Subject: [PATCH 2/3] Switch UTS namespace to use shadows
Posted by Pavel Emelianov on Tue, 20 Nov 2007 11:45:51 GMT
View Forum Message <> Reply to Message

The uts sysctl table contains two writable fields
(domainname and nodename), so split the table into
common (read-only) part and writable (shadowed).

This fixes the BUG! You may create a namespace and
then writing to /proc/sys/hostname will cause an
init_uts_ns overwrite.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/include/linux/utsname.h b/include/linux/utsname.h
index 923db99..7517b36 100644
--- a/include/linux/utsname.h
+++ b/include/linux/utsname.h
@@ -40,6 +40,7 @@ struct new_utsname {
 struct uts_namespace {
  struct kref kref;
  struct new_utsname name;
+ struct ctl_table_header *ctl_header;
 };
 extern struct uts_namespace init_uts_ns;

@@ -66,6 +67,9 @@ static inline struct new_utsname *init_utsname(void)
  return &init_uts_ns.name;
 }

+int clone_uts_sysctl(struct uts_namespace *ns);
+void free_uts_sysctl(struct uts_namespace *ns);
+
 extern struct rw_semaphore uts_sem;

 #endif /* __KERNEL__ */
diff --git a/kernel/utsname.c b/kernel/utsname.c
index 816d7b2..22e40bb 100644
--- a/kernel/utsname.c
+++ b/kernel/utsname.c
@@ -26,13 +26,21 @@ static struct uts_namespace *clone_uts_ns(struct uts_namespace
*old_ns)

  ns = kmalloc(sizeof(struct uts_namespace), GFP_KERNEL);
  if (!ns)
-  return ERR_PTR(-ENOMEM);
```

```
+  goto err_alloc;
+
+ if (clone_uts_sysctl(ns))
+  goto err_sysctl;

  down_read(&uts_sem);
  memcpy(&ns->name, &old_ns->name, sizeof(ns->name));
  up_read(&uts_sem);
  kref_init(&ns->kref);
  return ns;
+
+err_sysctl:
+ kfree(ns);
+err_alloc:
+ return ERR_PTR(-ENOMEM);
 }

 /*
@@ -62,5 +70,6 @@ void free_uts_ns(struct kref *kref)
  struct uts_namespace *ns;

  ns = container_of(kref, struct uts_namespace, kref);
+ free_uts_sysctl(ns);
  kfree(ns);
 }
diff --git a/kernel/utsname_sysctl.c b/kernel/utsname_sysctl.c
index c76c064..8a06f0b 100644
--- a/kernel/utsname_sysctl.c
+++ b/kernel/utsname_sysctl.c
@@ -75,6 +75,11 @@ static int sysctl_uts_string(ctl_table *table, int __user *name, int nlen,
 #define sysctl_uts_string NULL
 #endif

+static struct ctl_table_header *uts_sysctl_shadow(struct ctl_table_header *h)
+{
+ return current->nsproxy->uts_ns->ctl_header;
+}
+
 static struct ctl_table uts_kern_table[] = {
  {
   .ctl_name = KERN_OSTYPE,
@@ -103,6 +108,20 @@ static struct ctl_table uts_kern_table[] = {
   .proc_handler = proc_do_uts_string,
   .strategy = sysctl_uts_string,
  },
+ {}
+};
+
```

```
+static struct ctl_table uts_root_table[] = {
+ {
+  .ctl_name = CTL_KERN,
+  .procname = "kernel",
+  .mode  = 0555,
+  .child  = uts_kern_table,
+ },
+ {}
+};
+
+static struct ctl_table uts_kern_table_sh[] = {
  {
   .ctl_name = KERN_NODENAME,
   .procname = "hostname",
@@ -124,19 +143,44 @@ static struct ctl_table uts_kern_table[] = {
  {}
 };

-static struct ctl_table uts_root_table[] = {
+static struct ctl_table uts_root_table_sh[] = {
  {
   .ctl_name = CTL_KERN,
   .procname = "kernel",
   .mode  = 0555,
-  .child  = uts_kern_table,
+  .child  = uts_kern_table_sh,
  },
  {}
 };

+int clone_uts_sysctl(struct uts_namespace *ns)
+{
+ struct ctl_table_header *h;
+ struct ctl_table *tbl;
+
+ h = create_sysctl_shadow(init_uts_ns.ctl_header);
+ if (h == NULL)
+  return -ENOMEM;
+
+ tbl = h->ctl_table->child;
+
+ tbl[0].data = ns->name.nodename;
+ tbl[1].data = ns->name.domainname;
+
+ ns->ctl_header = h;
+ return 0;
+}
+
```

```
+void free_uts_sysctl(struct uts_namespace *ns)
+{
+ free_sysctl_shadow(ns->ctl_header);
+}
+
 static int __init utsname_sysctl_init(void)
 {
  register_sysctl_table(uts_root_table);
+ init_uts_ns.ctl_header = register_sysctl_table_shadow(uts_root_table_sh,
+   uts_sysctl_shadow);
  return 0;
 }
```