
Subject: [PATCH RFC] capabilities: introduce per-process capability bounding set
Posted by [serue](#) on Fri, 14 Sep 2007 18:52:57 GMT
[View Forum Message](#) <> [Reply to Message](#)

>From bb9e794d1f2e50b59f853a5d7fea925641c68c08 Mon Sep 17 00:00:00 2001
From: sergeh@us.ibm.com <sergeh@us.ibm.com>
Date: Tue, 11 Sep 2007 13:51:10 -0700
Subject: [PATCH 1/1] capabilities: introduce per-process capability bounding set

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. It is inherited at fork from parent. Noone can add elements, CAP_SYS_ADMIN is required to remove them. Perhaps a new capability should be introduced to control the ability to remove capabilities, in order to help prevent running a privileged app with enough privs to be dangerous but not enough to be successful.

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.

Signed-off-by: Serge Hallyn <serue@us.ibm.com>

include/linux/capability.h | 12 ++++++++
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 | 10 ++++++++
kernel/sysctl.c | 35 -----
kernel/sysctl_check.c | 7 -----
security/commoncap.c | 6 ++----
11 files changed, 30 insertions(+), 56 deletions(-)

diff --git a/include/linux/capability.h b/include/linux/capability.h
index 7a8d7ad..25b83bb 100644
--- a/include/linux/capability.h
+++ b/include/linux/capability.h
@@ -197,7 +197,6 @@ typedef __u32 kernel_cap_t;
#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 */
@@ -332,6 +331,17 @@ typedef __u32 kernel_cap_t;
#define CAP_INIT_EFF_SET   to_cap_t(~0 & ~CAP_TO_MASK(CAP_SETPCAP))
#define CAP_INIT_INH_SET   to_cap_t(0)

#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_TO_MASK(x) (1 << (x))
#define cap_raise(c, flag) (cap_t(c) |= CAP_TO_MASK(flag))
#define cap_lower(c, flag) (cap_t(c) &= ~CAP_TO_MASK(flag))
diff --git a/include/linux/init_task.h b/include/linux/init_task.h
index 1ac10c0..20e91ea 100644
--- a/include/linux/init_task.h
+++ b/include/linux/init_task.h
@@ -146,6 +146,7 @@ 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..a7de023 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_GET_CAPBSET 23
+#define PR_SET_CAPBSET 24
+

```

```

#endif /* _LINUX_PRCTL_H */
diff --git a/include/linux/sched.h b/include/linux/sched.h
index f6cf87e..f964743 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -973,7 +973,7 @@ 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 13d48fd..4a62edc 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 e99171f..3771782 100644
--- a/include/linux/sysctl.h
+++ b/include/linux/sysctl.h
@@ -103,7 +103,6 @@ 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 */

@@ -968,8 +967,6 @@ 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 a966c53..7331d62 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1072,6 +1072,7 @@ 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;
+ container_fork(p);
diff --git a/kernel/sys.c b/kernel/sys.c
index 787b73e..53c09fb 100644
--- a/kernel/sys.c
+++ b/kernel/sys.c
@@ -1740,6 +1740,16 @@ asmlinkage long sys_prctl(int option, unsigned long arg2, unsigned
long arg3,
    case PR_SET_SECCOMP:
        error = prctl_set_seccomp(arg2);
        break;
+ case PR_GET_CAPBSET:
+ error = put_user(current->cap_bset, (unsigned long __user *)arg2);
+ break;
+ case PR_SET_CAPBSET:
+ if (!capable(CAP_SYS_ADMIN))
+ return -EPERM;
+ if (!cap_issubset(arg2, current->cap_bset))
+ return -EINVAL;
+ current->cap_bset = arg2;
+ break;

    default:
        error = -EINVAL;
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index dc2378d..b46e4a9 100644
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -377,15 +377,6 @@ 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,
@@ -1915,26 +1906,6 @@ 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
- */
@@ -2348,12 +2319,6 @@ 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 3c9ef5a..41c7f16 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" },

@@ -1535,9 +1531,6 @@ 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 43f9027..46e03e3 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 */
@@ -307,7 +304,8 @@ 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);
--
1.5.1
```

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

Subject: Re: [PATCH RFC] capabilities: introduce per-process capability bounding set
Posted by [Andrew Morgan](#) on Thu, 20 Sep 2007 04:16:58 GMT
[View Forum Message](#) <> [Reply to Message](#)

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Serge E. Hallyn wrote:
> + case PR_GET_CAPBSET:
> + error = put_user(current->cap_bset, (unsigned long __user *)arg2);
> + break;
> + case PR_SET_CAPBSET:
> + if (!capable(CAP_SYS_ADMIN))
> + return -EPERM;
> + if (!cap_issubset(arg2, current->cap_bset))
> + return -EINVAL;
> + current->cap_bset = arg2;
> + break;

You need to pass the capability magic value in both get and set directions... Otherwise, you'll not be able to tell what vintage of cap_bset you are manipulating.

Cheers

Andrew
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.6 (GNU/Linux)

iD8DBQFG8fQ0QheEq9QabfIRApzJAKCUSxj72X4F++kNGy29oO6FE/OGAgCelrBw
dzyfE/XF2FI71WQvIwu/E9s=
=hkFZ
-----END PGP SIGNATURE-----

Containers mailing list

Subject: Re: [PATCH RFC] capabilities: introduce per-process capability bounding set

Posted by [serue](#) on Mon, 24 Sep 2007 20:55:43 GMT

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

Quoting Andrew Morgan (morgan@kernel.org):

> -----BEGIN PGP SIGNED MESSAGE-----

> Hash: SHA1

>

> Serge E. Hallyn wrote:

> > + case PR_GET_CAPBSET:

> > + error = put_user(current->cap_bset, (unsigned long __user *)arg2);

> > + break;

> > + case PR_SET_CAPBSET:

> > + if (!capable(CAP_SYS_ADMIN))

> > + return -EPERM;

> > + if (!cap_issubset(arg2, current->cap_bset))

> > + return -EINVAL;

> > + current->cap_bset = arg2;

> > + break;

>

> You need to pass the capability magic value in both get and set

> directions... Otherwise, you'll not be able to tell what vintage of

> cap_bset you are manipulating.

Great point, thanks. How about this version?

thanks,

-serge

>From 1236d211e2e1d41a8a60b9577e3d4b509a17de92 Mon Sep 17 00:00:00 2001

From: sergeh@us.ibm.com <hallyn@kernel.(none)>

Date: Mon, 24 Sep 2007 13:02:11 -0700

Subject: [PATCH 1/1] capabilities: introduce per-process capability bounding set (v2)

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. It is inherited at fork from parent. Noone can add elements, CAP_SYS_ADMIN is

required to remove them. Perhaps a new capability should be introduced to control the ability to remove capabilities, in order to help prevent running a privileged app with enough privs to be dangerous but not enough to be successful.

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 hacky test program will get and set the bounding set. For instance

```
./bset get
(lists capabilities in bset)
./bset strset cap_sys_admin
(starts shell with new bset)
(use capset or setuid binary to try to increase caps)
```

```
=====
bset.c:
=====
```

```
unsigned long newval;
int cmd_getbcap;
```

```
char *captable[] = {
"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"};

```

```

char *inttocap(unsigned long v)

```

```

{
    char *str = NULL;
    int i;

    str = malloc(1);
    str[0] = '\0';
    for (i=0; i<31; i++) {
        if (v & (1 << (i+1))) {
            char *tmp = captable[i];
            str = realloc(str, strlen(str)+2+strlen(tmp));
            sprintf(str+strlen(str), "%s", tmp);
        }
    }
    return str;
}

```

```

int getbcap(void)

```

```

{
    unsigned long bcap;
    int ret;
    unsigned long ver;

    ret = prctl(PR_GET_CAPBSET, &ver, &bcap);
    if (ret == -1)
        perror("prctl");
    if (ver != _LINUX_CAPABILITY_VERSION)
        printf("wrong capability version: %lu not %lu\n",
            ver, _LINUX_CAPABILITY_VERSION);
    printf("prctl get_bcap returned %lu (ret %d)\n", bcap, ret);
    printf("that is %s\n", inttocap(bcap));
    return ret;
}

```

```

int setbcap(unsigned long val)

```

```

{
    int ret;

```

```

ret = prctl(PR_SET_CAPBSET, _LINUX_CAPABILITY_VERSION, val);
return ret;
}

```

```

int usage(char *me)
{
    printf("Usage: %s get\n", me);
    printf("    %s set <newval>\n", me);
    printf("    %s strset capability_string\n", me);
    printf("        capability_string is for instance:\n");
    printf("        cap_sys_admin,cap_mknod,cap_dac_override\n");
    return 1;
}

```

```

unsigned long captoint(char *cap)
{
    if (strcmp(cap, "cap_dac_override") == 0)
        return 1;
    else if (strcmp(cap, "cap_dac_read_search") == 0)
        return 2;
    else if (strcmp(cap, "cap_fowner") == 0)
        return 3;
    else if (strcmp(cap, "cap_fsetid") == 0)
        return 4;
    else if (strcmp(cap, "cap_kill") == 0)
        return 5;
    else if (strcmp(cap, "cap_setgid") == 0)
        return 6;
    else if (strcmp(cap, "cap_setuid") == 0)
        return 7;
    else if (strcmp(cap, "cap_setpcap") == 0)
        return 8;
    else if (strcmp(cap, "cap_linux_immutable") == 0)
        return 9;
    else if (strcmp(cap, "cap_net_bind_service") == 0)
        return 10;
    else if (strcmp(cap, "cap_net_broadcast") == 0)
        return 11;
    else if (strcmp(cap, "cap_net_admin") == 0)
        return 12;
    else if (strcmp(cap, "cap_net_raw") == 0)
        return 13;
    else if (strcmp(cap, "cap_ipc_lock") == 0)
        return 14;
    else if (strcmp(cap, "cap_ipc_owner") == 0)
        return 15;
    else if (strcmp(cap, "cap_sys_module") == 0)

```

```

    return 16;
else if (strcmp(cap, "cap_sys_rawio") == 0)
    return 17;
else if (strcmp(cap, "cap_sys_chroot") == 0)
    return 18;
else if (strcmp(cap, "cap_sys_ptrace") == 0)
    return 19;
else if (strcmp(cap, "cap_sys_pacct") == 0)
    return 20;
else if (strcmp(cap, "cap_sys_admin") == 0)
    return 21;
else if (strcmp(cap, "cap_sys_boot") == 0)
    return 22;
else if (strcmp(cap, "cap_sys_nice") == 0)
    return 23;
else if (strcmp(cap, "cap_sys_resource") == 0)
    return 24;
else if (strcmp(cap, "cap_sys_time") == 0)
    return 25;
else if (strcmp(cap, "cap_sys_tty_config") == 0)
    return 26;
else if (strcmp(cap, "cap_mknod") == 0)
    return 27;
else if (strcmp(cap, "cap_lease") == 0)
    return 28;
else if (strcmp(cap, "cap_audit_write") == 0)
    return 29;
else if (strcmp(cap, "cap_audit_control") == 0)
    return 30;
else if (strcmp(cap, "cap_setfcap") == 0)
    return 31;
}

```

```

unsigned long parse_cap_string(char *capstring)
{
    unsigned long tmp, newval = 0;
    char *token = strtok(capstring, ",");

    while (token) {
        tmp = captoint(token);
        if (tmp < 0)
            return -1;
        newval |= 1<<tmp;
        token = strtok(NULL, ",");
    }
    return newval;
}

```

```

int read_args(int argc, char *argv[])
{
    if (strcmp(argv[1], "get") == 0) {
        cmd_getbcap = 1;
        return 0;
    }
    if (strcmp(argv[1], "strset") == 0) {
        newval = parse_cap_string(argv[2]);
        if (newval < 0)
            return newval;
        return 0;
    }
    if (strcmp(argv[1], "set") != 0)
        return 1;
    if (argc != 3)
        return 1;
    newval = strtoul(argv[2], NULL, 10);
    return 0;
}

```

```

int main(int argc, char *argv[])
{
    int ret;

```

```

    if (argc<2)
        return usage(argv[0]);

```

```

    if ((ret=read_args(argc, argv)))
        return ret;

```

```

    if (cmd_getbcap)
        return getbcap();

```

```

    ret = setbcap(newval);
    if (ret != 0)
        return ret;
    return execl("/bin/bash", "/bin/bash", NULL);
}

```

=====

Changelog:

As suggested by Andrew Morgan, send the capability version along with the bset for prctl(PR_SET_CAPBSET) and PR_GET_CAPBSET)

Signed-off-by: Serge Hallyn <serue@us.ibm.com>

include/linux/capability.h | 12 ++++++++--

```

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              | 17 ++++++
kernel/sysctl.c           | 35 -----
kernel/sysctl_check.c     | 7 -----
security/commoncap.c      | 6 ++----
11 files changed, 37 insertions(+), 56 deletions(-)

```

```
diff --git a/include/linux/capability.h b/include/linux/capability.h
```

```
index 7a8d7ad..25b83bb 100644
```

```
--- a/include/linux/capability.h
```

```
+++ b/include/linux/capability.h
```

```
@@ -197,7 +197,6 @@ typedef __u32 kernel_cap_t;
```

```
#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 */
```

```
@@ -332,6 +331,17 @@ typedef __u32 kernel_cap_t;
```

```
#define CAP_INIT_EFF_SET    to_cap_t(~0 & ~CAP_TO_MASK(CAP_SETPCAP))
```

```
#define CAP_INIT_INH_SET    to_cap_t(0)
```

```
+#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_TO_MASK(x) (1 << (x))
```

```
#define cap_raise(c, flag) (cap_t(c) |= CAP_TO_MASK(flag))
```

```
#define cap_lower(c, flag) (cap_t(c) &= ~CAP_TO_MASK(flag))
```

```
diff --git a/include/linux/init_task.h b/include/linux/init_task.h
```

```
index a3f2541..c2f4285 100644
```

```
--- a/include/linux/init_task.h
```

```
+++ b/include/linux/init_task.h
```

```
@@ -148,6 +148,7 @@ 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..a7de023 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_GET_CAPBSET 23
+#define PR_SET_CAPBSET 24
+
#endif /* _LINUX_PRCTL_H */
diff --git a/include/linux/sched.h b/include/linux/sched.h
index 12ef317..e9f5c36 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -963,7 +963,7 @@ 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 ff3f857..e2d2f06 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 e99171f..3771782 100644
--- a/include/linux/sysctl.h
+++ b/include/linux/sysctl.h
@@ -103,7 +103,6 @@ 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 */

@@ -968,8 +967,6 @@ 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 f85731a..df13a01 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1082,6 +1082,7 @@ 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 787b73e..3ea61a5 100644
--- a/kernel/sys.c
+++ b/kernel/sys.c
@@ -1740,6 +1740,23 @@ asmlinkage long sys_prctl(int option, unsigned long arg2, unsigned
long arg3,
    case PR_SET_SECCOMP:
        error = prctl_set_seccomp(arg2);
        break;
+ case PR_GET_CAPBSET:
+     error = put_user(_LINUX_CAPABILITY_VERSION,
+         (__u32 __user *)arg2);
+     if (error)
+         break;
+     error = put_user(current->cap_bset,

```

```

+ (unsigned long __user *)arg3);
+ break;
+ case PR_SET_CAPBSET:
+ if (!capable(CAP_SYS_ADMIN))
+ return -EPERM;
+ if (arg2 != _LINUX_CAPABILITY_VERSION)
+ return -EINVAL;
+ if (!cap_issubset(arg3, current->cap_bset))
+ return -EINVAL;
+ current->cap_bset = arg3;
+ break;

default:
error = -EINVAL;
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index 3ad2139..652a31e 100644
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -366,15 +366,6 @@ 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,
@@ -1885,26 +1876,6 @@ 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
 */
@@ -2318,12 +2289,6 @@ 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 3c9ef5a..41c7f16 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" },

@@ -1535,9 +1531,6 @@ int sysctl_check_table(struct ctl_table *table)
(table->strategy == sysctl_ms_jiffies) ||
(table->proc_handler == proc_doststring) ||
(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 43f9027..46e03e3 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 */
@@ -307,7 +304,8 @@ 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);
--
1.5.1

```

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