## NAME

selinux_status_open, selinux_status_close, selinux_status_updated, selinux_status_getenforce, selinux_status_policyload and selinux_status_deny_unknown − reference the SELinux kernel status without invocation of system calls

## SYNOPSIS

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

**int selinux_status_open(int** *fallback*);

**void selinux_status_close(void);**

**int selinux_status_updated(void);**

**int selinux_status_getenforce(void);**

**int selinux_status_policyload(void);**

**int selinux_status_deny_unknown(void);**

## DESCRIPTION

Linux 2.6.37 or later provides a SELinux kernel status page; being mostly placed on */sys/fs/selinux/status* entry. It enables userspace applications to mmap this page with read-only mode, then it informs some status without system call invocations.

In some cases that a userspace application tries to apply heavy frequent access control; such as row-level security in databases, it will face unignorable cost to communicate with kernel space to check invalidation of userspace avc.

These functions provides applications a way to know some kernel events without system-call invocation or worker thread for monitoring.

**selinux_status_open**() tries to **open**(2) */sys/fs/selinux/status* and **mmap**(2) it in read-only mode. The file-descriptor and pointer to the page shall be stored internally; Don't touch them directly. Set 1 on the*fallback* argument to handle a case of older kernels without kernel status page support. In this case, this function tries to open a netlink socket using **avc_netlink_open**(3) and overwrite corresponding callbacks (setenforce and policyload). Thus, we need to pay attention to the interaction with these interfaces, when fallback mode is enabled.

**selinux_status_close**() unmap the kernel status page and close its file descriptor, or close the netlink socket if fallbacked.

**selinux_status_updated**() processes status update events. There are two kinds of status updates. **setenforce** events will change the effective enforcing state used within the AVC, and **policyload** events will result in a cache flush.

This function returns 0 if there have been no updates since the last call, 1 if there have been updates since the last call, or −1 on error.

**selinux_status_getenforce**() returns 0 if SELinux is running in permissive mode, 1 if enforcing mode, or −1 on error. Same as**security_getenf orce**(3) except with or without system call invocation.

**selinux_status_policyload**() returns times of policy reloaded on the running system, or −1 on error. Note that it is not a reliable value on fallback-mode until it receive the first event message via netlink socket. Thus, don't use this value to know actual times of policy reloaded.

**selinux_status_deny_unknown**() returns 0 if SELinux treats policy queries on undefined object classes or permissions as being allowed, 1 if such queries are denied, or −1 on error.

Also note that these interfaces are not thread-safe, so you have to protect them from concurrent calls using exclusive locks when multiple threads are performing.

## RETURN VALUE

**selinux_status_open**() returns 0 or 1 on success. 1 means we are ready to use these interfaces, but netlink socket was opened as fallback instead of the kernel status page. On error, −1 shall be returned.

Any other functions with a return value shall return its characteristic value as described above, or −1 on errors.

## SEE ALSO

**mmap**(2), **avc_netlink_open**(3), **security_getenforce**(3), **security_deny_unknown**(3)