

[PATCH] Stacker – single-use static slots

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-08/0840.html>

serue_at_us.ibm.com

Date: 08/03/05

Date: Wed, 3 Aug 2005 11:45:16 -0500
To: James Morris <jmorris@redhat.com>

Quoting James Morris (jmorris@redhat.com):

> *On Wed, 27 Jul 2005, James Morris wrote:*

>

> > *Calls to security_get_value() etc. can then be very fast and simple for*
> > *the common case, where the security blob is a pointer offset by an index*
> > *in a small array. The arbitrarily sized hlist would then be a fallback*
> > *with a higher performance hit.*

>

> *Also, the static slots could be single-use only, so once a module*
> *registers for one, it's permanently assigned (even if the module unloads),*
> *to avoid locking overhead. Module registration can just be test & set a*
> *bitmap.*

Oops, I didn't notice this last part, so module unload does unregister the slot, but this shouldn't affect performance. However I may get rid of it still since the non-stacker mod_reg_security is not protected. Or, I may just throw in a spinlock, since this is clearly an infrequent call.

The attached patch implements your idea on top of my previous patchset. Following is performance data on a 16-way ppc. dbench and tbench were run 50 times, kernbench and reaim 10 times each; results are mean +/- 95% confidence half-interval. The 'static slot' kernel had a single static slot for selinux, plus the (unused in this case) shared struct hlist_head security.

The improvements are very slight, but at least in the case of dbench they appear to be significant. I'm particularly surprised by the reaim results, as I would have expected some punishment for the extra memory use.

dbench (throughput, higher is better):
plain stacker: 1450.847800 +/- 12.666275
static slot: 1471.137400 +/- 5.905005

kernbench (time, lower is better):
plain stacker: 52.977000 +/- 0.101730

Linux-Kernel: [PATCH] Stacker – single-use static slots

static slot: 52.976000 +/- 0.129828

tbench (throughput, higher is better):

plain stacker: 140.335360 +/- 2.622280

static slot: 142.455340 +/- 2.957549

reaim (number of jobs vs number of children forked, higher is better)

plain stacker:

1 100542.857000 3296.286475
3 284142.855000 16481.432373
5 480857.144000 26914.062669
7 622200.000000 23074.003061
9 799971.426000 29666.576010
11 949692.861000 27194.361719
13 1065535.715000 53564.650500
15 1147500.000000 0.000000

static slot:

1 100542.857000 3296.286475
3 288514.284000 16148.439818
5 488142.858000 25175.800349
7 612000.000000 0.000000
9 799971.426000 29666.576010
11 949692.861000 27194.361719
13 1093950.001000 49092.813099
15 1147500.000000 0.000000

thank,
–serge

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

```
--
fs/inode.c | 2
include/linux/binfmts.h | 1
include/linux/fs.h | 4
include/linux/ipc.h | 1
include/linux/msg.h | 1
include/linux/sched.h | 1
include/linux/security-stack.h | 2
include/linux/security.h | 74 ++++++--
include/net/sock.h | 1
ipc/msg.c | 2
ipc/msgutil.c | 2
ipc/sem.c | 2
ipc/shm.c | 2
kernel/fork.c | 2
security/Kconfig | 12 +
security/cap_stack.c | 4
security/capability.c | 4
security/root_plug.c | 4
security/seclvl.c | 22 +-
security/security.c | 79 ++++++--
security/selinux/hooks.c | 375 ++++++-----
security/selinux/selinuxfs.c | 11 -
```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

security/stacker.c          |    2
23 files changed, 443 insertions(+), 167 deletions(-)
Index: linux-2.6.12/include/linux/security.h
=====
--- linux-2.6.12.orig/include/linux/security.h    2005-08-01 20:00:51.000000000 -0500
+++ linux-2.6.12/include/linux/security.h        2005-08-01 20:23:52.000000000 -0500
@@ -44,24 +44,65 @@ struct security_list {
};

+static inline void *__get_value(void *head, int idx)
+{
+    void **p = head + sizeof(struct hlist_head);
+#if 0
+    printk(KERN_NOTICE "__get_value: %s (%d): head %lx p %lx idx %d returning %lx at %lx\n",
+           __FUNCTION__, __LINE__, (long)head, (long)p, idx, (long)p[idx], (long)&p[idx]);
+#endif
+    return p[idx];
+}
+
+static inline void __set_value(void *head, int idx, void *v)
+{
+    void **p = head + sizeof(struct hlist_head);
+    p[idx] = v;
+#if 0
+    printk(KERN_NOTICE "%s (%d): hd %lx, p %lx, idx %d,"
+           "v %lx, p[idx] %lx at %lx\n",
+           __FUNCTION__, __LINE__, (long) (head),
+           (long) p, idx, (long) (v),
+           (long)p[idx], (long)&p[idx]);
+#endif
+}
+
+/*
+ * These #defines present more convenient interfaces to
+ * LSMs for using the security{g,d,s}et_value functions.
+ */
+#define security_get_value_type(head, id, type) ( { \
-    struct security_list *v = security_get_value((head), id); \
-    v ? hlist_entry(v, type, lsm_list) : NULL; } )
-
+#define security_set_value_type(head, id, value) \
-    security_set_value((head), id, &(value)->lsm_list);
-
+#define security_add_value_type(head, id, value) \
-    security_add_value((head), id, &(value)->lsm_list);
-
+#define security_del_value_type(head, id, type) ( { \
-    struct security_list *v; \
-    v = security_del_value((head), id); \
-    v ? hlist_entry(v, type, lsm_list) : NULL; } )
+#define security_get_value_type(head, id, type, idx) (idx>=0) ? \
+    (type *)__get_value((head), idx) \
+    : ( { \
+        struct security_list *v = security_get_value((head), id); \
+        v ? hlist_entry(v, type, lsm_list) : NULL; \
+    } )
+
+#define security_set_value_type(head, id, value, idx) \
+    do { \
+        if (idx>=0) { \
+            __set_value((head), idx, (value)); \

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

+         } else { \
+             security_set_value((head), id, &(value)->lsm_list); \
+         } \
+     } while (0);
+
+#define security_add_value_type(head, id, value, idx) \
+     do { \
+         if (idx>=0) { \
+             __set_value((head), idx, (value)); \
+         } else { \
+             security_add_value((head), id, &(value)->lsm_list); \
+         } \
+     } while (0);
+
+#define security_del_value_type(head, id, type, idx) (idx>=0) ? \
+     (type *)__get_value((head), idx) \
+     : ( { \
+         struct security_list *v; \
+         v = security_del_value((head), id); \
+         v ? hlist_entry(v, type, lsm_list) : NULL; \
+     } )

/* security_disown_value is really only to be used by stacker */
extern void security_disown_value(struct hlist_head *);
@@ -2022,9 +2063,10 @@ static inline int security_netlink_recv(

/* prototypes */
extern int security_init      (void);
-extern int register_security (struct security_operations *ops);
+extern int register_security (struct security_operations *ops, int *idx);
extern int unregister_security (struct security_operations *ops);
-extern int mod_reg_security  (const char *name, struct security_operations *ops);
+extern int mod_reg_security  (const char *name,
+                             struct security_operations *ops, int *idx);
extern int mod_unreg_security (const char *name, struct security_operations *ops);
extern struct dentry *securityfs_create_file(const char *name, mode_t mode,
                                           struct dentry *parent, void *data,
Index: linux-2.6.12/security/Kconfig
=====
--- linux-2.6.12.orig/security/Kconfig 2005-08-01 20:00:51.000000000 -0500
+++ linux-2.6.12/security/Kconfig      2005-08-01 20:24:11.000000000 -0500
@@ -112,5 +112,17 @@ config SECURITY_STACKER
     help
         Stack multiple LSMs.

+config SECURITY_STACKER_NUMFIELDS
+     int "Number of security fields to reserve"
+     depends on SECURITY_STACKER
+     default 1
+     help
+         This option reserves extra space in each kernel object
+         for security information. This may speed up modules which
+         make a lot of use of these fields, as these accesses can
+         be lock-free. However each kernel object requires one
+         extra byte for each field reserved here, which can slow
+         the system down.
+
     endmenu

Index: linux-2.6.12/fs/inode.c
=====
--- linux-2.6.12.orig/fs/inode.c       2005-08-01 20:00:50.000000000 -0500

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

+++ linux-2.6.12/fs/inode.c      2005-08-01 20:24:25.000000000 -0500
@@ -134,6 +134,8 @@ static struct inode *alloc_inode(struct
        inode->i_cdev = NULL;
        inode->i_rdev = 0;
        INIT_HLIST_HEAD(&inode->i_security);
+
+       memset(&inode->i_security_p, 0,
+             CONFIG_SECURITY_STACKER_NUMFIELDS*sizeof(void *));
        inode->dirtyed_when = 0;
        if (security_inode_alloc(inode)) {
            if (inode->i_sb->s_op->destroy_inode)
Index: linux-2.6.12/include/linux/binfmts.h
=====
--- linux-2.6.12.orig/include/linux/binfmts.h      2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/include/linux/binfmts.h          2005-08-01 20:24:41.000000000 -0500
@@ -30,6 +30,7 @@ struct linux_binprm{
        int e_uid, e_gid;
        kernel_cap_t cap_inheritable, cap_permitted, cap_effective;
        struct hlist_head security;
+
+       void * security_p[CONFIG_SECURITY_STACKER_NUMFIELDS];
        int argc, envc;
        char * filename;          /* Name of binary as seen by procps */
        char * interp;           /* Name of the binary really executed. Most
Index: linux-2.6.12/include/linux/fs.h
=====
--- linux-2.6.12.orig/include/linux/fs.h           2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/include/linux/fs.h               2005-08-01 20:24:55.000000000 -0500
@@ -486,6 +486,7 @@ struct inode {
        atomic_t          i_writecount;
        struct hlist_head i_security;
+
+       void              *i_security_p[CONFIG_SECURITY_STACKER_NUMFIELDS];
        union {
            void          *generic_ip;
        } u;
@@ -560,6 +561,7 @@ struct fown_struct {
        int pid;          /* pid or -pgrp where SIGIO should be sent */
        uid_t uid, euid;  /* uid/euid of process setting the owner */
        struct hlist_head security;
+
+       void * security_p[CONFIG_SECURITY_STACKER_NUMFIELDS];
        int signum;      /* posix.1b rt signal to be delivered on IO */
    };
@@ -597,6 +599,7 @@ struct file {
        size_t          f_maxcount;
        unsigned long   f_version;
        struct hlist_head f_security;
+
+       void              *f_security_p[CONFIG_SECURITY_STACKER_NUMFIELDS];

        /* needed for tty driver, and maybe others */
        void              *private_data;
@@ -786,6 +789,7 @@ struct super_block {
        int              s_need_sync_fs;
        atomic_t         s_active;
        struct hlist_head s_security;
+
+       void              *s_security_p[CONFIG_SECURITY_STACKER_NUMFIELDS];
        struct xattr_handler **s_xattr;

        struct list_head s_inodes;          /* all inodes */
Index: linux-2.6.12/include/linux/ipc.h
=====
--- linux-2.6.12.orig/include/linux/ipc.h         2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/include/linux/ipc.h             2005-08-01 20:25:09.000000000 -0500

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

@@ -67,6 +67,7 @@ struct kern_ipc_perm
    mode_t          mode;
    unsigned long   seq;
    struct hlist_head security;
+   void           *security_p[CONFIG_SECURITY_STACKER_NUMFIELDS];
};

#ifdef /* __KERNEL__ */
Index: linux-2.6.12/include/linux/msg.h
=====
--- linux-2.6.12.orig/include/linux/msg.h      2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/include/linux/msg.h      2005-08-01 20:25:28.000000000 -0500
@@ -71,6 +71,7 @@ struct msg_msg {
    int m_ts;          /* message text size */
    struct msg_msgseg* next;
    struct hlist_head security;
+   void * security_p[CONFIG_SECURITY_STACKER_NUMFIELDS];
    /* the actual message follows immediately */
};

Index: linux-2.6.12/include/linux/sched.h
=====
--- linux-2.6.12.orig/include/linux/sched.h    2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/include/linux/sched.h    2005-08-01 20:25:56.000000000 -0500
@@ -722,6 +722,7 @@ struct task_struct {
    sigset_t *notifier_mask;

    struct hlist_head security;
+   void *security_p[CONFIG_SECURITY_STACKER_NUMFIELDS];
    struct audit_context *audit_context;
    seccomp_t seccomp;

Index: linux-2.6.12/include/net/sock.h
=====
--- linux-2.6.12.orig/include/net/sock.h      2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/include/net/sock.h      2005-08-01 20:26:11.000000000 -0500
@@ -241,6 +241,7 @@ struct sock {
    __u32          sk_sndmsg_off;
    int           sk_write_pending;
    struct hlist_head sk_security;
+   void         *sk_security_p[CONFIG_SECURITY_STACKER_NUMFIELDS];
    void         (*sk_state_change)(struct sock *sk);
    void         (*sk_data_ready)(struct sock *sk, int bytes);
    void         (*sk_write_space)(struct sock *sk);

Index: linux-2.6.12/ipc/msg.c
=====
--- linux-2.6.12.orig/ipc/msg.c 2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/ipc/msg.c      2005-08-01 20:26:27.000000000 -0500
@@ -100,6 +100,8 @@ static int newque (key_t key, int msgflg
    msq->q_perm.key = key;

    INIT_HLIST_HEAD(&msq->q_perm.security);
+   memset(&msq->q_perm.security_p, 0,
+         CONFIG_SECURITY_STACKER_NUMFIELDS*sizeof(void *));
    retval = security_msg_queue_alloc(msq);
    if (retval) {
        ipc_rcu_putref(msq);
Index: linux-2.6.12/ipc/msgutil.c
=====
--- linux-2.6.12.orig/ipc/msgutil.c 2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/ipc/msgutil.c 2005-08-01 20:26:41.000000000 -0500
@@ -42,6 +42,8 @@ struct msg_msg *load_msg(const void __us

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

msg->next = NULL;
INIT_HLIST_HEAD(&msg->security);
+   memset(&msg->security_p, 0,
+         CONFIG_SECURITY_STACKER_NUMFIELDS*sizeof(void *));

    if (copy_from_user(msg + 1, src, alen)) {
        err = -EFAULT;
Index: linux-2.6.12/ipc/sem.c
=====
--- linux-2.6.12.orig/ipc/sem.c 2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/ipc/sem.c      2005-08-01 20:26:57.000000000 -0500
@@ -179,6 +179,8 @@ static int newary (key_t key, int nsems,
    sma->sem_perm.key = key;

    INIT_HLIST_HEAD(&sma->sem_perm.security);
+   memset(&sma->sem_perm.security_p, 0,
+         CONFIG_SECURITY_STACKER_NUMFIELDS*sizeof(void *));
    retval = security_sem_alloc(sma);
    if (retval) {
        ipc_rcu_putref(sma);
Index: linux-2.6.12/ipc/shm.c
=====
--- linux-2.6.12.orig/ipc/shm.c 2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/ipc/shm.c      2005-08-01 20:27:10.000000000 -0500
@@ -200,6 +200,8 @@ static int newseg (key_t key, int shmflg
    shp->mlock_user = NULL;

    INIT_HLIST_HEAD(&shp->shm_perm.security);
+   memset(&shp->shm_perm.security_p, 0,
+         CONFIG_SECURITY_STACKER_NUMFIELDS*sizeof(void *));
    error = security_shm_alloc(shp);
    if (error) {
        ipc_rcu_putref(shp);
Index: linux-2.6.12/kernel/fork.c
=====
--- linux-2.6.12.orig/kernel/fork.c      2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/kernel/fork.c 2005-08-01 20:27:22.000000000 -0500
@@ -942,6 +942,8 @@ static task_t *copy_process(unsigned lon
    p->lock_depth = -1;          /* -1 = no lock */
    do_posix_clock_monotonic_gettime(&p->start_time);
    INIT_HLIST_HEAD(&p->security);
+   memset(&p->security_p, 0,
+         CONFIG_SECURITY_STACKER_NUMFIELDS*sizeof(void *));
    p->io_context = NULL;
    p->io_wait = NULL;
    p->audit_context = NULL;
Index: linux-2.6.12/security/security.c
=====
--- linux-2.6.12.orig/security/security.c 2005-08-01 20:00:51.000000000 -0500
+++ linux-2.6.12/security/security.c     2005-08-01 20:27:40.000000000 -0500
@@ -20,6 +20,9 @@

#define SECURITY_FRAMEWORK_VERSION      "1.0.0"

+static struct security_operations
+   *security_field_owners[CONFIG_SECURITY_STACKER_NUMFIELDS];
+
fastcall struct security_list *
security_get_value(struct hlist_head *head, int security_id)
{
@@ -49,12 +52,12 @@ fastcall void

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

security_set_value(struct hlist_head *head, int security_id,
                  struct security_list *obj_node)
{
-
    obj_node->security_id = security_id;
    hlist_add_head(&obj_node->list, head);
}

static DEFINE_SPINLOCK(stacker_value_spinlock);
+static DEFINE_SPINLOCK(security_field_spinlock);

/*
 * Used outside of security_*_alloc hooks, so we need to
@@ -65,7 +68,6 @@ fastcall void
security_add_value(struct hlist_head *head, int security_id,
                  struct security_list *obj_node)
{
-
    spin_lock(&stacker_value_spinlock);
    obj_node->security_id = security_id;
    hlist_add_head_rcu(&obj_node->list, head);
@@ -86,9 +88,13 @@ security_add_value(struct hlist_head *he
 * XXX TODO - switch this to take a type, and deref
 * obj->lsm_list.list here.
 */
-int security_unlink_value(struct hlist_node *n)
+int security_unlink_value(struct hlist_node *n, int idx)
{
    int ret = 0;
+
+    if (idx >= 0)
+        return 0;
+
    spin_lock(&stacker_value_spinlock);
    if (n->pprev == LIST_POISON2) {
        ret = 1;
@@ -223,7 +229,7 @@ int __init security_init(void)
 * If there is already a security module registered with the kernel,
 * an error will be returned. Otherwise 0 is returned on success.
 */
-int register_security(struct security_operations *ops)
+int register_security(struct security_operations *ops, int *idx)
{
    if (security_ops != &dummy_security_ops)
        return -EAGAIN;
@@ -234,6 +240,21 @@ int register_security(struct security_op
    return -EINVAL;
}

+    spin_lock(&security_field_spinlock);
+    if (idx && *idx) {
+        int i;
+
+        *idx = -1;
+        for (i=0; i<CONFIG_SECURITY_STACKER_NUMFIELDS; i++) {
+            if (security_field_owners[i] == NULL) {
+                security_field_owners[i] = ops;
+                *idx = i;
+                break;
+            }
+        }
+    }
+}

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

+     spin_unlock(&security_field_spinlock);
+
+     security_ops = ops;
+
+     return 0;
@@ -252,6 +273,8 @@ int register_security(struct security_op
 */
int unregister_security(struct security_operations *ops)
{
+     int i;
+
+     if (ops != security_ops) {
+         printk(KERN_INFO "%s: trying to unregister "
+             "a security_ops structure that is not "
@@ -261,6 +284,15 @@ int unregister_security(struct security_

+     security_ops = &dummy_security_ops;
+
+     spin_lock(&security_field_spinlock);
+     for (i=0; i<CONFIG_SECURITY_STACKER_NUMFIELDS; i++) {
+         if (security_field_owners[i] == ops) {
+             security_field_owners[i] = NULL;
+             break;
+         }
+     }
+     spin_unlock(&security_field_spinlock);
+
+     return 0;
}

@@ -276,8 +308,11 @@ int unregister_security(struct security_
 * The return value depends on the currently loaded security module, with 0 as
 * success.
 */
-int mod_reg_security(const char *name, struct security_operations *ops)
+int mod_reg_security(const char *name, struct security_operations *ops,
+     int *idx)
{
+     int ret = 0;
+
+     if (!ops) {
+         printk(KERN_INFO "%s received NULL security operations",
+             __FUNCTION__);
@@ -290,7 +325,28 @@ int mod_reg_security(const char *name, s
+         return -EINVAL;
+     }
+
+     return security_ops->register_security(name, ops);
+     ret = security_ops->register_security(name, ops);
+
+     if (ret < 0)
+         goto out;
+
+     spin_lock(&security_field_spinlock);
+     if (idx && *idx) {
+         int i;
+
+         *idx = -1;
+         for (i=0; i<CONFIG_SECURITY_STACKER_NUMFIELDS; i++) {
+             if (security_field_owners[i] == NULL) {
+                 security_field_owners[i] = ops;
+                 *idx = i;

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

+                                     break;
+                                     }
+                                 }
+                                }
+                                spin_unlock(&security_field_spinlock);
+
+out:
+    return ret;
+}

/**
@@ -308,12 +364,23 @@ int mod_reg_security(const char *name, s
 */
int mod_unreg_security(const char *name, struct security_operations *ops)
{
+    int i;
+
+    if (ops == security_ops) {
+        printk(KERN_INFO "%s invalid attempt to unregister "
+               " primary security ops.\n", __FUNCTION__);
+        return -EINVAL;
+    }

+    spin_lock(&security_field_spinlock);
+    for (i=0; i<CONFIG_SECURITY_STACKER_NUMFIELDS;i++) {
+        if (security_field_owners[i] == ops) {
+            security_field_owners[i] = NULL;
+            break;
+        }
+    }
+    spin_unlock(&security_field_spinlock);

+    return security_ops->unregister_security(name, ops);
}

```

Index: linux-2.6.12/security/selinux/hooks.c

```

=====
--- linux-2.6.12.orig/security/selinux/hooks.c 2005-08-01 20:00:51.000000000 -0500
+++ linux-2.6.12/security/selinux/hooks.c      2005-08-01 20:27:56.000000000 -0500
@@ -78,6 +78,8 @@
 #define XATTR_SELINUX_SUFFIX "selinux"
 #define XATTR_NAME_SELINUX XATTR_SECURITY_PREFIX XATTR_SELINUX_SUFFIX

+int selinux_secidx;
+
+extern unsigned int policydb_loaded_version;
+extern int selinux_nlmsg_lookup(u16 sclass, u16 nlmsg_type, u32 *perm);

@@ -123,7 +125,8 @@ static int task_alloc_security(struct ta
 memset(tsec, 0, sizeof(struct task_security_struct));
 tsec->task = task;
 tsec->osid = tsec->sid = tsec->ptrace_sid = SECINITSID_UNLABELED;
- security_set_value_type(&task->security, SELINUX_LSM_ID, tsec);
+ security_set_value_type(&task->security, SELINUX_LSM_ID, tsec,
+                         selinux_secidx);

 return 0;
}

@@ -133,7 +136,8 @@ static void task_free_security(struct ta
 struct task_security_struct *tsec;

 tsec = security_del_value_type(&task->security, SELINUX_LSM_ID,

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);

        kfree(tsec);
}
@@ -144,7 +148,8 @@ static int inode_alloc_security(struct i
        struct inode_security_struct *isec;

        tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
        isec = kmalloc(sizeof(struct inode_security_struct), GFP_KERNEL);
        if (!isec)
            return -ENOMEM;
@@ -159,7 +164,8 @@ static int inode_alloc_security(struct i
        isec->task_sid = tsec->sid;
        else
            isec->task_sid = SECINITSID_UNLABELED;
-         security_set_value_type(&inode->i_security, SELINUX_LSM_ID, isec);
+         security_set_value_type(&inode->i_security, SELINUX_LSM_ID, isec,
+         selinux_secidx);

        return 0;
}
@@ -170,12 +176,14 @@ static void inode_free_security(struct i
        struct superblock_security_struct *sbsec;

        isec = security_del_value_type(&inode->i_security, SELINUX_LSM_ID,
-         struct inode_security_struct);
+         struct inode_security_struct,
+         selinux_secidx);
        if (!isec)
            return;

        sbsec = security_get_value_type(&inode->i_sb->s_security,
-         SELINUX_LSM_ID, struct superblock_security_struct);
+         SELINUX_LSM_ID, struct superblock_security_struct,
+         selinux_secidx);

        spin_lock(&sbsec->isec_lock);
        if (!list_empty(&isec->list))
@@ -191,7 +199,8 @@ static int file_alloc_security(struct fi
        struct file_security_struct *fsec;

        tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
        fsec = kmalloc(sizeof(struct file_security_struct), GFP_ATOMIC);
        if (!fsec)
            return -ENOMEM;
@@ -205,7 +214,8 @@ static int file_alloc_security(struct fi
        fsec->sid = SECINITSID_UNLABELED;
        fsec->fown_sid = SECINITSID_UNLABELED;
    }
-         security_set_value_type(&file->f_security, SELINUX_LSM_ID, fsec);
+         security_set_value_type(&file->f_security, SELINUX_LSM_ID, fsec,
+         selinux_secidx);

        return 0;

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```
}
@@ -215,7 +225,8 @@ static void file_free_security(struct fi
    struct file_security_struct *fsec;

    fsec = security_del_value_type(&file->f_security, SELINUX_LSM_ID,
-    struct file_security_struct);
+    struct file_security_struct,
+    selinux_secidx);

    kfree(fsec);
}
@@ -236,7 +247,8 @@ static int superblock_alloc_security(str
    sbsec->sb = sb;
    sbsec->sid = SECINITSID_UNLABELED;
    sbsec->def_sid = SECINITSID_FILE;
-    security_set_value_type(&sb->s_security, SELINUX_LSM_ID, sbsec);
+    security_set_value_type(&sb->s_security, SELINUX_LSM_ID, sbsec,
+    selinux_secidx);

    return 0;
}
@@ -246,7 +258,7 @@ static void superblock_free_security(str
    struct superblock_security_struct *sbsec;

    sbsec = security_del_value_type(&sb->s_security, SELINUX_LSM_ID,
-    struct superblock_security_struct);
+    struct superblock_security_struct, selinux_secidx);
    if (!sbsec)
        return;

@@ -273,7 +285,8 @@ static int sk_alloc_security(struct sock
    memset(ssec, 0, sizeof(*ssec));
    ssec->sk = sk;
    ssec->peer_sid = SECINITSID_UNLABELED;
-    security_set_value_type(&sk->sk_security, SELINUX_LSM_ID, ssec);
+    security_set_value_type(&sk->sk_security, SELINUX_LSM_ID, ssec,
+    selinux_secidx);

    return 0;
}
@@ -286,7 +299,7 @@ static void sk_free_security(struct sock
    return;

    ssec = security_del_value_type(&sk->sk_security, SELINUX_LSM_ID,
-    struct sk_security_struct);
+    struct sk_security_struct, selinux_secidx);

    kfree(ssec);
}
@@ -338,9 +351,11 @@ static int try_context_mount(struct supe
    struct superblock_security_struct *sbsec;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);
    sbsec = security_get_value_type(&sb->s_security, SELINUX_LSM_ID,
-    struct superblock_security_struct);
+    struct superblock_security_struct,
+    selinux_secidx);

    if (!data)
```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```
        goto out;
@@ -512,7 +527,8 @@ static int superblock_doinit(struct supe
    int rc = 0;

    sbsec = security_get_value_type(&sb->s_security, SELINUX_LSM_ID,
-    struct superblock_security_struct);
+    struct superblock_security_struct,
+    selinux_secidx);

    down(&sbsec->sem);
    if (sbsec->initialized)
@@ -748,7 +764,8 @@ static int inode_doinit_with_dentry(stru
    int hold_sem = 0;

    isec = security_get_value_type(&inode->i_security, SELINUX_LSM_ID,
-    struct inode_security_struct);
+    struct inode_security_struct,
+    selinux_secidx);

    if (isec->initialized)
        goto out;
@@ -759,7 +776,8 @@ static int inode_doinit_with_dentry(stru
    goto out;

    sbsec = security_get_value_type(&inode->i_sb->s_security,
-    SELINUX_LSM_ID, struct superblock_security_struct);
+    SELINUX_LSM_ID, struct superblock_security_struct,
+    selinux_secidx);
    if (!sbsec->initialized) {
        /* Defer initialization until selinux_complete_init,
        after the initial policy is loaded and the security
@@ -935,9 +953,11 @@ static int task_has_perm(struct task_str
    struct task_security_struct *tsec1, *tsec2;

    tsec1 = security_get_value_type(&tsk1->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);
    tsec2 = security_get_value_type(&tsk2->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);
    return avc_has_perm(tsec1->sid, tsec2->sid,
        SECClass_PROCESS, perms, NULL);
}
@@ -950,7 +970,8 @@ static int task_has_capability(struct ta
    struct avc_audit_data ad;

    tsec = security_get_value_type(&tsk->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);

    AVC_AUDIT_DATA_INIT(&ad,CAP);
    ad.tsk = tsk;
@@ -967,7 +988,8 @@ static int task_has_system(struct task_s
    struct task_security_struct *tsec;

    tsec = security_get_value_type(&tsk->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);
```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```
    return avc_has_perm(tsec->sid, SECINITSID_KERNEL,
                        SECCLASS_SYSTEM, perms, NULL);
@@ -986,9 +1008,11 @@ static int inode_has_perm(struct task_st
    struct avc_audit_data ad;

    tsec = security_get_value_type(&tsk->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);
    isec = security_get_value_type(&inode->i_security, SELINUX_LSM_ID,
-    struct inode_security_struct);
+    struct inode_security_struct,
+    selinux_secidx);

    if (!adp) {
        adp = &ad;
@@ -1036,9 +1060,11 @@ static inline int file_has_perm(struct t
    int rc;

    tsec = security_get_value_type(&tsk->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);
    fsec = security_get_value_type(&file->f_security, SELINUX_LSM_ID,
-    struct file_security_struct);
+    struct file_security_struct,
+    selinux_secidx);

    AVC_AUDIT_DATA_INIT(&ad, FS);
    ad.u.fs.mnt = mnt;
@@ -1073,11 +1099,14 @@ static int may_create(struct inode *dir,
    int rc;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);
    dsec = security_get_value_type(&dir->i_security, SELINUX_LSM_ID,
-    struct inode_security_struct);
+    struct inode_security_struct,
+    selinux_secidx);
    sbsec = security_get_value_type(&dir->i_sb->s_security, SELINUX_LSM_ID,
-    struct superblock_security_struct);
+    struct superblock_security_struct,
+    selinux_secidx);

    AVC_AUDIT_DATA_INIT(&ad, FS);
    ad.u.fs.dentry = dentry;
@@ -1123,11 +1152,14 @@ static int may_link(struct inode *dir,
    int rc;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);
    dsec = security_get_value_type(&dir->i_security, SELINUX_LSM_ID,
-    struct inode_security_struct);
+    struct inode_security_struct,
+    selinux_secidx);
    isec = security_get_value_type(&dentry->d_inode->i_security,
-    SELINUX_LSM_ID, struct inode_security_struct);
```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

+         SELINUX_LSM_ID, struct inode_security_struct,
+         selinux_secidx);

    AVC_AUDIT_DATA_INIT(&ad, FS);
    ad.u.fs.dentry = dentry;
@@ -1170,14 +1202,18 @@ static inline int may_rename(struct inode
    int rc;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
    old_dsec = security_get_value_type(&old_dir->i_security, SELINUX_LSM_ID,
-         struct inode_security_struct);
+         struct inode_security_struct,
+         selinux_secidx);
    old_isec = security_get_value_type(&old_dentry->d_inode->i_security,
-         SELINUX_LSM_ID, struct inode_security_struct);
+         SELINUX_LSM_ID, struct inode_security_struct,
+         selinux_secidx);
    old_is_dir = S_ISDIR(old_dentry->d_inode->i_mode);
    new_dsec = security_get_value_type(&new_dir->i_security, SELINUX_LSM_ID,
-         struct inode_security_struct);
+         struct inode_security_struct,
+         selinux_secidx);

    AVC_AUDIT_DATA_INIT(&ad, FS);

@@ -1206,7 +1242,8 @@ static inline int may_rename(struct inode
    return rc;
    if (new_dentry->d_inode) {
        new_isec = security_get_value_type(&new_dentry->d_inode->i_security,
-         SELINUX_LSM_ID, struct inode_security_struct);
+         SELINUX_LSM_ID, struct inode_security_struct,
+         selinux_secidx);
        new_is_dir = S_ISDIR(new_dentry->d_inode->i_mode);
        rc = avc_has_perm(tsec->sid, new_isec->sid,
            new_isec->sclass,
@@ -1228,9 +1265,11 @@ static int superblock_has_perm(struct ta
    struct superblock_security_struct *sbsec;

    tsec = security_get_value_type(&tsk->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
    sbsec = security_get_value_type(&sb->s_security, SELINUX_LSM_ID,
-         struct superblock_security_struct);
+         struct superblock_security_struct,
+         selinux_secidx);
    return avc_has_perm(tsec->sid, sbsec->sid, SECCLASS_FILESYSTEM,
        perms, ad);
}
@@ -1287,9 +1326,11 @@ static int inode_security_set_sid(struct
    struct superblock_security_struct *sbsec;

    isec = security_get_value_type(&inode->i_security, SELINUX_LSM_ID,
-         struct inode_security_struct);
+         struct inode_security_struct,
+         selinux_secidx);
    sbsec = security_get_value_type(&inode->i_sb->s_security,
-         SELINUX_LSM_ID, struct superblock_security_struct);
+         SELINUX_LSM_ID, struct superblock_security_struct,

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

+         selinux_secidx);

        if (!sbsec->initialized) {
            /* Defer initialization to selinux_complete_init. */
@@ -1319,11 +1360,14 @@ static int post_create(struct inode *dir
        int rc;

        tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
        dsec = security_get_value_type(&dir->i_security, SELINUX_LSM_ID,
-         struct inode_security_struct);
+         struct inode_security_struct,
+         selinux_secidx);
        sbsec = security_get_value_type(&dir->i_sb->s_security, SELINUX_LSM_ID,
-         struct superblock_security_struct);
+         struct superblock_security_struct,
+         selinux_secidx);

        inode = dentry->d_inode;
        if (!inode) {
@@ -1395,9 +1439,11 @@ static int selinux_ptrace(struct task_st
        int rc;

        psec = security_get_value_type(&parent->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
        csec = security_get_value_type(&child->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);

        rc = task_has_perm(parent, child, PROCESS_PTRACE);
        /* Save the SID of the tracing process for later use in apply_creds. */
@@ -1432,7 +1478,8 @@ static int selinux_sysctl(ctl_table *tab
        int rc;

        tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);

        rc = selinux_proc_get_sid(table->de, (op == 001) ?
            SECCLASS_DIR : SECCLASS_FILE, &tsid);
@@ -1537,7 +1584,8 @@ static int selinux_bprm_alloc_security(s
        bsec->sid = SECINITSID_UNLABELED;
        bsec->set = 0;

-         security_set_value_type(&bprm->security, SELINUX_LSM_ID, bsec);
+         security_set_value_type(&bprm->security, SELINUX_LSM_ID, bsec,
+         selinux_secidx);
        return 0;
    }

@@ -1552,15 +1600,18 @@ static int selinux_bprm_set_security(str
        int rc;

        bsec = security_get_value_type(&bprm->security, SELINUX_LSM_ID,
-         struct bprm_security_struct);
+         struct bprm_security_struct,

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

+         selinux_secidx);

    if (bsec->set)
        return 0;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
    isec = security_get_value_type(&inode->i_security, SELINUX_LSM_ID,
-         struct inode_security_struct);
+         struct inode_security_struct,
+         selinux_secidx);

    /* Default to the current task SID. */
    bsec->sid = tsec->sid;
@@ -1621,7 +1672,8 @@ static int selinux_bprm_secureexec (stru
    int atsecure = 0;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
    if (tsec->osid != tsec->sid) {
        /* Enable secure mode for SIDs transitions unless
           the noatsecure permission is granted between
@@ -1639,7 +1691,7 @@ static void selinux_bprm_free_security(s
    struct bprm_security_struct *bsec;

    bsec = security_del_value_type(&bprm->security, SELINUX_LSM_ID,
-         struct bprm_security_struct);
+         struct bprm_security_struct, selinux_secidx);
    kfree(bsec);
}

@@ -1736,9 +1788,11 @@ static void selinux_bprm_apply_creds(str
    int rc;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
    bsec = security_get_value_type(&bprm->security, SELINUX_LSM_ID,
-         struct bprm_security_struct);
+         struct bprm_security_struct,
+         selinux_secidx);
    sid = bsec->sid;

    tsec->osid = tsec->sid;
@@ -1782,9 +1836,11 @@ static void selinux_bprm_post_apply_cred
    int rc, i;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
    bsec = security_get_value_type(&bprm->security, SELINUX_LSM_ID,
-         struct bprm_security_struct);
+         struct bprm_security_struct,
+         selinux_secidx);

    if (bsec->unsafe) {

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```
force_sig_specific(SIGKILL, current);
@@ -2122,7 +2178,8 @@ static int selinux_inode_setxattr(struct
    }

    sbsec = security_get_value_type(&inode->i_sb->s_security,
-     SELINUX_LSM_ID, struct superblock_security_struct);
+     SELINUX_LSM_ID, struct superblock_security_struct,
+     selinux_secidx);
    if (sbsec->behavior == SECURITY_FS_USE_MNTPOINT)
        return -EOPNOTSUPP;

@@ -2133,9 +2190,11 @@ static int selinux_inode_setxattr(struct
    ad.u.fs.dentry = dentry;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-     struct task_security_struct);
+     struct task_security_struct,
+     selinux_secidx);
    isec = security_get_value_type(&inode->i_security, SELINUX_LSM_ID,
-     struct inode_security_struct);
+     struct inode_security_struct,
+     selinux_secidx);
    rc = avc_has_perm(tsec->sid, isec->sid, isec->sclass,
        FILE__RELABELFROM, &ad);
    if (rc)
@@ -2171,7 +2230,8 @@ static void selinux_inode_post_setxattr(
    int rc;

    isec = security_get_value_type(&inode->i_security, SELINUX_LSM_ID,
-     struct inode_security_struct);
+     struct inode_security_struct,
+     selinux_secidx);

    if (strcmp(name, XATTR_NAME_SELINUX)) {
        /* Not an attribute we recognize, so nothing to do. */
@@ -2195,7 +2255,8 @@ static int selinux_inode_getxattr (struc
    struct superblock_security_struct *sbsec;

    sbsec = security_get_value_type(&inode->i_sb->s_security,
-     SELINUX_LSM_ID, struct superblock_security_struct);
+     SELINUX_LSM_ID, struct superblock_security_struct,
+     selinux_secidx);

    if (sbsec->behavior == SECURITY_FS_USE_MNTPOINT)
        return -EOPNOTSUPP;
@@ -2243,7 +2304,8 @@ static int selinux_inode_getsecurity(str
    return -EOPNOTSUPP;

    isec = security_get_value_type(&inode->i_security, SELINUX_LSM_ID,
-     struct inode_security_struct);
+     struct inode_security_struct,
+     selinux_secidx);

    rc = security_sid_to_context(isec->sid, &context, &len);
    if (rc)
@@ -2273,7 +2335,8 @@ static int selinux_inode_setsecurity(str
    return -EOPNOTSUPP;

    isec = security_get_value_type(&inode->i_security, SELINUX_LSM_ID,
-     struct inode_security_struct);
+     struct inode_security_struct,
+     selinux_secidx);
```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```
        if (!value || !size)
            return -EACCES;
@@ -2512,9 +2575,11 @@ static int selinux_file_set_fowner(struct
    struct file_security_struct *fsec;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);
    fsec = security_get_value_type(&file->f_security, SELINUX_LSM_ID,
-    struct file_security_struct);
+    struct file_security_struct,
+    selinux_secidx);
    fsec->fown_sid = tsec->sid;

    return 0;
@@ -2532,9 +2597,11 @@ static int selinux_file_send_sigiotask(s
    file = (struct file *)((long)fown - offsetof(struct file, f_owner));

    tsec = security_get_value_type(&tsk->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);
    fsec = security_get_value_type(&file->f_security, SELINUX_LSM_ID,
-    struct file_security_struct);
+    struct file_security_struct,
+    selinux_secidx);

    if (!signum)
        perm = signal_to_av(SIGIO); /* as per send_sigio_to_task */
@@ -2563,13 +2630,15 @@ static int selinux_task_alloc_security(s
    int rc;

    tsec1 = security_get_value_type(&current->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);

    rc = task_alloc_security(tsk);
    if (rc)
        return rc;
    tsec2 = security_get_value_type(&tsk->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);

    tsec2->osid = tsec1->osid;
    tsec2->sid = tsec1->sid;
@@ -2699,7 +2768,8 @@ static void selinux_task_reparent_to_ini
    struct task_security_struct *tsec;

    tsec = security_get_value_type(&p->security, SELINUX_LSM_ID,
-    struct task_security_struct);
+    struct task_security_struct,
+    selinux_secidx);
    tsec->osid = tsec->sid;
    tsec->sid = SECINITSID_KERNEL;
    return;
@@ -2712,9 +2782,11 @@ static void selinux_task_to_inode(struct
    struct inode_security_struct *isec;
```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```
tsec = security_get_value_type(&p->security, SELINUX_LSM_ID,
-     struct task_security_struct);
+     struct task_security_struct,
+     selinux_secidx);
isec = security_get_value_type(&inode->i_security, SELINUX_LSM_ID,
-     struct inode_security_struct);
+     struct inode_security_struct,
+     selinux_secidx);

isec->sid = tsec->sid;
isec->initialized = 1;
@@ -2883,9 +2955,11 @@ static int socket_has_perm(struct task_s
int err = 0;

tsec = security_get_value_type(&task->security, SELINUX_LSM_ID,
-     struct task_security_struct);
-     isec = security_get_value_type(&SOCK_INODE(sock)->i_security, SELINUX_LSM_ID,
-     struct inode_security_struct);
+     struct task_security_struct,
+     selinux_secidx);
+     isec = security_get_value_type(&SOCK_INODE(sock)->i_security,
+     SELINUX_LSM_ID, struct inode_security_struct,
+     selinux_secidx);

if (isec->sid == SECINITSID_KERNEL)
    goto out;
@@ -2908,7 +2982,8 @@ static int selinux_socket_create(int fam
    goto out;

tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-     struct task_security_struct);
+     struct task_security_struct,
+     selinux_secidx);
err = avc_has_perm(tsec->sid, tsec->sid,
    socket_type_to_security_class(family, type,
    protocol), SOCKET__CREATE, NULL);
@@ -2924,10 +2999,12 @@ static void selinux_socket_post_create(s
struct task_security_struct *tsec;

isec = security_get_value_type(&SOCK_INODE(sock)->i_security,
-     SELINUX_LSM_ID, struct inode_security_struct);
+     SELINUX_LSM_ID, struct inode_security_struct,
+     selinux_secidx);

tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-     struct task_security_struct);
+     struct task_security_struct,
+     selinux_secidx);
isec->sclass = socket_type_to_security_class(family, type, protocol);
isec->sid = kern ? SECINITSID_KERNEL : tsec->sid;
isec->initialized = 1;
@@ -2966,9 +3043,11 @@ static int selinux_socket_bind(struct so
u32 sid, node_perm, addrlen;

tsec = security_get_value_type(&current->security,
-     SELINUX_LSM_ID, struct task_security_struct);
+     SELINUX_LSM_ID, struct task_security_struct,
+     selinux_secidx);
isec = security_get_value_type(&SOCK_INODE(sock)->i_security,
-     SELINUX_LSM_ID, struct inode_security_struct);
+     SELINUX_LSM_ID, struct inode_security_struct,
+     selinux_secidx);
```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

        if (family == PF_INET) {
            addr4 = (struct sockaddr_in *)address;
@@ -3047,7 +3126,8 @@ static int selinux_socket_connect(struct
    * If a TCP socket, check name_connect permission for the port.
    */
    isec = security_get_value_type(&SOCK_INODE(sock)->i_security,
-        SELINUX_LSM_ID, struct inode_security_struct);
+        SELINUX_LSM_ID, struct inode_security_struct,
+        selinux_secidx);
    if (isec->sclass == SECCLASS_TCP_SOCKET) {
        struct sock *sk = sock->sk;
        struct avc_audit_data ad;
@@ -3102,10 +3182,12 @@ static int selinux_socket_accept(struct
    return err;

    newisec = security_get_value_type(&SOCK_INODE(newsock)->i_security,
-        SELINUX_LSM_ID, struct inode_security_struct);
+        SELINUX_LSM_ID, struct inode_security_struct,
+        selinux_secidx);

    isec = security_get_value_type(&SOCK_INODE(sock)->i_security,
-        SELINUX_LSM_ID, struct inode_security_struct);
+        SELINUX_LSM_ID, struct inode_security_struct,
+        selinux_secidx);
    newisec->sclass = isec->sclass;
    newisec->sid = isec->sid;
    newisec->initialized = 1;
@@ -3162,9 +3244,11 @@ static int selinux_socket_unix_stream_co
    int err;

    isec = security_get_value_type(&SOCK_INODE(sock)->i_security,
-        SELINUX_LSM_ID, struct inode_security_struct);
+        SELINUX_LSM_ID, struct inode_security_struct,
+        selinux_secidx);
    other_isec = security_get_value_type(&SOCK_INODE(other)->i_security,
-        SELINUX_LSM_ID, struct inode_security_struct);
+        SELINUX_LSM_ID, struct inode_security_struct,
+        selinux_secidx);

    AVC_AUDIT_DATA_INIT(&ad,NET);
    ad.u.net.sk = other->sk;
@@ -3177,12 +3261,14 @@ static int selinux_socket_unix_stream_co

    /* connecting socket */
    ssec = security_get_value_type(&sock->sk->sk_security, SELINUX_LSM_ID,
-        struct sk_security_struct);
+        struct sk_security_struct,
+        selinux_secidx);
    ssec->peer_sid = other_isec->sid;

    /* server child socket */
    ssec = security_get_value_type(&newsk->sk_security, SELINUX_LSM_ID,
-        struct sk_security_struct);
+        struct sk_security_struct,
+        selinux_secidx);
    ssec->peer_sid = isec->sid;

    return 0;
@@ -3197,9 +3283,11 @@ static int selinux_socket_unix_may_send(
    int err;

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

    isec = security_get_value_type(&SOCK_INODE(sock)->i_security,
-       SELINUX_LSM_ID, struct inode_security_struct);
+       SELINUX_LSM_ID, struct inode_security_struct,
+       selinux_secidx);
    other_isec = security_get_value_type(&SOCK_INODE(other)->i_security,
-       SELINUX_LSM_ID, struct inode_security_struct);
+       SELINUX_LSM_ID, struct inode_security_struct,
+       selinux_secidx);

    AVC_AUDIT_DATA_INIT(&ad,NET);
    ad.u.net.sk = other->sk;
@@ -3240,7 +3328,8 @@ static int selinux_socket_sock_rcv_skb(s
        if (inode) {
            struct inode_security_struct *isec;
            isec = security_get_value_type(&inode->i_security,
-             SELINUX_LSM_ID, struct inode_security_struct);
+             SELINUX_LSM_ID, struct inode_security_struct,
+             selinux_secidx);
            sock_sid = isec->sid;
            sock_class = isec->sclass;
        }
@@ -3324,14 +3413,16 @@ static int selinux_socket_getpeersec(str
    struct inode_security_struct *isec;

    isec = security_get_value_type(&SOCK_INODE(sock)->i_security,
-       SELINUX_LSM_ID, struct inode_security_struct);
+       SELINUX_LSM_ID, struct inode_security_struct,
+       selinux_secidx);
    if (isec->sclass != SECCLASS_UNIX_STREAM_SOCKET) {
        err = -ENOPROTOOPT;
        goto out;
    }

    ssec = security_get_value_type(&sock->sk->sk_security, SELINUX_LSM_ID,
-       struct sk_security_struct);
+       struct sk_security_struct,
+       selinux_secidx);

    err = security_sid_to_context(ssec->peer_sid, &scontext, &scontext_len);
    if (err)
@@ -3373,7 +3464,8 @@ static int selinux_nlmsg_perm(struct soc
    struct inode_security_struct *isec;

    isec = security_get_value_type(&SOCK_INODE(sock)->i_security,
-       SELINUX_LSM_ID, struct inode_security_struct);
+       SELINUX_LSM_ID, struct inode_security_struct,
+       selinux_secidx);

    if (skb->len < NLMSG_SPACE(0)) {
        err = -EINVAL;
@@ -3440,7 +3532,8 @@ static unsigned int selinux_ip_postroute
    goto out;

    isec = security_get_value_type(&inode->i_security, SELINUX_LSM_ID,
-       struct inode_security_struct);
+       struct inode_security_struct,
+       selinux_secidx);

    switch (isec->sclass) {
    case SECCLASS_UDP_SOCKET:
@@ -3547,7 +3640,8 @@ static int selinux_netlink_send(struct s
    int err = 0;

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

tsec = security_get_value_type(&current->security,
-     SELINUX_LSM_ID, struct task_security_struct);
+     SELINUX_LSM_ID, struct task_security_struct,
+     selinux_secidx);

    avd.allowed = 0;
    avc_has_perm_noaudit(tsec->sid, tsec->sid,
@@ -3568,7 +3662,8 @@ static int ipc_alloc_security(struct tas
    struct ipc_security_struct *isec;

    tsec = security_get_value_type(&task->security, SELINUX_LSM_ID,
-     struct task_security_struct);
+     struct task_security_struct,
+     selinux_secidx);

    isec = kmalloc(sizeof(struct ipc_security_struct), GFP_KERNEL);
    if (!isec)
@@ -3582,7 +3677,8 @@ static int ipc_alloc_security(struct tas
    } else {
        isec->sid = SECINITSID_UNLABELED;
    }
-     security_set_value_type(&perm->security, SELINUX_LSM_ID, isec);
+     security_set_value_type(&perm->security, SELINUX_LSM_ID, isec,
+     selinux_secidx);

    return 0;
}
@@ -3592,7 +3688,7 @@ static void ipc_free_security(struct ker
    struct ipc_security_struct *isec;

    isec = security_del_value_type(&perm->security, SELINUX_LSM_ID,
-     struct ipc_security_struct);
+     struct ipc_security_struct, selinux_secidx);

    kfree(isec);
}
@@ -3608,7 +3704,8 @@ static int msg_msg_alloc_security(struct
    memset(msec, 0, sizeof(struct msg_security_struct));
    msec->msg = msg;
    msec->sid = SECINITSID_UNLABELED;
-     security_set_value_type(&msg->security, SELINUX_LSM_ID, msec);
+     security_set_value_type(&msg->security, SELINUX_LSM_ID, msec,
+     selinux_secidx);

    return 0;
}
@@ -3618,7 +3715,8 @@ static void msg_msg_free_security(struct
    struct msg_security_struct *msec;

    msec = security_del_value_type(&msg->security, SELINUX_LSM_ID,
-     struct msg_security_struct);
+     struct msg_security_struct,
+     selinux_secidx);

    kfree(msec);
}
@@ -3631,9 +3729,11 @@ static int ipc_has_perm(struct kern_ipc_
    struct avc_audit_data ad;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-     struct task_security_struct);

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```
+         struct task_security_struct,
+         selinux_secidx);
-     isec = security_get_value_type(&ipc_perms->security, SELINUX_LSM_ID,
+         struct ipc_security_struct);
+         struct ipc_security_struct,
+         selinux_secidx);

    AVC_AUDIT_DATA_INIT(&ad, IPC);
    ad.u.ipc_id = ipc_perms->key;
@@ -3664,9 +3764,11 @@ static int selinux_msg_queue_alloc_secur
        return rc;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
-     isec = security_get_value_type(&msq->q_perm.security, SELINUX_LSM_ID,
+         struct ipc_security_struct);
+         struct ipc_security_struct,
+         selinux_secidx);

    AVC_AUDIT_DATA_INIT(&ad, IPC);
    ad.u.ipc_id = msq->q_perm.key;
@@ -3692,9 +3794,11 @@ static int selinux_msg_queue_associate(s
    struct avc_audit_data ad;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
-     isec = security_get_value_type(&msq->q_perm.security, SELINUX_LSM_ID,
+         struct ipc_security_struct);
+         struct ipc_security_struct,
+         selinux_secidx);

    AVC_AUDIT_DATA_INIT(&ad, IPC);
    ad.u.ipc_id = msq->q_perm.key;
@@ -3740,11 +3844,14 @@ static int selinux_msg_queue_msgsnd(stru
    int rc;

    tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
-     isec = security_get_value_type(&msq->q_perm.security, SELINUX_LSM_ID,
+         struct ipc_security_struct);
+         struct ipc_security_struct,
+         selinux_secidx);
-     msec = security_get_value_type(&msg->security, SELINUX_LSM_ID,
+         struct msg_security_struct);
+         struct msg_security_struct,
+         selinux_secidx);

    /*
     * First time through, need to assign label to the message
@@ -3791,11 +3898,14 @@ static int selinux_msg_queue_msgrcv(stru
    int rc;

    tsec = security_get_value_type(&target->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

-     isec = security_get_value_type(&msg->q_perm.security, SELINUX_LSM_ID,
+     struct ipc_security_struct);
+     struct ipc_security_struct,
+     selinux_secidx);
-     msec = security_get_value_type(&msg->security, SELINUX_LSM_ID,
+     struct msg_security_struct);
+     struct msg_security_struct,
+     selinux_secidx);

-     AVC_AUDIT_DATA_INIT(&ad, IPC);
-     ad.u.ipc_id = msg->q_perm.key;
@@ -3821,9 +3931,11 @@ static int selinux_shm_alloc_security(st
-     return rc;

-     tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
+     struct task_security_struct);
+     struct task_security_struct,
+     selinux_secidx);
-     isec = security_get_value_type(&shp->shm_perm.security, SELINUX_LSM_ID,
+     struct ipc_security_struct);
+     struct ipc_security_struct,
+     selinux_secidx);

-     AVC_AUDIT_DATA_INIT(&ad, IPC);
-     ad.u.ipc_id = shp->shm_perm.key;
@@ -3849,9 +3961,11 @@ static int selinux_shm_associate(struct
-     struct avc_audit_data ad;

-     tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
+     struct task_security_struct);
+     struct task_security_struct,
+     selinux_secidx);
-     isec = security_get_value_type(&shp->shm_perm.security, SELINUX_LSM_ID,
+     struct ipc_security_struct);
+     struct ipc_security_struct,
+     selinux_secidx);

-     AVC_AUDIT_DATA_INIT(&ad, IPC);
-     ad.u.ipc_id = shp->shm_perm.key;
@@ -3919,9 +4033,11 @@ static int selinux_sem_alloc_security(st
-     return rc;

-     tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
+     struct task_security_struct);
+     struct task_security_struct,
+     selinux_secidx);
-     isec = security_get_value_type(&sma->sem_perm.security, SELINUX_LSM_ID,
+     struct ipc_security_struct);
+     struct ipc_security_struct,
+     selinux_secidx);

-     AVC_AUDIT_DATA_INIT(&ad, IPC);
-     ad.u.ipc_id = sma->sem_perm.key;
@@ -3947,9 +4063,11 @@ static int selinux_sem_associate(struct
-     struct avc_audit_data ad;

-     tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
+     struct task_security_struct);
+     struct task_security_struct,
+     selinux_secidx);
-     isec = security_get_value_type(&sma->sem_perm.security, SELINUX_LSM_ID,
+     struct ipc_security_struct);
```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

+         struct ipc_security_struct,
+         selinux_secidx);

        AVC_AUDIT_DATA_INIT(&ad, IPC);
        ad.u.ipc_id = sma->sem_perm.key;
@@ -4064,7 +4182,8 @@ static int selinux_getprocattr(struct ta
        return -ERANGE;

        tsec = security_get_value_type(&p->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);

        if (!strcmp(name, "current"))
            sid = tsec->sid;
@@ -4140,7 +4259,8 @@ static int selinux_setprocattr(struct ta
        checks and may_create for the file creation checks. The
        operation will then fail if the context is not permitted. */
        tsec = security_get_value_type(&p->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
        if (!strcmp(name, "exec"))
            tsec->exec_sid = sid;
        else if (!strcmp(name, "fscreate"))
@@ -4359,13 +4479,15 @@ static __init int selinux_init(void)
        if (task_alloc_security(current))
            panic("SELinux: Failed to initialize initial task.\n");
        tsec = security_get_value_type(&current->security, SELINUX_LSM_ID,
-         struct task_security_struct);
+         struct task_security_struct,
+         selinux_secidx);
        tsec->osid = tsec->sid = SECINITSID_KERNEL;

        avc_init();

-         if (register_security (&selinux_ops)) {
-             if (mod_reg_security( MY_NAME, &selinux_ops)) {
+         selinux_secidx = 1;
+         if (register_security (&selinux_ops, &selinux_secidx)) {
+             if (mod_reg_security( MY_NAME, &selinux_ops, &selinux_secidx)) {
                    printk(KERN_ERR "%s: Failed to register with primary LSM.\n",
                        __FUNCTION__);
                    panic("SELinux: Unable to register with kernel.\n");
@@ -4375,6 +4497,7 @@ static __init int selinux_init(void)
            }
            secondary = 1;
        }
+         printk(KERN_NOTICE "got selinux_secidx %d\n", selinux_secidx);

        if (selinux_enforcing) {
            printk(KERN_INFO "SELinux: Starting in enforcing mode\n");
Index: linux-2.6.12/security/selinux/selinuxfs.c
=====
--- linux-2.6.12.orig/security/selinux/selinuxfs.c      2005-08-01 20:00:51.000000000 -0500
+++ linux-2.6.12/security/selinux/selinuxfs.c      2005-08-01 20:28:09.000000000 -0500
@@ -35,6 +35,8 @@
#include "objsec.h"
#include "conditional.h"

+extern int selinux_secidx;
+

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```
unsigned int selinux_checkreqprot = CONFIG_SECURITY_SELINUX_CHECKREQPROT_VALUE;

static int __init checkreqprot_setup(char *str)
@@ -61,7 +63,8 @@ static int task_has_security(struct task
    struct task_security_struct *tsec;

    tsec = security_get_value_type(&tsk->security, SELINUX_LSM_ID,
-        struct task_security_struct);
+        struct task_security_struct,
+        selinux_secidx);
    if (!tsec)
        return -EACCES;

@@ -985,7 +988,8 @@ static int sel_make_bools(void)
    goto err;
}
    isec = security_get_value_type(&inode->i_security,
-        SELINUX_LSM_ID, struct inode_security_struct);
+        SELINUX_LSM_ID, struct inode_security_struct,
+        selinux_secidx);
    if ((ret = security_genfs_sid("selinuxfs", page, SECCLASS_FILE, &sid)))
        goto err;
    isec->sid = sid;
@@ -1271,7 +1275,8 @@ static int sel_fill_super(struct super_b
    if (!inode)
        goto out;
    isec = security_get_value_type(&inode->i_security, SELINUX_LSM_ID,
-        struct inode_security_struct);
+        struct inode_security_struct,
+        selinux_secidx);
    isec->sid = SECINITSID_DEVNULL;
    isec->sclass = SECCLASS_CHR_FILE;
    isec->initialized = 1;
Index: linux-2.6.12/security/seclvl.c
=====
--- linux-2.6.12.orig/security/seclvl.c 2005-08-01 20:00:51.000000000 -0500
+++ linux-2.6.12/security/seclvl.c      2005-08-01 20:28:28.000000000 -0500
@@ -38,6 +38,8 @@
    #define SHA1_DIGEST_SIZE 20
    #define SECLVL_LSM_ID 0xF45

+static int seclvl_secidx;
+
+/**
+ * Module parameter that defines the initial secure level.
+ *
@@ -506,7 +508,7 @@ static void seclvl_inode_free(struct ino
    struct seclvl_i_sec *isec;

    isec = security_del_value_type(&inode->i_security, SECLVL_LSM_ID,
-        struct seclvl_i_sec);
+        struct seclvl_i_sec, seclvl_secidx);
    if (isec) {
        spin_lock(&seclvl_ichain_lock);
        list_del(&isec->chain);
@@ -546,7 +548,8 @@ static void seclvl_bd_release(struct ino

    if (inode && S_ISBLK(inode->i_mode)) {
        isec = security_get_value_type(&inode->i_security,
-        SECLVL_LSM_ID, struct seclvl_i_sec);
+        SECLVL_LSM_ID, struct seclvl_i_sec,
+        seclvl_secidx);
```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

        if (!isec)
            return;
        spin_lock(&isec->spinlock);
@@ -574,12 +577,12 @@ seclvl_inode_get_or_alloc(struct inode *
    struct seclvl_i_sec *isec;

    isec = security_get_value_type(&inode->i_security,
-        SECLVL_LSM_ID, struct seclvl_i_sec);
+        SECLVL_LSM_ID, struct seclvl_i_sec, seclvl_secidx);
    if (isec)
        return isec;
    spin_lock(&seclvl_ichain_lock);
    isec = security_get_value_type(&inode->i_security,
-        SECLVL_LSM_ID, struct seclvl_i_sec);
+        SECLVL_LSM_ID, struct seclvl_i_sec, seclvl_secidx);
    if (isec)
        goto out;
    isec = kmalloc(sizeof(struct seclvl_i_sec), GFP_KERNEL);
@@ -588,7 +591,8 @@ seclvl_inode_get_or_alloc(struct inode *
    spin_lock_init(&isec->spinlock);
    INIT_LIST_HEAD(&isec->chain);
    list_add(&isec->chain, &seclvl_ichain);
-    security_add_value_type(&inode->i_security, SECLVL_LSM_ID, isec);
+    security_add_value_type(&inode->i_security, SECLVL_LSM_ID, isec,
+        seclvl_secidx);

    out:
        spin_unlock(&seclvl_ichain_lock);
@@ -781,12 +785,13 @@ static int __init seclvl_init(void)
    goto exit;
}
/* register ourselves with the security framework */
- if (register_security(&seclvl_ops)) {
+ seclvl_secidx = 1; /* yes we want to use a security field */
+ if (register_security(&seclvl_ops, &seclvl_secidx)) {
    seclvl_printk(0, KERN_ERR,
        "seclvl: Failure registering with the "
        "kernel.\n");
    /* try registering with primary module */
-    rc = mod_reg_security(MY_NAME, &seclvl_ops);
+    rc = mod_reg_security(MY_NAME, &seclvl_ops, &seclvl_secidx);
    if (rc) {
        seclvl_printk(0, KERN_ERR, "seclvl: Failure "
            "registering with primary security "
@@ -795,6 +800,7 @@ static int __init seclvl_init(void)
    } /* if primary module registered */
    secondary = 1;
} /* if we registered ourselves with the security framework */
+ printk(KERN_NOTICE "got seclvl_secidx %d\n", seclvl_secidx);
    if ((rc = doSysfsRegistrations())) {
        seclvl_printk(0, KERN_ERR, "Error registering with sysfs\n");
        goto exit;
@@ -821,7 +827,7 @@ static void free_ichain(void)
    struct seclvl_i_sec *isec, *n;

    list_for_each_entry_safe(isec, n, &seclvl_ichain, chain) {
-        security_unlink_value(&isec->lsm_list.list);
+        security_unlink_value(&isec->lsm_list.list, seclvl_secidx);
    }

    synchronize_rcu();
Index: linux-2.6.12/security/capability.c

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

```

=====
--- linux-2.6.12.orig/security/capability.c      2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/security/capability.c      2005-08-01 20:28:42.000000000 -0500
@@ -66,9 +66,9 @@ static int __init capability_init (void)
     return 0;
}
/* register ourselves with the security framework */
- if (register_security (&capability_ops)) {
+ if (register_security (&capability_ops, NULL)) {
    /* try registering with primary module */
- if (mod_reg_security (MY_NAME, &capability_ops)) {
+ if (mod_reg_security (MY_NAME, &capability_ops, NULL)) {
    printk (KERN_INFO "Failure registering capabilities "
            "with primary security module.\n");
    return -EINVAL;
Index: linux-2.6.12/security/root_plug.c
=====
--- linux-2.6.12.orig/security/root_plug.c      2005-08-01 20:00:50.000000000 -0500
+++ linux-2.6.12/security/root_plug.c      2005-08-01 20:28:53.000000000 -0500
@@ -104,11 +104,11 @@ static struct security_operations rootpl
 static int __init rootplug_init (void)
 {
    /* register ourselves with the security framework */
- if (register_security (&rootplug_security_ops)) {
+ if (register_security (&rootplug_security_ops, NULL)) {
    printk (KERN_INFO
            "Failure registering Root Plug module with the kernel\n");
    /* try registering with primary module */
- if (mod_reg_security (MY_NAME, &rootplug_security_ops)) {
+ if (mod_reg_security (MY_NAME, &rootplug_security_ops, NULL)) {
    printk (KERN_INFO "Failure registering Root Plug "
            " module with primary security module.\n");
    return -EINVAL;
Index: linux-2.6.12/security/cap_stack.c
=====
--- linux-2.6.12.orig/security/cap_stack.c      2005-08-01 20:00:51.000000000 -0500
+++ linux-2.6.12/security/cap_stack.c      2005-08-01 20:33:21.000000000 -0500
@@ -62,9 +62,9 @@ static int __init capability_init (void)
     return 0;
}
/* register ourselves with the security framework */
- if (register_security (&capability_ops)) {
+ if (register_security (&capability_ops, NULL)) {
    /* try registering with primary module */
- if (mod_reg_security (MY_NAME, &capability_ops)) {
+ if (mod_reg_security (MY_NAME, &capability_ops, NULL)) {
    printk (KERN_INFO "Failure registering capabilities "
            "with primary security module.\n");
    return -EINVAL;
Index: linux-2.6.12/security/stacker.c
=====
--- linux-2.6.12.orig/security/stacker.c      2005-08-01 20:00:51.000000000 -0500
+++ linux-2.6.12/security/stacker.c      2005-08-01 20:29:42.000000000 -0500
@@ -1629,7 +1629,7 @@ static int __init stacker_init (void)
    INIT_LIST_HEAD(&default_module.lsm_list);
    list_add_tail(&default_module.lsm_list, &stacked_modules);

- if (register_security (&stacker_ops)) {
+ if (register_security (&stacker_ops, NULL)) {
    /*
     * stacking stacker is just a stupid idea, so don't ask
     * the current module to load us.

```

Linux-Kernel: [PATCH] Stacker – single-use static slots

Index: linux-2.6.12/include/linux/security-stack.h

=====

--- linux-2.6.12.orig/include/linux/security-stack.h 2005-08-01 20:00:51.000000000 -0500

+++ linux-2.6.12/include/linux/security-stack.h 2005-08-01 20:34:18.000000000 -0500

```
@@ -19,7 +19,7 @@ extern fastcall struct security_list *se
    extern fastcall struct security_list *security_add_value(
        struct hlist_head *head,
        int security_id, struct security_list *obj_node);
-extern int security_unlink_value(struct hlist_node *n);
+extern int security_unlink_value(struct hlist_node *n, int idx);
    extern fastcall struct security_list *security_del_value(
        struct hlist_head *head,
        int security_id);
```

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>