of course wouldn't preclude also tagging inodes in later patches.

If you do object, then I can jump straight to tagging inodes with a
container, though that seems more likely to interfere conceptually
with any filesystems which are uid namespace aware.

thanks,
-serge

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by Herbert Poetzl on Thu, 09 Nov 2006 00:42:44 GMT
View Forum Message <> Reply to Message

On Wed, Nov 08, 2006 at 03:54:49PM -0600, Serge E. Hallyn wrote:
> Quoting Herbert Poetzl (herbert@13thfloor.at):
> > On Wed, Nov 08, 2006 at 01:34:09PM -0700, Eric W. Biederman wrote:
> > > Trond Myklebust <trond.myklebust@fys.uio.no> writes:
> > >
> > > > On Wed, 2006-11-08 at 01:52 +0100, Herbert Poetzl wrote:
> > > >> On Mon, Nov 06, 2006 at 10:18:14PM -0600, Serge E. Hallyn wrote:
> > > >> > Cedric has previously sent out a patchset
> > > >> > (http://lists.osdl.org/pipermail/containers/2006-August/000078.html)
> > > >> > impplementing the very basics of a user namespace. It ignores
> > > >> > filesystem access checks, so that uid 502 in one namespace could
> > > >> > access files belonging to uid 502 in another namespace, if the
> > > >> > containers were so set up.
> > > >> >
> > > >> > This isn't necessarily bad, since proper container setup should
> > > >> > prevent problems. However there has been concern, so here is a
> > > >> > patchset which takes one course in addressing the concern.
> > > >> >
> > > >> > It adds a user namespace pointer to every superblock, and to
> > > >> > enhances fsuid equivalence checks with a (inode->i_sb->s_uid_ns ==
> > > >> > current->nsproxy->uid_ns) comparison.
> > > >>
> > > >> I don't consider that a good idea as it means that a filesystem
> > > >> (or to be precise, a superblock) can only belong to one specific
> > > >> namespace, which is not very useful for shared setups
> > > >>
> > > >> Linux-VServer provides a mechanism to do per inode (and per
> > > >> nfs mount) tagging for similar 'security' and more important
> > > >> for disk space accounting and limiting, which permits to have
> > > >> different disk limits, quota and access on a shared partition
> > > >>
> > > >> i.e. I do not like it
> > > >
> > > > Indeed. I discussed this with Eric at the kernel summit this
> > > > summer and explained my reservations. As far as I'm concerned,
> > > > tagging superblocks with a container label is an unacceptable

> > > > hack since it completely breaks NFS caching semantics.
>
> So from your pov the same objection would apply to tagging vfsmounts,
> or not?
>
> What is the scenario where the caching is broken? It can't be multiple
> clients accessing the same NFS export from the same NFS service
> container, since that would just be an erroneous setup, right?
>
> > > As I recall there are two basic issues.
> > >
> > > Putting the default on the mount structure instead of the
> > > superblock for filesystems that are not uid namespaces aware
> > > sounded reasonable, and allowed certain classes of sharing between
> > > namespaces where they agreed on a subset of the uids (especially
> > > for read-only data).
> >
> > yes, that is especially interesting for --bind mounts
> > when you 'know' that you will dedicate a certain
> > sub-tree to one context/guest
>
> Ok, so you wouldn't object to a patch which tagged vfsmounts?

I would not object a vfsmount based tagging iif that would
still allow untagged vfsmounts where the the 'tagging' can
be inode based (either uid/gid or xattr or internal)

> I guess a NULL vfsmnt->user_ns pointer would mean ignore user_ns and
> only apply uid checks (useful for ro bind mount of /usr into multiple
> containers).

might as well work for our purpose, but it brings up another
question, regarding the 'control' over this feature, because
natrually it doesn't make too much sense if a context based
disk limit can be circumvented by unsharing the namespace and
doing a --bind mount :)

> That of course wouldn't preclude also tagging inodes in later patches.
>
> If you do object, then I can jump straight to tagging inodes with a
> container, though that seems more likely to interfere conceptually
> with any filesystems which are uid namespace aware.

like virtual filesystems and?

I think it would be interesting to discuss filesystem level
context tagging sooner or later (not sure this is the right
time for that though, first working bind mounts would be

really great :)

TIA,
Herbert

> thanks,
> -serge
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by ebiederm on Thu, 09 Nov 2006 13:26:13 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> So from your pov the same objection would apply to tagging vfsmounts,
> or not?

No. The issue is that the NFS server merges different mounts to the
same nfs server into the same superblock.

> What is the scenario where the caching is broken? It can't be multiple
> clients accessing the same NFS export from the same NFS service container,
> since that would just be an erroneous setup, right?

>
>> > As I recall there are two basic issues.
>> >
>> > Putting the default on the mount structure instead of the superblock
>> > for filesystems that are not uid namespaces aware sounded reasonable,
>> > and allowed certain classes of sharing between namespaces where they
>> > agreed on a subset of the uids (especially for read-only data).
>>
>> yes, that is especially interesting for --bind mounts
>> when you 'know' that you will dedicate a certain
>> sub-tree to one context/guest
>
> Ok, so you wouldn't object to a patch which tagged vfsmounts?
>
> I guess a NULL vfsmnt->user_ns pointer would mean ignore user_ns and

> only apply uid checks (useful for ro bind mount of /usr into multiple
> containers).

Bind mounts are peculiar. But I think as long as you charged the to
the context in which they happen (don't do the bind until after you switch
the user_ns. You should be fine.

> That of course wouldn't preclude also tagging inodes in later patches.
>
> If you do object, then I can jump straight to tagging inodes with a
> container, though that seems more likely to interfere conceptually
> with any filesystems which are uid namespace aware.

I'm pretty certain tagging inodes is the wrong approach. You want
a callback that allows the filesystem to make that determination,
a uid namespace aware filesystem.

Remote filesystems will be able to do things like tell you a particular
file is owned by "user@domain" which can get translated into a uid, uid_ns pair.

Where tagging the inode becomes a problem is when things like joe@domain1 is
fred@domain2, and treats those two users the same. I don't know if anything
actually supports that today but that is an interesting case to handle.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by serue on Thu, 09 Nov 2006 16:50:09 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> > So from your pov the same objection would apply to tagging vfsmounts,
> > or not?
>
> No. The issue is that the NFS server merges different mounts to the
> same nfs server into the same superblock.
>
> > What is the scenario where the caching is broken? It can't be multiple
> > clients accessing the same NFS export from the same NFS service container,
> > since that would just be an erroneous setup, right?
>

> >
> >> > As I recall there are two basic issues.
> >> >
> >> > Putting the default on the mount structure instead of the superblock
> >> > for filesystems that are not uid namespaces aware sounded reasonable,
> >> > and allowed certain classes of sharing between namespaces where they
> >> > agreed on a subset of the uids (especially for read-only data).
> >>
> >> yes, that is especially interesting for --bind mounts
> >> when you 'know' that you will dedicate a certain
> >> sub-tree to one context/guest
> >
> > Ok, so you wouldn't object to a patch which tagged vfsmounts?
> >
> > I guess a NULL vfsmnt->user_ns pointer would mean ignore user_ns and
> > only apply uid checks (useful for ro bind mount of /usr into multiple
> > containers).
>
> Bind mounts are peculiar. But I think as long as you charged the to
> the context in which they happen (don't do the bind until after you switch
> the user_ns. You should be fine.

Presumably container setup would be somewhat like system boot - you'd
start with a shared / filesystem, unshare user namespace, construct your
new /, pivot_root, and unmount /old_root, so you end up with all
vfsmounts accessible from the container having the correct user_ns.

-serge

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by Herbert Poetzl on Thu, 09 Nov 2006 17:17:01 GMT
View Forum Message <> Reply to Message

On Thu, Nov 09, 2006 at 10:50:09AM -0600, Serge E. Hallyn wrote:
> Quoting Eric W. Biederman (ebiederm@xmission.com):
> > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> >
> > > So from your pov the same objection would apply to tagging vfsmounts,
> > > or not?
> >
> > No. The issue is that the NFS server merges different mounts to the
> > same nfs server into the same superblock.
> >

> > > What is the scenario where the caching is broken? It can't be
> > > multiple clients accessing the same NFS export from the same NFS
> > > service container, since that would just be an erroneous setup,
> > > right?
> >
> > >
> > >> > As I recall there are two basic issues.
> > >> >
> > >> > Putting the default on the mount structure instead of the
> > >> > superblock for filesystems that are not uid namespaces aware
> > >> > sounded reasonable, and allowed certain classes of sharing
> > >> > between namespaces where they agreed on a subset of the uids
> > >> > (especially for read-only data).
> > >>
> > >> yes, that is especially interesting for --bind mounts
> > >> when you 'know' that you will dedicate a certain
> > >> sub-tree to one context/guest
> > >
> > > Ok, so you wouldn't object to a patch which tagged vfsmounts?
> > >
> > > I guess a NULL vfsmnt->user_ns pointer would mean ignore user_ns and
> > > only apply uid checks (useful for ro bind mount of /usr into multiple
> > > containers).
> >
> > Bind mounts are peculiar. But I think as long as you charged
> > the to the context in which they happen (don't do the bind
> > until after you switch the user_ns. You should be fine.
>
> Presumably container setup would be somewhat like system boot - you'd
> start with a shared / filesystem, unshare user namespace, construct your
> new /, pivot_root, and unmount /old_root, so you end up with all
> vfsmounts accessible from the container having the correct user_ns.

well, once again that is a very narrow view to the
real picture, what about the following cases:

 - folks who _share_ certain filesystems between different
   guests (maybe for cooperation or just readonly to save
   resource)

 - folks who still want a way to access and or
   andminsitrate the guests (without going through
   ssh or whatever, e.g. for bulk updates)

 - prestructured setups (like build roots) which require
   pre configured mounts to work ...

best,

Herbert

> -serge
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by serue on Thu, 09 Nov 2006 17:35:49 GMT
View Forum Message <> Reply to Message

Quoting Herbert Poetzl (herbert@13thfloor.at):
> On Thu, Nov 09, 2006 at 10:50:09AM -0600, Serge E. Hallyn wrote:
> > Quoting Eric W. Biederman (ebiederm@xmission.com):
> > > "Serge E. Hallyn" <serue@us.ibm.com> writes:
> > >
> > > > So from your pov the same objection would apply to tagging vfsmounts,
> > > > or not?
> > >
> > > No. The issue is that the NFS server merges different mounts to the
> > > same nfs server into the same superblock.
> > >
> > > > What is the scenario where the caching is broken? It can't be
> > > > multiple clients accessing the same NFS export from the same NFS
> > > > service container, since that would just be an erroneous setup,
> > > > right?
> > >
> > > >
> > > >> > As I recall there are two basic issues.
> > > >> >
> > > >> > Putting the default on the mount structure instead of the
> > > >> > superblock for filesystems that are not uid namespaces aware
> > > >> > sounded reasonable, and allowed certain classes of sharing
> > > >> > between namespaces where they agreed on a subset of the uids
> > > >> > (especially for read-only data).
> > > >>
> > > >> yes, that is especially interesting for --bind mounts
> > > >> when you 'know' that you will dedicate a certain
> > > >> sub-tree to one context/guest
> > > >
> > > > Ok, so you wouldn't object to a patch which tagged vfsmounts?
> > > >

> > > > I guess a NULL vfsmnt->user_ns pointer would mean ignore user_ns and
> > > > only apply uid checks (useful for ro bind mount of /usr into multiple
> > > > containers).
> > >
> > > Bind mounts are peculiar. But I think as long as you charged
> > > the to the context in which they happen (don't do the bind
> > > until after you switch the user_ns. You should be fine.
> >
> > Presumably container setup would be somewhat like system boot - you'd
> > start with a shared / filesystem, unshare user namespace, construct your
> > new /, pivot_root, and unmount /old_root, so you end up with all
> > vfsmounts accessible from the container having the correct user_ns.
>
> well, once again that is a very narrow view to the

why thanks

> real picture, what about the following cases:
>
> - folks who _share_ certain filesystems between different
>   guests (maybe for cooperation or just readonly to save
>   resource)

They can just mount --bind the same tree into multiple containers.
Or, they can use a shared filesystem like the initial /. (I intend for
vfsmount->mnt_user_ns == NULL to mean ignore user namespace checks.)

> - folks who still want a way to access and or
>   andminsitrate the guests (without going through
>   ssh or whatever, e.g. for bulk updates)

In addition to the shared mounts, Cedric has a bind_ns which lets you
enter another namespace. I think he's sent that patch out to the
containers list, but if he hasn't I expect he will be soon.

> - prestructured setups (like build roots) which require
>   pre configured mounts to work ...

i don't see why having the container setup script set these
up is a restriction here.

-serge

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC] [PATCH 1/4] uid_ns: introduce inode uid check helper
Posted by Cedric Le Goater on Thu, 09 Nov 2006 20:05:43 GMT
View Forum Message <> Reply to Message

> @@ -184,7 +184,7 @@ int generic_permission(struct inode *ino
> {
> umode_t mode = inode->i_mode;
>
> - if (current->fsuid == inode->i_uid)
> + if (inode_task_same_uid(inode, current))
> mode >>= 6;
> else {

Looking at the source of the above code in the email, I get :

@@ -184,7 +184,7 @@ int generic_permission(struct inode *ino
{
umode_t mode =3D inode->i_mode;
=

- if (current->fsuid =3D=3D inode->i_uid)
+ if (inode_task_same_uid(inode, current))
mode >>=3D 6;
else {

Where are those ugly '=3D' coming from ? is it my mailer or do you
get them also ?

thanks,

C.

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by Cedric Le Goater on Thu, 09 Nov 2006 20:12:15 GMT
View Forum Message <> Reply to Message

> In addition to the shared mounts, Cedric has a bind_ns which lets you
> enter another namespace. I think he's sent that patch out to the
> containers list, but if he hasn't I expect he will be soon.

When updated for the latest -mm, I will.

C.

Subject: Re: [RFC] [PATCH 1/4] uid_ns: introduce inode uid check helper
Posted by Sam Vilain on Sun, 12 Nov 2006 22:43:12 GMT
View Forum Message <> Reply to Message

Cedric Le Goater wrote:
> @@ -184,7 +184,7 @@ int generic_permission(struct inode *ino
> {
> umode_t mode =3D inode->i_mode;
> =
>
> - if (current->fsuid =3D=3D inode->i_uid)
> + if (inode_task_same_uid(inode, current))
> mode >>=3D 6;
> else {
>
>
>
> Where are those ugly '=3D' coming from ? is it my mailer or do you
> get them also ?
>

The message is Content-Transfer-Encoding: quoted-printable; that's where
they're coming from.

Why you're seeing them is a harder question, especially given I didn't,
and I appear to be using the same mailer as you...

Sam.

Subject: Re: [RFC] [PATCH 0/4] uid_ns: introduction
Posted by serue on Thu, 23 Nov 2006 03:09:54 GMT
View Forum Message <> Reply to Message

Quoting Herbert Poetzl (herbert@13thfloor.at):
> On Wed, Nov 08, 2006 at 03:54:49PM -0600, Serge E. Hallyn wrote:
> > Quoting Herbert Poetzl (herbert@13thfloor.at):
> > > On Wed, Nov 08, 2006 at 01:34:09PM -0700, Eric W. Biederman wrote:
> > > > Trond Myklebust <trond.myklebust@fys.uio.no> writes:
> > > >
> > > > > On Wed, 2006-11-08 at 01:52 +0100, Herbert Poetzl wrote:
> > > > >> On Mon, Nov 06, 2006 at 10:18:14PM -0600, Serge E. Hallyn wrote:
> > > > >> > Cedric has previously sent out a patchset
> > > > >> > (http://lists.osdl.org/pipermail/containers/2006-August/000078.html)
> > > > >> > impplementing the very basics of a user namespace. It ignores
> > > > >> > filesystem access checks, so that uid 502 in one namespace could
> > > > >> > access files belonging to uid 502 in another namespace, if the
> > > > >> > containers were so set up.
> > > > >> >
> > > > >> > This isn't necessarily bad, since proper container setup should
> > > > >> > prevent problems. However there has been concern, so here is a
> > > > >> > patchset which takes one course in addressing the concern.
> > > > >> >
> > > > >> > It adds a user namespace pointer to every superblock, and to
> > > > >> > enhances fsuid equivalence checks with a (inode->i_sb->s_uid_ns ==
> > > > >> > current->nsproxy->uid_ns) comparison.
> > > > >>
> > > > >> I don't consider that a good idea as it means that a filesystem
> > > > >> (or to be precise, a superblock) can only belong to one specific
> > > > >> namespace, which is not very useful for shared setups
> > > > >>
> > > > >> Linux-VServer provides a mechanism to do per inode (and per
> > > > >> nfs mount) tagging for similar 'security' and more important
> > > > >> for disk space accounting and limiting, which permits to have
> > > > >> different disk limits, quota and access on a shared partition
> > > > >>
> > > > >> i.e. I do not like it
> > > > >
> > > > > Indeed. I discussed this with Eric at the kernel summit this
> > > > > summer and explained my reservations. As far as I'm concerned,
> > > > > tagging superblocks with a container label is an unacceptable
> > > > > hack since it completely breaks NFS caching semantics.
> > >
> > > So from your pov the same objection would apply to tagging vfsmounts,
> > > or not?
> >
> > What is the scenario where the caching is broken? It can't be multiple
> > clients accessing the same NFS export from the same NFS service
> > container, since that would just be an erroneous setup, right?
> >
> > > > As I recall there are two basic issues.
> > > >

> > > > Putting the default on the mount structure instead of the
> > > > superblock for filesystems that are not uid namespaces aware
> > > > sounded reasonable, and allowed certain classes of sharing between
> > > > namespaces where they agreed on a subset of the uids (especially
> > > > for read-only data).
> > >
> > > yes, that is especially interesting for --bind mounts
> > > when you 'know' that you will dedicate a certain
> > > sub-tree to one context/guest
> >
> > Ok, so you wouldn't object to a patch which tagged vfsmounts?
>
> I would not object a vfsmount based tagging iif that would
> still allow untagged vfsmounts where the the 'tagging' can
> be inode based (either uid/gid or xattr or internal)
>
> > I guess a NULL vfsmnt->user_ns pointer would mean ignore user_ns and
> > only apply uid checks (useful for ro bind mount of /usr into multiple
> > containers).
>
> might as well work for our purpose, but it brings up another
> question, regarding the 'control' over this feature, because
> natrually it doesn't make too much sense if a context based
> disk limit can be circumvented by unsharing the namespace and
> doing a --bind mount :)

One quick and dirty solution would be to only let the initial namespace
do shared-ns mounts. Another would be to refuse doing a shared-ns bind
mount based on a non-shared mount. A third would be to introduce a new
capability letting you do the shared-ns mount.

Any other ideas?

I'm partial to the second, as it's nice and simple, so barring better
suggestions I'll plan on implementing that in the next patchset I send
out.

thanks,
-serge

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers