## NAME

selinux_set_mapping − establish dynamic object class and permission mapping

## SYNOPSIS

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

```
struct security_class_mapping {
        const char *name;
        const char *perms[];
};
```

**int selinux_set_mapping(struct security_class_mapping *_map_);**

## DESCRIPTION

**selinux_set_mapping**() establishes a mapping from a user-provided ordering of object classes and permissions to the numbers actually used by the loaded system policy. If using this function, applications should also set a **SELINUX_CB_POLICYLOAD** callback via **selinux_set_callback(3)** that calls this function again upon a policy reload to re-create the mapping in case the class or permission values change in the new policy. Generally it is preferred to instead use**selinux_check_access(3)** instead of **a vc_has_perm(3)** or **security_compute_av(3)** and not use this function at all.

After the mapping is established, all libselinux functions that operate on class and permission values take the user-provided numbers, which are determined as follows:

The *map* argument consists of an array of **security_class_mapping** structures, which must be terminated by a structure having a NULL name field. Except for this last structure, the *name* field should refer to the string name of an object class, and the corresponding *perms* field should refer to an array of permission bit names terminated by a NULL string.

The object classes named in the mapping and the bit indexes of each set of permission bits named in the mapping are numbered in order starting from 1. These numbers are the values that should be passed to subsequent libselinux calls.

## RETURN VALUE

Zero is returned on success. On error, −1 is returned and *errno* is set appropriately.

## ERRORS

**EINVAL**

One of the class or permission names requested in the mapping is not present in the loaded policy.

**ENOMEM**

An attempt to allocate memory failed.

## EXAMPLE

```
struct security_class_mapping map[] = {
      { "file", { "create", "unlink", "read", "write", NULL } },
      { "socket", { "bind", NULL } },
      { "process", { "signal", NULL } },
      { NULL }
};

if (selinux_set_mapping(map) < 0)
      exit(1);
```

In this example, after the call has succeeded, classes **file**, **socket**, and **process** will be identified by 1, 2 and 3, respectively. Permissions*cr eate*, *unlink*, *read*, and *write* (for the **file** class) will be identified by 1, 2, 4, and 8 respectively. Classes and permissions not listed in the mapping cannot be used.

## AUTHOR

Originally Eamon Walsh. Updated by Stephen Smalley <sds@tycho.nsa.gov>

## SEE ALSO

**selinux_check_access**(3), **selinux_set_callback**(3), **avc_has_perm**(3), **selinux**(8)