
Subject: [PATCH 9/9] namespaces: utsname: implement CLONE_NEWUTS flag
Posted by [serue](#) on Thu, 18 May 2006 15:51:14 GMT

[View Forum Message](#) <> [Reply to Message](#)

Implement a CLONE_NEWUTS flag, and use it at clone and sys_unshare.

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

```
include/linux/sched.h | 1 +
include/linux/utsname.h | 7 ++++++
kernel/fork.c | 20 ++++++++-----
kernel/nsproxy.c | 2 +-
kernel/utsname.c | 53 +++++++++++++++++++++++++++++++++++++++++++++++++++++
5 files changed, 79 insertions(+), 4 deletions(-)
```

```
8d097a939e5b69f068665d99a68d2deb13811d75
diff --git a/include/linux/sched.h b/include/linux/sched.h
index 3332d5e..55671b2 100644
```

```
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -62,6 +62,7 @@ struct exec_domain;
#define CLONE_UNTRACED 0x00800000 /* set if the tracing process can't force
CLONE_PTRACE on this clone */
#define CLONE_CHILD_SETTID 0x01000000 /* set the TID in the child */
#define CLONE_STOPPED 0x02000000 /* Start in stopped state */
+#define CLONE_NEWUTS 0x04000000 /* New utsname group? */
```

```
/*
 * List of flags we want to share for kernel threads,
diff --git a/include/linux/utsname.h b/include/linux/utsname.h
index 15dafa9..0d500fe 100644
```

```
--- a/include/linux/utsname.h
+++ b/include/linux/utsname.h
@@ -47,6 +47,8 @@ static inline void get_uts_ns(struct uts
}
```

```
#ifdef CONFIG_UTS_NS
+extern int unshare_utsname(unsigned long unshare_flags,
+ struct uts_namespace **new_uts);
extern int copy_utsname(int flags, struct task_struct *tsk);
extern void free_uts_ns(struct kref *kref);
```

```
@@ -64,6 +66,11 @@ static inline void exit_utsname(struct t
}
```

```
#else
```

```

+static inline int unshare_utsname(unsigned long unshare_flags,
+ struct uts_namespace **new_uts)
+{
+ return -EINVAL;
+}
static inline int copy_utsname(int flags, struct task_struct *tsk)
{
return 0;
diff --git a/kernel/fork.c b/kernel/fork.c
index ddab7e7..cdc549e 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1560,13 +1560,14 @@ asmlinkage long sys_unshare(unsigned long
struct files_struct *fd, *new_fd = NULL;
struct sem_undo_list *new_ulist = NULL;
struct nsproxy *new_nsproxy, *old_nsproxy;
+ struct uts_namespace *uts, *new_uts = NULL;

check_unshare_flags(&unshare_flags);

/* Return -EINVAL for all unsupported flags */
err = -EINVAL;
if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
- CLONE_VM|CLONE_FILES|CLONE_SYSVSEM))
+ CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|CLONE_NEWUTS))
goto bad_unshare_out;

if ((err = unshare_thread(unshare_flags)))
@@ -1583,14 +1584,17 @@ asmlinkage long sys_unshare(unsigned long
goto bad_unshare_cleanup_vm;
if ((err = unshare_semundo(unshare_flags, &new_ulist)))
goto bad_unshare_cleanup_fd;
+ if ((err = unshare_utsname(unshare_flags, &new_uts)))
+ goto bad_unshare_cleanup_semundo;

- if (new_fs || new_ns || new_sigh || new_mm || new_fd || new_ulist) {
+ if (new_fs || new_ns || new_sigh || new_mm || new_fd || new_uts ||
+ new_ulist) {

old_nsproxy = current->nsproxy;
new_nsproxy = dup_namespaces(old_nsproxy);
if (!new_nsproxy) {
err = -ENOMEM;
- goto bad_unshare_cleanup_semundo;
+ goto bad_unshare_cleanup_uts;
}

task_lock(current);

```

```
@@ -1629,10 +1633,20 @@ asmlinkage long sys_unshare(unsigned lon
    new_fd = fd;
}
```

```
+ if (new_uts) {
+   uts = current->nsproxy->uts_ns;
+   current->nsproxy->uts_ns = new_uts;
+   new_uts = uts;
+ }
+
+   task_unlock(current);
+   put_nsproxy(old_nsproxy);
+ }
```

```
+bad_unshare_cleanup_uts:
```

```
+ if (new_uts)
+   put_uts_ns(new_uts);
```

```
+
```

```
bad_unshare_cleanup_semundo:
```

```
bad_unshare_cleanup_fd:
```

```
if (new_fd)
```

```
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
```

```
index d2c6e94..f958551 100644
```

```
--- a/kernel/nsproxy.c
```

```
+++ b/kernel/nsproxy.c
```

```
@@ -80,7 +80,7 @@ int copy_namespaces(int flags, struct ta
```

```
    get_nsproxy(old_ns);
```

```
- if (!(flags & CLONE_NEWNS))
```

```
+ if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS)))
```

```
    return 0;
```

```
    new_ns = clone_namespaces(old_ns);
```

```
diff --git a/kernel/utsname.c b/kernel/utsname.c
```

```
index 2818c9b..2c45490 100644
```

```
--- a/kernel/utsname.c
```

```
+++ b/kernel/utsname.c
```

```
@@ -16,6 +16,41 @@
```

```
#include <linux/version.h>
```

```
/*
```

```
+ * Clone a new ns copying an original utsname, setting refcount to 1
```

```
+ * @old_ns: namespace to clone
```

```
+ * Return NULL on error (failure to kcalloc), new ns otherwise
```

```
+ */
```

```
+struct uts_namespace *clone_uts_ns(struct uts_namespace *old_ns)
```

```
+{
```

```

+ struct uts_namespace *ns;
+
+ ns = kmalloc(sizeof(struct uts_namespace), GFP_KERNEL);
+ if (ns) {
+ memcpy(&ns->name, &old_ns->name, sizeof(ns->name));
+ kref_init(&ns->kref);
+ }
+ return ns;
+}
+
+/*
+ * unshare the current process' utsname namespace.
+ * called only in sys_unshare()
+ */
+int unshare_utsname(unsigned long unshare_flags, struct uts_namespace **new_uts)
+{
+ if (unshare_flags & CLONE_NEWUTS) {
+ if (!capable(CAP_SYS_ADMIN))
+ return -EPERM;
+
+ *new_uts = clone_uts_ns(current->nsproxy->uts_ns);
+ if (!*new_uts)
+ return -ENOMEM;
+ }
+
+ return 0;
+}
+
+/*
+ * Copy task tsk's utsname namespace, or clone it if flags
+ * specifies CLONE_NEWUTS. In latter case, changes to the
+ * utsname of this process won't be seen by parent, and vice
@@ -24,6 +59,7 @@
int copy_utsname(int flags, struct task_struct *tsk)
{
    struct uts_namespace *old_ns = tsk->nsproxy->uts_ns;
+ struct uts_namespace *new_ns;
    int err = 0;

    if (!old_ns)
@@ -31,6 +67,23 @@ int copy_utsname(int flags, struct task_

    get_uts_ns(old_ns);

+ if (!(flags & CLONE_NEWUTS))
+ return 0;
+
+ if (!capable(CAP_SYS_ADMIN)) {

```

```
+ err = -EPERM;
+ goto out;
+ }
+
+ new_ns = clone_uts_ns(old_ns);
+ if (!new_ns) {
+ err = -ENOMEM;
+ goto out;
+ }
+ tsk->nsproxy->uts_ns = new_ns;
+
+out:
+ put_uts_ns(old_ns);
+ return err;
+ }
```

```
--
1.1.6
```
