Subject: [RFC] [PATCH 0/3] user ns and vfs: Introduction
Posted by serue on Wed, 15 Nov 2006 17:40:19 GMT
View Forum Message <> Reply to Message

From: Serge E. Hallyn <serue@us.ibm.com>
Subject: [RFC] [PATCH 0/3] user ns and vfs: Introduction

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.

This patchset adds assigns each vfsmount to the user namespace
of the process which did the mount. It introduces a userns-shared
mount flag mainly to allow a filesystem to be used by a container
while it is setting up. It could also be used along with read-only
bind mounts to share, for instance, /usr among mutiple containers.

This patchset replaces the previous one, which annotated the
superblock.

Is this direction in which we want to go? For instance, would we
want to allow the notion of a uidmap so that user 500 (hallyn)'s
files on the host system are owned by uid 0 in a container which
hallyn started? It's my impression that that could only be cleanly
done with either a stackable filesystem to give us fresh inodes
inside the container. Also, would a uidmap map uid's as stored on
disk to the mapped uids, or would we want to only support a uidmap
for the whole user namespace?

My own impression is that we are better off enfocing isolation than
trying to provide actual uid mapping, but please argue and discuss.

thanks,
-serge

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

Subject: [RFC] [PATCH 1/3] user ns: add user_namespace ptr to vfsmount
Posted by serue on Wed, 15 Nov 2006 17:41:21 GMT
View Forum Message <> Reply to Message

From: Serge Hallyn <serue@us.ibm.com>
Subject: [RFC] [PATCH 1/3] user ns: add user_namespace ptr to vfsmount

Add user_namespace ptr to vfsmount, and define a helper to compare it
to the task's user_ns.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
---
 fs/namespace.c | 4 ++++
 include/linux/mount.h | 2 ++
 include/linux/sched.h | 13 +++++++++++++
 3 files changed, 19 insertions(+), 0 deletions(-)

diff --git a/fs/namespace.c b/fs/namespace.c
index e199769..cfd4584 100644
--- a/fs/namespace.c
+++ b/fs/namespace.c
@@ -25,6 +25,7 @@ #include <linux/namei.h>
 #include <linux/security.h>
 #include <linux/mount.h>
 #include <linux/ramfs.h>
+#include <linux/user.h>
 #include <asm/uaccess.h>
 #include <asm/unistd.h>
 #include "pnode.h"
@@ -56,6 +57,8 @@ struct vfsmount *alloc_vfsmnt(const char
  struct vfsmount *mnt = kmem_cache_alloc(mnt_cache, GFP_KERNEL);
  if (mnt) {
   memset(mnt, 0, sizeof(struct vfsmount));
+ mnt->mnt_user_ns = current->nsproxy->user_ns;
+ get_user_ns(mnt->mnt_user_ns);
   atomic_set(&mnt->mnt_count, 1);
   atomic_set(&mnt->mnt_writers, 0);
   INIT_LIST_HEAD(&mnt->mnt_hash);
@@ -109,6 +112,7 @@ void free_vfsmnt(struct vfsmount *mnt)
  if (mnt->mnt_sb)
   list_del(&mnt->mnt_sb_list);
  spin_unlock(&vfsmount_lock);
+ put_user_ns(mnt->mnt_user_ns);
  kfree(mnt->mnt_devname);
  kmem_cache_free(mnt_cache, mnt);
 }
diff --git a/include/linux/mount.h b/include/linux/mount.h
index 8402137..827793f 100644
--- a/include/linux/mount.h

```
+++ b/include/linux/mount.h
@@ -21,6 +21,7 @@ struct super_block;
 struct vfsmount;
 struct dentry;
 struct mnt_namespace;
+struct user_namespace;

 #define MNT_NOSUID 0x01
 #define MNT_NODEV 0x02
@@ -57,6 +58,7 @@ struct vfsmount {
  struct list_head mnt_slave; /* slave list entry */
  struct vfsmount *mnt_master; /* slave is on master->mnt_slave_list */
  struct mnt_namespace *mnt_ns; /* containing namespace */
+ struct user_namespace *mnt_user_ns; /* namespace for uid interpretation */
  int mnt_pinned;
 };

diff --git a/include/linux/sched.h b/include/linux/sched.h
index 47fbd23..b72357d 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -82,6 +82,8 @@ #include <linux/param.h>
 #include <linux/resource.h>
 #include <linux/timer.h>
 #include <linux/hrtimer.h>
+#include <linux/nsproxy.h>
+#include <linux/mount.h>

 #include <asm/processor.h>

@@ -1583,6 +1585,17 @@ extern int cond_resched(void);
 extern int cond_resched_lock(spinlock_t * lock);
 extern int cond_resched_softirq(void);

+static inline int task_mnt_same_uid(struct task_struct *tsk,
+      struct vfsmount *mnt)
+{
+ if (tsk->nsproxy == init_task.nsproxy)
+  return 1;
+ if (mnt->mnt_user_ns == tsk->nsproxy->user_ns);
+  return 1;
+ return 0;
+}
+
+
 /*
  * Does a critical section need to be broken due to another
  * task waiting?:
```

--
1.4.1

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

Subject: [RFC] [PATCH 2/3] user ns: hook permission
Posted by serue on Wed, 15 Nov 2006 17:41:38 GMT
View Forum Message <> Reply to Message

From: Serge Hallyn <serue@us.ibm.com>
Subject: [RFC] [PATCH 2/3] user ns: hook permission

Hook permission to check vfsmnt->user_ns against current.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
---
 fs/namei.c | 4 ++++
 1 files changed, 4 insertions(+), 0 deletions(-)

diff --git a/fs/namei.c b/fs/namei.c
index dc4ff80..edf7c16 100644
--- a/fs/namei.c
+++ b/fs/namei.c
@@ -246,6 +246,8 @@ int permission(struct inode *inode, int
    return -EACCES;
  }

+ if (nd && !task_mnt_same_uid(current, nd->mnt))
+ return -EACCES;

  /*
   * MAY_EXEC on regular files requires special handling: We override
@@ -433,6 +435,8 @@ static int exec_permission_lite(struct i
 {
  umode_t mode = inode->i_mode;

+ if (!task_mnt_same_uid(current, nd->mnt))
+ return -EACCES;
  if (inode->i_op && inode->i_op->permission)
   return -EAGAIN;

--
1.4.1

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

Subject: [RFC] [PATCH 3/3] user ns: implement shared mounts
Posted by serue on Wed, 15 Nov 2006 17:41:51 GMT
View Forum Message <> Reply to Message

From: Serge E. Hallyn <serue@us.ibm.com>
Subject: [RFC] [PATCH 3/3] user ns: implement shared mounts

Implement shared-ns mounts, which allow containers in different user
namespaces to share mounts. Without this, containers can obviously
never even be started.

Here is a sample smount.c (based on Miklos' version) which only
does a bind mount of arg1 onto arg2, but making the destination
a shared-ns mount.

```
int main(int argc, char *argv[])
{
 int type;
 if(argc != 3) {
  fprintf(stderr, "usage: %s src dest", argv[0]);
  return 1;
 }

 fprintf(stdout, "%s %s %s\n", argv[0], argv[1], argv[2]);

 type = MS_SHARE_NS | MS_BIND;
 setfsuid(getuid());

 if(mount(argv[1], argv[2], "none", type, "") == -1) {
  perror("mount");
  return 1;
 }
 return 0;
}
```

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

```
 fs/namespace.c       |   16 ++++++++++++----
 fs/pnode.h           |    1 +
 include/linux/fs.h   |    1 +
 include/linux/mount.h |    1 +
 include/linux/sched.h |    2 ++
```

```
 5 files changed, 17 insertions(+), 4 deletions(-)

diff --git a/fs/namespace.c b/fs/namespace.c
index cfd4584..ef91a48 100644
--- a/fs/namespace.c
+++ b/fs/namespace.c
@@ -281,6 +281,8 @@ static struct vfsmount *clone_mnt(struct
   }
   if (flag & CL_MAKE_SHARED)
    set_mnt_shared(mnt);
+  if (flag & CL_SHARE_NS)
+   mnt->mnt_flags |= MNT_SHARE_NS;

   /* stick the duplicate mount on the same expiry list
    * as the original if that was on one */
@@ -392,6 +394,7 @@ static int show_vfsmnt(struct seq_file *
   { MNT_NOSUID, ",nosuid" },
   { MNT_NODEV, ",nodev" },
   { MNT_NOEXEC, ",noexec" },
+  { MNT_SHARE_NS, ",share_uidns" },
   { MNT_NOATIME, ",noatime" },
   { MNT_NODIRATIME, ",nodiratime" },
   { 0, NULL }
@@ -981,11 +984,14 @@ static int do_change_type(struct nameida
 /*
  * do loopback mount.
  */
-static int do_loopback(struct nameidata *nd, char *old_name, int recurse)
+static int do_loopback(struct nameidata *nd, char *old_name, int recurse,
+       int uidns_share)
 {
  struct nameidata old_nd;
  struct vfsmount *mnt = NULL;
  int err = mount_is_safe(nd);
+ int flag = (uidns_share ? CL_SHARE_NS : 0);
+
  if (err)
   return err;
  if (!old_name || !*old_name)
@@ -1004,9 +1010,9 @@ static int do_loopback(struct nameidata

  err = -ENOMEM;
  if (recurse)
-  mnt = copy_tree(old_nd.mnt, old_nd.dentry, 0);
+  mnt = copy_tree(old_nd.mnt, old_nd.dentry, flag);
  else
-  mnt = clone_mnt(old_nd.mnt, old_nd.dentry, 0);
+  mnt = clone_mnt(old_nd.mnt, old_nd.dentry, flag);
```

```
  if (!mnt)
   goto out;
@@ -1517,6 +1523,8 @@ long do_mount(char *dev_name, char *dir_
   mnt_flags |= MNT_NOATIME;
  if (flags & MS_NODIRATIME)
   mnt_flags |= MNT_NODIRATIME;
+ if (flags & MS_SHARE_NS)
+  mnt_flags |= MNT_SHARE_NS;

  flags &= ~(MS_NOSUID | MS_NOEXEC | MS_NODEV | MS_ACTIVE |
      MS_NOATIME | MS_NODIRATIME);
@@ -1534,7 +1542,7 @@ long do_mount(char *dev_name, char *dir_
   retval = do_remount(&nd, flags & ~MS_REMOUNT, mnt_flags,
         data_page);
  else if (flags & MS_BIND)
-  retval = do_loopback(&nd, dev_name, flags & MS_REC);
+  retval = do_loopback(&nd, dev_name, flags & MS_REC, flags & MS_SHARE_NS);
  else if (flags & (MS_SHARED | MS_PRIVATE | MS_SLAVE | MS_UNBINDABLE))
   retval = do_change_type(&nd, flags);
  else if (flags & MS_MOVE)
diff --git a/fs/pnode.h b/fs/pnode.h
index d45bd8e..eb62f4c 100644
--- a/fs/pnode.h
+++ b/fs/pnode.h
@@ -22,6 +22,7 @@ #define CL_SLAVE       0x02
 #define CL_COPY_ALL   0x04
 #define CL_MAKE_SHARED   0x08
 #define CL_PROPAGATION   0x10
+#define CL_SHARE_NS   0x20

 static inline void set_mnt_shared(struct vfsmount *mnt)
 {
diff --git a/include/linux/fs.h b/include/linux/fs.h
index 730b2ac..961e6cb 100644
--- a/include/linux/fs.h
+++ b/include/linux/fs.h
@@ -121,6 +121,7 @@ #define MS_PRIVATE (1<<18) /* change to
 #define MS_SLAVE (1<<19) /* change to slave */
 #define MS_SHARED (1<<20) /* change to shared */
 #define MS_FROZEN (1<<21) /* Frozen by freeze_filesystems() */
+#define MS_SHARE_NS (1<<22) /* ignore user namespaces for permission */
 #define MS_ACTIVE (1<<30)
 #define MS_NOUSER (1<<31)

diff --git a/include/linux/mount.h b/include/linux/mount.h
index 827793f..5258677 100644
--- a/include/linux/mount.h
```

```
+++ b/include/linux/mount.h
@@ -36,6 +36,7 @@ #define MNT_SHRINKABLE 0x100
 #define MNT_SHARED 0x1000 /* if the vfsmount is a shared mount */
 #define MNT_UNBINDABLE 0x2000 /* if the vfsmount is a unbindable mount */
 #define MNT_PNODE_MASK 0x3000 /* propogation flag mask */
+#define MNT_SHARE_NS 0x4000  /* ignore user namespaces for permission */

 struct vfsmount {
  struct list_head mnt_hash;
diff --git a/include/linux/sched.h b/include/linux/sched.h
index b72357d..27c83ac 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1590,6 +1590,8 @@ static inline int task_mnt_same_uid(stru
 {
  if (tsk->nsproxy == init_task.nsproxy)
   return 1;
+ if (mnt->mnt_flags & MNT_SHARE_NS)
+  return 1;
  if (mnt->mnt_user_ns == tsk->nsproxy->user_ns);
   return 1;
  return 0;
--
1.4.1
```

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

Subject: Re: [RFC] [PATCH 0/3] user ns and vfs: Introduction
Posted by serue on Fri, 17 Nov 2006 15:19:24 GMT
View Forum Message <> Reply to Message

Quoting Serge E. Hallyn (serue@us.ibm.com):
> From: Serge E. Hallyn <serue@us.ibm.com>
> Subject: [RFC] [PATCH 0/3] user ns and vfs: Introduction
>
> 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.

Oh, and the real question, which i forgot to ask - for those
who objected to Cedric's patchset on the grounds of lack of file access

controls, does this patchset address your concerns?

It seems to me it provides isolation to those who want it, while leaving
the door open to a uid mapping solution (whether in a stackable fs, a
global-uidaware fs, or whatever) in the future.

thanks,
-serge

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