****

18817 Mon Jan 21 16:23:51 2019
new/usr/src/cmd/coreadm/coreadm.c
10127 coreadm is mis-using strcpy()

****

```
/*
 * 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 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright (c) 2018, Joyent, Inc.
 */

#include <stdio.h>
#include <fcntl.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <limits.h>
#include <libintl.h>
#include <locale.h>
#include <sys/stat.h>
#include <sys/corectl.h>
#include <libproc.h>
#include <libscf.h>
#include <libscf_priv.h>
#include <assert.h>

#define E_SUCCESS       0               /* Exit status for success */
#define E_ERROR         1               /* Exit status for error */
#define E_USAGE         2               /* Exit status for usage error */

static  const   char    PATH_CONFIG[] = "/etc/coreadm.conf";
static  const   char    PATH_CONFIG_OLD[] = "/etc/coreadm.conf.old";

#define COREADM_INST_NAME       "system/coreadm:default"
#define COREADM_INST_FMRI       \
    SCF_FMRI_SVC_PREFIX SCF_FMRI_SERVICE_PREFIX COREADM_INST_NAME

#define CONFIG_PARAMS           "config_params"
#define GLOBAL_ENABLED          "global_enabled"
#define PROCESS_ENABLED         "process_enabled"
#define GLOBAL_SETID_ENABLED    "global_setid_enabled"
#define PROCESS_SETID_ENABLED   "process_setid_enabled"
#define GLOBAL_LOG_ENABLED      "global_log_enabled"
#define GLOBAL_PATTERN          "global_pattern"
```

```
#define GLOBAL_CONTENT          "global_content"
#define INIT_PATTERN            "init_pattern"
#define INIT_CONTENT            "init_content"

static  char            *command;
static  uint64_t        options;
static  int             alloptions;
static  char            *glob_pattern;
static  char            gpattern[PATH_MAX];
static  core_content_t  glob_content = CC_CONTENT_INVALID;
static  char            *init_pattern;
static  char            ipattern[PATH_MAX];
static  core_content_t  init_content = CC_CONTENT_INVALID;
static  char            *proc_pattern;
static  size_t          proc_size;
static  core_content_t  proc_content = CC_CONTENT_INVALID;

static  int             report_settings(void);
static  int             do_processes(int, char **);
static  int             do_modify(boolean_t);
static  int             do_update(void);
static  int             do_legacy(void);

static scf_propvec_t prop_gpattern = { GLOBAL_PATTERN, NULL, SCF_TYPE_ASTRING };
static scf_propvec_t prop_gcontent = { GLOBAL_CONTENT, NULL, SCF_TYPE_ASTRING };
static scf_propvec_t prop_ipattern = { INIT_PATTERN, NULL, SCF_TYPE_ASTRING };
static scf_propvec_t prop_icontent = { INIT_CONTENT, NULL, SCF_TYPE_ASTRING };
static scf_propvec_t prop_option[] = {
    { GLOBAL_ENABLED, NULL, SCF_TYPE_BOOLEAN, NULL, CC_GLOBAL_PATH },
    { PROCESS_ENABLED, NULL, SCF_TYPE_BOOLEAN, NULL, CC_PROCESS_PATH },
    { GLOBAL_SETID_ENABLED, NULL, SCF_TYPE_BOOLEAN, NULL, CC_GLOBAL_SETID },
    { PROCESS_SETID_ENABLED, NULL, SCF_TYPE_BOOLEAN, NULL, CC_PROCESS_SETID },
    { GLOBAL_LOG_ENABLED, NULL, SCF_TYPE_BOOLEAN, NULL, CC_GLOBAL_LOG },
    { NULL }
};
```

______***unchanged_portion_omitted_***

```
static int
read_legacy(void)
{
        FILE *fp;
        int line;
        char buf[BUFSIZE];
        char name[BUFSIZE], value[BUFSIZE];
        int n, len;

        /* defaults */
        alloptions = CC_OPTIONS;
        options = CC_PROCESS_PATH;
        gpattern[0] = '\0';
        (void) strcpy(ipattern, "core");
        glob_content = init_content = CC_CONTENT_DEFAULT;

        glob_pattern = gpattern;
        init_pattern = ipattern;

        if ((fp = fopen(PATH_CONFIG, "r")) == NULL)
                return (0);

        for (line = 1; fgets(buf, sizeof (buf), fp) != NULL; line++) {
                /*
                 * Skip comment lines and empty lines.
                 */
                if (buf[0] == '#' || buf[0] == '\n')
                        continue;
                /*
```

```
		 * Look for "name=value", with optional whitespace on either
		 * side, terminated by a newline, and consuming the whole line.
		 */
		/* LINTED - unbounded string specifier */
		n = sscanf(buf, " %[^=]=%s \n%n", name, value, &len);
		if (n >= 1 && name[0] != '\0' &&
		    (n == 1 || len == strlen(buf))) {
			if (n == 1)
				value[0] = '\0';
			if (strcmp(name, "COREADM_GLOB_PATTERN") == 0) {
				(void) strlcpy(gpattern, value,
				    sizeof (gpattern));
				(void) strcpy(gpattern, value);
				continue;
			}
			if (strcmp(name, "COREADM_GLOB_CONTENT") == 0) {
				(void) proc_str2content(value, &glob_content);
				continue;
			}
			if (strcmp(name, "COREADM_INIT_PATTERN") == 0) {
				(void) strlcpy(ipattern, value,
				    sizeof (ipattern));
				(void) strcpy(ipattern, value);
				continue;
			}
			if (strcmp(name, "COREADM_INIT_CONTENT") == 0) {
				(void) proc_str2content(value, &init_content);
				continue;
			}
			if (strcmp(name, "COREADM_GLOB_ENABLED") == 0) {
				if (yes(name, value, line))
					options |= CC_GLOBAL_PATH;
				continue;
			}
			if (strcmp(name, "COREADM_PROC_ENABLED") == 0) {
				if (yes(name, value, line))
					options |= CC_PROCESS_PATH;
				else
					options &= ~CC_PROCESS_PATH;
				continue;
			}
			if (strcmp(name, "COREADM_GLOB_SETID_ENABLED") == 0) {
				if (yes(name, value, line))
					options |= CC_GLOBAL_SETID;
				continue;
			}
			if (strcmp(name, "COREADM_PROC_SETID_ENABLED") == 0) {
				if (yes(name, value, line))
					options |= CC_PROCESS_SETID;
				continue;
			}
			if (strcmp(name, "COREADM_GLOB_LOG_ENABLED") == 0) {
				if (yes(name, value, line))
					options |= CC_GLOBAL_LOG;
				continue;
			}
			(void) fprintf(stderr, gettext(
			    "\"%s\", line %d: warning: invalid token: %s\n"),
			    PATH_CONFIG, line, name);
		} else {
			(void) fprintf(stderr,
			    gettext("\"%s\", line %d: syntax error\n"),
			    PATH_CONFIG, line);
		}
	}
	(void) fclose(fp);
```

```
	return (1);
}
```

______***unchanged_portion_omitted_***