Subject: [PATCH 1/1] capabilities: introduce per-process capability bounding set (v8)
Posted by serue on Mon, 19 Nov 2007 21:25:19 GMT
View Forum Message <> Reply to Message

Andrew, this version follows all of your suggestions. Definately nicer
userspace interface. thanks

-serge

>From b7c210160e3c210d63eca532289ca1c9caf1bd87 Mon Sep 17 00:00:00 2001
From: Serge E. Hallyn <serue@us.ibm.com>
Date: Mon, 19 Nov 2007 13:54:05 -0500
Subject: [PATCH 1/1] capabilities: introduce per-process capability bounding set (v8)

The capability bounding set is a set beyond which capabilities
cannot grow. Currently cap_bset is per-system. It can be
manipulated through sysctl, but only init can add capabilities.
Root can remove capabilities. By default it includes all caps
except CAP_SETPCAP.

This patch makes the bounding set per-process when file
capabilities are enabled. It is inherited at fork from parent.
Noone can add elements, CAP_SETPCAP is required to remove them.

One example use of this is to start a safer container. For
instance, until device namespaces or per-container device
whitelists are introduced, it is best to take CAP_MKNOD away
from a container.

The following test program will get and set the bounding
set. For instance

./bset get
(lists capabilities in bset)
./bset drop cap_net_raw
(starts shell with new bset)
(use capset, setuid binary, or binary with
file capabilities to try to increase caps)

```
************************************************************
cap_bound.c
************************************************************
#include <sys/prctl.h>
#include <linux/capability.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
```

```
 #include <stdlib.h>
 #include <string.h>

 #ifndef PR_CAPBSET_READ
 #define PR_CAPBSET_READ 23
 #endif

 #ifndef PR_CAPBSET_DROP
 #define PR_CAPBSET_DROP 24
 #endif

int usage(char *me)
{
 printf("Usage: %s get\n", me);
 printf("       %s drop <capability>\n", me);
 return 1;
}

 #define numcaps 32
char *captable[numcaps] = {
 "cap_chown",
 "cap_dac_override",
 "cap_dac_read_search",
 "cap_fowner",
 "cap_fsetid",
 "cap_kill",
 "cap_setgid",
 "cap_setuid",
 "cap_setpcap",
 "cap_linux_immutable",
 "cap_net_bind_service",
 "cap_net_broadcast",
 "cap_net_admin",
 "cap_net_raw",
 "cap_ipc_lock",
 "cap_ipc_owner",
 "cap_sys_module",
 "cap_sys_rawio",
 "cap_sys_chroot",
 "cap_sys_ptrace",
 "cap_sys_pacct",
 "cap_sys_admin",
 "cap_sys_boot",
 "cap_sys_nice",
 "cap_sys_resource",
 "cap_sys_time",
 "cap_sys_tty_config",
 "cap_mknod",
```

```
 "cap_lease",
 "cap_audit_write",
 "cap_audit_control",
 "cap_setfcap"
};

int getbcap(void)
{
 int comma=0;
 unsigned long i;
 int ret;

 printf("i know of %d capabilities\n", numcaps);
 printf("capability bounding set:");
 for (i=0; i<numcaps; i++) {
  ret = prctl(PR_CAPBSET_READ, i);
  if (ret < 0)
   perror("prctl");
  else if (ret==1)
   printf("%s%s", (comma++) ? ", " : " ", captable[i]);
 }
 printf("\n");
 return 0;
}

int capdrop(char *str)
{
 unsigned long i;

 int found=0;
 for (i=0; i<numcaps; i++) {
  if (strcmp(captable[i], str) == 0) {
   found=1;
   break;
  }
 }
 if (!found)
  return 1;
 if (prctl(PR_CAPBSET_DROP, i)) {
  perror("prctl");
  return 1;
 }
 return 0;
}

int main(int argc, char *argv[])
{
 if (argc<2)
```

```
  return usage(argv[0]);
 if (strcmp(argv[1], "get")==0)
  return getbcap();
 if (strcmp(argv[1], "drop")!=0 || argc<3)
  return usage(argv[0]);
 if (capdrop(argv[2])) {
  printf("unknown capability\n");
  return 1;
 }
 return execl("/bin/bash", "/bin/bash", NULL);
}
************************************************************

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>
---
 include/linux/capability.h |   25 +++++++++++++++++++++++-
 include/linux/init_task.h  |    1 +
 include/linux/prctl.h      |    4 ++++
 include/linux/sched.h      |    2 +-
 include/linux/security.h   |    5 -----
 include/linux/sysctl.h     |    3 ---
 kernel/fork.c              |    1 +
 kernel/sys.c               |    7 +++++++
 kernel/sysctl.c            |   35 -----------------------------------
 kernel/sysctl_check.c      |    7 -------
 security/commoncap.c       |   29 +++++++++++++++++++++++++++++----
 11 files changed, 63 insertions(+), 56 deletions(-)

diff --git a/include/linux/capability.h b/include/linux/capability.h
index a1d93da..94638b2 100644
--- a/include/linux/capability.h
+++ b/include/linux/capability.h
@@ -153,6 +153,7 @@ typedef struct kernel_cap_struct {
  *   remove any capability in your permitted set from any pid
  * With VFS support for capabilities (neither of above, but)
  *   Add any capability to the current process' inheritable set
+ *   Allow taking bits out of capability bounding set
  */

 #define CAP_SETPCAP          8
@@ -202,7 +203,6 @@ typedef struct kernel_cap_struct {
 #define CAP_IPC_OWNER        15

 /* Insert and remove kernel modules - modify kernel without limit */
-/* Modify cap_bset */
 #define CAP_SYS_MODULE       16

 /* Allow ioperm/iopl access */
```

```
@@ -314,6 +314,10 @@ typedef struct kernel_cap_struct {

 #define CAP_SETFCAP      31

+#define CAP_NUM_CAPS         32
+
+#define cap_valid(x) ((x) >= 0 && (x) < CAP_NUM_CAPS)
+
 /*
  * Bit location of each capability (used by user-space library and kernel)
  */
@@ -350,6 +354,17 @@ typedef struct kernel_cap_struct {

 #define CAP_INIT_INH_SET    CAP_EMPTY_SET

+#ifdef CONFIG_SECURITY_FILE_CAPABILITIES
+/*
+ * Because of the reduced scope of CAP_SETPCAP when filesystem
+ * capabilities are in effect, it is safe to allow this capability to
+ * be available in the default configuration.
+ */
+# define CAP_INIT_BSET  CAP_FULL_SET
+#else
+# define CAP_INIT_BSET  CAP_INIT_EFF_SET
+#endif
+
 # define cap_clear(c)         do { (c) = __cap_empty_set; } while (0)
 # define cap_set_full(c)      do { (c) = __cap_full_set; } while (0)
 # define cap_set_init_eff(c)  do { (c) = __cap_init_eff_set; } while (0)
@@ -465,6 +480,14 @@ extern const kernel_cap_t __cap_init_eff_set;
 int capable(int cap);
 int __capable(struct task_struct *t, int cap);

+#ifdef CONFIG_SECURITY_FILE_CAPABILITIES
+extern long cap_prctl_drop(unsigned long cap);
+#else
+#include <linux/errno.h>
+static inline long cap_prctl_drop(unsigned long cap)
+{ return -EINVAL; }
+#endif
+
 #endif /* __KERNEL__ */

 #endif /* !_LINUX_CAPABILITY_H */
diff --git a/include/linux/init_task.h b/include/linux/init_task.h
index cae35b6..5c84d14 100644
--- a/include/linux/init_task.h
+++ b/include/linux/init_task.h
```

```
@@ -147,6 +147,7 @@ extern struct group_info init_groups;
  .cap_effective = CAP_INIT_EFF_SET,    \
  .cap_inheritable = CAP_INIT_INH_SET,    \
  .cap_permitted = CAP_FULL_SET,     \
+ .cap_bset  = CAP_INIT_BSET,    \
  .keep_capabilities = 0,      \
  .user  = INIT_USER,     \
  .comm  = "swapper",     \
diff --git a/include/linux/prctl.h b/include/linux/prctl.h
index e2eff90..3800639 100644
--- a/include/linux/prctl.h
+++ b/include/linux/prctl.h
@@ -63,4 +63,8 @@
 #define PR_GET_SECCOMP 21
 #define PR_SET_SECCOMP 22

+/* Get/set the capability bounding set */
+#define PR_CAPBSET_READ 23
+#define PR_CAPBSET_DROP 24
+
 #endif /* _LINUX_PRCTL_H */
diff --git a/include/linux/sched.h b/include/linux/sched.h
index 1d17f7c..bf51a16 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1041,7 +1041,7 @@ struct task_struct {
  uid_t uid,euid,suid,fsuid;
  gid_t gid,egid,sgid,fsgid;
  struct group_info *group_info;
- kernel_cap_t   cap_effective, cap_inheritable, cap_permitted;
+ kernel_cap_t   cap_effective, cap_inheritable, cap_permitted, cap_bset;
  unsigned keep_capabilities:1;
  struct user_struct *user;
 #ifdef CONFIG_KEYS
diff --git a/include/linux/security.h b/include/linux/security.h
index f771ad8..04b18f1 100644
--- a/include/linux/security.h
+++ b/include/linux/security.h
@@ -34,11 +34,6 @@
 #include <linux/xfrm.h>
 #include <net/flow.h>

-/*
- * Bounding set
- */
-extern kernel_cap_t cap_bset;
-
 extern unsigned securebits;
```

```
 struct ctl_table;
diff --git a/include/linux/sysctl.h b/include/linux/sysctl.h
index 4f5047d..fa900cb 100644
--- a/include/linux/sysctl.h
+++ b/include/linux/sysctl.h
@@ -102,7 +102,6 @@ enum
  KERN_NODENAME=7,
  KERN_DOMAINNAME=8,

- KERN_CAP_BSET=14, /* int: capability bounding set */
  KERN_PANIC=15,  /* int: panic timeout */
  KERN_REALROOTDEV=16, /* real root device to mount after initrd */

@@ -962,8 +961,6 @@ extern int proc_dostring(struct ctl_table *, int, struct file *,
     void __user *, size_t *, loff_t *);
 extern int proc_dointvec(struct ctl_table *, int, struct file *,
     void __user *, size_t *, loff_t *);
-extern int proc_dointvec_bset(struct ctl_table *, int, struct file *,
-         void __user *, size_t *, loff_t *);
 extern int proc_dointvec_minmax(struct ctl_table *, int, struct file *,
     void __user *, size_t *, loff_t *);
 extern int proc_dointvec_jiffies(struct ctl_table *, int, struct file *,
diff --git a/kernel/fork.c b/kernel/fork.c
index 5639b3e..9e4a5e1 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1087,6 +1087,7 @@ static struct task_struct *copy_process(unsigned long clone_flags,
 #ifdef CONFIG_SECURITY
  p->security = NULL;
 #endif
+ p->cap_bset = current->cap_bset;
  p->io_context = NULL;
  p->audit_context = NULL;
  cgroup_fork(p);
diff --git a/kernel/sys.c b/kernel/sys.c
index 4c77ed2..bed55dc 100644
--- a/kernel/sys.c
+++ b/kernel/sys.c
@@ -1742,6 +1742,13 @@ asmlinkage long sys_prctl(int option, unsigned long arg2, unsigned
long arg3,
    error = prctl_set_seccomp(arg2);
    break;

+  case PR_CAPBSET_READ:
+   if (!cap_valid(arg2))
+    return -EINVAL;
+   return !!cap_raised(current->cap_bset, arg2);
```

```
+  case PR_CAPBSET_DROP:
+   return cap_prctl_drop(arg2);
+
    default:
     error = -EINVAL;
     break;
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index 489b0d1..d858819 100644
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -383,15 +383,6 @@ static struct ctl_table kern_table[] = {
    .proc_handler = &proc_dointvec_taint,
   },
  #endif
-#ifdef CONFIG_SECURITY_CAPABILITIES
- {
-  .procname = "cap-bound",
-  .data  = &cap_bset,
-  .maxlen  = sizeof(kernel_cap_t),
-  .mode  = 0600,
-  .proc_handler = &proc_dointvec_bset,
- },
-#endif /* def CONFIG_SECURITY_CAPABILITIES */
  #ifdef CONFIG_BLK_DEV_INITRD
   {
    .ctl_name = KERN_REALROOTDEV,
@@ -1910,26 +1901,6 @@ static int do_proc_dointvec_bset_conv(int *negp, unsigned long
*lvalp,
   return 0;
  }

-#ifdef CONFIG_SECURITY_CAPABILITIES
-/*
- * init may raise the set.
- */
-
-int proc_dointvec_bset(struct ctl_table *table, int write, struct file *filp,
-    void __user *buffer, size_t *lenp, loff_t *ppos)
-{
- int op;
-
- if (write && !capable(CAP_SYS_MODULE)) {
-  return -EPERM;
- }
-
- op = is_global_init(current) ? OP_SET : OP_AND;
- return do_proc_dointvec(table,write,filp,buffer,lenp,ppos,
-     do_proc_dointvec_bset_conv,&op);
```

```
-}
-#endif /* def CONFIG_SECURITY_CAPABILITIES */
-
 /*
  * Taint values can only be increased
  */
@@ -2343,12 +2314,6 @@ int proc_dointvec(struct ctl_table *table, int write, struct file *filp,
  return -ENOSYS;
 }

-int proc_dointvec_bset(struct ctl_table *table, int write, struct file *filp,
-   void __user *buffer, size_t *lenp, loff_t *ppos)
-{
- return -ENOSYS;
-}
-
 int proc_dointvec_minmax(struct ctl_table *table, int write, struct file *filp,
       void __user *buffer, size_t *lenp, loff_t *ppos)
 {
diff --git a/kernel/sysctl_check.c b/kernel/sysctl_check.c
index 8f5baac..526fa36 100644
--- a/kernel/sysctl_check.c
+++ b/kernel/sysctl_check.c
@@ -38,10 +38,6 @@ static struct trans_ctl_table trans_kern_table[] = {
  { KERN_NODENAME,  "hostname" },
  { KERN_DOMAINNAME,  "domainname" },

-#ifdef CONFIG_SECURITY_CAPABILITIES
- { KERN_CAP_BSET,  "cap-bound" },
-#endif /* def CONFIG_SECURITY_CAPABILITIES */
-
  { KERN_PANIC,   "panic" },
  { KERN_REALROOTDEV,  "real-root-dev" },

@@ -1522,9 +1518,6 @@ int sysctl_check_table(struct ctl_table *table)
       (table->strategy == sysctl_ms_jiffies) ||
       (table->proc_handler == proc_dostring) ||
       (table->proc_handler == proc_dointvec) ||
-#ifdef CONFIG_SECURITY_CAPABILITIES
-      (table->proc_handler == proc_dointvec_bset) ||
-#endif /* def CONFIG_SECURITY_CAPABILITIES */
       (table->proc_handler == proc_dointvec_minmax) ||
       (table->proc_handler == proc_dointvec_jiffies) ||
       (table->proc_handler == proc_dointvec_userhz_jiffies) ||
diff --git a/security/commoncap.c b/security/commoncap.c
index 3a95990..b6745f4 100644
--- a/security/commoncap.c
+++ b/security/commoncap.c
```

```
@@ -36,9 +36,6 @@
 # define CAP_INIT_BSET  CAP_INIT_EFF_SET
 #endif /* def CONFIG_SECURITY_FILE_CAPABILITIES */

-kernel_cap_t cap_bset = CAP_INIT_BSET;    /* systemwide capability bound */
-EXPORT_SYMBOL(cap_bset);
-
 /* Global security state */

 unsigned securebits = SECUREBITS_DEFAULT; /* systemwide security settings */
@@ -330,7 +327,8 @@ void cap_bprm_apply_creds (struct linux_binprm *bprm, int unsafe)
  /* Derived from fs/exec.c:compute_creds. */
  kernel_cap_t new_permitted, working;

- new_permitted = cap_intersect (bprm->cap_permitted, cap_bset);
+ new_permitted = cap_intersect (bprm->cap_permitted,
+     current->cap_bset);
  working = cap_intersect (bprm->cap_inheritable,
      current->cap_inheritable);
  new_permitted = cap_combine (new_permitted, working);
@@ -565,6 +563,29 @@ int cap_task_kill(struct task_struct *p, struct siginfo *info,

  return -EPERM;
 }
+
+/*
+ * called from kernel/sys.c for prctl(PR_CABSET_DROP)
+ * done without task_capability_lock() because it introduces
+ * no new races - i.e. only another task doing capget() on
+ * this task could get inconsistent info.  There can be no
+ * racing writer bc a task can only change its own caps.
+ */
+long cap_prctl_drop(unsigned long cap)
+{
+ if (!capable(CAP_SETPCAP))
+  return -EPERM;
+ if (!cap_valid(cap))
+  return -EINVAL;
+ cap_lower(current->cap_bset, cap);
+ current->cap_effective = cap_intersect(current->cap_effective,
+  current->cap_bset);
+ current->cap_permitted = cap_intersect(current->cap_permitted,
+  current->cap_bset);
+ current->cap_inheritable = cap_intersect(current->cap_inheritable,
+  current->cap_bset);
+ return 0;
+}
 #else
```

```
 int cap_task_setscheduler (struct task_struct *p, int policy,
       struct sched_param *lp)
--
1.5.1.1.GIT
```