Subject: [RFC][PATCH 7/7]: Enable cloning PTY namespaces
Posted by Sukadev Bhattiprolu on Tue, 08 Apr 2008 22:00:54 GMT
View Forum Message <> Reply to Message

From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Subject: [RFC][PATCH 7/7]: Enable cloning PTY namespaces

Enable cloning PTY namespaces.

Note:
 We are out of clone_flags! This patch depends on Cedric Le Goater's
 clone64() patchset.

Changelog[v2]:
 [Serge Hallyn]: Use rcu to access sb->s_fs_info.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Signed-off-by: Serge Hallyn <serue@us.ibm.com>
Signed-off-by: Matt Helsley <matthltc@us.ibm.com>

```
---
 fs/devpts/inode.c         |   84 ++++++++++++++++++++++++++++++++++++++++++++++--
 include/linux/devpts_fs.h |   22 ++++++++++--
 include/linux/init_task.h |    1
 include/linux/nsproxy.h   |    2 +
 include/linux/sched.h     |    1
 kernel/fork.c             |    2 -
 kernel/nsproxy.c          |   17 ++++++++-
 7 files changed, 122 insertions(+), 7 deletions(-)

Index: 2.6.25-rc8-mm1/include/linux/sched.h
===================================================================
--- 2.6.25-rc8-mm1.orig/include/linux/sched.h 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/include/linux/sched.h 2008-04-08 14:27:41.000000000 -0700
@@ -28,6 +28,7 @@
 #define CLONE_NEWPID  0x20000000 /* New pid namespace */
 #define CLONE_NEWNET  0x40000000 /* New network namespace */
 #define CLONE_IO  0x80000000 /* Clone io context */
+#define CLONE_NEWPTS  0x0000000200000000ULL /* Clone pts ns */

 /*
  * Scheduling policies
Index: 2.6.25-rc8-mm1/include/linux/nsproxy.h
===================================================================
--- 2.6.25-rc8-mm1.orig/include/linux/nsproxy.h 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/include/linux/nsproxy.h 2008-04-08 14:27:41.000000000 -0700
@@ -8,6 +8,7 @@ struct mnt_namespace;
 struct uts_namespace;
```

```
 struct ipc_namespace;
 struct pid_namespace;
+struct pts_namespace;

 /*
  * A structure to contain pointers to all per-process
@@ -29,6 +30,7 @@ struct nsproxy {
  struct pid_namespace *pid_ns;
  struct user_namespace *user_ns;
  struct net       *net_ns;
+ struct pts_namespace *pts_ns;
 };
 extern struct nsproxy init_nsproxy;

Index: 2.6.25-rc8-mm1/include/linux/init_task.h
===================================================================
--- 2.6.25-rc8-mm1.orig/include/linux/init_task.h 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/include/linux/init_task.h 2008-04-08 14:27:41.000000000 -0700
@@ -78,6 +78,7 @@ extern struct nsproxy init_nsproxy;
  .mnt_ns  = NULL,      \
  INIT_NET_NS(net_ns)                                     \
  INIT_IPC_NS(ipc_ns)      \
+ .pts_ns  = &init_pts_ns,     \
  .user_ns = &init_user_ns,    \
 }

Index: 2.6.25-rc8-mm1/include/linux/devpts_fs.h
===================================================================
--- 2.6.25-rc8-mm1.orig/include/linux/devpts_fs.h 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/include/linux/devpts_fs.h 2008-04-08 14:27:41.000000000 -0700
@@ -31,7 +31,7 @@ extern struct pts_namespace init_pts_ns;

 static inline struct pts_namespace *current_pts_ns(void)
 {
- return &init_pts_ns;
+ return current->nsproxy->pts_ns;
 }

 static inline struct pts_namespace *pts_ns_from_inode(struct inode *inode)
@@ -61,7 +61,8 @@ struct tty_struct *devpts_get_tty(struct
 /* unlink */
 void devpts_pty_kill(struct pts_namespace *pts_ns, int number);

-static inline void free_pts_ns(struct kref *ns_kref) { }
+extern struct pts_namespace *new_pts_ns(void);
+extern void free_pts_ns(struct kref *kref);

 static inline struct pts_namespace *get_pts_ns(struct pts_namespace *ns)
```

```
 {
@@ -75,6 +76,15 @@ static inline void put_pts_ns(struct pts
   kref_put(&ns->kref, free_pts_ns);
 }

+static inline struct pts_namespace *copy_pts_ns(u64 flags,
+             struct pts_namespace *old_ns)
+{
+      if (flags & CLONE_NEWPTS)
+             return new_pts_ns();
+      else
+             return get_pts_ns(old_ns);
+}
+
 #else

 /* Dummy stubs in the no-pty case */
@@ -90,6 +100,14 @@ static inline struct pts_namespace *get_
 }

 static inline void put_pts_ns(struct pts_namespace *ns) { }
+
+static inline struct pts_namespace *copy_pts_ns(u64 flags,
+             struct pts_namespace *old_ns)
+{
+      if (flags & CLONE_NEWPTS)
+             return ERR_PTR(-EINVAL);
+      return old_ns;
+}
 #endif


Index: 2.6.25-rc8-mm1/fs/devpts/inode.c
===================================================================
--- 2.6.25-rc8-mm1.orig/fs/devpts/inode.c 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/fs/devpts/inode.c 2008-04-08 14:33:04.000000000 -0700
@@ -27,6 +27,7 @@

 extern int pty_limit;   /* Config limit on Unix98 ptys */
 static DECLARE_MUTEX(allocated_ptys_lock);
+static struct file_system_type devpts_fs_type;

 static struct {
   int setuid;
@@ -119,6 +120,57 @@ static const struct super_operations dev
   .show_options = devpts_show_options,
 };
```

```
+struct pts_namespace *new_pts_ns(void)
+{
+ struct pts_namespace *ns;
+
+ ns = kmalloc(sizeof(*ns), GFP_KERNEL);
+ if (!ns)
+  return ERR_PTR(-ENOMEM);
+
+ kref_init(&ns->kref);
+
+ ns->mnt = kern_mount_data(&devpts_fs_type, ns);
+ if (IS_ERR(ns->mnt)) {
+  kfree(ns);
+  return ERR_PTR(PTR_ERR(ns->mnt));
+ }
+
+ idr_init(&ns->allocated_ptys);
+
+ printk(KERN_NOTICE "Created pts-ns 0x%p\n", ns);
+
+ return ns;
+}
+
+void free_pts_ns(struct kref *ns_kref)
+{
+ struct pts_namespace *ns;
+
+ ns = container_of(ns_kref, struct pts_namespace, kref);
+
+ /*
+  * Clear s_fs_info here rather than in ->kill_sb(), since the pts_ns
+  * is invalid real soon now, but the ->kill_sb() will not happen
+  * until the last mntput(). And if some one is accessing this
+  * devpts mount from an ancestor pts ns, we may not be holding
+  * the last reference to this mnt.
+  *
+  * After clearing the pts_ns is NULL here, any acceses from parent
+  * will fail.
+  */
+ rcu_assign_pointer(ns->mnt->mnt_sb->s_fs_info, NULL);
+ mntput(ns->mnt);
+
+ /*
+  * TODO:
+  *      idr_remove_all(&ns->allocated_ptys); introduced in 2.6.23
+  */
+ idr_destroy(&ns->allocated_ptys);
+ kfree(ns);
```

```
+
+ printk(KERN_NOTICE "Freed pts-ns 0x%p\n", ns);
+}

 static int devpts_mknod(struct inode *dir, struct dentry *dentry,
    int mode, dev_t rdev)
@@ -217,7 +269,16 @@ static int devpts_test_sb(struct super_b

 static int devpts_set_sb(struct super_block *sb, void *data)
 {
- sb->s_fs_info = data;
+ /*
+  * new_pts_ns() mounts the pts namespace and free_pts_ns()
+  * drops the reference to the mount. i.e the s_fs_info is
+  * cleared and vfsmnt is released _before_ pts_namespace
+  * is freed.
+  *
+  * So we don't need a reference to the pts_namespace here
+  * (Getting a reference here will also cause circular reference).
+  */
+ rcu_assign_pointer(sb->s_fs_info, data);
  return set_anon_super(sb, NULL);
 }

@@ -232,7 +293,7 @@ static int devpts_get_sb(struct file_sys
  if (flags & MS_KERNMOUNT)
   ns = data;
  else
-  ns = &init_pts_ns;
+  ns = current_pts_ns();

  /* hereafter we're very simlar to get_sb_nodev */
  sb = sget(fs_type, devpts_test_sb, devpts_set_sb, ns);
@@ -286,6 +347,13 @@ int devpts_new_index(struct pts_namespac
  int index;
  int idr_ret;

+ /*
+  * If pts_ns is NULL, this must be an access from an ancestor-pts-ns
+  * which happened just as this pts-ns was freed. Fail the access
+  * from parent-pts-ns.
+  */
+ if (!pts_ns)
+  return -EAGAIN;
 retry:
  if (!idr_pre_get(&pts_ns->allocated_ptys, GFP_KERNEL)) {
   return -ENOMEM;
@@ -311,6 +379,7 @@ retry:
```

```
 void devpts_kill_index(struct pts_namespace *pts_ns, int idx)
 {
+ BUG_ON(pts_ns == NULL);

  down(&allocated_ptys_lock);
  idr_remove(&pts_ns->allocated_ptys, idx);
@@ -330,6 +399,7 @@ int devpts_pty_new( struct pts_namespace
  /* We're supposed to be given the slave end of a pty */
  BUG_ON(driver->type != TTY_DRIVER_TYPE_PTY);
  BUG_ON(driver->subtype != PTY_TYPE_SLAVE);
+ BUG_ON(pts_ns == NULL);

  mnt = pts_ns->mnt;
  root = mnt->mnt_root;
@@ -370,6 +440,14 @@ struct tty_struct *devpts_get_tty(struct
  struct dentry *dentry;
  struct tty_struct *tty;

+ /*
+  * If pts_ns is NULL, this must be an access from an ancestor-pts-ns
+  * which happened just as this pts-ns was freed. Fail the access
+  * from parent-pts-ns.
+  */
+ if (!pts_ns)
+  return ERR_PTR(-EAGAIN);
+
  mnt = pts_ns->mnt;

  dentry = get_node(mnt->mnt_root, number);
@@ -391,6 +469,8 @@ void devpts_pty_kill(struct pts_namespac
  struct dentry *dentry;
  struct dentry *root;

+ BUG_ON(pts_ns == NULL);
+
  root = pts_ns->mnt->mnt_root;

  dentry = get_node(root, number);
Index: 2.6.25-rc8-mm1/kernel/fork.c
===================================================================
--- 2.6.25-rc8-mm1.orig/kernel/fork.c 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/kernel/fork.c 2008-04-08 14:27:41.000000000 -0700
@@ -1714,7 +1714,7 @@ static long do_unshare(u64 unshare_flags
  if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
     CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
     CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER|
-    CLONE_NEWNET))
```

```
+    CLONE_NEWNET|CLONE_NEWPTS))
   goto bad_unshare_out;

  if ((err = unshare_thread(unshare_flags)))
Index: 2.6.25-rc8-mm1/kernel/nsproxy.c
===================================================================
--- 2.6.25-rc8-mm1.orig/kernel/nsproxy.c 2008-04-08 13:38:08.000000000 -0700
+++ 2.6.25-rc8-mm1/kernel/nsproxy.c 2008-04-08 14:27:41.000000000 -0700
@@ -21,6 +21,7 @@
 #include <linux/utsname.h>
 #include <linux/pid_namespace.h>
 #include <net/net_namespace.h>
+#include <linux/devpts_fs.h>
 #include <linux/ipc_namespace.h>

 static struct kmem_cache *nsproxy_cachep;
@@ -93,8 +94,17 @@ static struct nsproxy *create_new_namesp
   goto out_net;
  }

+ new_nsp->pts_ns = copy_pts_ns(flags, tsk->nsproxy->pts_ns);
+ if (IS_ERR(new_nsp->pts_ns)) {
+  err = PTR_ERR(new_nsp->pts_ns);
+  goto out_pts;
+ }
+
  return new_nsp;

+out_pts:
+ if (new_nsp->net_ns)
+  put_net(new_nsp->net_ns);
 out_net:
  if (new_nsp->user_ns)
   put_user_ns(new_nsp->user_ns);
@@ -131,7 +141,8 @@ int copy_namespaces(u64 flags, struct ta
  get_nsproxy(old_ns);

  if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
-    CLONE_NEWUSER | CLONE_NEWPID | CLONE_NEWNET)))
+    CLONE_NEWUSER | CLONE_NEWPID | CLONE_NEWNET |
+    CLONE_NEWPTS)))
   return 0;

  if (!capable(CAP_SYS_ADMIN)) {
@@ -170,6 +181,8 @@ void free_nsproxy(struct nsproxy *ns)
   put_pid_ns(ns->pid_ns);
  if (ns->user_ns)
   put_user_ns(ns->user_ns);
```

```
+ if (ns->pts_ns)
+  put_pts_ns(ns->pts_ns);
  put_net(ns->net_ns);
  kmem_cache_free(nsproxy_cachep, ns);
 }
@@ -184,7 +197,7 @@ int unshare_nsproxy_namespaces(u64 unsha
  int err = 0;

  if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
-         CLONE_NEWUSER | CLONE_NEWNET)))
+         CLONE_NEWUSER | CLONE_NEWNET | CLONE_NEWPTS)))
   return 0;

  if (!capable(CAP_SYS_ADMIN))
```