```
**********************************************************
   10633 Tue Jan  6 20:35:58 2015
new/usr/src/cmd/sgs/libelf/common/clscook.c
5507 libelf may overflow data buffer when translating data to memory representat
**********************************************************
```

```
/*
 * 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) 1988 AT&T */
/*        All Rights Reserved   */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * This stuff used to live in cook.c, but was moved out to
 * facilitate dual (Elf32 and Elf64) compilation.  See block
 * comment in cook.c for more info.
 */

#include <string.h>
#include <ar.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/sysmacros.h>
#endif /* ! codereview */
#include "decl.h"
#include "member.h"
#include "msg.h"

/*
 * This module is compiled twice, the second time having
 * -D_ELF64 defined.  The following set of macros, along
 * with machelf.h, represent the differences between the
 * two compilations.  Be careful *not* to add any class-
 * dependent code (anything that has elf32 or elf64 in the
 * name) to this code without hiding it behind a switch-
 * able macro like these.
 */
#if     defined(_ELF64)
#define Snode           Snode64
#define ELFCLASS        ELFCLASS64
#define ElfField        Elf64
#define _elf_snode_init _elf64_snode_init
```



```
#define _elf_prepscan   _elf64_prepscan
#define _elf_cookscn    _elf64_cookscn
#define _elf_mtype      _elf64_mtype
#define _elf_msize      _elf64_msize
#define elf_fsize       elf64_fsize
#define _elf_snode      _elf64_snode
#define _elf_ehdr       _elf64_ehdr
#define elf_xlatetom    elf64_xlatetom
#define _elf_phdr       _elf64_phdr
#define _elf_shdr       _elf64_shdr
#define _elf_prepscn    _elf64_prepscn

#else  /* Elf32 */
#define Snode           Snode32
#define ELFCLASS        ELFCLASS32
#define ElfField        Elf32
#define _elf_snode_init _elf32_snode_init
#define _elf_prepscan   _elf32_prepscan
#define _elf_cookscn    _elf32_cookscn
#define _elf_mtype      _elf32_mtype
#define _elf_msize      _elf32_msize
#define elf_fsize       elf32_fsize
#define _elf_snode      _elf32_snode
#define _elf_ehdr       _elf32_ehdr
#define elf_xlatetom    elf32_xlatetom
#define _elf_phdr       _elf32_phdr
#define _elf_shdr       _elf32_shdr
#define _elf_prepscn    _elf32_prepscn

#endif /* _ELF64 */


static Okay
_elf_prepscn(Elf *elf, size_t cnt)
{
        NOTE(ASSUMING_PROTECTED(*elf))
        Elf_Scn *       s;
        Elf_Scn *       end;

        if (cnt == 0)
                return (OK_YES);

        if ((s = malloc(cnt * sizeof (Elf_Scn))) == 0) {
                _elf_seterr(EMEM_SCN, errno);
                return (OK_NO);
        }
        NOTE(NOW_INVISIBLE_TO_OTHER_THREADS(*s))
        elf->ed_scntabsz = cnt;
        end = s + cnt;
        elf->ed_hdscn = s;
        do {
                *s = _elf_snode_init.sb_scn;
                s->s_elf = elf;
                s->s_next = s + 1;
                s->s_index = s - elf->ed_hdscn;
                s->s_shdr = (Shdr*)s->s_elf->ed_shdr + s->s_index;
                ELFMUTEXINIT(&s->s_mutex);

                /*
                 * Section has not yet been cooked!
                 *
                 * We don't cook a section until it's data is actually
                 * referenced.
                 */
                s->s_myflags = 0;
        } while (++s < end);
```

```
127         elf->ed_tlscn = --s;
128         s->s_next = 0;

130         /*
131          * Section index SHN_UNDEF (0) does not and cannot
132          * have a data buffer.  Fix it here.  Also mark the
133          * initial section as being allocated for the block
134          */

136         s = elf->ed_hdscn;
137         s->s_myflags = SF_ALLOC;
138         s->s_hdnode = 0;
139         s->s_tlnode = 0;
140         NOTE(NOW_VISIBLE_TO_OTHER_THREADS(*s))
141         return (OK_YES);
142 }


145 Okay
146 _elf_cookscn(Elf_Scn * s)
147 {
148         NOTE(ASSUMING_PROTECTED(*s, *(s->s_elf)))
149         Elf *                   elf;
150         Shdr *                  sh;
151         register Dnode *        d = &s->s_dnode;
152         size_t                  fsz, msz;
153         unsigned                work;

155         NOTE(NOW_INVISIBLE_TO_OTHER_THREADS(*d))
156         s->s_hdnode = s->s_tlnode = d;
157         s->s_err = 0;
158         s->s_shflags = 0;
159         s->s_uflags = 0;


162         /*
163          * Prepare d_data for inspection, but don't actually
164          * translate data until needed.  Leave the READY
165          * flag off.  NOBITS sections see zero size.
166          */
167         elf = s->s_elf;
168         sh = s->s_shdr;

170         d->db_scn = s;
171         d->db_off = sh->sh_offset;
172         d->db_data.d_align = sh->sh_addralign;
173         d->db_data.d_version = elf->ed_version;
174         ELFACCESSDATA(work, _elf_work)
175         d->db_data.d_type = _elf_mtype(elf, sh->sh_type, work);
176         d->db_data.d_buf = 0;
177         d->db_data.d_off = 0;
178         fsz = elf_fsize(d->db_data.d_type, 1, elf->ed_version);
179         msz = _elf_msize(d->db_data.d_type, elf->ed_version);
180         d->db_data.d_size = MAX(sh->sh_size, (sh->sh_size / fsz) * msz);
 42         d->db_data.d_size = (sh->sh_size / fsz) * msz;
181         d->db_shsz = sh->sh_size;
182         d->db_raw = 0;
183         d->db_buf = 0;
184         d->db_uflags = 0;
185         d->db_myflags = 0;
186         d->db_next = 0;

188         if (sh->sh_type != SHT_NOBITS)
189                 d->db_fsz = sh->sh_size;
190         else
```



```
191                 d->db_fsz = 0;

193         s->s_myflags |= SF_READY;

195         NOTE(NOW_VISIBLE_TO_OTHER_THREADS(*d))
196         return (OK_YES);
197 }
```

_____***unchanged_portion_omitted_***