Subject: Re: [ckrm-tech] [RFC][PATCH 4/7] UBC: syscalls (user interface)
Posted by dev on Fri, 18 Aug 2006 11:43:38 GMT
View Forum Message <> Reply to Message

Matt Helsley wrote:

[... snip ...]
>>--- ./kernel/ub/sys.c.ubsys 2006-07-28 18:52:18.000000000 +0400
>>+++ ./kernel/ub/sys.c 2006-08-03 16:14:23.000000000 +0400
>>@@ -0,0 +1,126 @@
>>+/*
>>+ * kernel/ub/sys.c
>>+ *
>>+ * Copyright (C) 2006 OpenVZ. SWsoft Inc
>>+ *
>>+ */
>>+
>>+#include <linux/config.h>
>>+#include <linux/sched.h>
>>+#include <asm/uaccess.h>
>>+
>>+#include <ub/beancounter.h>
>>+#include <ub/task.h>
>>+
>>+#ifndef CONFIG_USER_RESOURCE
>
>
> Get rid of the #ifdef since this file should only be compiled if
> CONFIG_USER_RESOURCE=y anyway.
>
>
>>+asmlinkage long sys_getluid(void)
>>+{
>>+ return -ENOSYS;
>>+}
>>+
>>+asmlinkage long sys_setluid(uid_t uid)
>>+{
>>+ return -ENOSYS;
>>+}
>>+
>>+asmlinkage long sys_setublimit(uid_t uid, unsigned long resource,
>>+ unsigned long *limits)
>>+{
>>+ return -ENOSYS;
>>+}
>
>

> Looks to me like you want to add:
>
> cond_syscall(sys_getluid);
> ...
>
> in kernel/sys_ni.c and then you won't have to worry about making these
> empty functions.
Good note. Thanks, will do it!

>>+#else /* CONFIG_USER_RESOURCE */
>>+
>>+/*
>>+ * The (rather boring) getluid syscall
>>+ */
>>+asmlinkage long sys_getluid(void)
>>+{
>>+ struct user_beancounter *ub;
>>+
>>+ ub = get_exec_ub();
>>+ if (ub == NULL)
>>+ return -EINVAL;
>>+
>>+ return ub->ub_uid;
>>+}
>>+
>>+/*
>>+ * The setluid syscall
>>+ */
>>+asmlinkage long sys_setluid(uid_t uid)
>>+{
>>+ int error;
>>+ struct user_beancounter *ub;
>>+ struct task_beancounter *task_bc;
>>+
>>+ task_bc = &current->task_bc;
>>+
>>+ /* You may not disown a setluid */
>>+ error = -EINVAL;
>>+ if (uid == (uid_t)-1)
>>+ goto out;
>>+
>>+ /* You may only set an ub as root */
>>+ error = -EPERM;
>>+ if (!capable(CAP_SETUID))
>>+ goto out;
>
>
> With resource groups you don't necessarily have to be root -- just the

> owner of the group and task.
the question is - who is the owner of group?
user, user group or who?
Both are bad, since the same user can run inside the container and thus
container will be potentially controllable/breakable from inside.

> Filesystems and appropriate share representations offer a way to give
> regular users the ability to manage their resources without requiring
> CAP_FOO.
not sure what you propose...

we can introduce the following rules:

containers (UB) can be created by process with SETUID cap only.
subcontainers (SUB) can be created by any process.

what do you think?

```
>>+ /* Ok - set up a beancounter entry for this user */
>>+ error = -ENOBUFS;
>>+ ub = beancounter_findcreate(uid, NULL, UB_ALLOC);
>>+ if (ub == NULL)
>>+  goto out;
>>+
>>+ /* install bc */
>>+ put_beancounter(task_bc->exec_ub);
>>+ task_bc->exec_ub = ub;
>>+ put_beancounter(task_bc->fork_sub);
>>+ task_bc->fork_sub = get_beancounter(ub);
>>+ error = 0;
>>+out:
>>+ return error;
>>+}
>>+
>>+/*
>>+ * The setbeanlimit syscall
>>+ */
>>+asmlinkage long sys_setublimit(uid_t uid, unsigned long resource,
>>+  unsigned long *limits)
>>+{
>>+ int error;
>>+ unsigned long flags;
>>+ struct user_beancounter *ub;
>>+ unsigned long new_limits[2];
>>+
>>+ error = -EPERM;
```

>>+ if(!capable(CAP_SYS_RESOURCE))
>>+ goto out;
>
>
> Again, a filesystem interface would give us more flexibility when it
> comes to allowing users to manage their resources while still preventing
> them from exceeding limits.
we can have 2 different root users with uid = 0 in 2 different containers.

> I doubt you really want to give owners of a container CAP_SYS_RESOURCE
> and CAP_USER (i.e. total control over resource management) just to allow
> them to manage their subset of the resources.
The origin idea is that administator of the node can manage user
resources only. Users can't, since otherwise they can increase the limits.
But we can allow them to manage sub beancoutners imho...

>>+ error = -EINVAL;
>>+ if (resource >= UB_RESOURCES)
>>+ goto out;
>>+
>>+ error = -EFAULT;
>>+ if (copy_from_user(&new_limits, limits, sizeof(new_limits)))
>>+ goto out;
>>+
>>+ error = -EINVAL;
>>+ if (new_limits[0] > UB_MAXVALUE || new_limits[1] > UB_MAXVALUE)
>>+ goto out;
>>+
>>+ error = -ENOENT;
>>+ ub = beancounter_findcreate(uid, NULL, 0);
>>+ if (ub == NULL)
>>+ goto out;
>>+
>>+ spin_lock_irqsave(&ub->ub_lock, flags);
>>+ ub->ub_parms[resource].barrier = new_limits[0];
>>+ ub->ub_parms[resource].limit = new_limits[1];
>>+ spin_unlock_irqrestore(&ub->ub_lock, flags);
>>+
>>+ put_beancounter(ub);
>>+ error = 0;
>>+out:
>>+ return error;
>>+}
>>+#endif
>>
>> ---------- ----------
>>Using Tomcat but need to do more? Need to support web services, security?
>>Get stuff done quickly with pre-integrated technology to make your job easier

>>Download IBM WebSphere Application Server v.1.0.1 based on Apache Geronimo
>> http://sel.as-us.falkag.net/sel?cmd=lnk&kid=120709&b id=263057&dat=121642
>>_____
>>ckrm-tech mailing list
>>https://lists.sourceforge.net/lists/listinfo/ckrm-tech
>
>
>