
Subject: Re: [PATCH 8/9] user namespaces: convert several capable() calls
Posted by [ebiederm](#) on Fri, 18 Feb 2011 01:51:25 GMT
[View Forum Message](#) <> [Reply to Message](#)

"Serge E. Hallyn" <serge@hallyn.com> writes:

> CAP_IPC_OWNER and CAP_IPC_LOCK can be checked against current_user_ns(),
> because the resource comes from current's own ipc namespace.
>
> setuid/setgid are to uids in own namespace, so again checks can be
> against current_user_ns().

Some nits below. But this generally looks good if a little bit all over
the map for a single patch.

Acked-by: "Eric W. Biederman" <ebiederm@xmission.com>

> Changelog:
> Jan 11: Use task_ns_capable() in place of sched_capable().
> Jan 11: Use nsown_capable() as suggested by Bastian Blank.
> Jan 11: Clarify (hopefully) some logic in futex and sched.c
> Feb 15: use ns_capable for ipc, not nsown_capable
>
> Signed-off-by: Serge E. Hallyn <serge.hallyn@canonical.com>
> ---
> ipc/shm.c | 2 +-
> ipc/util.c | 5 +++--
> kernel/futex.c | 11 ++++++++--
> kernel/futex_compat.c | 11 ++++++++--
> kernel/groups.c | 2 +-
> kernel/sched.c | 9 ++++++--
> kernel/uid16.c | 2 +-
> 7 files changed, 32 insertions(+), 10 deletions(-)
>
> diff --git a/ipc/shm.c b/ipc/shm.c
> index 7d3bb22..e91e2e9 100644
> --- a/ipc/shm.c
> +++ b/ipc/shm.c
> @@ -773,7 +773,7 @@ SYSCALL_DEFINE3(shmctl, int, shmid, int, cmd, struct shmid_ds
__user *, buf)
>
> audit_ipc_obj(&(shp->shm_perm));
>
> - if (!capable(CAP_IPC_LOCK)) {
> + if (!ns_capable(ns->user_ns, CAP_IPC_LOCK)) {
> uid_t euid = current_euid();
> err = -EPERM;

```

> if (euid != shp->shm_perm.uid &&
> diff --git a/ipc/util.c b/ipc/util.c
> index 69a0cc1..8e7ec6a 100644
> --- a/ipc/util.c
> +++ b/ipc/util.c
> @@ -627,7 +627,7 @@ int ipcperms (struct kern_ipc_perm *ipcp, short flag)
>     granted_mode >= 3;
>     /* is there some bit set in requested_mode but not in granted_mode? */
>     if ((requested_mode & ~granted_mode & 0007) &&
>         !capable(CAP_IPC_OWNER))
> +     !ns_capable(current->nsproxy->ipc_ns->user_ns, CAP_IPC_OWNER))
>     return -1;

```

Serge can we please modify the code to pass the ns down from ipcget_public to ipcperms. It is passed in and dropping the value and going back to current to get it just feels wrong.

Strictly speaking this code is correct but it requires an audit of all of the callers to know that, which is unfortunate.

```

> return security_ipc_permission(ipcp, flag);
> @@ -800,7 +800,8 @@ struct kern_ipc_perm *ipcctl_pre_down(struct ipc_ids *ids, int id, int
> cmd,
>
>     euid = current_euid();
>     if (euid == ipcp->cuid ||
>         euid == ipcp->uid || capable(CAP_SYS_ADMIN))
> +     euid == ipcp->uid ||
> +     ns_capable(current->nsproxy->ipc_ns->user_ns, CAP_SYS_ADMIN))
>     return ipcp;

```

Like the other ipc call can we please pass the ipc_ns into ipcctl_pre_down.

The code as constructed appears correct but because we are passing the namespace into the caller of this function always using current to get the ipc_ns seems confusing and unnecessary.

```

> err = -EPERM;
> diff --git a/kernel/futex.c b/kernel/futex.c
> index b766d28..1e876f1 100644
> --- a/kernel/futex.c
> +++ b/kernel/futex.c
> @@ -2421,10 +2421,19 @@ SYSCALL_DEFINE3(get_robust_list, int, pid,
>     goto err_unlock;
>     ret = -EPERM;
>     pcred = __task_cred(p);
> +     /* If victim is in different user_ns, then uids are not
> +     comparable, so we must have CAP_SYS_PTRACE */

```

```

> + if (cred->user->user_ns != pcred->user->user_ns) {
> +   if (!ns_capable(pcred->user->user_ns, CAP_SYS_PTRACE))
> +     goto err_unlock;
> +   goto ok;
> + }
> + /* If victim is in same user_ns, then uids are comparable */
>   if (cred->euid != pcred->euid &&
>       cred->euid != pcred->uid &&
> -   !capable(CAP_SYS_PTRACE))
> +   !ns_capable(pcred->user->user_ns, CAP_SYS_PTRACE))
>   goto err_unlock;
> +ok:
>   head = p->robust_list;
>   rcu_read_unlock();
> }
> diff --git a/kernel/futex_compat.c b/kernel/futex_compat.c
> index a7934ac..5f9e689 100644
> --- a/kernel/futex_compat.c
> +++ b/kernel/futex_compat.c
> @@ -153,10 +153,19 @@ compat_sys_get_robust_list(int pid, compat_uptr_t __user
*head_ptr,
>   goto err_unlock;
>   ret = -EPERM;
>   pcred = __task_cred(p);
> + /* If victim is in different user_ns, then uids are not
> +   comparable, so we must have CAP_SYS_PTRACE */
> + if (cred->user->user_ns != pcred->user->user_ns) {
> +   if (!ns_capable(pcred->user->user_ns, CAP_SYS_PTRACE))
> +     goto err_unlock;
> +   goto ok;
> + }
> + /* If victim is in same user_ns, then uids are comparable */
>   if (cred->euid != pcred->euid &&
>       cred->euid != pcred->uid &&
> -   !capable(CAP_SYS_PTRACE))
> +   !ns_capable(pcred->user->user_ns, CAP_SYS_PTRACE))
>   goto err_unlock;
> +ok:
>   head = p->compat_robust_list;
>   rcu_read_unlock();
> }
> diff --git a/kernel/groups.c b/kernel/groups.c
> index 253dc0f..1cc476d 100644
> --- a/kernel/groups.c
> +++ b/kernel/groups.c
> @@ -233,7 +233,7 @@ SYSCALL_DEFINE2(setgroups, int, gidsetsize, gid_t __user *,
grouplist)
>   struct group_info *group_info;

```

```

> int retval;
>
> - if (!capable(CAP_SETGID))
> + if (!nsown_capable(CAP_SETGID))
> return -EPERM;
> if ((unsigned)gidsetsize > NGROUPS_MAX)
> return -EINVAL;
> diff --git a/kernel/sched.c b/kernel/sched.c
> index 18d38e4..dc12bc2 100644
> --- a/kernel/sched.c
> +++ b/kernel/sched.c
> @@ -4761,8 +4761,11 @@ static bool check_same_owner(struct task_struct *p)
>
> rcu_read_lock();
> pcred = __task_cred(p);
> - match = (cred->euid == pcred->euid ||
> - cred->euid == pcred->uid);
> + if (cred->user->user_ns == pcred->user->user_ns)
> + match = (cred->euid == pcred->euid ||
> + cred->euid == pcred->uid);
> + else
> + match = false;
> rcu_read_unlock();
> return match;
> }
> @@ -5088,7 +5091,7 @@ long sched_setaffinity(pid_t pid, const struct cpumask *in_mask)
> goto out_free_cpus_allowed;
> }
> retval = -EPERM;
> - if (!check_same_owner(p) && !capable(CAP_SYS_NICE))
> + if (!check_same_owner(p) && !task_ns_capable(p, CAP_SYS_NICE))
> goto out_unlock;
>
> retval = security_task_setscheduler(p);
> diff --git a/kernel/uid16.c b/kernel/uid16.c
> index 4192098..51c6e89 100644
> --- a/kernel/uid16.c
> +++ b/kernel/uid16.c
> @@ -189,7 +189,7 @@ SYSCALL_DEFINE2(setgroups16, int, gidsetsize, old_gid_t __user *,
grouplist)
> struct group_info *group_info;
> int retval;
>
> - if (!capable(CAP_SETGID))
> + if (!nsown_capable(CAP_SETGID))
> return -EPERM;
> if ((unsigned)gidsetsize > NGROUPS_MAX)
> return -EINVAL;

```

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