Subject: Re: [ckrm-tech] [RFC][PATCH 4/7] UBC: syscalls (user interface)
Posted by Matt Helsley on Fri, 18 Aug 2006 02:31:40 GMT
View Forum Message <> Reply to Message

On Wed, 2006-08-16 at 19:39 +0400, Kirill Korotaev wrote:
> Add the following system calls for UB management:
> 1. sys_getluid - get current UB id
> 2. sys_setluid - changes exec_ and fork_ UBs on current
> 3. sys_setublimit - set limits for resources consumtions
>
> Signed-Off-By: Pavel Emelianov <xemul@sw.ru>
> Signed-Off-By: Kirill Korotaev <dev@sw.ru>
>
> ---
> arch/i386/kernel/syscall_table.S | 3
> arch/ia64/kernel/entry.S | 3
> arch/sparc/kernel/systbls.S | 2
> arch/sparc64/kernel/systbls.S | 2
> include/asm-i386/unistd.h | 5 +
> include/asm-ia64/unistd.h | 5 +
> include/asm-powerpc/systbl.h | 3
> include/asm-powerpc/unistd.h | 5 +
> include/asm-sparc/unistd.h | 3
> include/asm-sparc64/unistd.h | 3
> include/asm-x86_64/unistd.h | 8 ++
> kernel/ub/Makefile | 1
> kernel/ub/sys.c | 126 ++
> 13 files changed, 163 insertions(+), 6 deletions(-)
>
> --- ./arch/i386/kernel/syscall_table.S.ubsys 2006-07-10 12:39:10.000000000 +0400
> +++ ./arch/i386/kernel/syscall_table.S 2006-07-31 14:36:59.000000000 +0400
> @@ -317,3 +317,6 @@ ENTRY(sys_call_table)
> .long sys_vmsplice
> .long sys_move_pages
> .long sys_getcpu
> + .long sys_getluid
> + .long sys_setluid
> + .long sys_setublimit /* 320 */
> --- ./arch/ia64/kernel/entry.S.ubsys 2006-07-10 12:39:10.000000000 +0400
> +++ ./arch/ia64/kernel/entry.S 2006-07-31 15:25:36.000000000 +0400
> @@ -1610,5 +1610,8 @@ sys_call_table:
> data8 sys_sync_file_range // 1300
> data8 sys_tee
> data8 sys_vmsplice
> + daat8 sys_getluid
> + data8 sys_setluid
> + data8 sys_setublimit // 1305
>

> .org sys_call_table + 8*NR_syscalls // guard against failures to increase NR_syscalls
> --- ./arch/sparc/kernel/systbls.S.arsys 2006-07-10 12:39:10.000000000 +0400
> +++ ./arch/sparc/kernel/systbls.S 2006-08-10 17:07:15.000000000 +0400
> @@ -78,7 +78,7 @@ sys_call_table:
> /*285*/ .long sys_mkdirat, sys_mknodat, sys_fchownat, sys_futimesat, sys_fstatat64
> /*290*/ .long sys_unlinkat, sys_renameat, sys_linkat, sys_symlinkat, sys_readlinkat
> /*295*/ .long sys_fchmodat, sys_faccessat, sys_pselect6, sys_ppoll, sys_unshare
> -/*300*/ .long sys_set_robust_list, sys_get_robust_list
> +/*300*/ .long sys_set_robust_list, sys_get_robust_list, sys_getluid, sys_setluid, sys_setublimit
>
> #ifdef CONFIG_SUNOS_EMUL
> /* Now the SunOS syscall table. */
> --- ./arch/sparc64/kernel/systbls.S.arsys 2006-07-10 12:39:11.000000000 +0400
> +++ ./arch/sparc64/kernel/systbls.S 2006-08-10 17:08:52.000000000 +0400
> @@ -79,7 +79,7 @@ sys_call_table32:
> .word sys_mkdirat, sys_mknodat, sys_fchownat, compat_sys_futimesat, compat_sys_fstatat64
> /*290*/ .word sys_unlinkat, sys_renameat, sys_linkat, sys_symlinkat, sys_readlinkat
> .word sys_fchmodat, sys_faccessat, compat_sys_pselect6, compat_sys_ppoll, sys_unshare
> -/*300*/ .word compat_sys_set_robust_list, compat_sys_get_robust_list
> +/*300*/ .word compat_sys_set_robust_list, compat_sys_get_robust_list, sys_getluid,
sys_setluid, sys_setublimit
>
> #endif /* CONFIG_COMPAT */
>
> --- ./include/asm-i386/unistd.h.ubsys 2006-07-10 12:39:19.000000000 +0400
> +++ ./include/asm-i386/unistd.h 2006-07-31 15:56:31.000000000 +0400
> @@ -323,10 +323,13 @@
> #define __NR_vmsplice 316
> #define __NR_move_pages 317
> #define __NR_getcpu 318
> +#define __NR_getluid 319
> +#define __NR_setluid 320
> +#define __NR_setublimit 321
>
> #ifdef __KERNEL__
>
> -#define NR_syscalls 318
> +#define NR_syscalls 322
> #include <linux/err.h>
>
> /*
> --- ./include/asm-ia64/unistd.h.ubsys 2006-07-10 12:39:19.000000000 +0400
> +++ ./include/asm-ia64/unistd.h 2006-07-31 15:57:23.000000000 +0400
> @@ -291,11 +291,14 @@
> #define __NR_sync_file_range 1300
> #define __NR_tee 1301
> #define __NR_vmsplice 1302
> +#define __NR_getluid 1303

> +#define __NR_setluid   1304
> +#define __NR_setublimit   1305
>
> #ifdef __KERNEL__
>
>
> -#define NR_syscalls   279 /* length of syscall table */
> +#define NR_syscalls   282 /* length of syscall table */
>
> #define __ARCH_WANT_SYS_RT_SIGACTION
>
> --- ./include/asm-powerpc/systbl.h.arsys 2006-07-10 12:39:19.000000000 +0400
> +++ ./include/asm-powerpc/systbl.h 2006-08-10 17:05:53.000000000 +0400
> @@ -304,3 +304,6 @@ SYSCALL_SPU(fchmodat)
> SYSCALL_SPU(faccessat)
> COMPAT_SYS_SPU(get_robust_list)
> COMPAT_SYS_SPU(set_robust_list)
> +SYSCALL(sys_getluid)
> +SYSCALL(sys_setluid)
> +SYSCALL(sys_setublimit)
> --- ./include/asm-powerpc/unistd.h.arsys 2006-07-10 12:39:19.000000000 +0400
> +++ ./include/asm-powerpc/unistd.h 2006-08-10 17:06:28.000000000 +0400
> @@ -323,10 +323,13 @@
> #define __NR_faccessat 298
> #define __NR_get_robust_list 299
> #define __NR_set_robust_list 300
> +#define __NR_getluid 301
> +#define __NR_setluid 302
> +#define __NR_setublimit 303
>
> #ifdef __KERNEL__
>
> -#define __NR_syscalls 301
> +#define __NR_syscalls 304
>
> #define __NR__exit __NR_exit
> #define NR_syscalls __NR_syscalls
> --- ./include/asm-sparc/unistd.h.arsys 2006-07-10 12:39:19.000000000 +0400
> +++ ./include/asm-sparc/unistd.h 2006-08-10 17:08:19.000000000 +0400
> @@ -318,6 +318,9 @@
> #define __NR_unshare 299
> #define __NR_set_robust_list 300
> #define __NR_get_robust_list 301
> +#define __NR_getluid 302
> +#define __NR_setluid 303
> +#define __NR_setublimit 304
>
> #ifdef __KERNEL__

> /* WARNING: You MAY NOT add syscall numbers larger than 301, since
> --- ./include/asm-sparc64/unistd.h.arsys 2006-07-10 12:39:19.000000000 +0400
> +++ ./include/asm-sparc64/unistd.h 2006-08-10 17:09:24.000000000 +0400
> @@ -320,6 +320,9 @@
> #define __NR_unshare 299
> #define __NR_set_robust_list 300
> #define __NR_get_robust_list 301
> +#define __NR_getluid 302
> +#define __NR_setluid 303
> +#define __NR_setublimit 304
>
> #ifdef __KERNEL__
> /* WARNING: You MAY NOT add syscall numbers larger than 301, since
> --- ./include/asm-x86_64/unistd.h.ubsys 2006-07-10 12:39:19.000000000 +0400
> +++ ./include/asm-x86_64/unistd.h 2006-07-31 16:00:01.000000000 +0400
> @@ -619,10 +619,16 @@ __SYSCALL(__NR_sync_file_range, sys_sync
> __SYSCALL(__NR_vmsplice, sys_vmsplice)
> #define __NR_move_pages 279
> __SYSCALL(__NR_move_pages, sys_move_pages)
> +#define __NR_getluid 280
> +__SYSCALL(__NR_getluid, sys_getluid)
> +#define __NR_setluid 281
> +__SYSCALL(__NR_setluid, sys_setluid)
> +#define __NR_setublimit 282
> +__SYSCALL(__NR_setublimit, sys_setublimit)
>
> #ifdef __KERNEL__
>
> -#define __NR_syscall_max __NR_move_pages
> +#define __NR_syscall_max __NR_setublimit
> #include <linux/err.h>
>
> #ifndef __NO_STUBS
> --- ./kernel/ub/Makefile.ubsys 2006-07-28 14:08:37.000000000 +0400
> +++ ./kernel/ub/Makefile 2006-08-01 11:08:39.000000000 +0400
> @@ -6,3 +6,4 @@
>
> obj-$(CONFIG_USER_RESOURCE) += beancounter.o
> obj-$(CONFIG_USER_RESOURCE) += misc.o
> +obj-y += sys.o
> --- ./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.

> +#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.

Filesystems and appropriate share representations offer a way to give regular users the ability to manage their resources without requiring CAP_FOO.

```
> + /* 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.

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.

> + 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