```
***********************************************************
   59746 Tue Jan 15 10:25:28 2019
new/usr/src/uts/common/os/evchannels.c
10092 sysevent_evc_control() dereferences pointer before checking for NULL
***********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2003, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */

/*
 * This file contains the source of the general purpose event channel extension
 * to the sysevent framework. This implementation is made up mainly of four
 * layers of functionality: the event queues (evch_evq_*()), the handling of
 * channels (evch_ch*()), the kernel interface (sysevent_evc_*()) and the
 * interface for the sysevent pseudo driver (evch_usr*()).
 * Libsysevent.so uses the pseudo driver sysevent's ioctl to access the event
 * channel extensions. The driver in turn uses the evch_usr*() functions below.
 *
 * The interfaces for user land and kernel are declared in sys/sysevent.h
 * Internal data structures for event channels are defined in
 * sys/sysevent_impl.h.
 *
 * The basic data structure for an event channel is of type evch_chan_t.
 * All channels are maintained by a list named evch_list. The list head
 * is of type evch_dlist_t.
 */

#include <sys/types.h>
#include <sys/errno.h>
#include <sys/stropts.h>
#include <sys/debug.h>
#include <sys/ddi.h>
#include <sys/vmem.h>
#include <sys/cmn_err.h>
#include <sys/callb.h>
#include <sys/sysevent.h>
#include <sys/sysevent_impl.h>
#include <sys/sysmacros.h>
#include <sys/disp.h>
#include <sys/atomic.h>
#include <sys/door.h>
#include <sys/zone.h>
```



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

/* Back-off delay for door_ki_upcall */
#define EVCH_MIN_PAUSE  8
#define EVCH_MAX_PAUSE  128

#define GEVENT(ev)      ((evch_gevent_t *)((char *)ev - \
                            offsetof(evch_gevent_t, ge_payload)))

#define EVCH_EVQ_EVCOUNT(x)     ((&(x)->eq_eventq)->sq_count)
#define EVCH_EVQ_HIGHWM(x)      ((&(x)->eq_eventq)->sq_highwm)

#define CH_HOLD_PEND            1
#define CH_HOLD_PEND_INDEF      2

struct evch_globals {
        evch_dlist_t evch_list;
        kmutex_t evch_list_lock;
};
______unchanged_portion_omitted_

int
sysevent_evc_control(evchan_t *scp, int cmd, ...)
{
        va_list         ap;
        evch_chan_t     *chp;
        evch_chan_t     *chp = ((evch_bind_t *)scp)->bd_channel;
        uint32_t        *chlenp;
        uint32_t        chlen;
        uint32_t        ochlen;
        int             rc = 0;

        if (scp == NULL) {
                return (EINVAL);
        }

        chp = ((evch_bind_t *)scp)->bd_channel;

        va_start(ap, cmd);
        mutex_enter(&chp->ch_mutex);
        switch (cmd) {
        case EVCH_GET_CHAN_LEN:
                chlenp = va_arg(ap, uint32_t *);
                *chlenp = chp->ch_maxev;
                break;
        case EVCH_SET_CHAN_LEN:
                chlen = va_arg(ap, uint32_t);
                ochlen = chp->ch_maxev;
                chp->ch_maxev = min(chlen, evch_events_max);
                if (ochlen < chp->ch_maxev) {
                        cv_signal(&chp->ch_pubcv);
                }
                break;
        case EVCH_GET_CHAN_LEN_MAX:
                *va_arg(ap, uint32_t *) = evch_events_max;
                break;
        default:
                rc = EINVAL;
        }

        mutex_exit(&chp->ch_mutex);
        va_end(ap);
        return (rc);
}
______unchanged_portion_omitted_
```