```
**********************************************************
    9882 Thu Jan 17 15:08:14 2019
new/usr/src/cmd/audit/audit.c
10119 audit(1) gets NULL check wrong
**********************************************************
```

```
/*
 * 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) 1992, 2010, Oracle and/or its affiliates. All rights reserved.
 */

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

#include <fcntl.h>
#include <libscf.h>
#include <secdb.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/file.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <sys/param.h>
#include <unistd.h>
#include <bsm/audit.h>
#include <bsm/libbsm.h>
#include <locale.h>
#include <zone.h>
#include <audit_scf.h>

#if !defined(TEXT_DOMAIN)
#define TEXT_DOMAIN "SUNW_OST_OSCMD"
#endif

#define VERIFY -1

/* GLOBALS */
static char     *progname = "audit";
static char     *usage = "audit [-n] | [-s] | [-t] | [-v]";
static int      silent = 0;

static void     display_smf_error();
```



```
static boolean_t is_audit_config_ok();          /* config validation  */
static boolean_t is_valid_zone(boolean_t);      /* operation ok in this zone? */
static boolean_t contains_valid_dirs(char *);   /* p_dir contents validation */
static boolean_t validate_path(char *);         /* is it path to dir? */
static void     start_auditd();                 /* start audit daemon */
static int      sig_auditd(int);                /* send signal to auditd */

/*
 * audit() - This program serves as a general administrator's interface to
 *      the audit trail.  Only one option is valid at a time.
 *
 * input:
 *      audit -s
 *              - signal audit daemon to read audit configuration and
 *                start auditd if needed.
 *      audit -n
 *              - signal audit daemon to use next audit_binfile directory.
 *      audit -t
 *              - signal audit daemon to disable auditing.
 *      audit -T
 *              - signal audit daemon to temporarily disable auditing reporting
 *                no errors.
 *      audit -v
 *              - validate audit configuration parameters;
 *                Print errors or "configuration ok".
 *
 *
 * output:
 *
 * returns:     0 - command successful
 *              >0 - command failed
 */

int
main(int argc, char *argv[])
{
        int     c;

        /* Internationalization */
        (void) setlocale(LC_ALL, "");
        (void) textdomain(TEXT_DOMAIN);

        /* second or more options not allowed; please pick one */
        if (argc > 2) {
                (void) fprintf(stderr, gettext("usage: %s\n"), usage);
                exit(1);
        }

        /* first option required */
        if ((c = getopt(argc, argv, "nstTv")) == -1) {
                (void) fprintf(stderr, gettext("usage: %s\n"), usage);
                exit(1);
        }

        switch (c) {
        case 'n':
                if (!is_valid_zone(1))  /* 1 == display error if any */
                        exit(1);

                if (sig_auditd(SIGUSR1) != 0)
                        exit(1);
                break;
        case 's':
                if (!is_valid_zone(1))  /* 1 == display error if any */
                        exit(1);
                else if (!is_audit_config_ok())
```

```
128                         exit(1);

130                 start_auditd();
131                 return (0);
132         case 't':
133                 if (!is_valid_zone(0))  /* 0 == no error message display */
134                         exit(1);
135                 if (smf_disable_instance(AUDITD_FMRI, 0) != 0) {
136                         display_smf_error();
137                         exit(1);
138                 }
139                 break;
140         case 'T':
141                 silent = 1;
142                 if (!is_valid_zone(0))  /* 0 == no error message display */
143                         exit(1);
144                 if (smf_disable_instance(AUDITD_FMRI, SMF_TEMPORARY) != 0) {
145                         exit(1);
146                 }
147                 break;
148         case 'v':
149                 if (is_audit_config_ok()) {
150                         (void) fprintf(stderr, gettext("configuration ok\n"));
151                         exit(0);
152                 } else {
153                         exit(1);
154                 }
155                 break;
156         default:
157                 (void) fprintf(stderr, gettext("usage: %s\n"), usage);
158                 exit(1);
159         }

161         return (0);
162 }
```

*unchanged_portion_omitted_*

```
199 /*
200  * perform reasonableness check on audit configuration
201  */

203 static boolean_t
204 is_audit_config_ok() {
205         int                     state = B_TRUE; /* B_TRUE/B_FALSE = ok/not_ok */
206         char                    *cval_str;
207         int                     cval_int;
208         kva_t                   *kvlist;
209         scf_plugin_kva_node_t   *plugin_kva_ll;
210         scf_plugin_kva_node_t   *plugin_kva_ll_head;
211         boolean_t               one_plugin_enabled = B_FALSE;

213         /*
214          * There must be at least one active plugin configured; if the
215          * configured plugin is audit_binfile(5), then the p_dir must not be
216          * empty.
217          */
218         if (!do_getpluginconfig_scf(NULL, &plugin_kva_ll)) {
219                 (void) fprintf(stderr,
220                     gettext("Could not get plugin configuration.\n"));
221                 exit(1);
222         }

224         plugin_kva_ll_head = plugin_kva_ll;

226         while (plugin_kva_ll != NULL) {
227                 kvlist = plugin_kva_ll->plugin_kva;
```



```
229                 if (!one_plugin_enabled) {
230                         cval_str = kva_match(kvlist, "active");
231                         if (atoi(cval_str) == 1) {
232                                 one_plugin_enabled = B_TRUE;
233                         }
234                 }

236                 if (strcmp((char *)&(*plugin_kva_ll).plugin_name,
237                     "audit_binfile") == 0) {
238                         cval_str = kva_match(kvlist, "p_dir");
239                         if (cval_str == NULL || cval_str[0] == '\0') {
235                         if (*cval_str == '\0' || cval_str == NULL) {
240                                 (void) fprintf(stderr,
241                                     gettext("%s: audit_binfile(5) \"p_dir:\" "
242                                     "attribute empty\n"), progname);
243                                 state = B_FALSE;
244                         } else if (!contains_valid_dirs(cval_str)) {
245                                 (void) fprintf(stderr,
246                                     gettext("%s: audit_binfile(5) \"p_dir:\" "
247                                     "attribute invalid\n"), progname);
248                                 state = B_FALSE;
249                         }

251                         cval_str = kva_match(kvlist, "p_minfree");
252                         cval_int = atoi(cval_str);
253                         if (cval_int < 0 || cval_int > 100) {
254                                 (void) fprintf(stderr,
255                                     gettext("%s: audit_binfile(5) "
256                                     "\"p_minfree:\" attribute invalid\n"),
257                                     progname);
258                                 state = B_FALSE;
259                         }
260                 }

262                 plugin_kva_ll = plugin_kva_ll->next;
263         }

265         plugin_kva_ll_free(plugin_kva_ll_head);

267         if (!one_plugin_enabled) {
268                 (void) fprintf(stderr, gettext("%s: no active plugin found\n"),
269                     progname);
270                 state = B_FALSE;
271         }

273         return (state);
274 }
```

*unchanged_portion_omitted_*