

---

Subject: [PATCH 1/4] userns: let clone\_uts\_ns() handle setting uts-&gt;user\_ns  
Posted by [serge](#) on Mon, 21 Feb 2011 04:01:57 GMT

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

---

To do so we need to pass in the task\_struct who'll get the utsname,  
so we can get its user\_ns.

Signed-off-by: Serge E. Hallyn <[serge.hallyn@canonical.com](mailto:serge.hallyn@canonical.com)>

```
---  
include/linux/utsname.h | 10 ++++++----  
kernel/nsproxy.c      |  7 +-----  
kernel/utsname.c      | 12 ++++++----  
3 files changed, 14 insertions(+), 15 deletions(-)
```

```
diff --git a/include/linux/utsname.h b/include/linux/utsname.h  
index 85171be..165b17b 100644  
--- a/include/linux/utsname.h  
+++ b/include/linux/utsname.h  
@@ -52,8 +52,9 @@ static inline void get_uts_ns(struct uts_namespace *ns)  
    kref_get(&ns->kref);  
}  
  
-extern struct uts_namespace *copy_utsname(unsigned long flags,  
-    struct uts_namespace *ns);  
+extern struct uts_namespace *copy_utsname(struct task_struct *tsk,  
+    unsigned long flags,  
+    struct uts_namespace *ns);  
extern void free_uts_ns(struct kref *kref);  
  
static inline void put_uts_ns(struct uts_namespace *ns)  
@@ -69,8 +70,9 @@ static inline void put_uts_ns(struct uts_namespace *ns)  
{  
}  
  
-static inline struct uts_namespace *copy_utsname(unsigned long flags,  
-    struct uts_namespace *ns)  
+static inline struct uts_namespace *copy_utsname(struct task_struct *tsk,  
+    unsigned long flags,  
+    struct uts_namespace *ns)  
{  
    if (flags & CLONE_NEWUTS)  
        return ERR_PTR(-EINVAL);  
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c  
index b6dbff2..ffa6b67 100644  
--- a/kernel/nsproxy.c  
+++ b/kernel/nsproxy.c  
@@ -69,16 +69,11 @@ static struct nsproxy *create_new_namespaces(unsigned long flags,  
    goto out_ns;
```

```

}

- new_nsp->uts_ns = copy_utsname(flags, tsk->nsproxy->uts_ns);
+ new_nsp->uts_ns = copy_utsname(tsk, flags, tsk->nsproxy->uts_ns);
if (IS_ERR(new_nsp->uts_ns)) {
    err = PTR_ERR(new_nsp->uts_ns);
    goto out_uts;
}
- if (new_nsp->uts_ns != tsk->nsproxy->uts_ns) {
- put_user_ns(new_nsp->uts_ns->user_ns);
- new_nsp->uts_ns->user_ns = task_cred_xxx(tsk, user)->user_ns;
- get_user_ns(new_nsp->uts_ns->user_ns);
- }

new_nsp->ipc_ns = copy_ipcs(flags, tsk->nsproxy->ipc_ns);
if (IS_ERR(new_nsp->ipc_ns)) {
diff --git a/kernel/utsname.c b/kernel/utsname.c
index a7b3a8d..9462580 100644
--- a/kernel/utsname.c
+++ b/kernel/utsname.c
@@ -31,7 +31,8 @@ static struct uts_namespace *create_uts_ns(void)
 * @old_ns: namespace to clone
 * Return NULL on error (failure to kmalloc), new ns otherwise
 */
-static struct uts_namespace *clone_uts_ns(struct uts_namespace *old_ns)
+static struct uts_namespace *clone_uts_ns(struct task_struct *tsk,
+    struct uts_namespace *old_ns)
{
    struct uts_namespace *ns;

@@ -41,8 +42,7 @@ static struct uts_namespace *clone_uts_ns(struct uts_namespace *old_ns)

down_read(&uts_sem);
memcpy(&ns->name, &old_ns->name, sizeof(ns->name));
- ns->user_ns = old_ns->user_ns;
- get_user_ns(ns->user_ns);
+ ns->user_ns = get_user_ns(task_cred_xxx(tsk, user)->user_ns);
    up_read(&uts_sem);
    return ns;
}
@@ -53,7 +53,9 @@ static struct uts_namespace *clone_uts_ns(struct uts_namespace *old_ns)
 * utsname of this process won't be seen by parent, and vice
 * versa.
 */
-struct uts_namespace *copy_utsname(unsigned long flags, struct uts_namespace *old_ns)
+struct uts_namespace *copy_utsname(struct task_struct *tsk,
+    unsigned long flags,
+    struct uts_namespace *old_ns)

```

```

{
    struct uts_namespace *new_ns;

@@ -63,7 +65,7 @@ struct uts_namespace *copy_utsname(unsigned long flags, struct
    uts_namespace *ol
    if (!(flags & CLONE_NEWUTS))
        return old_ns;

- new_ns = clone_uts_ns(old_ns);
+ new_ns = clone_uts_ns(tsk, old_ns);

    put_uts_ns(old_ns);
    return new_ns;
}

```

#### 1.7.0.4

---

Containers mailing list  
 Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---



---

Subject: [PATCH 2/4] userns: let copy\_ipcs handle setting ipc\_ns->user\_ns  
 Posted by [serge](#) on Mon, 21 Feb 2011 04:02:30 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

To do that, we have to pass in the task\_struct of the task which will own the ipc\_ns, so we can assign its user\_ns.

Signed-off-by: Serge E. Hallyn <[serge.hallyn@canonical.com](mailto:serge.hallyn@canonical.com)>

---

```

include/linux/ipc_namespace.h |  8 ++++++-
ipc/namespace.c            | 12 ++++++----
kernel/nsproxy.c           |  7 +-----
3 files changed, 13 insertions(+), 14 deletions(-)

```

```

diff --git a/include/linux/ipc_namespace.h b/include/linux/ipc_namespace.h
index 46d2eb4..9974429 100644
--- a/include/linux/ipc_namespace.h
+++ b/include/linux/ipc_namespace.h
@@ -92,7 +92,8 @@ static inline int mq_init_ns(struct ipc_namespace *ns) { return 0; }
#endif

```

```

#if defined(CONFIG_IPC_NS)
-extern struct ipc_namespace *copy_ipcs(unsigned long flags,
+extern struct ipc_namespace *copy_ipcs(struct task_struct *tsk,
+    unsigned long flags,
        struct ipc_namespace *ns);

```

```

static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)
{
@@ -103,8 +104,9 @@ static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace
*ns)

extern void put_ipc_ns(struct ipc_namespace *ns);
#else
-static inline struct ipc_namespace *copy_ipcs(unsigned long flags,
- struct ipc_namespace *ns)
+static inline struct ipc_namespace *copy_ipcs(struct task_struct *tsk,
+ unsigned long flags,
+ struct ipc_namespace *ns)
{
    if (flags & CLONE_NEWIPC)
        return ERR_PTR(-EINVAL);
diff --git a/ipc/namespace.c b/ipc/namespace.c
index aa18899..ee84882 100644
--- a/ipc/namespace.c
+++ b/ipc/namespace.c
@@ -15,7 +15,8 @@

#include "util.h"

-static struct ipc_namespace *create_ipc_ns(struct ipc_namespace *old_ns)
+static struct ipc_namespace *create_ipc_ns(struct task_struct *tsk,
+ struct ipc_namespace *old_ns)
{
    struct ipc_namespace *ns;
    int err;
@@ @ -44,17 +45,18 @@ static struct ipc_namespace *create_ipc_ns(struct ipc_namespace
*old_ns)
    ipcns_notify(IPCNS_CREATED);
    register_ipcns_notifier(ns);

- ns->user_ns = old_ns->user_ns;
- get_user_ns(ns->user_ns);
+ ns->user_ns = get_user_ns(task_cred_xxx(tsk, user)->user_ns);

    return ns;
}

-struct ipc_namespace *copy_ipcs(unsigned long flags, struct ipc_namespace *ns)
+struct ipc_namespace *copy_ipcs(struct task_struct *tsk,
+ unsigned long flags,
+ struct ipc_namespace *ns)
{
    if (!(flags & CLONE_NEWIPC))
        return get_ipc_ns(ns);

```

```

- return create_ipc_ns(ns);
+ return create_ipc_ns(tsk, ns);
}

/*
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index ffa6b67..b905ecc 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -75,16 +75,11 @@ static struct nsproxy *create_new_namespaces(unsigned long flags,
    goto out_uts;
}

- new_nsp->ipc_ns = copy_ipcs(flags, tsk->nsproxy->ipc_ns);
+ new_nsp->ipc_ns = copy_ipcs(tsk, flags, tsk->nsproxy->ipc_ns);
if (IS_ERR(new_nsp->ipc_ns)) {
    err = PTR_ERR(new_nsp->ipc_ns);
    goto out_ipc;
}
- if (new_nsp->ipc_ns != tsk->nsproxy->ipc_ns) {
- put_user_ns(new_nsp->ipc_ns->user_ns);
- new_nsp->ipc_ns->user_ns = task_cred_xxx(tsk, user)->user_ns;
- get_user_ns(new_nsp->ipc_ns->user_ns);
- }

new_nsp->pid_ns = copy_pid_ns(flags, task_active_pid_ns(tsk));
if (IS_ERR(new_nsp->pid_ns)) {
--
```

1.7.0.4

Containers mailing list  
 Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: [PATCH 3/4] Add the required user\_ns parameter to security\_capable  
 Posted by [serge](#) on Mon, 21 Feb 2011 04:02:47 GMT

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

Fixes a compile failure.

Signed-off-by: Serge E. Hallyn <[serge.hallyn@canonical.com](mailto:serge.hallyn@canonical.com)>

---

drivers/pci/pci-sysfs.c | 2 ++
 1 files changed, 1 insertions(+), 1 deletions(-)

diff --git a/drivers/pci/pci-sysfs.c b/drivers/pci/pci-sysfs.c

```
index ea25e5b..90a6b04 100644
--- a/drivers/pci/pci-sysfs.c
+++ b/drivers/pci/pci-sysfs.c
@@ -369,7 +369,7 @@ pci_read_config(struct file *filp, struct kobject *kobj,
 u8 *data = (u8*) buf;

 /* Several chips lock up trying to read undefined config space */
- if (security_capable(filp->f_cred, CAP_SYS_ADMIN) == 0) {
+ if (security_capable(&init_user_ns, filp->f_cred, CAP_SYS_ADMIN) == 0) {
    size = dev->cfg_size;
} else if (dev->hdr_type == PCI_HEADER_TYPE_CARDBUS) {
    size = 128;
--
```

1.7.0.4

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: [PATCH 4/4] userns: uts and ipc: fix checkpatch warning  
Posted by [serge](#) on Mon, 21 Feb 2011 04:05:27 GMT

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

---

As pointed out by Andrew Morton (and checkpatch), init/version.c  
(and ipc/msgutil.c) should not have an extern declaration for  
init\_user\_ns. Instead, move those to ipc\_namespace.h and utsname.h.

Signed-off-by: Serge E. Hallyn <[serge.hallyn@canonical.com](mailto:serge.hallyn@canonical.com)>

---

```
include/linux/ipc_namespace.h |  3 ++
include/linux/utsname.h      |  1 +
init/version.c              |  1 -
ipc/msgutil.c               |  2 --
4 files changed, 3 insertions(+), 4 deletions(-)
```

```
diff --git a/include/linux/ipc_namespace.h b/include/linux/ipc_namespace.h
index 9974429..ebd4a93 100644
--- a/include/linux/ipc_namespace.h
+++ b/include/linux/ipc_namespace.h
@@ -15,6 +15,8 @@
```

```
#define IPCNS_CALLBACK_PRI 0

+struct user_namespace;
+extern struct user_namespace init_user_ns;
```

```

struct ipc_ids {
    int in_use;
@@ -24,7 +26,6 @@ struct ipc_ids {
    struct idr ipcs_idr;
};

-struct user_namespace;
struct ipc_namespace {
    atomic_t count;
    struct ipc_ids ids[3];
diff --git a/include/linux/utsname.h b/include/linux/utsname.h
index 165b17b..69957ca 100644
--- a/include/linux/utsname.h
+++ b/include/linux/utsname.h
@@ -38,6 +38,7 @@ struct new_utsname {
#include <linux/err.h>

struct user_namespace;
+extern struct user_namespace init_user_ns;

struct uts_namespace {
    struct kref kref;
diff --git a/init/version.c b/init/version.c
index 97bb86f..86fe0cc 100644
--- a/init/version.c
+++ b/init/version.c
@@ -21,7 +21,6 @@ extern int version_string(LINUX_VERSION_CODE);
int version_string(LINUX_VERSION_CODE);
#endif

-extern struct user_namespace init_user_ns;
struct uts_namespace init_uts_ns = {
    .kref = {
        .refcount = ATOMIC_INIT(2),
diff --git a/ipc/msgutil.c b/ipc/msgutil.c
index d91ff4b..8b5ce5d 100644
--- a/ipc/msgutil.c
+++ b/ipc/msgutil.c
@@ -20,8 +20,6 @@ 

#define DEFINE_SPINLOCK(mq_lock);

-extern struct user_namespace init_user_ns;
-
/*
 * The next 2 defines are here bc this is the only file
 * compiled when either CONFIG_SYSVIPC and CONFIG_POSIX_MQUEUE
--
```

## 1.7.0.4

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

Subject: Re: [PATCH 1/4] userns: let clone\_uts\_ns() handle setting uts-&gt;user\_ns  
Posted by [Daniel Lezcano](#) on Mon, 21 Feb 2011 10:03:50 GMT

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

On 02/21/2011 05:01 AM, Serge E. Hallyn wrote:

> To do so we need to pass in the task\_struct who'll get the utsname,  
> so we can get its user\_ns.  
>  
> Signed-off-by: Serge E. Hallyn<[serge.hallyn@canonical.com](mailto:serge.hallyn@canonical.com)>  
> ---

```
> include/linux/utsname.h | 10 ++++++----  
> kernel/nsproxy.c      |  7 +-----  
> kernel/utsname.c      | 12 ++++++----  
> 3 files changed, 14 insertions(+), 15 deletions(-)  
>  
> diff --git a/include/linux/utsname.h b/include/linux/utsname.h  
> index 85171be..165b17b 100644  
> --- a/include/linux/utsname.h  
> +++ b/include/linux/utsname.h  
> @@ -52,8 +52,9 @@ static inline void get_uts_ns(struct uts_namespace *ns)  
>   kref_get(&ns->kref);  
> }  
>  
> -extern struct uts_namespace *copy_utsname(unsigned long flags,  
> -    struct uts_namespace *ns);  
> +extern struct uts_namespace *copy_utsname(struct task_struct *tsk,  
> +    unsigned long flags,  
> +    struct uts_namespace *ns);
```

Why don't we pass 'user\_ns' instead of 'tsk' ? that will look  
semantically clearer for the caller no ?  
(example below).

```
> extern void free_uts_ns(struct kref *kref);  
>  
> static inline void put_uts_ns(struct uts_namespace *ns)  
> @@ -69,8 +70,9 @@ static inline void put_uts_ns(struct uts_namespace *ns)  
> {
```

```

> }
>
> -static inline struct uts_namespace *copy_utsname(unsigned long flags,
> -    struct uts_namespace *ns)
> +static inline struct uts_namespace *copy_utsname(struct task_struct *tsk,
> +    unsigned long flags,
> +    struct uts_namespace *ns)
> {
>     if (flags & CLONE_NEWUTS)
>         return ERR_PTR(-EINVAL);
> diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
> index b6dbff2..ffa6b67 100644
> --- a/kernel/nsproxy.c
> +++ b/kernel/nsproxy.c
> @@ -69,16 +69,11 @@ static struct nsproxy *create_new_namespaces(unsigned long flags,
>     goto out_ns;
> }
>
> - new_nsp->uts_ns = copy_utsname(flags, tsk->nsproxy->uts_ns);
> + new_nsp->uts_ns = copy_utsname(tsk, flags, tsk->nsproxy->uts_ns);
>     if (IS_ERR(new_nsp->uts_ns)) {
>         err = PTR_ERR(new_nsp->uts_ns);
>         goto out_uts;
>     }
>
...
new_nsp->uts_ns = copy_utsname(flags, tsk->nsproxy->uts_ns, task_cred_xxx(tsk,
user)->user_ns);

...
> - if (new_nsp->uts_ns != tsk->nsproxy->uts_ns) {
> -     put_user_ns(new_nsp->uts_ns->user_ns);
> -     new_nsp->uts_ns->user_ns = task_cred_xxx(tsk, user)->user_ns;
> -     get_user_ns(new_nsp->uts_ns->user_ns);
> - }
>
>     new_nsp->ipc_ns = copy_ipcs(flags, tsk->nsproxy->ipc_ns);
>     if (IS_ERR(new_nsp->ipc_ns)) {
> diff --git a/kernel/utsname.c b/kernel/utsname.c
> index a7b3a8d..9462580 100644
> --- a/kernel/utsname.c
> +++ b/kernel/utsname.c
> @@ -31,7 +31,8 @@ static struct uts_namespace *create_uts_ns(void)
>     * @old_ns: namespace to clone
>     * Return NULL on error (failure to kmalloc), new ns otherwise
>     */

```

```

> -static struct uts_namespace *clone_uts_ns(struct uts_namespace *old_ns)
> +static struct uts_namespace *clone_uts_ns(struct task_struct *tsk,
> +    struct uts_namespace *old_ns)
> {
>     struct uts_namespace *ns;
>
> @@ -41,8 +42,7 @@ static struct uts_namespace *clone_uts_ns(struct uts_namespace
> *old_ns)
>
>     down_read(&uts_sem);
>     memcpy(&ns->name,&old_ns->name, sizeof(ns->name));
> - ns->user_ns = old_ns->user_ns;
> - get_user_ns(ns->user_ns);
> + ns->user_ns = get_user_ns(task_cred_xxx(tsk, user)->user_ns);
>     up_read(&uts_sem);
>     return ns;
> }
> @@ -53,7 +53,9 @@ static struct uts_namespace *clone_uts_ns(struct uts_namespace
> *old_ns)
>     * utsname of this process won't be seen by parent, and vice
>     * versa.
>     */
> -struct uts_namespace *copy_utsname(unsigned long flags, struct uts_namespace *old_ns)
> +struct uts_namespace *copy_utsname(struct task_struct *tsk,
> +    unsigned long flags,
> +    struct uts_namespace *old_ns)
> {
>     struct uts_namespace *new_ns;
>
> @@ -63,7 +65,7 @@ struct uts_namespace *copy_utsname(unsigned long flags, struct
> uts_namespace *ol
>     if (!(flags & CLONE_NEWUTS))
>         return old_ns;
>
> - new_ns = clone_uts_ns(old_ns);
> + new_ns = clone_uts_ns(tsk, old_ns);
>
>     put_uts_ns(old_ns);
>     return new_ns;

```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [PATCH 2/4] userns: let copy\_ipcs handle setting ipc\_ns->user\_ns

Posted by [Daniel Lezcano](#) on Mon, 21 Feb 2011 10:05:26 GMT

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

---

On 02/21/2011 05:02 AM, Serge E. Hallyn wrote:

> To do that, we have to pass in the task\_struct of the task which  
> will own the ipc\_ns, so we can assign its user\_ns.

>  
> Signed-off-by: Serge E. Hallyn<serge.hallyn@canonical.com>  
> ---  
> include/linux/ipc\_namespace.h | 8 +++++++  
> ipc/namespace.c | 12 ++++++----  
> kernel/nsproxy.c | 7 +-----  
> 3 files changed, 13 insertions(+), 14 deletions(-)  
>  
> diff --git a/include/linux/ipc\_namespace.h b/include/linux/ipc\_namespace.h  
> index 46d2eb4..9974429 100644  
> --- a/include/linux/ipc\_namespace.h  
> +++ b/include/linux/ipc\_namespace.h  
> @@ -92,7 +92,8 @@ static inline int mq\_init\_ns(struct ipc\_namespace \*ns) { return 0; }  
> #endif  
>  
> #if defined(CONFIG\_IPC\_NS)  
> -extern struct ipc\_namespace \*copy\_ipcs(unsigned long flags,  
> +extern struct ipc\_namespace \*copy\_ipcs(struct task\_struct \*tsk,  
> + unsigned long flags,  
> struct ipc\_namespace \*ns);

Same comment than patch 1/4

---

Containers mailing list

[Containers@lists.linux-foundation.org](mailto:Containers@lists.linux-foundation.org)

<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: Re: [PATCH 1/4] userns: let clone\_uts\_ns() handle setting uts-&gt;user\_ns

Posted by [Oleg Nesterov](#) on Mon, 21 Feb 2011 13:41:29 GMT

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

---

On 02/21, Daniel Lezcano wrote:

>  
> On 02/21/2011 05:01 AM, Serge E. Hallyn wrote:  
>> To do so we need to pass in the task\_struct who'll get the utsname,  
>> so we can get its user\_ns.  
>>  
>> -extern struct uts\_namespace \*copy\_utsname(unsigned long flags,  
>> - struct uts\_namespace \*ns);  
>> +extern struct uts\_namespace \*copy\_utsname(struct task\_struct \*tsk,  
>> + unsigned long flags,

```
>> +     struct uts_namespace *ns);
>
> Why don't we pass 'user_ns' instead of 'tsk' ? that will look
> semantically clearer for the caller no ?
> (example below).
> ...
>
> new_nsp->uts_ns = copy_utsname(flags, tsk->nsproxy->uts_ns, task_cred_xxx(tsk,
user)->user_ns);
```

To me tsk looks more readable, I mean

```
new_nsp->uts_ns = copy_utsname(flags, tsk);
```

copy\_utsname() can find both uts\_ns and user\_ns looking at task\_struct.

But this is cosmetic and up to you and Serge.

But, I think it makes sense to pass "tsk" argument to copy\_pid\_ns() as well.  
This way we can remove some CLONE\_PIDNS code in copy\_process(), and this  
looks like a nice cleanup (even if minor) to me.

Oleg.

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: Re: [PATCH 1/4] userns: let clone\_uts\_ns() handle setting uts-&gt;user\_ns  
Posted by [serge](#) on Mon, 21 Feb 2011 13:58:03 GMT

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

---

Quoting Oleg Nesterov (oleg@redhat.com):

> On 02/21, Daniel Lezcano wrote:

> >

> > On 02/21/2011 05:01 AM, Serge E. Hallyn wrote:

> >> To do so we need to pass in the task\_struct who'll get the utsname,  
> >> so we can get its user\_ns.

> >>

> >> -extern struct uts\_namespace \*copy\_utsname(unsigned long flags,  
> >> - struct uts\_namespace \*ns);  
> >> +extern struct uts\_namespace \*copy\_utsname(struct task\_struct \*tsk,  
> >> + unsigned long flags,  
> >> + struct uts\_namespace \*ns);  
> >

```
>> Why don't we pass 'user_ns' instead of 'tsk' ? that will look
>> semantically clearer for the caller no ?
>> (example below).
>> ...
>>
>> new_nsp->uts_ns = copy_utsname(flags, tsk->nsproxy->uts_ns, task_cred_xxx(tsk,
user)->user_ns);
>
> To me tsk looks more readable, I mean
>
> new_nsp->uts_ns = copy_utsname(flags, tsk);
>
> copy_utsname() can find both uts_ns and user_ns looking at task_struct.
```

Uh, yeah. I should remove the 'ns' argument there shouldn't I.

Daniel, does that sway your opinion then?

thanks,  
-serge

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: Re: [PATCH 1/4] userns: let clone\_uts\_ns() handle setting uts-&gt;user\_ns  
Posted by [Daniel Lezcano](#) on Mon, 21 Feb 2011 14:23:30 GMT

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

---

On 02/21/2011 02:58 PM, Serge E. Hallyn wrote:  
> Quoting Oleg Nesterov (oleg@redhat.com):  
>> On 02/21, Daniel Lezcano wrote:  
>>> On 02/21/2011 05:01 AM, Serge E. Hallyn wrote:  
>>>> To do so we need to pass in the task\_struct who'll get the utsname,  
>>>> so we can get its user\_ns.  
>>>  
>>>> -extern struct uts\_namespace \*copy\_utsname(unsigned long flags,  
>>>> - struct uts\_namespace \*ns);  
>>>> +extern struct uts\_namespace \*copy\_utsname(struct task\_struct \*tsk,  
>>>> + unsigned long flags,  
>>>> + struct uts\_namespace \*ns);  
>> Why don't we pass 'user\_ns' instead of 'tsk' ? that will look  
>> semantically clearer for the caller no ?  
>> (example below).  
>> ...  
>>  
>>> new\_nsp->uts\_ns = copy\_utsname(flags, tsk->nsproxy->uts\_ns, task\_cred\_xxx(tsk,

```
user)->user_ns);
>> To me tsk looks more readable, I mean
>>
>> new_nsp->uts_ns = copy_utsname(flags, tsk);
>>
>> copy_utsname() can find both uts_ns and user_ns looking at task_struct.
> Uh, yeah. I should remove the 'ns' argument there shouldn't I.
>
> Daniel, does that sway your opinion then?
```

Well, I prefer to pass the needed parameters to a function. AFAICS, 'tsk' is not really needed but 'user\_ns'.

But it is a detail, so if passing the tsk parameter in the other copy\_\* functions helps to cleanup, that will be consistent.

So I am fine with that.

---

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: [PATCH 1/4] userns: let clone\_uts\_ns() handle setting uts-&gt;user\_ns

Posted by [serge](#) on Thu, 24 Feb 2011 00:21:41 GMT

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

---

To do so we need to pass in the task\_struct who'll get the utsname, so we can get its user\_ns.

Changelog:

Feb 23: As per Oleg's coment, just pass in tsk.

Signed-off-by: Serge E. Hallyn <[serge.hallyn@canonical.com](mailto:serge.hallyn@canonical.com)>

---

```
include/linux/utsname.h |  6 +-----
kernel/nsproxy.c      |  7 +-----
kernel/utsname.c      | 12 ++++++-----
3 files changed, 11 insertions(+), 14 deletions(-)
```

```
diff --git a/include/linux/utsname.h b/include/linux/utsname.h
index 85171be..21b4566 100644
--- a/include/linux/utsname.h
+++ b/include/linux/utsname.h
@@ -53,7 +53,7 @@ static inline void get_uts_ns(struct uts_namespace *ns)
}

extern struct uts_namespace *copy_utsname(unsigned long flags,
```

- struct uts\_namespace \*ns);

```

+     struct task_struct *tsk);
extern void free_uts_ns(struct kref *kref);

static inline void put_uts_ns(struct uts_namespace *ns)
@@ -70,12 +70,12 @@ static inline void put_uts_ns(struct uts_namespace *ns)
}

static inline struct uts_namespace *copy_utsname(unsigned long flags,
-     struct uts_namespace *ns)
+     struct task_struct *tsk)
{
    if (flags & CLONE_NEWUTS)
        return ERR_PTR(-EINVAL);

-    return ns;
+    return tsk->nsproxy->uts_ns;
}
#endif

diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index b6dbff2..ac8a56e 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -69,16 +69,11 @@ static struct nsproxy *create_new_namespaces(unsigned long flags,
    goto out_ns;
}

- new_nsp->uts_ns = copy_utsname(flags, tsk->nsproxy->uts_ns);
+ new_nsp->uts_ns = copy_utsname(flags, tsk);
if (IS_ERR(new_nsp->uts_ns)) {
    err = PTR_ERR(new_nsp->uts_ns);
    goto out_uts;
}
- if (new_nsp->uts_ns != tsk->nsproxy->uts_ns) {
-     put_user_ns(new_nsp->uts_ns->user_ns);
-     new_nsp->uts_ns->user_ns = task_cred_xxx(tsk, user)->user_ns;
-     get_user_ns(new_nsp->uts_ns->user_ns);
- }

    new_nsp->ipc_ns = copy_ipcs(flags, tsk->nsproxy->ipc_ns);
    if (IS_ERR(new_nsp->ipc_ns)) {
diff --git a/kernel/utsname.c b/kernel/utsname.c
index a7b3a8d..4464617 100644
--- a/kernel/utsname.c
+++ b/kernel/utsname.c
@@ -31,7 +31,8 @@ static struct uts_namespace *create_uts_ns(void)
 * @old_ns: namespace to clone
 * Return NULL on error (failure to kmalloc), new ns otherwise

```

```

*/
-static struct uts_namespace *clone_uts_ns(struct uts_namespace *old_ns)
+static struct uts_namespace *clone_uts_ns(struct task_struct *tsk,
+    struct uts_namespace *old_ns)
{
    struct uts_namespace *ns;

@@ -41,8 +42,7 @@ static struct uts_namespace *clone_uts_ns(struct uts_namespace *old_ns)

    down_read(&uts_sem);
    memcpy(&ns->name, &old_ns->name, sizeof(ns->name));
- ns->user_ns = old_ns->user_ns;
- get_user_ns(ns->user_ns);
+ ns->user_ns = get_user_ns(task_cred_xxx(tsk, user)->user_ns);
    up_read(&uts_sem);
    return ns;
}
@@ -53,8 +53,10 @@ static struct uts_namespace *clone_uts_ns(struct uts_namespace
*old_ns)
 * utsname of this process won't be seen by parent, and vice
 * versa.
 */
-struct uts_namespace *copy_utsname(unsigned long flags, struct uts_namespace *old_ns)
+struct uts_namespace *copy_utsname(unsigned long flags,
+    struct task_struct *tsk)
{
+ struct uts_namespace *old_ns = tsk->nsproxy->uts_ns;
    struct uts_namespace *new_ns;

    BUG_ON(!old_ns);
@@ -63,7 +65,7 @@ struct uts_namespace *copy_utsname(unsigned long flags, struct
uts_namespace *ol
    if (!(flags & CLONE_NEWUTS))
        return old_ns;

- new_ns = clone_uts_ns(old_ns);
+ new_ns = clone_uts_ns(tsk, old_ns);

    put_uts_ns(old_ns);
    return new_ns;
--
```

1.7.0.4

---

Containers mailing list  
 Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---



---

Subject: [PATCH 2/4] userns: let copy\_ipcs handle setting ipc\_ns->user\_ns  
Posted by [serge](#) on Thu, 24 Feb 2011 00:22:04 GMT

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

To do that, we have to pass in the task\_struct of the task which will own the ipc\_ns, so we can assign its user\_ns.

Changelog:

Feb 23: As per Oleg comment, just pass in tsk. To get the ipc\_ns from the nsproxy we need to include nsproxy.h

Signed-off-by: Serge E. Hallyn <[serge.hallyn@canonical.com](mailto:serge.hallyn@canonical.com)>

```
---  
include/linux/ipc_namespace.h |  7 ++++++  
ipc/namespace.c           | 13 +++++++-----  
kernel/nsproxy.c          |  7 +-----  
3 files changed, 13 insertions(+), 14 deletions(-)  
  
diff --git a/include/linux/ipc_namespace.h b/include/linux/ipc_namespace.h  
index 46d2eb4..c079d09 100644  
--- a/include/linux/ipc_namespace.h  
+++ b/include/linux/ipc_namespace.h  
@@ -5,6 +5,7 @@  
 #include <linux/idr.h>  
 #include <linux/rwsem.h>  
 #include <linux/notifier.h>  
+#include <linux/nsproxy.h>  
  
/*  
 * ipc namespace events  
@@ -93,7 +94,7 @@ static inline int mq_init_ns(struct ipc_namespace *ns) { return 0; }  
  
#if defined(CONFIG_IPC_NS)  
extern struct ipc_namespace *copy_ipcs(unsigned long flags,  
-         struct ipc_namespace *ns);  
+         struct task_struct *tsk);  
static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)  
{  
    if (ns)  
@@ -104,12 +105,12 @@ static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace  
*ns)  
extern void put_ipc_ns(struct ipc_namespace *ns);  
#else  
static inline struct ipc_namespace *copy_ipcs(unsigned long flags,  
-         struct ipc_namespace *ns)  
+         struct task_struct *tsk)  
{  
    if (flags & CLONE_NEWIPC)  
        return ERR_PTR(-EINVAL);
```

```

- return ns;
+ return tsk->nsproxy->ipc_ns;
}

static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)
diff --git a/ipc/namespace.c b/ipc/namespace.c
index aa18899..3c3e522 100644
--- a/ipc/namespace.c
+++ b/ipc/namespace.c
@@ -15,7 +15,8 @@

#include "util.h"

-static struct ipc_namespace *create_ipc_ns(struct ipc_namespace *old_ns)
+static struct ipc_namespace *create_ipc_ns(struct task_struct *tsk,
+    struct ipc_namespace *old_ns)
{
    struct ipc_namespace *ns;
    int err;
@@ -44,17 +45,19 @@ static struct ipc_namespace *create_ipc_ns(struct ipc_namespace
*old_ns)
    ipcns_notify(IPCNS_CREATED);
    register_ipcns_notifier(ns);

- ns->user_ns = old_ns->user_ns;
- get_user_ns(ns->user_ns);
+ ns->user_ns = get_user_ns(task_cred_xxx(tsk, user)->user_ns);

    return ns;
}

-struct ipc_namespace *copy_ipcs(unsigned long flags, struct ipc_namespace *ns)
+struct ipc_namespace *copy_ipcs(unsigned long flags,
+    struct task_struct *tsk)
{
+ struct ipc_namespace *ns = tsk->nsproxy->ipc_ns;
+
if (!(flags & CLONE_NEWIPC))
    return get_ipc_ns(ns);
- return create_ipc_ns(ns);
+ return create_ipc_ns(tsk, ns);
}

/*
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index ac8a56e..a05d191 100644
--- a/kernel/nsproxy.c

```

```
+++ b/kernel/nsproxy.c
@@ -75,16 +75,11 @@ static struct nsproxy *create_new_namespaces(unsigned long flags,
    goto out_uts;
}

- new_nsp->ipc_ns = copy_ipcs(flags, tsk->nsproxy->ipc_ns);
+ new_nsp->ipc_ns = copy_ipcs(flags, tsk);
if (IS_ERR(new_nsp->ipc_ns)) {
    err = PTR_ERR(new_nsp->ipc_ns);
    goto out_ipc;
}
- if (new_nsp->ipc_ns != tsk->nsproxy->ipc_ns) {
- put_user_ns(new_nsp->ipc_ns->user_ns);
- new_nsp->ipc_ns->user_ns = task_cred_xxx(tsk, user)->user_ns;
- get_user_ns(new_nsp->ipc_ns->user_ns);
- }

new_nsp->pid_ns = copy_pid_ns(flags, task_active_pid_ns(tsk));
if (IS_ERR(new_nsp->pid_ns)) {
--
```

#### 1.7.0.4

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: [PATCH 5/4] Clean up capability.h and capability.c  
Posted by [serge](#) on Thu, 24 Feb 2011 00:22:49 GMT

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

---

Convert macros to functions to let type safety do its thing. Switch some functions from ints to more appropriate bool. Move all forward declarations together to top of the #ifdef \_\_KERNEL\_\_ section. Use kernel-doc format for comments.

Some macros couldn't be converted because they use functions from security.h which sometimes are extern and sometimes static inline, and we don't want to #include security.h in capability.h.

Also add a real current\_user\_ns function (and convert the existing macro to \_current\_user\_ns() so we can use it in capability.h without #including cred.h.

Signed-off-by: Serge E. Hallyn <[serge.hallyn@canonical.com](mailto:serge.hallyn@canonical.com)>

--

include/linux/capability.h | 38 ++++++-----

```

include/linux/cred.h      |  4 +++
kernel/capability.c      | 20 ++++++-----+
kernel/cred.c            |  5 +++++
4 files changed, 46 insertions(+), 21 deletions(-)

```

```

diff --git a/include/linux/capability.h b/include/linux/capability.h
index bc0f262..688462f 100644
--- a/include/linux/capability.h
+++ b/include/linux/capability.h
@@ -368,6 +368,17 @@ struct cpu_vfs_cap_data {

#ifndef __KERNEL__

+struct dentry;
+struct user_namespace;
+
+extern struct user_namespace init_user_ns;
+
+struct user_namespace *current_user_ns(void);
+
+extern const kernel_cap_t __cap_empty_set;
+extern const kernel_cap_t __cap_full_set;
+extern const kernel_cap_t __cap_init_eff_set;
+
/*
 * Internal kernel functions only
 */
@@ -530,10 +541,6 @@ static inline kernel_cap_t cap_raise_nfsd_set(const kernel_cap_t a,
    cap_intersect(permitted, __cap_nfsd_set));
}

-extern const kernel_cap_t __cap_empty_set;
-extern const kernel_cap_t __cap_full_set;
-extern const kernel_cap_t __cap_init_eff_set;
-
/***
 * has_capability - Determine if a task has a superior capability available
 * @t: The task in question
@@ -560,18 +567,25 @@ extern const kernel_cap_t __cap_init_eff_set;
 * Note that this does not set PF_SUPERPRIV on the task.
 */
#define has_capability_noaudit(t, cap) \
- (security_real_capable_noaudit((t), &init_user_ns, (cap)) == 0)
+ (security_real_capable_noaudit((t), &init_user_ns, (cap)) == 0)

-struct user_namespace;
-extern struct user_namespace init_user_ns;
-extern int capable(int cap);

```

```

-extern int ns_capable(struct user_namespace *ns, int cap);
-extern int task_ns_capable(struct task_struct *t, int cap);
+extern bool capable(int cap);
+extern bool ns_capable(struct user_namespace *ns, int cap);
+extern bool task_ns_capable(struct task_struct *t, int cap);

#define nsown_capable(cap) (ns_capable(current_user_ns(), (cap)))
+/*
+ * nsown_capable - Check superior capability to one's own user_ns
+ * @cap: The capability in question
+ *
+ * Return true if the current task has the given superior capability
+ * targeted at its own user namespace.
+ */
+static inline bool nsown_capable(int cap)
+{
+    return ns_capable(current_user_ns(), cap);
+}

/* audit system wants to get cap info from files as well */
-struct dentry;
extern int get_vfs_caps_from_disk(const struct dentry *dentry, struct cpu_vfs_cap_data
*cpu_caps);

#endif /* __KERNEL__ */
diff --git a/include/linux/cred.h b/include/linux/cred.h
index 4aaeab3..9aeeb0b 100644
--- a/include/linux/cred.h
+++ b/include/linux/cred.h
@@ -354,9 +354,11 @@ static inline void put_cred(const struct cred *_cred)
#define current_fsgid() (current_cred_xxx(fsgid))
#define current_cap() (current_cred_xxx(cap_effective))
#define current_user() (current_cred_xxx(user))
#define current_user_ns() (current_cred_xxx(user)->user_ns)
+#define _current_user_ns() (current_cred_xxx(user)->user_ns)
#define current_security() (current_cred_xxx(security))

+extern struct user_namespace *current_user_ns(void);
+
#define current_uid_gid(_uid, _gid) \
do { \
    const struct cred *__cred; \
diff --git a/kernel/capability.c b/kernel/capability.c
index 916658c..0a3d2c8 100644
--- a/kernel/capability.c
+++ b/kernel/capability.c
@@ -300,7 +300,7 @@ error:
 * This sets PF_SUPERPRIV on the task if the capability is available on the

```

```

 * assumption that it's about to be used.
 */
-int capable(int cap)
+bool capable(int cap)
{
    return ns_capable(&init_user_ns, cap);
}
@@ -317,7 +317,7 @@ EXPORT_SYMBOL(capable);
 * This sets PF_SUPERPRIV on the task if the capability is available on the
 * assumption that it's about to be used.
 */
-int ns_capable(struct user_namespace *ns, int cap)
+bool ns_capable(struct user_namespace *ns, int cap)
{
    if (unlikely(!cap_valid(cap))) {
        printk(KERN_CRIT "capable() called with invalid cap=%u\n", cap);
@@ -326,17 +326,21 @@ int ns_capable(struct user_namespace *ns, int cap)

    if (security_capable(ns, current_cred(), cap) == 0) {
        current->flags |= PF_SUPERPRIV;
-    return 1;
+    return true;
    }
-    return 0;
+    return false;
    }
EXPORT_SYMBOL(ns_capable);

/*
- * does current have capability 'cap' to the user namespace of task
- * 't'. Return true if it does, false otherwise.
+*/
+ * task_ns_capable - Determine whether current task has a superior
+ * capability targeted at a specific task's user namespace.
+ * @t: The task whose user namespace is targeted.
+ * @cap: The capability in question.
+ *
+ * Return true if it does, false otherwise.
*/
-int task_ns_capable(struct task_struct *t, int cap)
+bool task_ns_capable(struct task_struct *t, int cap)
{
    return ns_capable(task_cred_xxx(t, user)->user_ns, cap);
}

diff --git a/kernel/cred.c b/kernel/cred.c
index 3a9d6dd..e447fa2 100644
--- a/kernel/cred.c
+++ b/kernel/cred.c

```

```
@@ -741,6 +741,11 @@ int set_create_files_as(struct cred *new, struct inode *inode)
}
EXPORT_SYMBOL(set_create_files_as);

+struct user_namespace *current_user_ns(void)
+{
+ return _current_user_ns();
+}
+
#endif CONFIG_DEBUG_CREDENTIALS

bool creds_are_invalid(const struct cred *cred)
--
```

1.7.0.4

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: Re: [PATCH 1/4] userns: let clone\_uts\_ns() handle setting uts-&gt;user\_ns  
Posted by [Daniel Lezcano](#) on Thu, 24 Feb 2011 09:54:45 GMT

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

---

On 02/24/2011 01:21 AM, Serge E. Hallyn wrote:  
> To do so we need to pass in the task\_struct who'll get the utsname,  
> so we can get its user\_ns.  
>  
> Changelog:  
> Feb 23: As per Oleg's comment, just pass in tsk.  
>  
> Signed-off-by: Serge E. Hallyn<serge.hallyn@canonical.com>

Acked-by: Daniel Lezcano <daniel.lezcano@free.fr>

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: Re: [PATCH 2/4] userns: let copy\_ipcs handle setting ipc\_ns-&gt;user\_ns  
Posted by [Daniel Lezcano](#) on Thu, 24 Feb 2011 09:55:24 GMT

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

---

On 02/24/2011 01:22 AM, Serge E. Hallyn wrote:

> To do that, we have to pass in the task\_struct of the task which  
> will own the ipc\_ns, so we can assign its user\_ns.  
>  
> Changelog:  
> Feb 23: As per Oleg comment, just pass in tsk. To get the  
> ipc\_ns from the nsproxy we need to include nsproxy.h  
>  
> Signed-off-by: Serge E. Hallyn<serge.hallyn@canonical.com>  
> ---

Acked-by: Daniel Lezcano <daniel.lezcano@free.fr>

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---

---

Subject: Re: [PATCH 5/4] Clean up capability.h and capability.c

Posted by [Daniel Lezcano](#) on Thu, 24 Feb 2011 09:56:52 GMT

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

---

On 02/24/2011 01:22 AM, Serge E. Hallyn wrote:

> Convert macros to functions to let type safety do its thing. Switch  
> some functions from ints to more appropriate bool. Move all forward  
> declarations together to top of the #ifdef \_\_KERNEL\_\_ section. Use  
> kernel-doc format for comments.  
>  
> Some macros couldn't be converted because they use functions from  
> security.h which sometimes are extern and sometimes static inline,  
> and we don't want to #include security.h in capability.h.  
>  
> Also add a real current\_user\_ns function (and convert the existing  
> macro to \_current\_user\_ns() so we can use it in capability.h  
> without #including cred.h.  
>  
> Signed-off-by: Serge E. Hallyn<serge.hallyn@canonical.com>  
> ---

Acked-by: Daniel Lezcano <daniel.lezcano@free.fr>

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---