NAME

dtrace_audit - A DTrace provider for tracing audit(4) events

SYNOPSIS

audit:event:aue_*:commit(*char *eventname, struct audit_record *ar*);

audit:event:aue_*:bsm(*char *eventname, struct audit_record *ar, const void *, size_t*);

To compile this module into the kernel, place the following in your kernel configuration file:

options DTAUDIT

Alternatively, to load the module at boot time, place the following line in loader.conf(5):

dtaudit_load="YES"

DESCRIPTION

The DTrace **dtaudit** provider allows users to trace events in the kernel security auditing subsystem, audit(4). audit(4) provides detailed logging of a configurable set of security-relevant system calls, including key arguments (such as file paths) and return values that are copied race-free as the system call proceeds. The **dtaudit** provider allows DTrace scripts to selectively enable in-kernel audit-record capture for system calls, and then access those records in either the in-kernel format or BSM format (audit.log(5)) when the system call completes. While the in-kernel audit record data structure is subject to change as the kernel changes over time, it is a much more friendly interface for use in D scripts than either those available via the DTrace system-call provider or the BSM trail itself.

Configuration

The **dtaudit** provider relies on audit(4) being compiled into the kernel. **dtaudit** probes become available only once there is an event-to-name mapping installed in the kernel, normally done by auditd(8) during the boot process, if audit is enabled in rc.conf(5):

auditd_enable="YES"

If **dtaudit** probes are required earlier in boot -- for example, in single-user mode -- or without enabling audit(4), they can be preloaded in the boot loader by adding this line to loader.conf(5).

audit_event_load="YES"

Probes

The **audit:event:aue_*:commit**() probes fire synchronously during system-call return, giving access to

two arguments: a *char* * audit event name, and the *struct audit_record* * in-kernel audit record. Because the probe fires in system-call return, the user thread has not yet regained control, and additional information from the thread and process remains available for capture by the script.

The **audit:event:aue_*:bsm**() probes fire asynchronously from system-call return, following BSM conversion and just prior to being written to disk, giving access to four arguments: a *char* * audit event name, the *struct audit_record* * in-kernel audit record, a *const void* * pointer to the converted BSM record, and a *size_t* for the length of the BSM record.

IMPLEMENTATION NOTES

When a set of **dtaudit** probes are registered, corresponding in-kernel audit records will be captured and their probes will fire regardless of whether the audit(4) subsystem itself would have captured the record for the purposes of writing it to the audit trail, or for delivery to a auditpipe(4). In-kernel audit records allocated only because of enabled dtaudit(4) probes will not be unnecessarily written to the audit trail or enabled pipes.

SEE ALSO

dtrace(1), audit(4), audit.log(5), loader.conf(5), rc.conf(5), auditd(8)

HISTORY

The **dtaudit** provider first appeared in FreeBSD 12.0.

AUTHORS

This software and this manual page were developed by BAE Systems, the University of Cambridge Computer Laboratory, and Memorial University under DARPA/AFRL contract (FA8650-15-C-7558) ("CADETS"), as part of the DARPA Transparent Computing (TC) research program. The **dtaudit** provider and this manual page were written by Robert Watson <*rwatson@FreeBSD.org*>.

BUGS

Because audit(4) maintains its primary event-to-name mapping database in userspace, that database must be loaded into the kernel before **dtaudit** probes become available.

dtaudit is only able to provide access to system-call audit events, not the full scope of userspace events, such as those relating to login, password change, and so on.