Subject: [PATCH 0/9] namespaces: Introduction
Posted by serue on Thu, 18 May 2006 15:47:00 GMT
View Forum Message <> Reply to Message

This patchset introduces a per-process utsname namespace. These can
be used by openvz, vserver, and application migration to virtualize and
isolate utsname info (i.e. hostname). More resources will follow, until
hopefully most or all vserver and openvz functionality can be implemented
by controlling resource namespaces from userspace.

Previous utsname submissions placed a pointer to the utsname namespace
straight in the task_struct. This patchset (and the last one) moves
it and the filesystem namespace pointer into struct nsproxy, which is
shared by processes sharing all namespaces. The intent is to keep
the taskstruct smaller as the number of namespaces grows.

Changes:
- the reference count on fs namespace and uts namespace now
refers to the number of nsproxies pointing to it
- some consolidation of namespace cloning and exit code to
clean up kernel/{fork,exit}.c
- passed ltp and ltpstress on smp power, x86, and x86-64
boxes.

Subject: [PATCH 4/9] namespaces: utsname: switch to using uts namespaces
Posted by serue on Thu, 18 May 2006 15:49:36 GMT
View Forum Message <> Reply to Message

Replace references to system_utsname to the per-process uts namespace
where appropriate. This includes things like uname.

Changes: Per Eric Biederman's comments, use the per-process uts namespace
for ELF_PLATFORM, sunrpc, and parts of net/ipv4/ipconfig.c

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

```
arch/alpha/kernel/osf_sys.c         |   24 ++++++++++++------------
arch/i386/kernel/sys_i386.c         |   12 ++++++------
arch/ia64/sn/kernel/sn2/sn_hwperf.c |    2 +-
arch/m32r/kernel/sys_m32r.c         |    2 +-
arch/mips/kernel/linux32.c          |    2 +-
arch/mips/kernel/syscall.c          |   18 +++++++++---------
arch/mips/kernel/sysirix.c          |   12 ++++++------
arch/parisc/hpux/sys_hpux.c         |   22 ++++++++++++----------
arch/powerpc/kernel/syscalls.c      |   14 +++++++-------
```

```
 arch/sh/kernel/sys_sh.c            |    2 +-
 arch/sh64/kernel/sys_sh64.c        |    2 +-
 arch/sparc/kernel/sys_sparc.c      |    4 ++--
 arch/sparc/kernel/sys_sunos.c      |   10 +++++-----
 arch/sparc64/kernel/sys_sparc.c    |    4 ++--
 arch/sparc64/kernel/sys_sunos32.c  |   10 +++++-----
 arch/sparc64/solaris/misc.c        |    6 +++---
 arch/um/drivers/mconsole_kern.c    |    6 +++---
 arch/um/kernel/syscall_kern.c      |   12 ++++++------
 arch/um/sys-x86_64/syscalls.c      |    2 +-
 arch/x86_64/ia32/sys_ia32.c        |   10 +++++-----
 arch/x86_64/kernel/sys_x86_64.c    |    2 +-
 arch/xtensa/kernel/syscalls.c      |    2 +-
 drivers/char/random.c              |    4 ++--
 fs/cifs/connect.c                  |   28 ++++++++++++++++++----------
 fs/exec.c                          |    2 +-
 fs/lockd/clntproc.c                |    4 ++--
 fs/lockd/mon.c                     |    2 +-
 fs/lockd/svclock.c                 |    2 +-
 fs/lockd/xdr.c                     |    2 +-
 fs/nfs/nfsroot.c                   |    2 +-
 include/asm-i386/elf.h             |    2 +-
 include/linux/lockd/lockd.h        |    2 +-
 kernel/sys.c                       |   14 +++++++-------
 net/ipv4/ipconfig.c                |   14 +++++++-------
 net/sunrpc/clnt.c                  |    4 ++--
 35 files changed, 131 insertions(+), 131 deletions(-)

9ee063adf4d2287583dbb0a71d1d5f80d7ae011f
diff --git a/arch/alpha/kernel/osf_sys.c b/arch/alpha/kernel/osf_sys.c
index 31afe3d..b793b96 100644
--- a/arch/alpha/kernel/osf_sys.c
+++ b/arch/alpha/kernel/osf_sys.c
@@ -402,15 +402,15 @@ osf_utsname(char __user *name)

  down_read(&uts_sem);
  error = -EFAULT;
- if (copy_to_user(name + 0, system_utsname.sysname, 32))
+ if (copy_to_user(name + 0, utsname()->sysname, 32))
   goto out;
- if (copy_to_user(name + 32, system_utsname.nodename, 32))
+ if (copy_to_user(name + 32, utsname()->nodename, 32))
   goto out;
- if (copy_to_user(name + 64, system_utsname.release, 32))
+ if (copy_to_user(name + 64, utsname()->release, 32))
   goto out;
- if (copy_to_user(name + 96, system_utsname.version, 32))
+ if (copy_to_user(name + 96, utsname()->version, 32))
```

```
   goto out;
- if (copy_to_user(name + 128, system_utsname.machine, 32))
+ if (copy_to_user(name + 128, utsname()->machine, 32))
   goto out;

  error = 0;
@@ -449,8 +449,8 @@ osf_getdomainname(char __user *name, int

  down_read(&uts_sem);
  for (i = 0; i < len; ++i) {
-  __put_user(system_utsname.domainname[i], name + i);
-  if (system_utsname.domainname[i] == '\0')
+  __put_user(utsname()->domainname[i], name + i);
+  if (utsname()->domainname[i] == '\0')
    break;
  }
  up_read(&uts_sem);
@@ -608,11 +608,11 @@ asmlinkage long
 osf_sysinfo(int command, char __user *buf, long count)
 {
  static char * sysinfo_table[] = {
-  system_utsname.sysname,
-  system_utsname.nodename,
-  system_utsname.release,
-  system_utsname.version,
-  system_utsname.machine,
+  utsname()->sysname,
+  utsname()->nodename,
+  utsname()->release,
+  utsname()->version,
+  utsname()->machine,
   "alpha", /* instruction set architecture */
   "dummy", /* hardware serial number */
   "dummy", /* hardware manufacturer */
diff --git a/arch/i386/kernel/sys_i386.c b/arch/i386/kernel/sys_i386.c
index 8fdb1fb..4af731d 100644
--- a/arch/i386/kernel/sys_i386.c
+++ b/arch/i386/kernel/sys_i386.c
@@ -210,7 +210,7 @@ asmlinkage int sys_uname(struct old_utsn
  if (!name)
   return -EFAULT;
  down_read(&uts_sem);
- err=copy_to_user(name, &system_utsname, sizeof (*name));
+ err=copy_to_user(name, utsname(), sizeof (*name));
  up_read(&uts_sem);
  return err?-EFAULT:0;
 }
@@ -226,15 +226,15 @@ asmlinkage int sys_olduname(struct oldol
```

```
    down_read(&uts_sem);

- error = __copy_to_user(&name->sysname,&system_utsname.sysname,__OLD_UTS_LEN);
+ error = __copy_to_user(&name->sysname,&utsname()->sysname,__OLD_UTS_LEN);
  error |= __put_user(0,name->sysname+__OLD_UTS_LEN);
- error |= __copy_to_user(&name->nodename,&system_utsname.nodename,__OLD_UTS_LEN);
+ error |= __copy_to_user(&name->nodename,&utsname()->nodename,__OLD_UTS_LEN);
  error |= __put_user(0,name->nodename+__OLD_UTS_LEN);
- error |= __copy_to_user(&name->release,&system_utsname.release,__OLD_UTS_LEN);
+ error |= __copy_to_user(&name->release,&utsname()->release,__OLD_UTS_LEN);
  error |= __put_user(0,name->release+__OLD_UTS_LEN);
- error |= __copy_to_user(&name->version,&system_utsname.version,__OLD_UTS_LEN);
+ error |= __copy_to_user(&name->version,&utsname()->version,__OLD_UTS_LEN);
  error |= __put_user(0,name->version+__OLD_UTS_LEN);
- error |= __copy_to_user(&name->machine,&system_utsname.machine,__OLD_UTS_LEN);
+ error |= __copy_to_user(&name->machine,&utsname()->machine,__OLD_UTS_LEN);
  error |= __put_user(0,name->machine+__OLD_UTS_LEN);

  up_read(&uts_sem);
diff --git a/arch/ia64/sn/kernel/sn2/sn_hwperf.c b/arch/ia64/sn/kernel/sn2/sn_hwperf.c
index 739c948..a27b223 100644
--- a/arch/ia64/sn/kernel/sn2/sn_hwperf.c
+++ b/arch/ia64/sn/kernel/sn2/sn_hwperf.c
@@ -420,7 +420,7 @@ static int sn_topology_show(struct seq_f
    "coherency_domain %d, "
    "region_size %d\n",

-   partid, system_utsname.nodename,
+   partid, utsname()->nodename,
    shubtype ? "shub2" : "shub1",
    (u64)nasid_mask << nasid_shift, nasid_msb, nasid_shift,
    system_size, sharing_size, coher, region_size);
diff --git a/arch/m32r/kernel/sys_m32r.c b/arch/m32r/kernel/sys_m32r.c
index 670cb49..11412c0 100644
--- a/arch/m32r/kernel/sys_m32r.c
+++ b/arch/m32r/kernel/sys_m32r.c
@@ -206,7 +206,7 @@ asmlinkage int sys_uname(struct old_utsn
  if (!name)
   return -EFAULT;
  down_read(&uts_sem);
- err=copy_to_user(name, &system_utsname, sizeof (*name));
+ err=copy_to_user(name, utsname(), sizeof (*name));
  up_read(&uts_sem);
  return err?-EFAULT:0;
 }
diff --git a/arch/mips/kernel/linux32.c b/arch/mips/kernel/linux32.c
index a7d2bb3..66f999b 100644
```

```
--- a/arch/mips/kernel/linux32.c
+++ b/arch/mips/kernel/linux32.c
@@ -1040,7 +1040,7 @@ asmlinkage long sys32_newuname(struct ne
  int ret = 0;

  down_read(&uts_sem);
- if (copy_to_user(name,&system_utsname,sizeof *name))
+ if (copy_to_user(name,utsname(),sizeof *name))
   ret = -EFAULT;
  up_read(&uts_sem);

diff --git a/arch/mips/kernel/syscall.c b/arch/mips/kernel/syscall.c
index 2aeaa2f..8b13d57 100644
--- a/arch/mips/kernel/syscall.c
+++ b/arch/mips/kernel/syscall.c
@@ -232,7 +232,7 @@ out:
  */
 asmlinkage int sys_uname(struct old_utsname __user * name)
 {
- if (name && !copy_to_user(name, &system_utsname, sizeof (*name)))
+ if (name && !copy_to_user(name, utsname(), sizeof (*name)))
   return 0;
  return -EFAULT;
 }
@@ -249,15 +249,15 @@ asmlinkage int sys_olduname(struct oldol
  if (!access_ok(VERIFY_WRITE,name,sizeof(struct oldold_utsname)))
   return -EFAULT;

- error = __copy_to_user(&name->sysname,&system_utsname.sysname,__OLD_UTS_LEN);
+ error = __copy_to_user(&name->sysname,&utsname()->sysname,__OLD_UTS_LEN);
  error -= __put_user(0,name->sysname+__OLD_UTS_LEN);
- error -= __copy_to_user(&name->nodename,&system_utsname.nodename,__OLD_UTS_LEN);
+ error -= __copy_to_user(&name->nodename,&utsname()->nodename,__OLD_UTS_LEN);
  error -= __put_user(0,name->nodename+__OLD_UTS_LEN);
- error -= __copy_to_user(&name->release,&system_utsname.release,__OLD_UTS_LEN);
+ error -= __copy_to_user(&name->release,&utsname()->release,__OLD_UTS_LEN);
  error -= __put_user(0,name->release+__OLD_UTS_LEN);
- error -= __copy_to_user(&name->version,&system_utsname.version,__OLD_UTS_LEN);
+ error -= __copy_to_user(&name->version,&utsname()->version,__OLD_UTS_LEN);
  error -= __put_user(0,name->version+__OLD_UTS_LEN);
- error -= __copy_to_user(&name->machine,&system_utsname.machine,__OLD_UTS_LEN);
+ error -= __copy_to_user(&name->machine,&utsname()->machine,__OLD_UTS_LEN);
  error = __put_user(0,name->machine+__OLD_UTS_LEN);
  error = error ? -EFAULT : 0;

@@ -293,10 +293,10 @@ asmlinkage int _sys_sysmips(int cmd, lon
   return -EFAULT;
```

```
   down_write(&uts_sem);
-  strncpy(system_utsname.nodename, nodename, len);
+  strncpy(utsname()->nodename, nodename, len);
   nodename[__NEW_UTS_LEN] = '\0';
-  strlcpy(system_utsname.nodename, nodename,
-          sizeof(system_utsname.nodename));
+  strlcpy(utsname()->nodename, nodename,
+          sizeof(utsname()->nodename));
   up_write(&uts_sem);
   return 0;
  }
diff --git a/arch/mips/kernel/sysirix.c b/arch/mips/kernel/sysirix.c
index 5407b78..1b4e7e7 100644
--- a/arch/mips/kernel/sysirix.c
+++ b/arch/mips/kernel/sysirix.c
@@ -884,7 +884,7 @@ asmlinkage int irix_getdomainname(char _
  down_read(&uts_sem);
  if (len > __NEW_UTS_LEN)
   len = __NEW_UTS_LEN;
- err = copy_to_user(name, system_utsname.domainname, len) ? -EFAULT : 0;
+ err = copy_to_user(name, utsname()->domainname, len) ? -EFAULT : 0;
  up_read(&uts_sem);

  return err;
@@ -1127,11 +1127,11 @@ struct iuname {
 asmlinkage int irix_uname(struct iuname __user *buf)
 {
  down_read(&uts_sem);
- if (copy_from_user(system_utsname.sysname, buf->sysname, 65)
-     || copy_from_user(system_utsname.nodename, buf->nodename, 65)
-     || copy_from_user(system_utsname.release, buf->release, 65)
-     || copy_from_user(system_utsname.version, buf->version, 65)
-     || copy_from_user(system_utsname.machine, buf->machine, 65)) {
+ if (copy_from_user(utsname()->sysname, buf->sysname, 65)
+     || copy_from_user(utsname()->nodename, buf->nodename, 65)
+     || copy_from_user(utsname()->release, buf->release, 65)
+     || copy_from_user(utsname()->version, buf->version, 65)
+     || copy_from_user(utsname()->machine, buf->machine, 65)) {
   return -EFAULT;
  }
  up_read(&uts_sem);
diff --git a/arch/parisc/hpux/sys_hpux.c b/arch/parisc/hpux/sys_hpux.c
index 05273cc..9fc2c08 100644
--- a/arch/parisc/hpux/sys_hpux.c
+++ b/arch/parisc/hpux/sys_hpux.c
@@ -266,15 +266,15 @@ static int hpux_uname(struct hpux_utsnam

  down_read(&uts_sem);
```

```
- error = __copy_to_user(&name->sysname,&system_utsname.sysname,HPUX_UTSLEN-1);
+ error = __copy_to_user(&name->sysname,&utsname()->sysname,HPUX_UTSLEN-1);
  error |= __put_user(0,name->sysname+HPUX_UTSLEN-1);
- error |= __copy_to_user(&name->nodename,&system_utsname.nodename,HPUX_UTSLEN-1);
+ error |= __copy_to_user(&name->nodename,&utsname()->nodename,HPUX_UTSLEN-1);
  error |= __put_user(0,name->nodename+HPUX_UTSLEN-1);
- error |= __copy_to_user(&name->release,&system_utsname.release,HPUX_UTSLEN-1);
+ error |= __copy_to_user(&name->release,&utsname()->release,HPUX_UTSLEN-1);
  error |= __put_user(0,name->release+HPUX_UTSLEN-1);
- error |= __copy_to_user(&name->version,&system_utsname.version,HPUX_UTSLEN-1);
+ error |= __copy_to_user(&name->version,&utsname()->version,HPUX_UTSLEN-1);
  error |= __put_user(0,name->version+HPUX_UTSLEN-1);
- error |= __copy_to_user(&name->machine,&system_utsname.machine,HPUX_UTSLEN-1);
+ error |= __copy_to_user(&name->machine,&utsname()->machine,HPUX_UTSLEN-1);
  error |= __put_user(0,name->machine+HPUX_UTSLEN-1);

  up_read(&uts_sem);
@@ -373,8 +373,8 @@ int hpux_utssys(char *ubuf, int n, int t
   /*  TODO:  print a warning about using this?  */
   down_write(&uts_sem);
   error = -EFAULT;
-  if (!copy_from_user(system_utsname.sysname, ubuf, len)) {
-   system_utsname.sysname[len] = 0;
+  if (!copy_from_user(utsname()->sysname, ubuf, len)) {
+   utsname()->sysname[len] = 0;
    error = 0;
   }
   up_write(&uts_sem);
@@ -400,8 +400,8 @@ int hpux_utssys(char *ubuf, int n, int t
   /*  TODO:  print a warning about this?  */
   down_write(&uts_sem);
   error = -EFAULT;
-  if (!copy_from_user(system_utsname.release, ubuf, len)) {
-   system_utsname.release[len] = 0;
+  if (!copy_from_user(utsname()->release, ubuf, len)) {
+   utsname()->release[len] = 0;
    error = 0;
   }
   up_write(&uts_sem);
@@ -422,13 +422,13 @@ int hpux_getdomainname(char *name, int l

   down_read(&uts_sem);

- nlen = strlen(system_utsname.domainname) + 1;
+ nlen = strlen(utsname()->domainname) + 1;

  if (nlen < len)
```

```
   len = nlen;
  if(len > __NEW_UTS_LEN)
   goto done;
- if(copy_to_user(name, system_utsname.domainname, len))
+ if(copy_to_user(name, utsname()->domainname, len))
   goto done;
  err = 0;
 done:
diff --git a/arch/powerpc/kernel/syscalls.c b/arch/powerpc/kernel/syscalls.c
index 9b69d99..d358866 100644
--- a/arch/powerpc/kernel/syscalls.c
+++ b/arch/powerpc/kernel/syscalls.c
@@ -260,7 +260,7 @@ long ppc_newuname(struct new_utsname __u
  int err = 0;

  down_read(&uts_sem);
- if (copy_to_user(name, &system_utsname, sizeof(*name)))
+ if (copy_to_user(name, utsname(), sizeof(*name)))
   err = -EFAULT;
  up_read(&uts_sem);
  if (!err)
@@ -273,7 +273,7 @@ int sys_uname(struct old_utsname __user
  int err = 0;

  down_read(&uts_sem);
- if (copy_to_user(name, &system_utsname, sizeof(*name)))
+ if (copy_to_user(name, utsname(), sizeof(*name)))
   err = -EFAULT;
  up_read(&uts_sem);
  if (!err)
@@ -289,19 +289,19 @@ int sys_olduname(struct oldold_utsname _
   return -EFAULT;

  down_read(&uts_sem);
- error = __copy_to_user(&name->sysname, &system_utsname.sysname,
+ error = __copy_to_user(&name->sysname, &utsname()->sysname,
          __OLD_UTS_LEN);
  error |= __put_user(0, name->sysname + __OLD_UTS_LEN);
- error |= __copy_to_user(&name->nodename, &system_utsname.nodename,
+ error |= __copy_to_user(&name->nodename, &utsname()->nodename,
     __OLD_UTS_LEN);
  error |= __put_user(0, name->nodename + __OLD_UTS_LEN);
- error |= __copy_to_user(&name->release, &system_utsname.release,
+ error |= __copy_to_user(&name->release, &utsname()->release,
     __OLD_UTS_LEN);
  error |= __put_user(0, name->release + __OLD_UTS_LEN);
- error |= __copy_to_user(&name->version, &system_utsname.version,
+ error |= __copy_to_user(&name->version, &utsname()->version,
```

```
     __OLD_UTS_LEN);
  error |= __put_user(0, name->version + __OLD_UTS_LEN);
- error |= __copy_to_user(&name->machine, &system_utsname.machine,
+ error |= __copy_to_user(&name->machine, &utsname()->machine,
     __OLD_UTS_LEN);
  error |= override_machine(name->machine);
  up_read(&uts_sem);
diff --git a/arch/sh/kernel/sys_sh.c b/arch/sh/kernel/sys_sh.c
index 917b2f3..e4966b2 100644
--- a/arch/sh/kernel/sys_sh.c
+++ b/arch/sh/kernel/sys_sh.c
@@ -267,7 +267,7 @@ asmlinkage int sys_uname(struct old_utsn
  if (!name)
   return -EFAULT;
  down_read(&uts_sem);
- err=copy_to_user(name, &system_utsname, sizeof (*name));
+ err=copy_to_user(name, utsname(), sizeof (*name));
  up_read(&uts_sem);
  return err?-EFAULT:0;
 }
diff --git a/arch/sh64/kernel/sys_sh64.c b/arch/sh64/kernel/sys_sh64.c
index 58ff7d5..a8dc88c 100644
--- a/arch/sh64/kernel/sys_sh64.c
+++ b/arch/sh64/kernel/sys_sh64.c
@@ -279,7 +279,7 @@ asmlinkage int sys_uname(struct old_utsn
  if (!name)
   return -EFAULT;
  down_read(&uts_sem);
- err=copy_to_user(name, &system_utsname, sizeof (*name));
+ err=copy_to_user(name, utsname(), sizeof (*name));
  up_read(&uts_sem);
  return err?-EFAULT:0;
 }
diff --git a/arch/sparc/kernel/sys_sparc.c b/arch/sparc/kernel/sys_sparc.c
index 0cdfc9d..c8ad73c 100644
--- a/arch/sparc/kernel/sys_sparc.c
+++ b/arch/sparc/kernel/sys_sparc.c
@@ -470,13 +470,13 @@ asmlinkage int sys_getdomainname(char __

   down_read(&uts_sem);

- nlen = strlen(system_utsname.domainname) + 1;
+ nlen = strlen(utsname()->domainname) + 1;

  if (nlen < len)
   len = nlen;
  if (len > __NEW_UTS_LEN)
   goto done;
```

```
- if (copy_to_user(name, system_utsname.domainname, len))
+ if (copy_to_user(name, utsname()->domainname, len))
    goto done;
  err = 0;
 done:
diff --git a/arch/sparc/kernel/sys_sunos.c b/arch/sparc/kernel/sys_sunos.c
index 288de27..9f9206f 100644
--- a/arch/sparc/kernel/sys_sunos.c
+++ b/arch/sparc/kernel/sys_sunos.c
@@ -483,13 +483,13 @@ asmlinkage int sunos_uname(struct sunos_
 {
  int ret;
  down_read(&uts_sem);
- ret = copy_to_user(&name->sname[0], &system_utsname.sysname[0], sizeof(name->sname) -
1);
+ ret = copy_to_user(&name->sname[0], &utsname()->sysname[0], sizeof(name->sname) - 1);
  if (!ret) {
-  ret |= __copy_to_user(&name->nname[0], &system_utsname.nodename[0],
sizeof(name->nname) - 1);
+  ret |= __copy_to_user(&name->nname[0], &utsname()->nodename[0], sizeof(name->nname) -
1);
    ret |= __put_user('\0', &name->nname[8]);
-  ret |= __copy_to_user(&name->rel[0], &system_utsname.release[0], sizeof(name->rel) - 1);
-  ret |= __copy_to_user(&name->ver[0], &system_utsname.version[0], sizeof(name->ver) - 1);
-  ret |= __copy_to_user(&name->mach[0], &system_utsname.machine[0], sizeof(name->mach) -
1);
+  ret |= __copy_to_user(&name->rel[0], &utsname()->release[0], sizeof(name->rel) - 1);
+  ret |= __copy_to_user(&name->ver[0], &utsname()->version[0], sizeof(name->ver) - 1);
+  ret |= __copy_to_user(&name->mach[0], &utsname()->machine[0], sizeof(name->mach) - 1);
  }
  up_read(&uts_sem);
  return ret ? -EFAULT : 0;
diff --git a/arch/sparc64/kernel/sys_sparc.c b/arch/sparc64/kernel/sys_sparc.c
index 7a86913..0453bd2 100644
--- a/arch/sparc64/kernel/sys_sparc.c
+++ b/arch/sparc64/kernel/sys_sparc.c
@@ -707,13 +707,13 @@ asmlinkage long sys_getdomainname(char _

    down_read(&uts_sem);

- nlen = strlen(system_utsname.domainname) + 1;
+ nlen = strlen(utsname()->domainname) + 1;

         if (nlen < len)
                 len = nlen;
  if (len > __NEW_UTS_LEN)
    goto done;
- if (copy_to_user(name, system_utsname.domainname, len))
```

```
+ if (copy_to_user(name, utsname()->domainname, len))
   goto done;
  err = 0;
 done:
diff --git a/arch/sparc64/kernel/sys_sunos32.c b/arch/sparc64/kernel/sys_sunos32.c
index ae5b32f..ba98c47 100644
--- a/arch/sparc64/kernel/sys_sunos32.c
+++ b/arch/sparc64/kernel/sys_sunos32.c
@@ -439,16 +439,16 @@ asmlinkage int sunos_uname(struct sunos_
  int ret;

  down_read(&uts_sem);
- ret = copy_to_user(&name->sname[0], &system_utsname.sysname[0],
+ ret = copy_to_user(&name->sname[0], &utsname()->sysname[0],
       sizeof(name->sname) - 1);
- ret |= copy_to_user(&name->nname[0], &system_utsname.nodename[0],
+ ret |= copy_to_user(&name->nname[0], &utsname()->nodename[0],
        sizeof(name->nname) - 1);
  ret |= put_user('\0', &name->nname[8]);
- ret |= copy_to_user(&name->rel[0], &system_utsname.release[0],
+ ret |= copy_to_user(&name->rel[0], &utsname()->release[0],
        sizeof(name->rel) - 1);
- ret |= copy_to_user(&name->ver[0], &system_utsname.version[0],
+ ret |= copy_to_user(&name->ver[0], &utsname()->version[0],
        sizeof(name->ver) - 1);
- ret |= copy_to_user(&name->mach[0], &system_utsname.machine[0],
+ ret |= copy_to_user(&name->mach[0], &utsname()->machine[0],
        sizeof(name->mach) - 1);
  up_read(&uts_sem);
  return (ret ? -EFAULT : 0);
diff --git a/arch/sparc64/solaris/misc.c b/arch/sparc64/solaris/misc.c
index 5284996..5d0162a 100644
--- a/arch/sparc64/solaris/misc.c
+++ b/arch/sparc64/solaris/misc.c
@@ -239,7 +239,7 @@ asmlinkage int solaris_utssys(u32 buf, u
   /* Let's cheat */
   err  = set_utsfield(v->sysname, "SunOS", 1, 0);
   down_read(&uts_sem);
-  err |= set_utsfield(v->nodename, system_utsname.nodename,
+  err |= set_utsfield(v->nodename, utsname()->nodename,
         1, 1);
   up_read(&uts_sem);
   err |= set_utsfield(v->release, "2.6", 0, 0);
@@ -263,7 +263,7 @@ asmlinkage int solaris_utsname(u32 buf)
  /* Why should we not lie a bit? */
  down_read(&uts_sem);
  err  = set_utsfield(v->sysname, "SunOS", 0, 0);
- err |= set_utsfield(v->nodename, system_utsname.nodename, 1, 1);
```

```
+ err |= set_utsfield(v->nodename, utsname()->nodename, 1, 1);
  err |= set_utsfield(v->release, "5.6", 0, 0);
  err |= set_utsfield(v->version, "Generic", 0, 0);
  err |= set_utsfield(v->machine, machine(), 0, 0);
@@ -295,7 +295,7 @@ asmlinkage int solaris_sysinfo(int cmd,
  case SI_HOSTNAME:
   r = buffer + 256;
   down_read(&uts_sem);
-  for (p = system_utsname.nodename, q = buffer;
+  for (p = utsname()->nodename, q = buffer;
        q < r && *p && *p != '.'; *q++ = *p++);
   up_read(&uts_sem);
   *q = 0;
diff --git a/arch/um/drivers/mconsole_kern.c b/arch/um/drivers/mconsole_kern.c
index 6d7173f..244244a 100644
--- a/arch/um/drivers/mconsole_kern.c
+++ b/arch/um/drivers/mconsole_kern.c
@@ -106,9 +106,9 @@ void mconsole_version(struct mc_request
 {
  char version[256];

- sprintf(version, "%s %s %s %s %s", system_utsname.sysname,
-  system_utsname.nodename, system_utsname.release,
-  system_utsname.version, system_utsname.machine);
+ sprintf(version, "%s %s %s %s %s", utsname()->sysname,
+  utsname()->nodename, utsname()->release,
+  utsname()->version, utsname()->machine);
  mconsole_reply(req, version, 0, 0);
 }

diff --git a/arch/um/kernel/syscall_kern.c b/arch/um/kernel/syscall_kern.c
index 37d3978..d90e9ed 100644
--- a/arch/um/kernel/syscall_kern.c
+++ b/arch/um/kernel/syscall_kern.c
@@ -110,7 +110,7 @@ long sys_uname(struct old_utsname __user
  if (!name)
   return -EFAULT;
  down_read(&uts_sem);
- err=copy_to_user(name, &system_utsname, sizeof (*name));
+ err=copy_to_user(name, utsname(), sizeof (*name));
  up_read(&uts_sem);
  return err?-EFAULT:0;
 }
@@ -126,19 +126,19 @@ long sys_olduname(struct oldold_utsname

    down_read(&uts_sem);

- error = __copy_to_user(&name->sysname,&system_utsname.sysname,
```

```
+ error = __copy_to_user(&name->sysname,&utsname()->sysname,
          __OLD_UTS_LEN);
  error |= __put_user(0,name->sysname+__OLD_UTS_LEN);
- error |= __copy_to_user(&name->nodename,&system_utsname.nodename,
+ error |= __copy_to_user(&name->nodename,&utsname()->nodename,
     __OLD_UTS_LEN);
  error |= __put_user(0,name->nodename+__OLD_UTS_LEN);
- error |= __copy_to_user(&name->release,&system_utsname.release,
+ error |= __copy_to_user(&name->release,&utsname()->release,
     __OLD_UTS_LEN);
  error |= __put_user(0,name->release+__OLD_UTS_LEN);
- error |= __copy_to_user(&name->version,&system_utsname.version,
+ error |= __copy_to_user(&name->version,&utsname()->version,
     __OLD_UTS_LEN);
  error |= __put_user(0,name->version+__OLD_UTS_LEN);
- error |= __copy_to_user(&name->machine,&system_utsname.machine,
+ error |= __copy_to_user(&name->machine,&utsname()->machine,
     __OLD_UTS_LEN);
  error |= __put_user(0,name->machine+__OLD_UTS_LEN);

diff --git a/arch/um/sys-x86_64/syscalls.c b/arch/um/sys-x86_64/syscalls.c
index 6acee5c..3ad014e 100644
--- a/arch/um/sys-x86_64/syscalls.c
+++ b/arch/um/sys-x86_64/syscalls.c
@@ -21,7 +21,7 @@ asmlinkage long sys_uname64(struct new_u
 {
  int err;
  down_read(&uts_sem);
- err = copy_to_user(name, &system_utsname, sizeof (*name));
+ err = copy_to_user(name, utsname(), sizeof (*name));
  up_read(&uts_sem);
  if (personality(current->personality) == PER_LINUX32)
   err |= copy_to_user(&name->machine, "i686", 5);
diff --git a/arch/x86_64/ia32/sys_ia32.c b/arch/x86_64/ia32/sys_ia32.c
index f182b20..6e0a19d 100644
--- a/arch/x86_64/ia32/sys_ia32.c
+++ b/arch/x86_64/ia32/sys_ia32.c
@@ -801,13 +801,13 @@ asmlinkage long sys32_olduname(struct ol

    down_read(&uts_sem);

- error = __copy_to_user(&name->sysname,&system_utsname.sysname,__OLD_UTS_LEN);
+ error = __copy_to_user(&name->sysname,&utsname()->sysname,__OLD_UTS_LEN);
   __put_user(0,name->sysname+__OLD_UTS_LEN);
-  __copy_to_user(&name->nodename,&system_utsname.nodename,__OLD_UTS_LEN);
+  __copy_to_user(&name->nodename,&utsname()->nodename,__OLD_UTS_LEN);
   __put_user(0,name->nodename+__OLD_UTS_LEN);
-  __copy_to_user(&name->release,&system_utsname.release,__OLD_UTS_LEN);
```

```
+  __copy_to_user(&name->release,&utsname()->release,__OLD_UTS_LEN);
   __put_user(0,name->release+__OLD_UTS_LEN);
-  __copy_to_user(&name->version,&system_utsname.version,__OLD_UTS_LEN);
+  __copy_to_user(&name->version,&utsname()->version,__OLD_UTS_LEN);
   __put_user(0,name->version+__OLD_UTS_LEN);
   {
    char *arch = "x86_64";
@@ -830,7 +830,7 @@ long sys32_uname(struct old_utsname __us
  if (!name)
   return -EFAULT;
  down_read(&uts_sem);
- err=copy_to_user(name, &system_utsname, sizeof (*name));
+ err=copy_to_user(name, utsname(), sizeof (*name));
  up_read(&uts_sem);
  if (personality(current->personality) == PER_LINUX32)
   err |= copy_to_user(&name->machine, "i686", 5);
diff --git a/arch/x86_64/kernel/sys_x86_64.c b/arch/x86_64/kernel/sys_x86_64.c
index 6449ea8..76bf7c2 100644
--- a/arch/x86_64/kernel/sys_x86_64.c
+++ b/arch/x86_64/kernel/sys_x86_64.c
@@ -148,7 +148,7 @@ asmlinkage long sys_uname(struct new_uts
 {
  int err;
  down_read(&uts_sem);
- err = copy_to_user(name, &system_utsname, sizeof (*name));
+ err = copy_to_user(name, utsname(), sizeof (*name));
  up_read(&uts_sem);
  if (personality(current->personality) == PER_LINUX32)
   err |= copy_to_user(&name->machine, "i686", 5);
diff --git a/arch/xtensa/kernel/syscalls.c b/arch/xtensa/kernel/syscalls.c
index f20c649..30060c1 100644
--- a/arch/xtensa/kernel/syscalls.c
+++ b/arch/xtensa/kernel/syscalls.c
@@ -129,7 +129,7 @@ out:

 int sys_uname(struct old_utsname * name)
 {
- if (name && !copy_to_user(name, &system_utsname, sizeof (*name)))
+ if (name && !copy_to_user(name, utsname(), sizeof (*name)))
   return 0;
  return -EFAULT;
 }
diff --git a/drivers/char/random.c b/drivers/char/random.c
index 58f3512..a891421 100644
--- a/drivers/char/random.c
+++ b/drivers/char/random.c
@@ -888,8 +888,8 @@ static void init_std_data(struct entropy
```

```
  do_gettimeofday(&tv);
  add_entropy_words(r, (__u32 *)&tv, sizeof(tv)/4);
- add_entropy_words(r, (__u32 *)&system_utsname,
-     sizeof(system_utsname)/4);
+ add_entropy_words(r, (__u32 *)utsname(),
+     sizeof(*(utsname()))/4);
 }

 static int __init rand_initialize(void)
diff --git a/fs/cifs/connect.c b/fs/cifs/connect.c
index d2ec806..b6c0886 100644
--- a/fs/cifs/connect.c
+++ b/fs/cifs/connect.c
@@ -765,12 +765,12 @@ cifs_parse_mount_options(char *options,
  separator[1] = 0;

  memset(vol->source_rfc1001_name,0x20,15);
- for(i=0;i < strnlen(system_utsname.nodename,15);i++) {
+ for(i=0;i < strnlen(utsname()->nodename,15);i++) {
   /* does not have to be a perfect mapping since the field is
   informational, only used for servers that do not support
   port 445 and it can be overridden at mount time */
   vol->source_rfc1001_name[i] =
-   toupper(system_utsname.nodename[i]);
+   toupper(utsname()->nodename[i]);
  }
  vol->source_rfc1001_name[15] = 0;
  /* null target name indicates to use *SMBSERVR default called name
@@ -2077,7 +2077,7 @@ CIFSSessSetup(unsigned int xid, struct c
       32, nls_codepage);
   bcc_ptr += 2 * bytes_returned;
   bytes_returned =
-      cifs_strtoUCS((__le16 *) bcc_ptr, system_utsname.release,
+      cifs_strtoUCS((__le16 *) bcc_ptr, utsname()->release,
       32, nls_codepage);
   bcc_ptr += 2 * bytes_returned;
   bcc_ptr += 2;
@@ -2104,8 +2104,8 @@ CIFSSessSetup(unsigned int xid, struct c
   }
   strcpy(bcc_ptr, "Linux version ");
   bcc_ptr += strlen("Linux version ");
-  strcpy(bcc_ptr, system_utsname.release);
-  bcc_ptr += strlen(system_utsname.release) + 1;
+  strcpy(bcc_ptr, utsname()->release);
+  bcc_ptr += strlen(utsname()->release) + 1;
   strcpy(bcc_ptr, CIFS_NETWORK_OPSYS);
   bcc_ptr += strlen(CIFS_NETWORK_OPSYS) + 1;
  }
```

```
@@ -2346,7 +2346,7 @@ CIFSSpnegoSessSetup(unsigned int xid, st
       32, nls_codepage);
   bcc_ptr += 2 * bytes_returned;
   bytes_returned =
-      cifs_strtoUCS((__le16 *) bcc_ptr, system_utsname.release, 32,
+      cifs_strtoUCS((__le16 *) bcc_ptr, utsname()->release, 32,
       nls_codepage);
   bcc_ptr += 2 * bytes_returned;
   bcc_ptr += 2;
@@ -2371,8 +2371,8 @@ CIFSSpnegoSessSetup(unsigned int xid, st
   }
   strcpy(bcc_ptr, "Linux version ");
   bcc_ptr += strlen("Linux version ");
-  strcpy(bcc_ptr, system_utsname.release);
-  bcc_ptr += strlen(system_utsname.release) + 1;
+  strcpy(bcc_ptr, utsname()->release);
+  bcc_ptr += strlen(utsname()->release) + 1;
   strcpy(bcc_ptr, CIFS_NETWORK_OPSYS);
   bcc_ptr += strlen(CIFS_NETWORK_OPSYS) + 1;
  }
@@ -2622,7 +2622,7 @@ CIFSNTLMSSPNegotiateSessSetup(unsigned i
       32, nls_codepage);
   bcc_ptr += 2 * bytes_returned;
   bytes_returned =
-      cifs_strtoUCS((__le16 *) bcc_ptr, system_utsname.release, 32,
+      cifs_strtoUCS((__le16 *) bcc_ptr, utsname()->release, 32,
       nls_codepage);
   bcc_ptr += 2 * bytes_returned;
   bcc_ptr += 2; /* null terminate Linux version */
@@ -2639,8 +2639,8 @@ CIFSNTLMSSPNegotiateSessSetup(unsigned i
  } else {  /* ASCII */
   strcpy(bcc_ptr, "Linux version ");
   bcc_ptr += strlen("Linux version ");
-  strcpy(bcc_ptr, system_utsname.release);
-  bcc_ptr += strlen(system_utsname.release) + 1;
+  strcpy(bcc_ptr, utsname()->release);
+  bcc_ptr += strlen(utsname()->release) + 1;
   strcpy(bcc_ptr, CIFS_NETWORK_OPSYS);
   bcc_ptr += strlen(CIFS_NETWORK_OPSYS) + 1;
   bcc_ptr++; /* empty domain field */
@@ -3001,7 +3001,7 @@ CIFSNTLMSSPAuthSessSetup(unsigned int xi
       32, nls_codepage);
   bcc_ptr += 2 * bytes_returned;
   bytes_returned =
-      cifs_strtoUCS((__le16 *) bcc_ptr, system_utsname.release, 32,
+      cifs_strtoUCS((__le16 *) bcc_ptr, utsname()->release, 32,
       nls_codepage);
   bcc_ptr += 2 * bytes_returned;
```

```
   bcc_ptr += 2; /* null term version string */
@@ -3053,8 +3053,8 @@ CIFSNTLMSSPAuthSessSetup(unsigned int xi

   strcpy(bcc_ptr, "Linux version ");
   bcc_ptr += strlen("Linux version ");
-  strcpy(bcc_ptr, system_utsname.release);
-  bcc_ptr += strlen(system_utsname.release) + 1;
+  strcpy(bcc_ptr, utsname()->release);
+  bcc_ptr += strlen(utsname()->release) + 1;
   strcpy(bcc_ptr, CIFS_NETWORK_OPSYS);
   bcc_ptr += strlen(CIFS_NETWORK_OPSYS) + 1;
   bcc_ptr++; /* null domain */
diff --git a/fs/exec.c b/fs/exec.c
index 3a79d97..cbb3270 100644
--- a/fs/exec.c
+++ b/fs/exec.c
@@ -1337,7 +1337,7 @@ static void format_corename(char *corena
    case 'h':
     down_read(&uts_sem);
     rc = snprintf(out_ptr, out_end - out_ptr,
-          "%s", system_utsname.nodename);
+          "%s", utsname()->nodename);
     up_read(&uts_sem);
     if (rc > out_end - out_ptr)
      goto out;
diff --git a/fs/lockd/clntproc.c b/fs/lockd/clntproc.c
index f96e381..915e596 100644
--- a/fs/lockd/clntproc.c
+++ b/fs/lockd/clntproc.c
@@ -130,11 +130,11 @@ static void nlmclnt_setlockargs(struct n
  nlmclnt_next_cookie(&argp->cookie);
  argp->state   = nsm_local_state;
  memcpy(&lock->fh, NFS_FH(fl->fl_file->f_dentry->d_inode), sizeof(struct nfs_fh));
- lock->caller  = system_utsname.nodename;
+ lock->caller  = utsname()->nodename;
  lock->oh.data = req->a_owner;
  lock->oh.len  = snprintf(req->a_owner, sizeof(req->a_owner), "%u@%s",
     (unsigned int)fl->fl_u.nfs_fl.owner->pid,
-    system_utsname.nodename);
+    utsname()->nodename);
  lock->svid = fl->fl_u.nfs_fl.owner->pid;
  lock->fl.fl_start = fl->fl_start;
  lock->fl.fl_end = fl->fl_end;
diff --git a/fs/lockd/mon.c b/fs/lockd/mon.c
index 3fc683f..547aaa3 100644
--- a/fs/lockd/mon.c
+++ b/fs/lockd/mon.c
@@ -152,7 +152,7 @@ xdr_encode_common(struct rpc_rqst *rqstp
```

```
   */
  sprintf(buffer, "%u.%u.%u.%u", NIPQUAD(argp->addr));
  if (!(p = xdr_encode_string(p, buffer))
-  || !(p = xdr_encode_string(p, system_utsname.nodename)))
+  || !(p = xdr_encode_string(p, utsname()->nodename)))
   return ERR_PTR(-EIO);
  *p++ = htonl(argp->prog);
  *p++ = htonl(argp->vers);
diff --git a/fs/lockd/svclock.c b/fs/lockd/svclock.c
index 3ef7391..ec93c35 100644
--- a/fs/lockd/svclock.c
+++ b/fs/lockd/svclock.c
@@ -326,7 +326,7 @@ static int nlmsvc_setgrantargs(struct nl
 {
  locks_copy_lock(&call->a_args.lock.fl, &lock->fl);
  memcpy(&call->a_args.lock.fh, &lock->fh, sizeof(call->a_args.lock.fh));
- call->a_args.lock.caller = system_utsname.nodename;
+ call->a_args.lock.caller = utsname()->nodename;
  call->a_args.lock.oh.len = lock->oh.len;

  /* set default data area */
diff --git a/fs/lockd/xdr.c b/fs/lockd/xdr.c
index f22a376..4eec051 100644
--- a/fs/lockd/xdr.c
+++ b/fs/lockd/xdr.c
@@ -516,7 +516,7 @@ nlmclt_decode_res(struct rpc_rqst *req,
  */
 #define NLM_void_sz  0
 #define NLM_cookie_sz  1+XDR_QUADLEN(NLM_MAXCOOKIELEN)
-#define NLM_caller_sz  1+XDR_QUADLEN(sizeof(system_utsname.nodename))
+#define NLM_caller_sz  1+XDR_QUADLEN(sizeof(utsname()->nodename))
 #define NLM_netobj_sz  1+XDR_QUADLEN(XDR_MAX_NETOBJ)
 /* #define NLM_owner_sz  1+XDR_QUADLEN(NLM_MAXOWNER) */
 #define NLM_fhandle_sz  1+XDR_QUADLEN(NFS2_FHSIZE)
diff --git a/fs/nfs/nfsroot.c b/fs/nfs/nfsroot.c
index c0a754e..1d656a6 100644
--- a/fs/nfs/nfsroot.c
+++ b/fs/nfs/nfsroot.c
@@ -312,7 +312,7 @@ static int __init root_nfs_name(char *na
  /* Override them by options set on kernel command-line */
  root_nfs_parse(name, buf);

- cp = system_utsname.nodename;
+ cp = utsname()->nodename;
  if (strlen(buf) + strlen(cp) > NFS_MAXPATHLEN) {
   printk(KERN_ERR "Root-NFS: Pathname for remote directory too long.\n");
   return -1;
diff --git a/include/asm-i386/elf.h b/include/asm-i386/elf.h
```

```
index 4153d80..1b06c44 100644
--- a/include/asm-i386/elf.h
+++ b/include/asm-i386/elf.h
@@ -108,7 +108,7 @@ typedef struct user_fxsr_struct elf_fpxr
    For the moment, we have only optimizations for the Intel generations,
    but that could change... */

-#define ELF_PLATFORM  (system_utsname.machine)
+#define ELF_PLATFORM  (utsname()->machine)

 #ifdef __KERNEL__
 #define SET_PERSONALITY(ex, ibcs2) do { } while (0)
diff --git a/include/linux/lockd/lockd.h b/include/linux/lockd/lockd.h
index 995f89d..ac15b87 100644
--- a/include/linux/lockd/lockd.h
+++ b/include/linux/lockd/lockd.h
@@ -80,7 +80,7 @@ struct nlm_wait;
 /*
  * Memory chunk for NLM client RPC request.
  */
-#define NLMCLNT_OHSIZE  (sizeof(system_utsname.nodename)+10)
+#define NLMCLNT_OHSIZE  (sizeof(utsname()->nodename)+10)
 struct nlm_rqst {
  unsigned int  a_flags; /* initial RPC task flags */
  struct nlm_host * a_host;  /* host handle */
diff --git a/kernel/sys.c b/kernel/sys.c
index 0b6ec0e..bcaa48e 100644
--- a/kernel/sys.c
+++ b/kernel/sys.c
@@ -1671,7 +1671,7 @@ asmlinkage long sys_newuname(struct new_
  int errno = 0;

  down_read(&uts_sem);
- if (copy_to_user(name,&system_utsname,sizeof *name))
+ if (copy_to_user(name,utsname(),sizeof *name))
   errno = -EFAULT;
  up_read(&uts_sem);
  return errno;
@@ -1689,8 +1689,8 @@ asmlinkage long sys_sethostname(char __u
  down_write(&uts_sem);
  errno = -EFAULT;
  if (!copy_from_user(tmp, name, len)) {
-  memcpy(system_utsname.nodename, tmp, len);
-  system_utsname.nodename[len] = 0;
+  memcpy(utsname()->nodename, tmp, len);
+  utsname()->nodename[len] = 0;
   errno = 0;
  }
```

```
  up_write(&uts_sem);
@@ -1706,11 +1706,11 @@ asmlinkage long sys_gethostname(char __u
  if (len < 0)
   return -EINVAL;
  down_read(&uts_sem);
- i = 1 + strlen(system_utsname.nodename);
+ i = 1 + strlen(utsname()->nodename);
  if (i > len)
   i = len;
  errno = 0;
- if (copy_to_user(name, system_utsname.nodename, i))
+ if (copy_to_user(name, utsname()->nodename, i))
   errno = -EFAULT;
  up_read(&uts_sem);
  return errno;
@@ -1735,8 +1735,8 @@ asmlinkage long sys_setdomainname(char _
  down_write(&uts_sem);
  errno = -EFAULT;
  if (!copy_from_user(tmp, name, len)) {
-  memcpy(system_utsname.domainname, tmp, len);
-  system_utsname.domainname[len] = 0;
+  memcpy(utsname()->domainname, tmp, len);
+  utsname()->domainname[len] = 0;
   errno = 0;
  }
  up_write(&uts_sem);
diff --git a/net/ipv4/ipconfig.c b/net/ipv4/ipconfig.c
index cb8a92f..b9bdf0f 100644
--- a/net/ipv4/ipconfig.c
+++ b/net/ipv4/ipconfig.c
@@ -806,7 +806,7 @@ static void __init ic_do_bootp_ext(u8 *e
    }
    break;
   case 12: /* Host name */
-   ic_bootp_string(system_utsname.nodename, ext+1, *ext, __NEW_UTS_LEN);
+   ic_bootp_string(utsname()->nodename, ext+1, *ext, __NEW_UTS_LEN);
    ic_host_name_set = 1;
    break;
   case 15: /* Domain name (DNS) */
@@ -817,7 +817,7 @@ static void __init ic_do_bootp_ext(u8 *e
    ic_bootp_string(root_server_path, ext+1, *ext, sizeof(root_server_path));
    break;
   case 40: /* NIS Domain name (_not_ DNS) */
-   ic_bootp_string(system_utsname.domainname, ext+1, *ext, __NEW_UTS_LEN);
+   ic_bootp_string(utsname()->domainname, ext+1, *ext, __NEW_UTS_LEN);
    break;
  }
 }
```

```
@@ -1369,7 +1369,7 @@ static int __init ip_auto_config(void)
  printk(", mask=%u.%u.%u.%u", NIPQUAD(ic_netmask));
  printk(", gw=%u.%u.%u.%u", NIPQUAD(ic_gateway));
  printk(",\n     host=%s, domain=%s, nis-domain=%s",
-        system_utsname.nodename, ic_domain, system_utsname.domainname);
+        utsname()->nodename, ic_domain, utsname()->domainname);
  printk(",\n     bootserver=%u.%u.%u.%u", NIPQUAD(ic_servaddr));
  printk(", rootserver=%u.%u.%u.%u", NIPQUAD(root_server_addr));
  printk(", rootpath=%s", root_server_path);
@@ -1479,11 +1479,11 @@ static int __init ip_auto_config_setup(c
    case 4:
     if ((dp = strchr(ip, '.'))) {
      *dp++ = '\0';
-     strlcpy(system_utsname.domainname, dp,
-      sizeof(system_utsname.domainname));
+     strlcpy(utsname()->domainname, dp,
+      sizeof(utsname()->domainname));
     }
-    strlcpy(system_utsname.nodename, ip,
-     sizeof(system_utsname.nodename));
+    strlcpy(utsname()->nodename, ip,
+     sizeof(utsname()->nodename));
     ic_host_name_set = 1;
     break;
    case 5:
diff --git a/net/sunrpc/clnt.c b/net/sunrpc/clnt.c
index aa8965e..1d00e41 100644
--- a/net/sunrpc/clnt.c
+++ b/net/sunrpc/clnt.c
@@ -176,10 +176,10 @@ rpc_new_client(struct rpc_xprt *xprt, ch
  }

  /* save the nodename */
- clnt->cl_nodelen = strlen(system_utsname.nodename);
+ clnt->cl_nodelen = strlen(utsname()->nodename);
  if (clnt->cl_nodelen > UNX_MAXNODENAME)
   clnt->cl_nodelen = UNX_MAXNODENAME;
- memcpy(clnt->cl_nodename, system_utsname.nodename, clnt->cl_nodelen);
+ memcpy(clnt->cl_nodename, utsname()->nodename, clnt->cl_nodelen);
  return clnt;

 out_no_auth:
--
1.1.6
```

Subject: [PATCH 7/9] namespaces: utsname: sysctl hack

Sysctl uts patch. This clearly will need to be done another way, but
since sysctl itself needs to be container aware, 'the right thing' is
a separate patchset.

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

---

 kernel/sysctl.c | 20 ++++++++++----------
 1 files changed, 10 insertions(+), 10 deletions(-)

```
e03808f9c1b803ff67e396e806c062c97b4073aa
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index e82726f..ab36b41 100644
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -233,8 +233,8 @@ static ctl_table kern_table[] = {
  {
   .ctl_name = KERN_OSTYPE,
   .procname = "ostype",
-  .data  = system_utsname.sysname,
-  .maxlen  = sizeof(system_utsname.sysname),
+  .data  = init_uts_ns.name.sysname,
+  .maxlen  = sizeof(init_uts_ns.name.sysname),
   .mode  = 0444,
   .proc_handler = &proc_doutsstring,
   .strategy = &sysctl_string,
@@ -242,8 +242,8 @@ static ctl_table kern_table[] = {
  {
   .ctl_name = KERN_OSRELEASE,
   .procname = "osrelease",
-  .data  = system_utsname.release,
-  .maxlen  = sizeof(system_utsname.release),
+  .data  = init_uts_ns.name.release,
+  .maxlen  = sizeof(init_uts_ns.name.release),
   .mode  = 0444,
   .proc_handler = &proc_doutsstring,
   .strategy = &sysctl_string,
@@ -251,8 +251,8 @@ static ctl_table kern_table[] = {
  {
   .ctl_name = KERN_VERSION,
   .procname = "version",
-  .data  = system_utsname.version,
-  .maxlen  = sizeof(system_utsname.version),
+  .data  = init_uts_ns.name.version,
+  .maxlen  = sizeof(init_uts_ns.name.version),
```

```
   .mode  = 0444,
   .proc_handler = &proc_doutsstring,
   .strategy = &sysctl_string,
@@ -260,8 +260,8 @@ static ctl_table kern_table[] = {
  {
   .ctl_name = KERN_NODENAME,
   .procname = "hostname",
-  .data  = system_utsname.nodename,
-  .maxlen  = sizeof(system_utsname.nodename),
+  .data  = init_uts_ns.name.nodename,
+  .maxlen  = sizeof(init_uts_ns.name.nodename),
   .mode  = 0644,
   .proc_handler = &proc_doutsstring,
   .strategy = &sysctl_string,
@@ -269,8 +269,8 @@ static ctl_table kern_table[] = {
  {
   .ctl_name = KERN_DOMAINNAME,
   .procname = "domainname",
-  .data  = system_utsname.domainname,
-  .maxlen  = sizeof(system_utsname.domainname),
+  .data  = init_uts_ns.name.domainname,
+  .maxlen  = sizeof(init_uts_ns.name.domainname),
   .mode  = 0644,
   .proc_handler = &proc_doutsstring,
   .strategy = &sysctl_string,
--
1.1.6
```

Subject: Re: [PATCH 4/9] namespaces: utsname: switch to using uts namespaces
Posted by rdunlap on Fri, 19 May 2006 00:02:34 GMT
View Forum Message <> Reply to Message

On Thu, 18 May 2006 10:49:36 -0500 Serge E. Hallyn wrote:

> Replace references to system_utsname to the per-process uts namespace
> where appropriate. This includes things like uname.
>
> Changes: Per Eric Biederman's comments, use the per-process uts namespace
> for ELF_PLATFORM, sunrpc, and parts of net/ipv4/ipconfig.c
>
> Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
>
> ---
>
> 9ee063adf4d2287583dbb0a71d1d5f80d7ae011f
> diff --git a/arch/i386/kernel/sys_i386.c b/arch/i386/kernel/sys_i386.c
> index 8fdb1fb..4af731d 100644

> --- a/arch/i386/kernel/sys_i386.c
> +++ b/arch/i386/kernel/sys_i386.c
> @@ -210,7 +210,7 @@ asmlinkage int sys_uname(struct old_utsn
> if (!name)
> return -EFAULT;
> down_read(&uts_sem);
> - err=copy_to_user(name, &system_utsname, sizeof (*name));
> + err=copy_to_user(name, utsname(), sizeof (*name));

It would be really nice if you would fix spacing while you are here,
like a space a each side of '='.

and a space after ',' in the function calls below.

> up_read(&uts_sem);
> return err?-EFAULT:0;
> }
> @@ -226,15 +226,15 @@ asmlinkage int sys_olduname(struct oldol
>
> down_read(&uts_sem);
>
> - error = __copy_to_user(&name->sysname,&system_utsname.sysname,__OLD_UTS_LEN);
> + error = __copy_to_user(&name->sysname,&utsname()->sysname,__OLD_UTS_LEN);
> error |= __put_user(0,name->sysname+__OLD_UTS_LEN);
> - error |=
__copy_to_user(&name->nodename,&system_utsname.nodename,__OLD_UTS_LEN);
> + error |= __copy_to_user(&name->nodename,&utsname()->nodename,__OLD_UTS_LEN);
> error |= __put_user(0,name->nodename+__OLD_UTS_LEN);
> - error |= __copy_to_user(&name->release,&system_utsname.release,__OLD_UTS_LEN);
> + error |= __copy_to_user(&name->release,&utsname()->release,__OLD_UTS_LEN);
> error |= __put_user(0,name->release+__OLD_UTS_LEN);
> - error |= __copy_to_user(&name->version,&system_utsname.version,__OLD_UTS_LEN);
> + error |= __copy_to_user(&name->version,&utsname()->version,__OLD_UTS_LEN);
> error |= __put_user(0,name->version+__OLD_UTS_LEN);
> - error |= __copy_to_user(&name->machine,&system_utsname.machine,__OLD_UTS_LEN);
> + error |= __copy_to_user(&name->machine,&utsname()->machine,__OLD_UTS_LEN);
> error |= __put_user(0,name->machine+__OLD_UTS_LEN);
>
> up_read(&uts_sem);
> diff --git a/arch/m32r/kernel/sys_m32r.c b/arch/m32r/kernel/sys_m32r.c
> index 670cb49..11412c0 100644
> --- a/arch/m32r/kernel/sys_m32r.c
> +++ b/arch/m32r/kernel/sys_m32r.c
> @@ -206,7 +206,7 @@ asmlinkage int sys_uname(struct old_utsn
> if (!name)
> return -EFAULT;
> down_read(&uts_sem);
> - err=copy_to_user(name, &system_utsname, sizeof (*name));

> + err=copy_to_user(name, utsname(), sizeof (*name));

spacing

>   up_read(&uts_sem);
>   return err?-EFAULT:0;
> }
> diff --git a/arch/mips/kernel/linux32.c b/arch/mips/kernel/linux32.c
> index a7d2bb3..66f999b 100644
> --- a/arch/mips/kernel/linux32.c
> +++ b/arch/mips/kernel/linux32.c
> @@ -1040,7 +1040,7 @@ asmlinkage long sys32_newuname(struct ne
>   int ret = 0;
>
>   down_read(&uts_sem);
> - if (copy_to_user(name,&system_utsname,sizeof *name))
> + if (copy_to_user(name,utsname(),sizeof *name))

spacing

>   ret = -EFAULT;
>   up_read(&uts_sem);
>
> diff --git a/arch/mips/kernel/syscall.c b/arch/mips/kernel/syscall.c
> index 2aeaa2f..8b13d57 100644
> --- a/arch/mips/kernel/syscall.c
> +++ b/arch/mips/kernel/syscall.c
> @@ -232,7 +232,7 @@ out:
>   */
> asmlinkage int sys_uname(struct old_utsname __user * name)
> {
> - if (name && !copy_to_user(name, &system_utsname, sizeof (*name)))
> + if (name && !copy_to_user(name, utsname(), sizeof (*name)))

OK, here's my big comment/question. I want to see <nodename> increased to
256 bytes (per current POSIX), so each field of struct <variant>_utsname
needs be copied individually (I think) instead of doing a single
struct copy.

I've been working on this for the past few weeks (among other
things). Sorry about the timing.
I could send patches for this against mainline in a few days,
but I'll be glad to listen to how it would be easiest for all of us
to handle.

I'm probably a little over half done with my patches.
They will end up adding a lib/utsname.c that has functions for:

put_oldold_unmame() // to user
put_old_uname() // to user
put_new_uname() // to user
put_posix_uname() // to user

> return 0;
> return -EFAULT;
> }
> @@ -249,15 +249,15 @@ asmlinkage int sys_olduname(struct oldol
> if (!access_ok(VERIFY_WRITE,name,sizeof(struct oldold_utsname)))
> return -EFAULT;
>
> - error = __copy_to_user(&name->sysname,&system_utsname.sysname,__OLD_UTS_LEN);
> + error = __copy_to_user(&name->sysname,&utsname()->sysname,__OLD_UTS_LEN);
> error -= __put_user(0,name->sysname+__OLD_UTS_LEN);
> - error -=
__copy_to_user(&name->nodename,&system_utsname.nodename,__OLD_UTS_LEN);
> + error -= __copy_to_user(&name->nodename,&utsname()->nodename,__OLD_UTS_LEN);
> error -= __put_user(0,name->nodename+__OLD_UTS_LEN);
> - error -= __copy_to_user(&name->release,&system_utsname.release,__OLD_UTS_LEN);
> + error -= __copy_to_user(&name->release,&utsname()->release,__OLD_UTS_LEN);
> error -= __put_user(0,name->release+__OLD_UTS_LEN);
> - error -= __copy_to_user(&name->version,&system_utsname.version,__OLD_UTS_LEN);
> + error -= __copy_to_user(&name->version,&utsname()->version,__OLD_UTS_LEN);
> error -= __put_user(0,name->version+__OLD_UTS_LEN);
> - error -= __copy_to_user(&name->machine,&system_utsname.machine,__OLD_UTS_LEN);
> + error -= __copy_to_user(&name->machine,&utsname()->machine,__OLD_UTS_LEN);
> error = __put_user(0,name->machine+__OLD_UTS_LEN);
> error = error ? -EFAULT : 0;

spaces

> diff --git a/arch/parisc/hpux/sys_hpux.c b/arch/parisc/hpux/sys_hpux.c
> index 05273cc..9fc2c08 100644
> --- a/arch/parisc/hpux/sys_hpux.c
> +++ b/arch/parisc/hpux/sys_hpux.c
> @@ -266,15 +266,15 @@ static int hpux_uname(struct hpux_utsnam
>
> down_read(&uts_sem);
>
> - error = __copy_to_user(&name->sysname,&system_utsname.sysname,HPUX_UTSLEN-1);
> + error = __copy_to_user(&name->sysname,&utsname()->sysname,HPUX_UTSLEN-1);
> error |= __put_user(0,name->sysname+HPUX_UTSLEN-1);
> - error |=
__copy_to_user(&name->nodename,&system_utsname.nodename,HPUX_UTSLEN-1);

```
> + error |= __copy_to_user(&name->nodename,&utsname()->nodename,HPUX_UTSLEN-1);
>   error |= __put_user(0,name->nodename+HPUX_UTSLEN-1);
> - error |= __copy_to_user(&name->release,&system_utsname.release,HPUX_UTSLEN-1);
> + error |= __copy_to_user(&name->release,&utsname()->release,HPUX_UTSLEN-1);
>   error |= __put_user(0,name->release+HPUX_UTSLEN-1);
> - error |= __copy_to_user(&name->version,&system_utsname.version,HPUX_UTSLEN-1);
> + error |= __copy_to_user(&name->version,&utsname()->version,HPUX_UTSLEN-1);
>   error |= __put_user(0,name->version+HPUX_UTSLEN-1);
> - error |= __copy_to_user(&name->machine,&system_utsname.machine,HPUX_UTSLEN-1);
> + error |= __copy_to_user(&name->machine,&utsname()->machine,HPUX_UTSLEN-1);
>   error |= __put_user(0,name->machine+HPUX_UTSLEN-1);

spacing

>   up_read(&uts_sem);

> diff --git a/arch/sh/kernel/sys_sh.c b/arch/sh/kernel/sys_sh.c
> index 917b2f3..e4966b2 100644
> --- a/arch/sh/kernel/sys_sh.c
> +++ b/arch/sh/kernel/sys_sh.c
> @@ -267,7 +267,7 @@ asmlinkage int sys_uname(struct old_utsn
>   if (!name)
>    return -EFAULT;
>   down_read(&uts_sem);
> - err=copy_to_user(name, &system_utsname, sizeof (*name));
> + err=copy_to_user(name, utsname(), sizeof (*name));

spacing

>   up_read(&uts_sem);
>   return err?-EFAULT:0;
>  }
> diff --git a/arch/sh64/kernel/sys_sh64.c b/arch/sh64/kernel/sys_sh64.c
> index 58ff7d5..a8dc88c 100644
> --- a/arch/sh64/kernel/sys_sh64.c
> +++ b/arch/sh64/kernel/sys_sh64.c
> @@ -279,7 +279,7 @@ asmlinkage int sys_uname(struct old_utsn
>   if (!name)
>    return -EFAULT;
>   down_read(&uts_sem);
> - err=copy_to_user(name, &system_utsname, sizeof (*name));
> + err=copy_to_user(name, utsname(), sizeof (*name));

spacing

>   up_read(&uts_sem);
>   return err?-EFAULT:0;
>  }
```

> diff --git a/arch/sparc/kernel/sys_sunos.c b/arch/sparc/kernel/sys_sunos.c
> index 288de27..9f9206f 100644
> --- a/arch/sparc/kernel/sys_sunos.c
> +++ b/arch/sparc/kernel/sys_sunos.c
> @@ -483,13 +483,13 @@ asmlinkage int sunos_uname(struct sunos_
> {
>   int ret;
>   down_read(&uts_sem);
> - ret = copy_to_user(&name->sname[0], &system_utsname.sysname[0], sizeof(name->sname) -
1);
> + ret = copy_to_user(&name->sname[0], &utsname()->sysname[0], sizeof(name->sname) - 1);
>   if (!ret) {
> - ret |= __copy_to_user(&name->nname[0], &system_utsname.nodename[0],
sizeof(name->nname) - 1);
> + ret |= __copy_to_user(&name->nname[0], &utsname()->nodename[0], sizeof(name->nname)
- 1);
>   ret |= __put_user('\0', &name->nname[8]);
> - ret |= __copy_to_user(&name->rel[0], &system_utsname.release[0], sizeof(name->rel) - 1);
> - ret |= __copy_to_user(&name->ver[0], &system_utsname.version[0], sizeof(name->ver) - 1);
> - ret |= __copy_to_user(&name->mach[0], &system_utsname.machine[0], sizeof(name->mach)
- 1);
> + ret |= __copy_to_user(&name->rel[0], &utsname()->release[0], sizeof(name->rel) - 1);
> + ret |= __copy_to_user(&name->ver[0], &utsname()->version[0], sizeof(name->ver) - 1);
> + ret |= __copy_to_user(&name->mach[0], &utsname()->machine[0], sizeof(name->mach) - 1);

Oh, please limit to 80 column width while you are here (+ other places).

>   }
>   up_read(&uts_sem);
>   return ret ? -EFAULT : 0;

> diff --git a/arch/um/kernel/syscall_kern.c b/arch/um/kernel/syscall_kern.c
> index 37d3978..d90e9ed 100644
> --- a/arch/um/kernel/syscall_kern.c
> +++ b/arch/um/kernel/syscall_kern.c
> @@ -110,7 +110,7 @@ long sys_uname(struct old_utsname __user
>   if (!name)
>   return -EFAULT;
>   down_read(&uts_sem);
> - err=copy_to_user(name, &system_utsname, sizeof (*name));
> + err=copy_to_user(name, utsname(), sizeof (*name));

spacing

>   up_read(&uts_sem);
>   return err?-EFAULT:0;
> }
> @@ -126,19 +126,19 @@ long sys_olduname(struct oldold_utsname

>
>   down_read(&uts_sem);
>
> - error = __copy_to_user(&name->sysname,&system_utsname.sysname,
> + error = __copy_to_user(&name->sysname,&utsname()->sysname,
>   __OLD_UTS_LEN);
>   error |= __put_user(0,name->sysname+__OLD_UTS_LEN);
> - error |= __copy_to_user(&name->nodename,&system_utsname.nodename,
> + error |= __copy_to_user(&name->nodename,&utsname()->nodename,
>   __OLD_UTS_LEN);
>   error |= __put_user(0,name->nodename+__OLD_UTS_LEN);
> - error |= __copy_to_user(&name->release,&system_utsname.release,
> + error |= __copy_to_user(&name->release,&utsname()->release,
>   __OLD_UTS_LEN);
>   error |= __put_user(0,name->release+__OLD_UTS_LEN);
> - error |= __copy_to_user(&name->version,&system_utsname.version,
> + error |= __copy_to_user(&name->version,&utsname()->version,
>   __OLD_UTS_LEN);
>   error |= __put_user(0,name->version+__OLD_UTS_LEN);
> - error |= __copy_to_user(&name->machine,&system_utsname.machine,
> + error |= __copy_to_user(&name->machine,&utsname()->machine,
>   __OLD_UTS_LEN);
>   error |= __put_user(0,name->machine+__OLD_UTS_LEN);

spacing

> diff --git a/arch/x86_64/ia32/sys_ia32.c b/arch/x86_64/ia32/sys_ia32.c
> index f182b20..6e0a19d 100644
> --- a/arch/x86_64/ia32/sys_ia32.c
> +++ b/arch/x86_64/ia32/sys_ia32.c
> @@ -801,13 +801,13 @@ asmlinkage long sys32_olduname(struct ol
>
>   down_read(&uts_sem);
>
> - error = __copy_to_user(&name->sysname,&system_utsname.sysname,__OLD_UTS_LEN);
> + error = __copy_to_user(&name->sysname,&utsname()->sysname,__OLD_UTS_LEN);
>   __put_user(0,name->sysname+__OLD_UTS_LEN);
> - __copy_to_user(&name->nodename,&system_utsname.nodename,__OLD_UTS_LEN);
> + __copy_to_user(&name->nodename,&utsname()->nodename,__OLD_UTS_LEN);
>   __put_user(0,name->nodename+__OLD_UTS_LEN);
> - __copy_to_user(&name->release,&system_utsname.release,__OLD_UTS_LEN);
> + __copy_to_user(&name->release,&utsname()->release,__OLD_UTS_LEN);
>   __put_user(0,name->release+__OLD_UTS_LEN);
> - __copy_to_user(&name->version,&system_utsname.version,__OLD_UTS_LEN);
> + __copy_to_user(&name->version,&utsname()->version,__OLD_UTS_LEN);
>   __put_user(0,name->version+__OLD_UTS_LEN);

spacing

```
>    {
>     char *arch = "x86_64";
> @@ -830,7 +830,7 @@ long sys32_uname(struct old_utsname __us
>    if (!name)
>     return -EFAULT;
>    down_read(&uts_sem);
> - err=copy_to_user(name, &system_utsname, sizeof (*name));
> + err=copy_to_user(name, utsname(), sizeof (*name));
```

ditto

```
>    up_read(&uts_sem);
>    if (personality(current->personality) == PER_LINUX32)
>     err |= copy_to_user(&name->machine, "i686", 5);

> diff --git a/fs/cifs/connect.c b/fs/cifs/connect.c
> index d2ec806..b6c0886 100644
> --- a/fs/cifs/connect.c
> +++ b/fs/cifs/connect.c
> @@ -765,12 +765,12 @@ cifs_parse_mount_options(char *options,
>    separator[1] = 0;
>
>    memset(vol->source_rfc1001_name,0x20,15);
> - for(i=0;i < strnlen(system_utsname.nodename,15);i++) {
> + for(i=0;i < strnlen(utsname()->nodename,15);i++) {
```

spacing

```
>     /* does not have to be a perfect mapping since the field is
>     informational, only used for servers that do not support
>     port 445 and it can be overridden at mount time */
>     vol->source_rfc1001_name[i] =
> -    toupper(system_utsname.nodename[i]);
> +    toupper(utsname()->nodename[i]);
>    }
>    vol->source_rfc1001_name[15] = 0;
>    /* null target name indicates to use *SMBSERVR default called name

> diff --git a/kernel/sys.c b/kernel/sys.c
> index 0b6ec0e..bcaa48e 100644
> --- a/kernel/sys.c
> +++ b/kernel/sys.c
> @@ -1671,7 +1671,7 @@ asmlinkage long sys_newuname(struct new_
>    int errno = 0;
>
>    down_read(&uts_sem);
> - if (copy_to_user(name,&system_utsname,sizeof *name))
```

> + if (copy_to_user(name,utsname(),sizeof *name))

spacing

> errno = -EFAULT;
> up_read(&uts_sem);
> return errno;

Thanks,

~Randy

Subject: Re: [PATCH 4/9] namespaces: utsname: switch to using uts namespaces
Posted by serue on Fri, 19 May 2006 02:21:14 GMT
View Forum Message <> Reply to Message

Quoting Randy.Dunlap (rdunlap@xenotime.net):
> > --- a/arch/i386/kernel/sys_i386.c
> > +++ b/arch/i386/kernel/sys_i386.c
> > @@ -210,7 +210,7 @@ asmlinkage int sys_uname(struct old_utsn
> > if (!name)
> > return -EFAULT;
> > down_read(&uts_sem);
> > - err=copy_to_user(name, &system_utsname, sizeof (*name));
> > + err=copy_to_user(name, utsname(), sizeof (*name));
>
> It would be really nice if you would fix spacing while you are here,
> like a space a each side of '='.
>
> and a space after ',' in the function calls below.

Ok. Then in blocks like the following:

> > - error = __copy_to_user(&name->sysname,&system_utsname.sysname,__OLD_UTS_LEN);
> > + error = __copy_to_user(&name->sysname,&utsname()->sysname,__OLD_UTS_LEN);
> > error |= __put_user(0,name->sysname+__OLD_UTS_LEN);
> > - error |=
__copy_to_user(&name->nodename,&system_utsname.nodename,__OLD_UTS_LEN);
> > + error |= __copy_to_user(&name->nodename,&utsname()->nodename,__OLD_UTS_LEN);
> > error |= __put_user(0,name->nodename+__OLD_UTS_LEN);
> > - error |= __copy_to_user(&name->release,&system_utsname.release,__OLD_UTS_LEN);
> > + error |= __copy_to_user(&name->release,&utsname()->release,__OLD_UTS_LEN);
> > error |= __put_user(0,name->release+__OLD_UTS_LEN);
> > - error |= __copy_to_user(&name->version,&system_utsname.version,__OLD_UTS_LEN);
> > + error |= __copy_to_user(&name->version,&utsname()->version,__OLD_UTS_LEN);
> > error |= __put_user(0,name->version+__OLD_UTS_LEN);
> > - error |= __copy_to_user(&name->machine,&system_utsname.machine,__OLD_UTS_LEN);

> > + error |= __copy_to_user(&name->machine,&utsname()->machine,__OLD_UTS_LEN);
> >   error |= __put_user(0,name->machine+__OLD_UTS_LEN);

Should I leave it as is, to keep the consistent look? Change just the
lines I'm editing, making it inconsistent? Or change the whole block,
making my patch seem a bit larger than it really is, but giving the
nicest end result?

I suppose I could insert a separate patchset fixing up the spacing in
those blocks but making no real changes at all, then apply my patch on
top of that...?

> > --- a/arch/mips/kernel/syscall.c
> > +++ b/arch/mips/kernel/syscall.c
> > @@ -232,7 +232,7 @@ out:
> >   */
> > asmlinkage int sys_uname(struct old_utsname __user * name)
> > {
> > - if (name && !copy_to_user(name, &system_utsname, sizeof (*name)))
> > + if (name && !copy_to_user(name, utsname(), sizeof (*name)))
>
>
> OK, here's my big comment/question. I want to see <nodename> increased to
> 256 bytes (per current POSIX), so each field of struct <variant>_utsname
> needs be copied individually (I think) instead of doing a single
> struct copy.
>
> I've been working on this for the past few weeks (among other
> things). Sorry about the timing.
> I could send patches for this against mainline in a few days,
> but I'll be glad to listen to how it would be easiest for all of us
> to handle.
>
> I'm probably a little over half done with my patches.
> They will end up adding a lib/utsname.c that has functions for:
>   put_oldold_unmame() // to user
>   put_old_uname() // to user
>   put_new_uname() // to user
>   put_posix_uname() // to user

Ok, so long as these functions accept a utsname, we should be able to
just change what we pass in to these functions to being the namespace's
utsname, right? Or am I missing the really nasty part?

thanks,
-serge

Subject: Re: [PATCH 4/9] namespaces: utsname: switch to using uts namespaces
Posted by rdunlap on Fri, 19 May 2006 02:42:50 GMT
View Forum Message <> Reply to Message

On Thu, 18 May 2006 21:21:14 -0500 Serge E. Hallyn wrote:

> Quoting Randy.Dunlap (rdunlap@xenotime.net):
> > > --- a/arch/i386/kernel/sys_i386.c
> > > +++ b/arch/i386/kernel/sys_i386.c
> > > @@ -210,7 +210,7 @@ asmlinkage int sys_uname(struct old_utsn
> > > if (!name)
> > > return -EFAULT;
> > > down_read(&uts_sem);
> > > - err=copy_to_user(name, &system_utsname, sizeof (*name));
> > > + err=copy_to_user(name, utsname(), sizeof (*name));
> >
> > It would be really nice if you would fix spacing while you are here,
> > like a space a each side of '='.
> >
> > and a space after ',' in the function calls below.
>
> Ok. Then in blocks like the following:
>
> > > - error =
__copy_to_user(&name->sysname,&system_utsname.sysname,__OLD_UTS_LEN);
> > > + error = __copy_to_user(&name->sysname,&utsname()->sysname,__OLD_UTS_LEN);
> > > error |= __put_user(0,name->sysname+__OLD_UTS_LEN);
> > > - error |=
__copy_to_user(&name->nodename,&system_utsname.nodename,__OLD_UTS_LEN);
> > > + error |= __copy_to_user(&name->nodename,&utsname()->nodename,__OLD_UTS_LEN);
> > > error |= __put_user(0,name->nodename+__OLD_UTS_LEN);
> > > - error |= __copy_to_user(&name->release,&system_utsname.release,__OLD_UTS_LEN);
> > > + error |= __copy_to_user(&name->release,&utsname()->release,__OLD_UTS_LEN);
> > > error |= __put_user(0,name->release+__OLD_UTS_LEN);
> > > - error |= __copy_to_user(&name->version,&system_utsname.version,__OLD_UTS_LEN);
> > > + error |= __copy_to_user(&name->version,&utsname()->version,__OLD_UTS_LEN);
> > > error |= __put_user(0,name->version+__OLD_UTS_LEN);
> > > - error |=
__copy_to_user(&name->machine,&system_utsname.machine,__OLD_UTS_LEN);
> > > + error |= __copy_to_user(&name->machine,&utsname()->machine,__OLD_UTS_LEN);
> > > error |= __put_user(0,name->machine+__OLD_UTS_LEN);
>
> Should I leave it as is, to keep the consistent look? Change just the
> lines I'm editing, making it inconsistent? Or change the whole block,
> making my patch seem a bit larger than it really is, but giving the
> nicest end result?

I'd go for the latter, along with my other comment of breaking them
to fit into 80 columns also.

> I suppose I could insert a separate patchset fixing up the spacing in
> those blocks but making no real changes at all, then apply my patch on
> top of that...?
>
> > > --- a/arch/mips/kernel/syscall.c
> > > +++ b/arch/mips/kernel/syscall.c
> > > @@ -232,7 +232,7 @@ out:
> > > */
> > > asmlinkage int sys_uname(struct old_utsname __user * name)
> > > {
> > > - if (name && !copy_to_user(name, &system_utsname, sizeof (*name)))
> > > + if (name && !copy_to_user(name, utsname(), sizeof (*name)))
> >
> >
> > OK, here's my big comment/question. I want to see <nodename> increased to
> > 256 bytes (per current POSIX), so each field of struct <variant>_utsname
> > needs be copied individually (I think) instead of doing a single
> > struct copy.
> >
> > I've been working on this for the past few weeks (among other
> > things). Sorry about the timing.
> > I could send patches for this against mainline in a few days,
> > but I'll be glad to listen to how it would be easiest for all of us
> > to handle.
> >
> > I'm probably a little over half done with my patches.
> > They will end up adding a lib/utsname.c that has functions for:
> > put_oldold_unmame() // to user
> > put_old_uname() // to user
> > put_new_uname() // to user
> > put_posix_uname() // to user
>
> Ok, so long as these functions accept a utsname, we should be able to
> just change what we pass in to these functions to being the namespace's
> utsname, right? Or am I missing the really nasty part?

The nodename field changes from 65 chars (struct new_utsname) to 256 chars
(struct posix_utsname), and nodename is not the final field in the
struct, so it's no longer safe to do a simple struct copy. Each
field in the struct needs to be copied individually if the target is
not a struct posix_utsname. It's not rocket science.

~Randy

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by ebiederm on Fri, 19 May 2006 08:50:19 GMT
View Forum Message <> Reply to Message

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

> This patchset introduces a per-process utsname namespace. These can
> be used by openvz, vserver, and application migration to virtualize and
> isolate utsname info (i.e. hostname). More resources will follow, until
> hopefully most or all vserver and openvz functionality can be implemented
> by controlling resource namespaces from userspace.
>
> Previous utsname submissions placed a pointer to the utsname namespace
> straight in the task_struct. This patchset (and the last one) moves
> it and the filesystem namespace pointer into struct nsproxy, which is
> shared by processes sharing all namespaces. The intent is to keep
> the taskstruct smaller as the number of namespaces grows.

Previously you mentioned:
> BTW - a first set of comparison results showed nsproxy to have better
> dbench and tbench throughput, and worse kernbench performance. Which
> may make sense given that nsproxy results in lower memory usage but
> likely increased cache misses due to extra pointer dereference.

Is this still true? Or did our final reference counting tweak fix
the kernbench numbers?

I just want to be certain that we don't add an optimization,
that reduces performance.

> Changes:
> - the reference count on fs namespace and uts namespace now
> refers to the number of nsproxies pointing to it
> - some consolidation of namespace cloning and exit code to
> clean up kernel/{fork,exit}.c
> - passed ltp and ltpstress on smp power, x86, and x86-64
> boxes.

Nice.

Eric

Subject: Re: [PATCH 4/9] namespaces: utsname: switch to using uts namespaces
Posted by ebiederm on Fri, 19 May 2006 09:05:23 GMT
View Forum Message <> Reply to Message

"Randy.Dunlap" <rdunlap@xenotime.net> writes:

> On Thu, 18 May 2006 10:49:36 -0500 Serge E. Hallyn wrote:
>
>> Replace references to system_utsname to the per-process uts namespace
>> where appropriate. This includes things like uname.
>>
>> Changes: Per Eric Biederman's comments, use the per-process uts namespace
>> for ELF_PLATFORM, sunrpc, and parts of net/ipv4/ipconfig.c
>>
>> Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

>
> OK, here's my big comment/question. I want to see <nodename> increased to
> 256 bytes (per current POSIX), so each field of struct <variant>_utsname
> needs be copied individually (I think) instead of doing a single
> struct copy.

Where is it specified? Looking at the spec as SUSV3 I don't see a size
specified for nodename.

> I've been working on this for the past few weeks (among other
> things). Sorry about the timing.
> I could send patches for this against mainline in a few days,
> but I'll be glad to listen to how it would be easiest for all of us
> to handle.
>
> I'm probably a little over half done with my patches.
> They will end up adding a lib/utsname.c that has functions for:
>   put_oldold_uname() // to user
>   put_old_uname() // to user
>   put_new_uname() // to user
>   put_posix_uname() // to user

Sounds reasonable, if we really need a 256 byte nodename.

As long as they take a pointer to the appropriate utsname
structure these patches should not fundamentally conflict.

Eric

Subject: Re: [PATCH 4/9] namespaces: utsname: switch to using uts namespaces
Posted by ebiederm on Fri, 19 May 2006 11:58:22 GMT
View Forum Message <> Reply to Message

"Randy.Dunlap" <rdunlap@xenotime.net> writes:

> OK, here's my big comment/question. I want to see <nodename> increased to
> 256 bytes (per current POSIX), so each field of struct <variant>_utsname
> needs be copied individually (I think) instead of doing a single
> struct copy.
>
> I've been working on this for the past few weeks (among other
> things). Sorry about the timing.
> I could send patches for this against mainline in a few days,
> but I'll be glad to listen to how it would be easiest for all of us
> to handle.
>
> I'm probably a little over half done with my patches.
> They will end up adding a lib/utsname.c that has functions for:
>   put_oldold_unmame() // to user
>   put_old_uname() // to user
>   put_new_uname() // to user
>   put_posix_uname() // to user

Looking 256 at least makes sense to hold a dns fully qualified domain
name. So even if it isn't specified by posix is make sense.

Can we please make the structure we return to user space look something
like:

```
struct long_utsname {
 char *sysname;
 char *nodename;
 char *release;
 char *version;
 char *machine;
        char *domainname;
        char buf[0];
}

int sys_long_uname(char *buf, size_t bufsz);
```

So we don't hard code the maximum length of these strings into the user
interface, and can just return more by increasing our buffer size.

Eric

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by serue on Fri, 19 May 2006 13:30:41 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:

>
> > This patchset introduces a per-process utsname namespace. These can
> > be used by openvz, vserver, and application migration to virtualize and
> > isolate utsname info (i.e. hostname). More resources will follow, until
> > hopefully most or all vserver and openvz functionality can be implemented
> > by controlling resource namespaces from userspace.
> >
> > Previous utsname submissions placed a pointer to the utsname namespace
> > straight in the task_struct. This patchset (and the last one) moves
> > it and the filesystem namespace pointer into struct nsproxy, which is
> > shared by processes sharing all namespaces. The intent is to keep
> > the taskstruct smaller as the number of namespaces grows.
>
>
> Previously you mentioned:
> > BTW - a first set of comparison results showed nsproxy to have better
> > dbench and tbench throughput, and worse kernbench performance. Which
> > may make sense given that nsproxy results in lower memory usage but
> > likely increased cache misses due to extra pointer dereference.
>
> Is this still true? Or did our final reference counting tweak fix
> the kernbench numbers?

I haven't checked that. I'll start a new set of runs later this
morning, should get the results out saturday.

-serge

Subject: Re: [PATCH 4/9] namespaces: utsname: switch to using uts namespaces
Posted by rdunlap on Fri, 19 May 2006 17:37:42 GMT
View Forum Message <> Reply to Message

On Fri, 19 May 2006 03:05:23 -0600 Eric W. Biederman wrote:

> "Randy.Dunlap" <rdunlap@xenotime.net> writes:
>
> > On Thu, 18 May 2006 10:49:36 -0500 Serge E. Hallyn wrote:
> >
> >> Replace references to system_utsname to the per-process uts namespace
> >> where appropriate. This includes things like uname.
> >>
> >> Changes: Per Eric Biederman's comments, use the per-process uts namespace
> >> for ELF_PLATFORM, sunrpc, and parts of net/ipv4/ipconfig.c
> >>
> >> Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
>
> > >

> > OK, here's my big comment/question. I want to see <nodename> increased to
> > 256 bytes (per current POSIX), so each field of struct <variant>_utsname
> > needs be copied individually (I think) instead of doing a single
> > struct copy.
>
> Where is it specified? Looking at the spec as SUSV3 I don't see a size
> specified for nodename.

It's actually for hostname. It looks to me like they are used
interchangeably. yes/no?

gethostname:
 http://www.opengroup.org/onlinepubs/009695399/functions/geth ostname.html
sysconf:
 http://www.opengroup.org/onlinepubs/009695399/functions/sysc onf.html
unistd.h:
 http://www.opengroup.org/onlinepubs/009695399/basedefs/unist d.h.html
limits.h:
 http://www.opengroup.org/onlinepubs/009695399/basedefs/limit s.h.html

>From the latter:
{HOST_NAME_MAX}
    Maximum length of a host name (not including the terminating null) as returned from the
gethostname() function.
    Minimum Acceptable Value: {_POSIX_HOST_NAME_MAX}
(and)
{_POSIX_HOST_NAME_MAX}
    Maximum length of a host name (not including the terminating null) as returned from the
gethostname() function.
    Value: 255

> > I've been working on this for the past few weeks (among other
> > things). Sorry about the timing.
> > I could send patches for this against mainline in a few days,
> > but I'll be glad to listen to how it would be easiest for all of us
> > to handle.
> >
> > I'm probably a little over half done with my patches.
> > They will end up adding a lib/utsname.c that has functions for:
> >   put_oldold_uname() // to user
> >   put_old_uname() // to user
> >   put_new_uname() // to user
> >   put_posix_uname() // to user
>
> Sounds reasonable, if we really need a 256 byte nodename.
>

> As long as they take a pointer to the appropriate utsname
> structure these patches should not fundamentally conflict.

~Randy

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by serue on Fri, 19 May 2006 19:38:02 GMT
View Forum Message <> Reply to Message

Quoting Hua Zhong (hzhong@gmail.com):
> > snapshot/restart/migration worry me. If they require
> > complete serialisation of complex kernel data structures then
> > we have a problem, because it means that any time anyone
> > changes such a structure they need to update (and test) the
> > serialisation.
>
> Checkpoint/Restart/Migration could be very complicated if done at OS level (per process/process group/or any subset of an OS). But
> it is much simpler if done on virtual machine level (VMWare/Xen) because there is a natural and clear boundary, and doesn't get
> affected if the OS kernel internal changes.
>
> It's good to see some progress in supporting virtualization in Linux, but as Andrew put it, some big decisions need to be made
> up-front. One big question is actually how many virtualization technologies Linux should support? Particularly, does it need to
> support both OS-level virtualization and VM-level virtualization? And why? And to what degree?

Because migration can be used for more than one purpose. One such purpose is load-balancing large numbers of jobs. If you have large numbers of jobs, you do not want the resource overhead of a full OS for each migrateable job.

The reason it is deemed simpler at the vm level, as you point out, is that resources are naturally isolated. The same work which will prepare the kernel for vserver/openvz functionality will isolate kernel resources between vserver/containers, making c/r and migration at that level level much simpler.

thanks,
-serge

Subject: Re: [PATCH 0/9] namespaces: Introduction

Posted by John Kelly on Fri, 19 May 2006 19:45:43 GMT
View Forum Message <> Reply to Message

On Fri, 19 May 2006 11:28:08 -0700, "Hua Zhong" <hzhong@gmail.com>
wrote:

> how many virtualization technologies Linux should support?

> Particularly, does it need to support both OS-level virtualization

If users want it. I do.

> It seems at least the VM approach is much less risky. It might be helpful
> if someone could explain why we need both.

A better question is, why can't we have both?

I don't have unlimited memory and disk. I need to conserve my
resources as much as possible.

The one-kernel approach saves memory, leaving more for applications.
That's important to me. I don't need to run multiple kernels, and I
don't want to. I only want multiple secure operating environments.

The one-kernel approach also makes it easy to have all VPS in one disk
partition, without the performance penalty of file backed I/O.

If the VM approach is truly less risky, seems to me the Xen/VMware
developers should be able to succeed independently, despite changes
made for in-kernel virtualization.

I'm glad someone asked a question I could answer. :-)

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by John Kelly on Fri, 19 May 2006 20:23:43 GMT
View Forum Message <> Reply to Message

On Fri, 19 May 2006 15:45:43 -0400, John Kelly <jak@isp2dial.com>
wrote:

>If the VM approach is truly less risky, seems to me the Xen/VMware
>developers should be able to succeed independently, despite changes
>made for in-kernel virtualization.

OTOH, maybe the Xen and VMware developers will kill each other off,
and then we won't have to worry about Xen, or VMware.

;-)

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Alexey Kuznetsov on Fri, 19 May 2006 20:52:47 GMT
View Forum Message <> Reply to Message

Hello!

> > Migration of currently-open sockets (for example) would require storing of
> > a lot of state, wouldn't it?
>
> In a word, yes. :)

Yes. But, actually, it is not "for example". Socket state is really far more
complicated thing than all the rest. I would say, migration of another
objects is mostly trivial thing.

Actually, what Andrew worried about:

> snapshot/restart/migration worry me. If they require complete
> serialisation of complex kernel data structures then we have a problem,
> because it means that any time anyone changes such a structure they need to
> update (and test) the serialisation.

The answer is: after user space processes referring to objects are suspended,
surprizingly, not so much of places, which have trouble with serialization
remain. Actually, no serialization additional to existing one is required.
Sockets are the most complicated, to suspend networking state, after
processes are frozen, we have to:

1. Block access from network.
2. Stop socket timers.

Only after this we can make a coherent snapshot. But it is an exception,
most of objects are in coherent state (all the VM, files etc. etc),
when processes are frozen.

> I don't think the networking guys from either the OpenVZ project or IBM
> were cc'd on this. Alexey, Daniel, can you elaborate, or point us to
> any existing code?

http://git.openvz.org

linux-2.6-openvz/kernel/cpt/. Particularly, kernel/cpt/cpt_socket*.c.
Hairy, but straighforward.

Alexey

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by serue on Sun, 21 May 2006 16:27:59 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> > This patchset introduces a per-process utsname namespace. These can
> > be used by openvz, vserver, and application migration to virtualize and
> > isolate utsname info (i.e. hostname). More resources will follow, until
> > hopefully most or all vserver and openvz functionality can be implemented
> > by controlling resource namespaces from userspace.
> >
> > Previous utsname submissions placed a pointer to the utsname namespace
> > straight in the task_struct. This patchset (and the last one) moves
> > it and the filesystem namespace pointer into struct nsproxy, which is
> > shared by processes sharing all namespaces. The intent is to keep
> > the taskstruct smaller as the number of namespaces grows.
>
>
> Previously you mentioned:
> > BTW - a first set of comparison results showed nsproxy to have better
> > dbench and tbench throughput, and worse kernbench performance. Which
> > may make sense given that nsproxy results in lower memory usage but
> > likely increased cache misses due to extra pointer dereference.
>
> Is this still true? Or did our final reference counting tweak fix
> the kernbench numbers?
>
> I just want to be certain that we don't add an optimization,
> that reduces performance.

Here are the numbers with the basic patchsets. But I guess I should
do another round with adding 7 more void*'s to represent additional
namespaces.

(intervals are for 95% CI, tests were each run 15 times)

| with nsproxy | without nsproxy |
kernbench | 68.90 +/- 0.21 | 69.06 +/- 0.22 |
dbench | 386.0 +/- 26.6 | 388.4 +/- 21.0 |
tbench | 391.6 +/- 8.00 | 389.4 +/- 10.95 |

reaim with nsproxy

1 115600.000000 5512.441557
3 246985.712000 9375.780582
5 272309.092000 8029.833742
7 290020.000000 7288.367116
9 298591.580000 5557.531915
11 nan nan
13 nan nan
15 nan nan

reaim without nsproxy
1 110160.000000 5728.697311
3 246985.712000 9375.780582
5 262204.197333 11138.510652
7 288660.000000 6880.898412
9 300631.580000 4351.926692
11 nan nan
13 nan nan
15 nan nan

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by ebiederm on Sun, 21 May 2006 18:08:53 GMT
View Forum Message <> Reply to Message

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

>
> Here are the numbers with the basic patchsets. But I guess I should
> do another round with adding 7 more void*'s to represent additional
> namespaces.

I'm a little slow coming up to speed on these benchmarks.
dbench and tbench are measured in megabytes per second correct?
kernbench is the number of seconds it takes to compile a kernel?
reaim is measured in jobs per minute?

So if I read this right the differences are currently in
the noise levels, from your testing.

> (intervals are for 95% CI, tests were each run 15 times)
>
> | with nsproxy | without nsproxy |
> kernbench | 68.90 +/- 0.21 | 69.06 +/- 0.22 |
> dbench | 386.0 +/- 26.6 | 388.4 +/- 21.0 |
> tbench | 391.6 +/- 8.00 | 389.4 +/- 10.95 |
>
> reaim with nsproxy
> 1 115600.000000 5512.441557

> 3 246985.712000 9375.780582
> 5 272309.092000 8029.833742
> 7 290020.000000 7288.367116
> 9 298591.580000 5557.531915
> 11 nan nan
> 13 nan nan
> 15 nan nan
>
> reaim without nsproxy
> 1 110160.000000 5728.697311
> 3 246985.712000 9375.780582
> 5 262204.197333 11138.510652
> 7 288660.000000 6880.898412
> 9 300631.580000 4351.926692
> 11 nan nan
> 13 nan nan
> 15 nan nan

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by ebiederm on Sun, 21 May 2006 23:18:50 GMT
View Forum Message <> Reply to Message

Pavel Machek <pavel@ucw.cz> writes:

> Well, if pid #1 virtualization is only needed for pstree, we may want
> to fix pstree instead :-).

One thing that is not clear is if isolation by permission checks is any
easier to implement than isolation with a namespace.

Isolation at permission checks may actually be more expensive in terms
of execution time, and maintenance.

Eric

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by Herbert Poetzl on Sun, 21 May 2006 23:32:15 GMT
View Forum Message <> Reply to Message

On Sun, May 21, 2006 at 05:18:50PM -0600, Eric W. Biederman wrote:
> Pavel Machek <pavel@ucw.cz> writes:
>
> > Well, if pid #1 virtualization is only needed for pstree, we may want
> > to fix pstree instead :-).

yes, actually this and init itself (which uses the
pid to switch between init and telinit behaviour)
are the only two applications we found so far ...

and as far as I know, those work with non pid=1
values on other operating systems (inside containers)

a fix there would definitely be appreciated and
I think it would not hurt normal behaviour ...

> One thing that is not clear is if isolation by permission checks is
> any easier to implement than isolation with a namespace.

for the pid space, I'm not really sure if isolation
is really cheaper than virtualization, but for the
network space for example, a virtualization solution
which is as lightweigth as the isolation is probably
more challenging, although not impossible ...

> Isolation at permission checks may actually be more expensive in terms
> of execution time, and maintenance.

again, for the pid space, maintenance is quite low ..

best,
Herbert

> Eric

Subject: Re: [PATCH 4/9] namespaces: utsname: switch to using uts namespaces
Posted by Sam Vilain on Mon, 22 May 2006 00:19:20 GMT
View Forum Message <> Reply to Message

Serge E. Hallyn wrote:

>--- a/arch/alpha/kernel/osf_sys.c
>+++ b/arch/alpha/kernel/osf_sys.c
>@@ -402,15 +402,15 @@ osf_utsname(char __user *name)
>
> down_read(&uts_sem);
> error = -EFAULT;
>- if (copy_to_user(name + 0, system_utsname.sysname, 32))
>+ if (copy_to_user(name + 0, utsname()->sysname, 32))
>   goto out;
>diff --git a/arch/i386/kernel/sys_i386.c b/arch/i386/kernel/sys_i386.c
>index 8fdb1fb..4af731d 100644
>--- a/arch/i386/kernel/sys_i386.c

>+++ b/arch/i386/kernel/sys_i386.c
>@@ -210,7 +210,7 @@ asmlinkage int sys_uname(struct old_utsn
> if (!name)
>   return -EFAULT;
> down_read(&uts_sem);
>- err=copy_to_user(name, &system_utsname, sizeof (*name));
>+ err=copy_to_user(name, utsname(), sizeof (*name));
> up_read(&uts_sem);
> return err?-EFAULT:0;
> }
>
>

The semaphore (uts_sem) should be moved in the uts_ns structure, no?

It's probably low impact enough to keep it as it is, though. Just a tad
untidy.

Sam.

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by serue on Mon, 22 May 2006 12:10:24 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> >
> > Here are the numbers with the basic patchsets. But I guess I should
> > do another round with adding 7 more void*'s to represent additional
> > namespaces.
>
> I'm a little slow coming up to speed on these benchmarks.
> dbench and tbench are measured in megabytes per second correct?
> kernbench is the number of seconds it takes to compile a kernel?
> reaim is measured in jobs per minute?
>
> So if I read this right the differences are currently in
> the noise levels, from your testing.

Yup.

Adding 7 extra void*'s seems to affect only dbench, which
whose degration with the nsproxy falls outside the noise.
The odd thing isn't so much the degradation, but the widely
scattered values, compared to without nsproxy.

| with nsproxy | without nsproxy |
kernbench | 70.23 +/- 0.27 | 70.04 +/- 0.22 |
dbench | 367.1 +/- 32.6 | 410.0 +/- 2.96 |
tbench | 399.3 +/- 12.4 | 399.4 +/- 12.5 |

reaim with nsproxy
1 115600.000000 5512.441557
3 243099.998000 10876.225044
5 270002.798667 11800.545221
7 283291.578667 10897.147984
9 294530.528000 7095.760045
11 nan nan
13 nan nan
15 nan nan

reaim without nsproxy
1 114240.000000 5728.697311
3 254271.426000 11767.994417
5 279965.036000 12615.448140
7 281660.000000 9898.028733
9 302905.264000 5165.026561
11 nan nan
13 nan nan
15 nan nan

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by ebiederm on Mon, 22 May 2006 16:44:32 GMT
View Forum Message <> Reply to Message

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

> Quoting Eric W. Biederman (ebiederm@xmission.com):

> Yup.
>
> Adding 7 extra void*'s seems to affect only dbench, which
> whose degration with the nsproxy falls outside the noise.
> The odd thing isn't so much the degradation, but the widely
> scattered values, compared to without nsproxy.

Well dbench is filesystem heavy so it may have some need to
actually be using the filesystem namespace. Although I can't
think of why it would. Do you wan to track down that performance
regression or do you want to give up on that space optimization for
now?

Eric

Subject: Re: [PATCH 0/9] namespaces: Introduction
Posted by ebiederm on Mon, 22 May 2006 16:54:11 GMT
View Forum Message <> Reply to Message

Yep. I bungle my description pretty badly.

The key points.
- Simply messing with pid == 1 is not enough, you need to filter
  which pids are accessible.
- pid isolation by permission checks and pid isolation via
  pid visibility are competing implementations.
- pid isolation by permission checks (except for the pid == 1 case)
  can currently be implemented with a security module.

Eric

Subject: Re: [PATCH 4/9] namespaces: utsname: switch to using uts namespaces
Posted by Cedric Le Goater on Mon, 22 May 2006 19:43:37 GMT
View Forum Message <> Reply to Message

Randy.Dunlap wrote:
> On Thu, 18 May 2006 10:49:36 -0500 Serge E. Hallyn wrote:
>
>> Replace references to system_utsname to the per-process uts namespace
>> where appropriate. This includes things like uname.
>>
>> Changes: Per Eric Biederman's comments, use the per-process uts namespace
>> for ELF_PLATFORM, sunrpc, and parts of net/ipv4/ipconfig.c
>>
>> Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
>>
>> ---
>>
>> 9ee063adf4d2287583dbb0a71d1d5f80d7ae011f
>> diff --git a/arch/i386/kernel/sys_i386.c b/arch/i386/kernel/sys_i386.c
>> index 8fdb1fb..4af731d 100644
>> --- a/arch/i386/kernel/sys_i386.c
>> +++ b/arch/i386/kernel/sys_i386.c
>> @@ -210,7 +210,7 @@ asmlinkage int sys_uname(struct old_utsn
>> if (!name)
>> return -EFAULT;
>> down_read(&uts_sem);
>> - err=copy_to_user(name, &system_utsname, sizeof (*name));

>> + err=copy_to_user(name, utsname(), sizeof (*name));
>
> It would be really nice if you would fix spacing while you are here,
> like a space a each side of '='.
>
> and a space after ',' in the function calls below.

Here's a possible cleanup on top of serge's patchset as found in
2.6.17-rc4-mm3.

C.

Subject: Re: [PATCH 4/9] namespaces: utsname: switch to using uts namespaces
Posted by rdunlap on Mon, 22 May 2006 20:16:52 GMT
View Forum Message <> Reply to Message

On Mon, 22 May 2006 21:43:37 +0200 Cedric Le Goater wrote:

> Randy.Dunlap wrote:
> >>
> >> 9ee063adf4d2287583dbb0a71d1d5f80d7ae011f
> >> diff --git a/arch/i386/kernel/sys_i386.c b/arch/i386/kernel/sys_i386.c
> >> index 8fdb1fb..4af731d 100644
> >> --- a/arch/i386/kernel/sys_i386.c
> >> +++ b/arch/i386/kernel/sys_i386.c
> >> @@ -210,7 +210,7 @@ asmlinkage int sys_uname(struct old_utsn
> >>   if (!name)
> >>   return -EFAULT;
> >>   down_read(&uts_sem);
> >> - err=copy_to_user(name, &system_utsname, sizeof (*name));
> >> + err=copy_to_user(name, utsname(), sizeof (*name));
> >
> > It would be really nice if you would fix spacing while you are here,
> > like a space a each side of '='.
> >
> > and a space after ',' in the function calls below.
>
> Here's a possible cleanup on top of serge's patchset as found in
> 2.6.17-rc4-mm3.

Yes, thanks, looks good.

~Randy
