NAME

auditpipe - pseudo-device for live audit event tracking

SYNOPSIS

options AUDIT

DESCRIPTION

While audit trail files generated with audit(4) and maintained by auditd(8) provide a reliable long-term store for audit log information, current log files are owned by the audit daemon until terminated making them somewhat unwieldy for live monitoring applications such as host-based intrusion detection. For example, the log may be cycled and new records written to a new file without notice to applications that may be accessing the file.

The audit facility provides an audit pipe facility for applications requiring direct access to live BSM audit data for the purposes of real-time monitoring. Audit pipes are available via a clonable special device, */dev/auditpipe*, subject to the permissions on the device node, and provide a "tee" of the audit event stream. As the device is clonable, more than one instance of the device may be opened at a time; each device instance will provide independent access to all records.

The audit pipe device provides discrete BSM audit records; if the read buffer passed by the application is too small to hold the next record in the sequence, it will be dropped. Unlike audit data written to the audit trail, the reliability of record delivery is not guaranteed. In particular, when an audit pipe queue fills, records will be dropped. Audit pipe devices are blocking by default, but support non-blocking I/O, asynchronous I/O using SIGIO, and polled operation via select(2) and poll(2).

Applications may choose to track the global audit trail, or configure local preselection parameters independent of the global audit trail parameters.

Audit Pipe Queue Ioctls

The following ioctls retrieve and set various audit pipe record queue properties:

AUDITPIPE_GET_QLEN	Query the current number of records available for reading on the pipe.
AUDITPIPE_GET_QLIMIT	Retrieve the current maximum number of records that may be queued for reading on the pipe.
AUDITPIPE_SET_QLIMIT	Set the current maximum number of records that may be queued for reading on the pipe. The new limit must fall between the queue limit minimum and queue limit maximum

queryable using the following two ioctls.

AUDITPIPE_GET_QLIMIT_MIN Query the lowest possible maximum number of records that may be queued for reading on the pipe.

AUDITPIPE_GET_QLIMIT_MAX Query the highest possible maximum number of records that may be queued for reading on the pipe.

AUDITPIPE_FLUSH Flush all outstanding records on the audit pipe; useful after setting initial preselection properties to delete records queued during the configuration process which may not match the interests of the user process.

AUDITPIPE_GET_MAXAUDITDATA Query the maximum size of an audit record, which is a useful minimum size for a user space buffer intended to hold audit records read from the audit pipe.

Audit Pipe Preselection Mode Ioctls

By default, the audit pipe facility configures pipes to present records matched by the system-wide audit trail, configured by auditd(8). However, the preselection mechanism for audit pipes can be configured using alternative criteria, including pipe-local flags and naflags settings, as well as auid-specific selection masks. This allows applications to track events not captured in the global audit trail, as well as limit records presented to those of specific interest to the application.

The following ioctls configure the preselection mode on an audit pipe:

AUDITPIPE_GET_PRESELECT_MODE Return the current preselect mode on the audit pipe. The ioctl argument should be of type *int*.

AUDITPIPE_SET_PRESELECT_MODE Set the current preselection mode on the audit pipe. The ioctl argument should be of type *int*.

Possible preselection mode values are:

AUDITPIPE_PRESELECT_MODE_TRAIL Use the global audit trail preselection parameters to select records for the audit pipe.

AUDITPIPE_PRESELECT_MODE_LOCAL
Use local audit pipe preselection; this model is similar to the global audit trail configuration model, consisting of

global flags and naflags parameters, as well as a set of per-auid masks. These parameters are configured using further ioctls.

After changing the audit pipe preselection mode, records selected under earlier preselection configuration may still be in the audit pipe queue. The application may flush the current record queue after changing the configuration to remove possibly undesired records.

Audit Pipe Local Preselection Mode Ioctls

The following ioctls configure the preselection parameters used when an audit pipe is configured for the AUDITPIPE_PRESELECT_MODE_LOCAL preselection mode.

AUDITPIPE_GET_PRESELECT_FLAGS — Retrieve the current default preselection flags for attributable events on the pipe. These flags correspond to the *flags* field in audit_control(5). The ioctl argument should be of type *au_mask_t*.

AUDITPIPE_SET_PRESELECT_FLAGS — Set the current default preselection flags for attributable events on the pipe. These flags correspond to the *flags* field in audit_control(5). The ioctl argument should be of type *au_mask_t*.

AUDITPIPE_GET_PRESELECT_NAFLAGS — Retrieve the current default preselection flags for non-attributable events on the pipe. These flags correspond to the *naflags* field in audit_control(5). The ioctl argument should be of type *au_mask_t*.

AUDITPIPE_SET_PRESELECT_NAFLAGS — Set the current default preselection flags for non-attributable events on the pipe. These flags correspond to the *naflags* field in audit_control(5). The ioctl argument should be of type *au_mask_t*.

AUDITPIPE_GET_PRESELECT_AUID — Query the current preselection masks for a specific auid on the pipe. The ioctl argument should be of type *struct auditpipe_ioctl_preselect*. The auid to query is specified via the *ap_auid* field of type *au_id_t*; the mask will be returned via *ap_mask* of type *au_mask_t*.

AUDITPIPE_SET_PRESELECT_AUID — Set the current preselection masks for a specific auid on the pipe. Arguments are identical to

	AUDITPIPE_GET_PRESELECT_AUID, except that the caller should properly initialize the *ap_mask* field to hold the desired preselection mask.
AUDITPIPE_DELETE_PRESELECT_AUID	Delete the current preselection mask for a specific auid on the pipe. Once called, events associated with the specified auid will use the default flags mask. The ioctl argument should be of type *au_id_t*.
AUDITPIPE_FLUSH_PRESELECT_AUID	Delete all auid specific preselection specifications.

EXAMPLES

The praudit(1) utility may be directly executed on */dev/auditpipe* to review the default audit trail.

SEE ALSO

poll(2), select(2), audit(4), dtaudit(4), audit_control(5), audit(8), auditd(8)

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.

Support for kernel audit first appeared in FreeBSD 6.2.

AUTHORS

The audit pipe facility was designed and implemented by Robert Watson <*rwatson@FreeBSD.org*>.

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

BUGS

See the audit(4) manual page for information on audit-related bugs and limitations.

The configurable preselection mechanism mirrors the selection model present for the global audit trail. It might be desirable to provide a more flexible selection model.

The per-pipe audit event queue is fifo, with drops occurring if either the user thread provides in sufficient for the record on the queue head, or on enqueue if there is insufficient room. It might be desirable to support partial reads of records, which would be more compatible with buffered I/O as implemented in system libraries, and to allow applications to select which records are dropped, possibly

in the style of preselection.