```
**********************************************************
   14786 Mon Jan 21 16:30:20 2019
new/usr/src/cmd/cmd-crypto/cryptoadm/adm_kef_ioctl.c
10124 smatch fixes for cryptoadm
**********************************************************
```

```
   1 /*
   2  * CDDL HEADER START
   3  *
   4  * The contents of this file are subject to the terms of the
   5  * Common Development and Distribution License (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright (c) 2003, 2010, Oracle and/or its affiliates. All rights reserved.
  23  */
  24 /*
  25  * Copyright 2010 Nexenta Systems, Inc.  All rights reserved.
  26  * Copyright (c) 2018, Joyent, Inc.
  27  */

  29 #include <fcntl.h>
  30 #include <stdio.h>
  31 #include <stdlib.h>
  32 #include <strings.h>
  33 #include <unistd.h>
  34 #include <locale.h>
  35 #include <libgen.h>
  36 #include <sys/types.h>
  37 #include <sys/varargs.h>
  38 #include <zone.h>
  39 #include <sys/crypto/ioctladmin.h>
  40 #include "cryptoadm.h"

  42 #define DEFAULT_DEV_NUM 5
  43 #define DEFAULT_SOFT_NUM 10

  45 static crypto_get_soft_info_t *setup_get_soft_info(char *, int);

  47 /*
  48  * Prepare the argument for the LOAD_SOFT_CONFIG ioctl call for the
  49  * provider pointed by pent.  Return NULL if out of memory.
  50  */
  51 crypto_load_soft_config_t *
  52 setup_soft_conf(entry_t *pent)
  53 {
  54         crypto_load_soft_config_t       *pload_soft_conf;
  55         mechlist_t      *plist;
  56         uint_t          sup_count;
  57         size_t          extra_mech_size = 0;
  58         int             i;

  60         if (pent == NULL) {
  61                 return (NULL);
```



```
  62         }

  64         sup_count = pent->sup_count;
  65         if (sup_count > 1) {
  66                 extra_mech_size = sizeof (crypto_mech_name_t) *
  67                     (sup_count - 1);
  68         }

  70         pload_soft_conf = malloc(sizeof (crypto_load_soft_config_t) +
  71             extra_mech_size);
  72         if (pload_soft_conf == NULL) {
  73                 cryptodebug("out of memory.");
  74                 return (NULL);
  75         }

  77         (void) strlcpy(pload_soft_conf->sc_name, pent->name, MAXNAMELEN);
  78         pload_soft_conf->sc_count = sup_count;

  80         i = 0;
  81         plist =  pent->suplist;
  82         while (i < sup_count) {
  83                 (void) strlcpy(pload_soft_conf->sc_list[i++],
  84                     plist->name, CRYPTO_MAX_MECH_NAME);
  85                 plist = plist->next;
  86         }

  88         return (pload_soft_conf);
  89 }
_____unchanged_portion_omitted_


 250 /*
 251  * Get the device list from kernel.
 252  */
 253 int
 254 get_dev_list(crypto_get_dev_list_t **ppdevlist)
 255 {
 256         crypto_get_dev_list_t   *pdevlist;
 257         int                     fd = -1;
 258         int                     count = DEFAULT_DEV_NUM;

 260         pdevlist = malloc(sizeof (crypto_get_dev_list_t) +
 261             sizeof (crypto_dev_list_entry_t) * (count - 1));
 262         if (pdevlist == NULL) {
 263                 cryptodebug("out of memory.");
 264                 return (FAILURE);
 265         }

 267         if ((fd = open(ADMIN_IOCTL_DEVICE, O_RDONLY)) == -1) {
 268                 cryptoerror(LOG_STDERR, gettext("failed to open %s: %s"),
 269                     ADMIN_IOCTL_DEVICE, strerror(errno));
 270                 free(pdevlist);
 271                 return (FAILURE);
 272         }

 274         pdevlist->dl_dev_count = count;
 275         if (ioctl(fd, CRYPTO_GET_DEV_LIST, pdevlist) == -1) {
 276                 cryptodebug("CRYPTO_GET_DEV_LIST ioctl failed: %s",
 277                     strerror(errno));
 278                 free(pdevlist);
 279                 (void) close(fd);
 280                 return (FAILURE);
 281         }

 283         /* BUFFER is too small, get the number of devices and retry it. */
 284         if (pdevlist->dl_return_value == CRYPTO_BUFFER_TOO_SMALL) {
```

```
285                 count = pdevlist->dl_dev_count;
286                 free(pdevlist);
287                 pdevlist = malloc(sizeof (crypto_get_dev_list_t) +
288                     sizeof (crypto_dev_list_entry_t) * (count - 1));
289                 if (pdevlist == NULL) {
290                         cryptodebug("out of memory.");
291                         (void) close(fd);
292                         return (FAILURE);
293                 }

295                 if (ioctl(fd, CRYPTO_GET_DEV_LIST, pdevlist) == -1) {
296                         cryptodebug("CRYPTO_GET_DEV_LIST ioctl failed: %s",
297                             strerror(errno));
298                         free(pdevlist);
299                         (void) close(fd);
300                         return (FAILURE);
301                 }
302         }

304         if (pdevlist->dl_return_value != CRYPTO_SUCCESS) {
305                 cryptodebug("CRYPTO_GET_DEV_LIST ioctl failed, "
306                     "return_value = %d", pdevlist->dl_return_value);
307                 free(pdevlist);
308                 (void) close(fd);
309                 return (FAILURE);
310         }

312         *ppdevlist = pdevlist;
313         (void) close(fd);
314         return (SUCCESS);
315 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   41663 Mon Jan 21 16:30:20 2019
new/usr/src/cmd/cmd-crypto/cryptoadm/adm_uef.c
10124 smatch fixes for cryptoadm
**********************************************************
```

```
/*
 * 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 2010 Nexenta Systems, Inc.  All rights resrved.
 * Copyright (c) 2018, Joyent, Inc.
 */

#include <cryptoutil.h>
#include <fcntl.h>
#include <libintl.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <unistd.h>
#include <errno.h>
#include <dlfcn.h>
#include <link.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <security/cryptoki.h>
#include "cryptoadm.h"

#define HDR1 "                                    P\n"
#define HDR2 "                        S     V  K  a     U  D\n"
#define HDR3 "                        i     e  e  i     n  e\n"
#define HDR4 "                     S  g  V  r  y  r  W  w  r\n"
#define HDR5 "            E  D  D  i  n  e  i  G  G  r  r  i\n"
#define HDR6 "         H  n  e  i  g  +  r  +  e  e  a  a  v  E\n"
#define HDR7 "min  max  W  c  c  g  n  R  i  R  n  n  p  p  e  C\n"


static int err; /* To store errno which may be overwritten by gettext() */
static boolean_t is_in_policylist(midstr_t, umechlist_t *);
static char *uent2str(uentry_t *);
static boolean_t check_random(CK_SLOT_ID, CK_FUNCTION_LIST_PTR);

static void display_slot_flags(CK_FLAGS flags)
{
        (void) printf(gettext("Slot Flags: "));
        if (flags & CKF_TOKEN_PRESENT)
```



```
                (void) printf("CKF_TOKEN_PRESENT ");
        if (flags & CKF_REMOVABLE_DEVICE)
                (void) printf("CKF_REMOVABLE_DEVICE ");
        if (flags & CKF_HW_SLOT)
                (void) printf("CKF_HW_SLOT ");
        (void) printf("\n");
}
_____unchanged_portion_omitted_

/*
 * Converts the provided list of mechanism names in their string format to
 * their corresponding PKCS#11 mechanism IDs.
 *
 * The list of mechanism names to be converted is provided in the
 * "mlist" argument.  The list of converted mechanism IDs is returned
 * in the "pmech_list" argument.
 *
 * This function is called by list_metaslot_info() and
 * list_mechlist_for_lib() functions.
 */
int
convert_mechlist(CK_MECHANISM_TYPE **pmech_list, CK_ULONG *mech_count,
    mechlist_t *mlist)
{
        int i, n = 0;
        mechlist_t *p = mlist;

        while (p != NULL) {
                p = p->next;
                n++;
        }

        *pmech_list = malloc(n * sizeof (CK_MECHANISM_TYPE));
        if (*pmech_list == NULL) {
        if (pmech_list == NULL) {
                cryptodebug("out of memory");
                return (FAILURE);
        }
        p = mlist;
        for (i = 0; i < n; i++) {
                if (pkcs11_str2mech(p->name, &(*pmech_list[i])) != CKR_OK) {
                        free(*pmech_list);
                        return (FAILURE);
                }
                p = p->next;
        }
        *mech_count = n;
        return (SUCCESS);
}
_____unchanged_portion_omitted_
```

```
**********************************************************
   42685 Mon Jan 21 16:30:21 2019
new/usr/src/cmd/cmd-crypto/cryptoadm/cryptoadm.c
10124 smatch fixes for cryptoadm
**********************************************************
```

```
/*
 * 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.
 */

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <unistd.h>
#include <locale.h>
#include <libgen.h>
#include <sys/types.h>
#include <zone.h>
#include <sys/crypto/ioctladmin.h>
#include <cryptoutil.h>
#include "cryptoadm.h"

#define REQ_ARG_CNT     2

/* subcommand index */
enum subcommand_index {
        CRYPTO_LIST,
        CRYPTO_DISABLE,
        CRYPTO_ENABLE,
        CRYPTO_INSTALL,
        CRYPTO_UNINSTALL,
        CRYPTO_UNLOAD,
        CRYPTO_REFRESH,
        CRYPTO_START,
        CRYPTO_STOP,
        CRYPTO_HELP };

/*
 * TRANSLATION_NOTE
 * Command keywords are not to be translated.
 */
static char *cmd_table[] = {
```



```
        "list",
        "disable",
        "enable",
        "install",
        "uninstall",
        "unload",
        "refresh",
        "start",
        "stop",
        "--help" };

/* provider type */
enum provider_type_index {
        PROV_UEF_LIB,
        PROV_KEF_SOFT,
        PROV_KEF_HARD,
        METASLOT,
        PROV_BADNAME };

typedef struct {
        char cp_name[MAXPATHLEN];
        enum provider_type_index cp_type;
} cryptoadm_provider_t;
______unchanged_portion_omitted_



/*
 * Print a list all the the providers.
 * Called for "cryptoadm list" or "cryptoadm list -v" (no -m or -p).
 */
static int
list_simple_for_all(boolean_t verbose)
{
        uentrylist_t            *pliblist = NULL;
        uentrylist_t            *plibptr = NULL;
        entry_t                 *pent = NULL;
        crypto_get_dev_list_t   *pdevlist_kernel = NULL;
        int                     rc = SUCCESS;
        int                     i;

        /* get user-level providers */
        (void) printf(gettext("\nUser-level providers:\n"));
        if (get_pkcs11conf_info(&pliblist) != SUCCESS) {
                cryptoerror(LOG_STDERR, gettext(
                    "failed to retrieve the list of user-level providers."));
                rc = FAILURE;
        }

        for (plibptr = pliblist; plibptr != NULL; plibptr = plibptr->next) {
                /* skip metaslot and fips-140 entry */
                if ((strcmp(plibptr->puent->name, METASLOT_KEYWORD) != 0) &&
                    (strcmp(plibptr->puent->name, FIPS_KEYWORD) != 0)) {
                        (void) printf(gettext("Provider: %s\n"),
                            plibptr->puent->name);
                        if (verbose) {
                                (void) list_mechlist_for_lib(
                                    plibptr->puent->name, mecharglist, NULL,
                                    B_FALSE, verbose, B_FALSE);
                                (void) printf("\n");
                        }
                }
        }
        free_uentrylist(pliblist);

        /* get kernel software providers */
```

```
1285         (void) printf(gettext("\nKernel software providers:\n"));

1287         if (getzoneid() == GLOBAL_ZONEID) {
1288                 /* get kernel software providers from kernel ioctl */
1289                 crypto_get_soft_list_t          *psoftlist_kernel = NULL;
1290                 uint_t                          sl_soft_count;
1291                 char                            *psoftname;
1292                 entrylist_t                     *pdevlist_conf = NULL;
1293                 entrylist_t                     *psoftlist_conf = NULL;

1295                 if (get_soft_list(&psoftlist_kernel) == FAILURE) {
1296                         cryptoerror(LOG_ERR, gettext("Failed to retrieve the "
1297                             "software provider list from kernel."));
1298                         rc = FAILURE;
1299                 } else {
1300                         sl_soft_count = psoftlist_kernel->sl_soft_count;

1302                         if (get_kcfconf_info(&pdevlist_conf, &psoftlist_conf)
1303                             == FAILURE) {
1304                                 cryptoerror(LOG_ERR,
1305                                     "failed to retrieve the providers' "
1306                                     "information from file kcf.conf - %s.",
1307                                     _PATH_KCF_CONF);
1305                                 free(psoftlist_kernel);
1308                                 rc = FAILURE;
1309                         } else {

1311                                 for (i = 0,
1312                                     psoftname = psoftlist_kernel->sl_soft_names;
1313                                     i < sl_soft_count;
1314                                     ++i, psoftname += strlen(psoftname) + 1) {
1315                                         pent = getent_kef(psoftname,
1316                                             pdevlist_conf, psoftlist_conf);
1317                                         (void) printf("\t%s%s\n", psoftname,
1318                                             (pent == NULL) || (pent->load) ?
1319                                             "" : gettext(" (inactive)"));
1320                                 }
1321                                 free_entrylist(pdevlist_conf);
1322                                 free_entrylist(psoftlist_conf);
1323                         }
1324                         free(psoftlist_kernel);
1325                 }

1327         } else {
1328                 /* kcf.conf not there in non-global zone, use /dev/cryptoadm */
1329                 entrylist_t     *pdevlist_zone = NULL;
1330                 entrylist_t     *psoftlist_zone = NULL;
1331                 entrylist_t     *ptr;

1333                 if (get_admindev_info(&pdevlist_zone, &psoftlist_zone) !=
1334                     SUCCESS) {
1335                         cryptoerror(LOG_STDERR,
1336                             gettext("failed to retrieve the "
1337                             "list of kernel software providers.\n"));
1338                         rc = FAILURE;
1339                 }

1341                 ptr = psoftlist_zone;
1342                 while (ptr != NULL) {
1343                         (void) printf("\t%s\n", ptr->pent->name);
1344                         ptr = ptr->next;
1345                 }

1347                 free_entrylist(pdevlist_zone);
1348                 free_entrylist(psoftlist_zone);
1349         }
```



```
1351         /* get kernel hardware providers */
1352         (void) printf(gettext("\nKernel hardware providers:\n"));
1353         if (get_dev_list(&pdevlist_kernel) == FAILURE) {
1354                 cryptoerror(LOG_STDERR, gettext("failed to retrieve "
1355                     "the list of kernel hardware providers.\n"));
1356                 rc = FAILURE;
1357         } else {
1358                 for (i = 0; i < pdevlist_kernel->dl_dev_count; i++) {
1359                         (void) printf("\t%s/%d\n",
1360                             pdevlist_kernel->dl_devs[i].le_dev_name,
1361                             pdevlist_kernel->dl_devs[i].le_dev_instance);
1362                 }
1363         }
1364         free(pdevlist_kernel);

1366         return (rc);
1367 }
```

_____***unchanged_portion_omitted_***