NAME

avc_has_perm, avc_has_perm_noaudit, avc_audit, avc_entry_ref_init − obtain and audit SELinux access decisions

SYNOPSIS

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

void avc_entry_ref_init(struct avc_entry_ref ***_aeref_);**

int avc_has_perm(security_id_t _ssid_**, security_id_t** _tsid_**,**
security_class_t _tclass_**, access_vector_t** _requested_**,**
struct avc_entry_ref ***_aeref_, void ***_auditdata_**);**

int avc_has_perm_noaudit(security_id_t _ssid_**, security_id_t** _tsid_**,**
security_class_t _tclass_**, access_vector_t** _requested_**,**
struct avc_entry_ref ***_aeref_, struct av_decision ***_avd_**);**

void avc_audit(security_id_t _ssid_**, security_id_t** _tsid_**,**
security_class_t _tclass_**, access_vector_t** _requested_**,**
struct av_decision ***_avd_, int** _result_**, void ***_auditdata_**);**

DESCRIPTION

Direct use of these functions is generally discouraged in favor of the higher level interface **selinux_check_access(3)** since the latter automatically handles the dynamic mapping of class and permission names to their policy values and proper handling of allow_unknown.

When using any of the functions that take policy integer values for classes or permissions as inputs, use **string_to_security_class(3)** and **string_to_av_perm(3)** to map the class and permission names to their policy values. These values may change across a policy reload, so they should be re-acquired on every use or using a **SELINUX_CB_POLICYLOAD** callback set via **selinux_set_callback(3).**

An alternative approach is to use **selinux_set_mapping(3)** to create a mapping from class and permission index values used by the application to the policy values, thereby allowing the application to pass its own fixed constants for the classes and permissions to these functions and internally mapping them on demand. However, this also requires setting up a callback as above to address policy reloads.

avc_entry_ref_init() initializes an **avc_entry_ref** structure; see **ENTRY REFERENCES** below. This function may be implemented as a macro.

avc_has_perm() checks whether the _requested_ permissions are granted for subject SID _ssid_ and target SID _tsid_, interpreting the permissions based on _tclass_ and updating _aeref_, if non-NULL, to refer to a cache entry with the resulting decision. The granting or denial of permissions is audited in accordance with the policy. The_auditdata_ parameter is for supplemental auditing; see **a vc_audit**() below.

avc_has_perm_noaudit() behaves as **avc_has_perm**() without producing an audit message. The access decision is returned in _avd_ and can be passed to **avc_audit**() explicitly.

avc_audit() produces an audit message for the access query represented by _ssid_, _tsid_, _tclass_, and _requested_, with a decision represented by _avd_. Pass the value returned by **avc_has_perm_noaudit**() as _result_. The_auditdata_ parameter is passed to the user -supplied **func_audit** callback and can be used to add supplemental information to the audit message; see **avc_init**(3).

ENTRY REFERENCES

Entry references can be used to speed cache performance for repeated queries on the same subject and target. The userspace AVC will check the _aeref_ argument, if supplied, before searching the cache on a permission query. After a query is performed,*aer ef* will be updated to reference the cache entry for that query. A subsequent query on the same subject and target will then have the decision at hand without having to walk the cache.

After declaring an **avc_entry_ref** structure, use **avc_entry_ref_init**() to initialize it before passing it to **avc_has_perm**() or **avc_has_perm_noaudit**() for the first time. Using an uninitialized structure will produce undefined behavior.

RETURN VALUE

If requested permissions are granted, zero is returned. If requested permissions are denied or an error occurred, −1 is returned and *errno* is set appropriately.

In permissive mode, zero will be returned and *errno* unchanged even if permissions were denied. **avc_has_perm**() will still produce an audit message in this case.

ERRORS

EACCES
A requested permission was denied.

EINVAL
The *tclass* and/or the security contexts referenced by *ssid* and *tsid* are not recognized by the currently loaded policy.

ENOMEM
An attempt to allocate memory failed.

NOTES

Internal errors encountered by the userspace AVC may cause certain values of *errno* to be returned unexpectedly. For example, netlink socket errors may produce **EACCES** or **EINVAL**. Make sure that userspace object managers are granted appropriate access to netlink by the policy.

AUTHOR

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

SEE ALSO

selinux_check_access(3), string_to_security_class(3), string_to_av_perm(3), selinux_set_callback(3), selinux_set_mapping(3), avc_init(3), **avc_context_to_sid**(3), **avc_cache_stats**(3), **avc_add_callback**(3), **security_compute_av**(3), **selinux**(8)