NAME

security_load_policy − load a new SELinux policy

SYNOPSIS

#include <selinux/selinux.h>

int security_load_policy(void ****data*, size_t** *len***);**

int selinux_mkload_policy(int *preservebools***);**

int selinux_init_load_policy(int ****enforce*);**

DESCRIPTION

security_load_policy() loads a new policy, returns 0 for success and −1 for error.

selinux_mkload_policy() makes a policy image and loads it. This function provides a higher level interface for loading policy than **security_load_policy**(), internally determining the right policy version, locating and opening the policy file, mapping it into memory, manipulating it as needed for current boolean settings and/or local definitions, and then calling security_load_policy to load it. *preservebools* is a boolean flag indicating whether current policy boolean values should be preserved into the new policy (if 1) or reset to the saved policy settings (if 0). The former case is the default for policy reloads, while the latter case is an option for policy reloads but is primarily used for the initial policy load. **selinux_init_load_policy**() performs the initial policy load. This function determines the desired enforcing mode, sets the *enforce* argument accordingly for the caller to use, sets the SELinux kernel enforcing status to match it, and loads the policy. It also internally handles the initial selinuxfs mount required to perform these actions.

It should also be noted that after the initial policy load, the SELinux kernel code cannot anymore be disabled and the selinuxfs cannot be unmounted using a call to **security_disable**(3). Therefore, after the initial policy load, the only operational changes are those permitted by **security_setenforce**(3) (i.e. eventually setting the framework in permissive mode rather than in enforcing one).

RETURN VALUE

Returns zero on success or −1 on error.

AUTHOR

This manual page has been written by Guido Trentalancia <guido@trentalancia.com>

SEE ALSO

selinux(8), **security_disable**(3), **setenforce**(8)