NAME

cap_iab_init, cap_iab_dup, cap_iab_get_proc, cap_iab_get_pid, cap_iab_set_proc, cap_iab_to_text, cap_iab_from_text, cap_iab_get_vector, cap_iab_compare, cap_iab_set_vector, cap_iab_fill, cap_proc_root − inheritable IAB tuple support functions

SYNOPSIS

```
#include <sys/capability.h>

cap_iab_t cap_iab_init(void);
cap_iab_t cap_iab_dup(cap_iab_t iab);
cap_iab_t cap_iab_get_proc(void);
cap_iab_t cap_iab_get_pid(pid_t pid);
int cap_iab_set_proc(cap_iab_t iab);
char *cap_iab_to_text(cap_iab_t iab);
cap_iab_t cap_iab_from_text(const char *text);
cap_flag_value_t cap_iab_get_vector(cap_iab_t iab, cap_iab_vector_t vec,
    cap_value_t val);
int cap_iab_compare(cap_iab_t a, cap_iab_t b);
int cap_iab_set_vector(cap_iab_t iab, cap_iab_vector_t vec, cap_value_t val,
    cap_flag_value_t enable);
int cap_iab_fill(cap_iab_t iab, cap_iab_vector_t vec,
    cap_t set, cap_flag_t flag);
char *cap_proc_root(const char *root);
```

Link with *−lcap*.

DESCRIPTION

The functions defined in this man page concern the three naively inheritable process capability vectors: Inh, Amb and Bound. This *IAB* 3-tuple of capability vectors, captured in type *cap_iab_t* combine to pass capabilities from one process to another through **execve**(2) system calls. The convolution rules using the IAB tuple are a fail over set of rules when the executed file has no configured *file-capabilities*.

There are some constraints enforced by the kernel with respect to the three components of an IAB tuple and the Permitted process capability flag. They are: the Inh vector (**CAP_IAB_INH**) is entirely equal to the process Inheritable flag at all times; the Amb vector (**CAP_IAB_AMB**) contains no more capability values than the intersection of the Inh vector and the Permitted flag for the process; and the Bound (or *blocked*) vector (**CAP_IAB_BOUND**) is the twos-complement of the process bounding vector.

In some environments, it is considered desirable to *naively* inherit capabilities. That is pass capabilities, independent of the status of the executed binary, from parent to child through **exec*** system calls. The surviving capabilities become the Permitted flag for the post-exec process. This method of inheritance differs significantly from the handshake inheritance between a pre-exec* process and a file-capability bestowed executable of the traditional (POSIX.1e) capability mechanism.

The convolution rules for IAB style inheritance are: I'=I; A'=A&I; P'=A&I&P. Where P etc are the pre-exec values and P' etc are the post-exec values.

With an understanding of these convolution rules, we can explain how **libcap**(3) support for the IAB tuple is managed: the IAB API.

cap_iab_init() returns an empty IAB value. That is a *mostly-harmless* tuple. It will not block any Permitted file capabilities through exec, but it won't bestow any either. The returned *cap_iab_t* should be freed with **cap_free**(3).

cap_iab_dup() returns a copy of the specified IAB value. The returned cap_iab_t should be freed with **cap_free**(3).

cap_iab_get_proc() returns a copy of the IAB value for the current process. The returned cap_iab_t should be freed with **cap_free**(3).

cap_iab_get_pid() returns a copy of the IAB value for the specified process. The returned cap_iab_t should be freed with **cap_free**(3). This function defaults to searching **/proc/**<PID>**/status** for the IAB information, but that location can be overridden using the **cap_proc_root**() function.

cap_iab_set_proc() can be used to set the IAB value carried by the current process. Such a setting will fail if the process is insufficiently capable. To raise I bits not present in the P flag, or newly apply B masking, requires **CAP_SETPCAP** is present in the Permitted flag (the function will temporarily raise it in the Effective flag if needed). In all cases, raised A values must be present in the intersection of Permitted and Inh values.

cap_iab_to_text() will convert an IAB tuple to a canonical IAB text representation. The representation is slightly redundant but **libcap**(3)**will**try**to**generate as short a representation as it is able.

cap_iab_from_text() generates an IAB tuple from a text string (likely generated by the previous function). The returned IAB tuple should be freed with **cap_free**().

cap_iab_get_vector() can be used to determine the specific capability value of an IAB vector.

cap_iab_compare() can be used to compare two cap_iab_t tuples. When the return value is non-zero, the macro **CAP_IAB_DIFFERS**(*status*, *vector*) evaluates to non-zero if the returned status differs in its *vector* components.

cap_iab_set_vector() can be used to set a specific vector value to the enable setting.

cap_iab_fill() can be used to wholesale copy a *cap_t* flag value into the vec vector of the IAB tuple. Copying into Amb (**CAP_IAB_AMB**) in this way may implicitly raise Inh (**CAP_IAB_INH**) values in the IAB tuple. Similarly copying into the Inh vector may implicitly lower Amb values that are not present in the resulting Inh vector.

cap_proc_root() can be used to determine the current location queried by **cap_iab_get_pid**(). Returned values should be released with **cap_free**(3). If the argument to **cap_proc_root**() is not **NULL**, a copy of it will become the replacement for **/proc**. Note, this function is*not* thread safe with respect to concurrent calls to **cap_iab_get_pid**().

ERRORS

The functions returning *cap_iab_t* values or allocated memory in the form of a string return NULL on error.

Integer return values are -1 on error and 0 on success.

In the case of error consult *errno*.

NOTES

Unlike the traditional *cap_t* capability set, the IAB tuple, taken together, is incompatible with filesystem capabilities created via tools like **setcap**(8). That is, the Amb vector of the IAB tuple is rendered moot when an executable with a file capability is executed.

Further, there are libcap **cap_mode**(3)s that render the Amb vector and its method of process inheritance disabled.

HISTORY

The IAB format for inheritable variants of capabilities was first developed as the configuration syntax for the *pam_cap.so* Linux-PAM module in libcap-2.29. It was introduced to extend the *simple* comma separated list of process Inheritable capabilities, that the module could besow on an authenticated process tree, to include enforced limits on the Bounding vector and introduce support for the Amibient vector of capability bits.

While the Inheritable and Bounding vectors were anticipated by the POSIX.1e draft that introduced capabilities, the Ambient vector is a Linux invention, and incompatible with the POSIX.1e file capability model. As such, it was felt that trying to meld together all of the 5 capability vectors into one text representation was not going to work. Instead the *pam_cap.so* config syntax was generalized into a whole set of libcap functions for bundling together all three naively inheritable capabilities: the IAB tuple. The support for this debuted in libcap-2.33.

REPORTING BUGS

Please report bugs via:

https://bugzilla.kernel.org/buglist.cgi?component=libcap&list_id=1090757

SEE ALSO

libcap(3), **cap_launch**(3), **cap_init**(3), **capabilities**(7) and **errno**(3).