#### NAME

getaudit, getaudit\_addr - retrieve audit session state

#### **SYNOPSIS**

#include <bsm/audit.h>

int

**getaudit**(*auditinfo* t \**auditinfo*);

int

getaudit\_addr(auditinfo\_addr\_t \*auditinfo\_addr, u\_int length);

## DESCRIPTION

The getaudit() system call retrieves the active audit session state for the current process via the auditinfo\_t pointed to by auditinfo. The getaudit\_addr() system call retrieves extended state via *auditinfo\_addr* and *length*.

The *auditinfo\_t* data structure is defined as follows:

```
struct auditinfo {
```

	au_id_t	ai_auid;	/* Audit user ID */
	au_mask_t	ai_mask;	/* Audit masks */
	au_tid_t	ai_termid;	/* Terminal ID */
	au_asid_t	ai_asid;	/* Audit session ID */
	au_asflgs_t	ai_flags;	/* Audit session flags. */
};			
typedef	struct auditint	fo auditii	nfo_t;

The *ai\_auid* variable contains the audit identifier which is recorded in the audit log for each event the process caused.

The *au\_mask\_t* data structure defines the bit mask for auditing successful and failed events out of the predefined list of event classes. It is defined as follows:

```
struct au_mask {
         unsigned int am_success; /* success bits */
         unsigned int am_failure; /* failure bits */
};
typedef struct au_mask_au_mask_t;
```

The *au\_termid\_t* data structure defines the Terminal ID recorded with every event caused by the process. It is defined as follows:

The *ai\_asid* variable contains the audit session ID which is recorded with every event caused by the process.

The **getaudit\_addr**() system call uses the expanded *auditinfo\_addr\_t* data structure and supports Terminal IDs with larger addresses such as those used in IP version 6. It is defined as follows:

```
struct auditinfo_addr {
    au_id_t ai_auid; /* Audit user ID. */
    au_mask_t ai_mask; /* Audit masks. */
    au_tid_addr_t ai_termid; /* Terminal ID. */
    au_asid_t ai_asid; /* Audit session ID. */
};
```

typedef struct auditinfo\_addr auditinfo\_addr\_t;

The *au\_tid\_addr\_t* data structure which includes a larger address storage field and an additional field with the type of address stored:

```
struct au_tid_addr {
    dev_t at_port;
    u_int32_t at_type;
    u_int32_t at_addr[4];
};
typedef struct au_tid_addr au_tid_addr_t;
```

These system calls require an appropriate privilege to complete.

## **RETURN VALUES**

The **getaudit**() and **getaudit\_addr**() functions return the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

## ERRORS

# The **getaudit**() function will fail if:

[EFAULT]	A failure occurred while data transferred to or from the kernel failed.
[EINVAL]	Illegal argument was passed by a system call.
[EPERM]	The process does not have sufficient permission to complete the operation.
[EOVERFLOW]	The <i>length</i> argument indicates an overflow condition will occur.
[E2BIG]	The address is too big and, therefore, <b>getaudit_addr</b> () should be used instead.

## SEE ALSO

audit(2), auditon(2), getauid(2), setaudit(2), setauid(2), libbsm(3)

# HISTORY

The OpenBSM implementation was created by McAfee Research, the security division of McAfee Inc., under contract to Apple Computer Inc. in 2004. It was subsequently adopted by the TrustedBSD Project as the foundation for the OpenBSM distribution.

# AUTHORS

This software was created by McAfee Research, the security research division of McAfee, Inc., under contract to Apple Computer Inc. Additional authors include Wayne Salamon, Robert Watson, and SPARTA Inc.

The Basic Security Module (BSM) interface to audit records and audit event stream format were defined by Sun Microsystems.

This manual page was written by Robert Watson <rwatson@FreeBSD.org>.