NAME

auditon - configure system audit parameters

SYNOPSIS

#include <bsm/audit.h>

int
auditon(*int cmd, void *data, u_int length*);

DESCRIPTION

The **auditon**() system call is used to manipulate various audit control operations. The *data* argument should point to a structure whose type depends on the command. The *length* argument specifies the size of **data* in bytes. The *cmd* argument may be any of the following:

A_SETPOLICY — Set audit policy flags. The *data* argument must point to a *int* value set to one or more the following audit policy control values bitwise OR'ed together: AUDIT_CNT, AUDIT_AHLT, AUDIT_ARGV, and AUDIT_ARGE. If AUDIT_CNT is set, the system will continue even if it becomes low on space and discontinue logging events until the low space condition is remedied. If it is not set, audited events will block until the low space condition is remedied. Unaudited events, however, are unaffected. If AUDIT_AHLT is set, a panic(9) if it cannot write an event to the global audit log file. If AUDIT_ARGV is set, then the argument list passed to the execve(2) system call will be audited. If AUDIT_ARGE is set, then the environment variables passed to the execve(2) system call will be audited. The default policy is none of the audit policy control flags set.

A_SETKAUDIT — Set the host information. The *data* argument must point to a *auditinfo_addr_t* structure containing the host IP address information. After setting, audit records that are created as a result of kernel events will contain this information.

A_SETKMASK — Set the kernel preselection masks (success and failure). The *data* argument must point to a *au_mask_t* structure containing the mask values as defined in <*bsm/audit.h*>. These masks are used for non-attributable audit event preselection. The field *am_success* specifies which classes of successful audit events are to be logged to the audit trail. The field *am_failure* specifies which classes of failed audit events are to be logged. The value of both fields is the bitwise OR'ing of the audit event classes specified in *bsm/audit.h*. The various audit classes are described more fully in

audit_class(5).

A_SETQCTRL — Set kernel audit queue parameters. The *data* argument must point to a *au_qctrl_t* structure (defined in <*bsm/audit.h*>) containing the kernel audit queue control settings: *aq_hiwater*, *aq_lowater*, *aq_bufsz*, *aq_delay*, and *aq_minfree*. The field *aq_hiwater* defines the maximum number of audit record entries in the queue used to store the audit records ready for delivery to disk. New records are inserted at the tail of the queue and removed from the head. For new records which would exceed the high water mark, the calling thread is inserted into the wait queue, waiting for the audit queue to have enough space available as defined with the field *aq_lowater*. The field *aq_bufsz* defines the maximum length of the audit record that can be supplied with audit(2). The field *aq_delay* is unused. The field *aq_minfree* specifies the minimum amount of free blocks on the disk device used to store audit records. If the value of free blocks falls below the configured minimum amount, the kernel informs the audit daemon about low disk space. The value is to be specified in percent of free file system blocks. A value of 0 results in a disabling of the check. The default and maximum values (default/maximum) for the audit queue control parameters are:

aq_hiwater	100/10000 (audit records)
aq_lowater	10/aq_hiwater (audit records)
aq_bufsz	32767/1048576 (bytes)
aq_delay	(Not currently used.)

A_SETSTAT — Return ENOSYS. (Not implemented.)

A_SETUMASK — Return ENOSYS. (Not implemented.)

A_SETSMASK — Return ENOSYS. (Not implemented.)

A_SETCOND — Set the current auditing condition. The *data* argument must point to a *int* value containing the new audit condition, one of AUC_AUDITING, AUC_NOAUDIT, or AUC_DISABLED. If AUC_NOAUDIT is set, then auditing is temporarily suspended. If AUC_AUDITING is set, auditing is resumed. If AUC_DISABLED is set, the auditing system will shutdown, draining all audit records and closing out the audit trail file.

A_SETCLASS — Set the event class preselection mask for an audit event. The *data* argument must point to a *au_evclass_map_t* structure containing the audit event and

mask. The field *ec_number* is the audit event and *ec_class* is the audit class mask. See audit_event(5) for more information on audit event to class mapping.

A_SETPMASK — Set the preselection masks for a process. The *data* argument must point to a *auditpinfo_t* structure that contains the given process's audit preselection masks for both success and failure. The field *ap_pid* is the process id of the target process. The field *ap_mask* must point to a *au_mask_t* structure which holds the preselection masks as described in the A_SETKMASK section above.

A_SETFSIZE — Set the maximum size of the audit log file. The *data* argument must point to a *au_fstat_t* structure with the *af_filesz* field set to the maximum audit log file size. A value of 0 indicates no limit to the size.

A_GETCLASS — Return the event to class mapping for the designated audit event. The *data* argument must point to a *au_evclass_map_t* structure. See the A_SETCLASS section above for more information.

A_GETKAUDIT — Get the current host information. The *data* argument must point to a *auditinfo_addr_t* structure.

A_GETPINFO — Return the audit settings for a process. The *data* argument must point to a *auditpinfo_t* structure which will be set to contain *ap_auid* (the audit ID), *ap_mask* (the preselection mask), *ap_termid* (the terminal ID), and *ap_asid* (the audit session ID) of the given target process. The process ID of the target process is passed into the kernel using the *ap_pid* field. See the section A_SETPMASK above and getaudit(2) for more information.

A_GETPINFO_ADDR — Return the extended audit settings for a process. The *data* argument must point to a *auditpinfo_addr_t* structure which is similar to the *auditpinfo_t* structure described above. The exception is the *ap_termid* (the terminal ID) field which points to a *au_tid_addr_t* structure can hold much a larger terminal address and an address type. The process ID of the target process is passed into the kernel using the *ap_pid* field. See the section A_SETPMASK above and getaudit(2) for more information.

A_GETSINFO_ADDR — Return the extended audit settings for a session. The *data* argument must point to a *auditinfo_addr_t* structure. The audit session ID of the target session is passed into the kernel using the *ai_asid* field. See

getaudit_addr(2) for more information about the *auditinfo_addr_t* structure.

A_GETKMASK — Return the current kernel preselection masks. The *data* argument must point to a *au_mask_t* structure which will be set to the current kernel preselection masks for non-attributable events.

A_GETPOLICY — Return the current audit policy setting. The *data* argument must point to a *int* value which will be set to one of the current audit policy flags. The audit policy flags are described in the A_SETPOLICY section above.

A_GETQCTRL — Return the current kernel audit queue control parameters. The *data* argument must point to a *au_qctrl_t* structure which will be set to the current kernel audit queue control parameters. See the A_SETQCTL section above for more information.

A_GETFSIZE — Returns the maximum size of the audit log file. The *data* argument must point to a *au_fstat_t* structure. The *af_filesz* field will be set to the maximum audit log file size. A value of 0 indicates no limit to the size. The *af_currsz* field will be set to the current audit log file size.

A_GETCWD — Return ENOSYS. (Not implemented.)

A_GETCAR — Return ENOSYS. (Not implemented.)

A_GETSTAT — Return ENOSYS. (Not implemented.)

A_GETCOND — Return the current auditing condition. The *data* argument must point to a *int* value which will be set to the current audit condition, one of AUC_AUDITING, AUC_NOAUDIT or AUC_DISABLED. See the A_SETCOND section above for more information.

A_SENDTRIGGER — Send a trigger to the audit daemon. The *data* argument must point to a *int* value set to one of the acceptable trigger values: AUDIT_TRIGGER_LOW_SPACE (low disk space where the audit log resides), AUDIT_TRIGGER_OPEN_NEW (open a new audit log file), AUDIT_TRIGGER_READ_FILE (read the *audit_control* file), AUDIT_TRIGGER_CLOSE_AND_DIE (close the current log file and exit), AUDIT_TRIGGER_NO_SPACE (no disk space left for audit log file). AUDIT_TRIGGER_ROTATE_USER (request audit log file rotation). AUDIT_TRIGGER_INITIALIZE (initialize audit subsystem for Mac OS X

only). or AUDIT_TRIGGER_EXPIRE_TRAILS (request audit log file expiration).

RETURN VALUES

Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **auditon**() function will fail if:

[ENOSYS] Returned by options not yet implemented.

[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.

The A_SENDTRIGGER command is specific to the FreeBSD and Mac OS X implementations, and is not present in Solaris.

SEE ALSO

audit(2), auditctl(2), getaudit(2), getaudit_addr(2), getauid(2), setaudit(2), setaudit_addr(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 Tom Rhodes <trhodes@FreeBSD.org>, Robert Watson <rwatson@FreeBSD.org>, and Wayne Salamon <wsalamon@FreeBSD.org>.