NAME

avc_init − legacy userspace SELinux AVC setup

SYNOPSIS

#include <selinux/selinux.h>
#include <selinux/avc.h>

```
int avc_init(const char *msgprefix,
        const struct avc_memory_callback *mem_callbacks,
        const struct avc_log_callback *log_callbacks,
        const struct avc_thread_callback *thread_callbacks,
        const struct avc_lock_callback *lock_callbacks);
```

DESCRIPTION

avc_init() is deprecated; please use **avc_open**(3) in conjunction with **selinux_set_callback**(3) in all new code.

avc_init() initializes the userspace AVC and must be called before any other AVC operation can be performed. A non-NULL*msgpr efix* will be prepended to all audit messages produced by the userspace AVC. The default is ‘uavc’. The remaining arguments, if non-NULL, specify callbacks to be used by the userspace AVC.

CALLBACKS

The userspace AVC can be directed how to perform memory allocation, logging, thread creation, and locking via callback functions passed to **avc_init**(). The purpose of this functionality is to allow the userspace AVC to be smoothly integrated into existing userspace object managers.

Use an **avc_memory_callback** structure to specify alternate functions for dynamic memory allocation.

```
struct avc_memory_callback {
    void    *(*func_malloc)(size_t size);
    void    (*func_free)(void *ptr);
};
```

The two fields of the structure should be pointers to functions which behave as **malloc**(3) and **free**(3), which are used by default.

Use an **avc_log_callback** structure to specify alternate functions for logging.

```
struct avc_log_callback {
    void    (*func_log)(const char *fmt, ...);
    void    (*func_audit)(void *auditdata,
                          security_class_t class,
                          char *msgbuf, size_t msgbufsize);
};
```

The **func_log** callback should accept a **printf**(3) style format and arguments and log them as desired. The default behavior prints the message on the standard error. The**func_audit** callback should interpret the *auditdata* parameter for the given *class*, printing a human-readable interpretation to *msgbuf* using no more than *msgbufsize* characters. The default behavior is to ignore *auditdata*.

Use an **avc_thread_callback** structure to specify functions for starting and manipulating threads.

```
struct avc_thread_callback {
    void    *(*func_create_thread)(void (*run)(void));
    void    (*func_stop_thread)(void *thread);
};
```

The **func_create_thread** callback should create a new thread and return a pointer which references it. The thread should execute the *run* argument, which does not return under normal conditions. The **func_stop_thread** callback should cancel the running thread referenced by *thread*. By default, threading is not used; see **KERNEL STATUS PAGE** and **NETLINK NOTIFICATION** below.

Use an **avc_lock_callback** structure to specify functions to create, obtain, and release locks for use by threads.

```
struct avc_lock_callback {
      void    *(*func_alloc_lock)(void);
      void    (*func_get_lock)(void *lock);
      void    (*func_release_lock)(void *lock);
      void    (*func_free_lock)(void *lock);
};
```

The **func_alloc_lock** callback should create a new lock, returning a pointer which references it. The **func_get_lock** callback should obtain *lock*, blocking if necessary. The**func_r elease_lock** callback should release *lock*. The**func_fr ee_lock** callback should destroy *lock*, freeing any resources associated with it. The default behavior is not to perform any locking. Note that undefined behavior may result if threading is used without appropriate locking.

KERNEL STATUS PAGE

Linux kernel version 2.6.37 supports the SELinux kernel status page, enabling userspace applications to **mmap**(2) SELinux status state in read-only mode to avoid system calls during the cache hit code path.

avc_init() calls **selinux_status_open**(3) to initialize the selinux status state. If successfully initialized, the userspace AVC will default to single-threaded mode and ignore the **func_create_thread** and **func_stop_thread** callbacks. All callbacks set via **selinux_set_callback**(3) will still be honored.

avc_has_perm(3) and **selinux_check_access**(3) both check for status updates through calls to **selinux_status_updated**(3) at the start of each permission query and take the appropriate action.

Two status types are currently implemented. **setenf orce** events will change the effective enforcing state used within the AVC, and **policyload** events will result in a cache flush.

NETLINK NOTIFICATION

In the event that the kernel status page is not successfully **mmap**(2)'ed the AVC will default to the netlink fallback mechanism, which opens a netlink socket for receiving status updates. **setenf orce** and **policyload** events will have the same results as for the status page implementation, but all status update checks will now require a system call.

By default, **avc_open**(3) does not set threading or locking callbacks. In the fallback case, the userspace AVC checks for new netlink messages at the start of each permission query. If threading and locking callbacks are passed to **avc_init**(), a dedicated thread will be started to listen on the netlink socket. This may increase performance in the absence of the status page and will ensure that log messages are generated immediately rather than at the time of the next permission query.

RETURN VALUE

Functions with a return value return zero on success. On error, −1 is returned and *errno* is set appropriately.

NOTES

The *msgprefix* argument to **avc_init**() currently has a length limit of 15 characters and will be truncated if necessary.

If a provided **func_malloc** callback does not set *errno* appropriately on error, userspace AVC calls may exhibit the same behavior.

If a netlink thread has been created and an error occurs on the socket (such as an access error), the thread may terminate and cause the userspace AVC to return **EINVAL** on all further permission checks until **avc_destroy** is called.

AUTHOR

Eamon Walsh <ewalsh@tycho.nsa.gov>

SEE ALSO

avc_open(3), **selinux_status_open**(3), **selinux_status_updated**(3), **selinux_set_callback**(3), **selinux**(8)