

Re: [PATCH] modpost: detect unterminated device id lists

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-09/msg04511.html>

- *From:* Kees Cook <kees@xxxxxxxxxx>
 - *Date:* Sun, 16 Sep 2007 20:45:54 -0700
-

Hi Satyam,

On Mon, Sep 17, 2007 at 06:52:52AM +0530, Satyam Sharma wrote:

On 9/13/07, Kees Cook <kees@xxxxxxxxxx> wrote:

This patch against 2.6.23-rc6 will cause modpost to fail if any device id lists are incorrectly terminated, after reporting the offender.

Signed-off-by: Kees Cook <kees@xxxxxxxxxx>

Nice! :-)

BTW a very similar idea (but for a different problem) was discussed in: <http://lkml.org/lkml/2007/8/23/48>

I tried doing something about that, but gave up in between. For the device_id tables, a lot of infrastructure/code already exists in modpost, but no such luck for kobjects :- (Still, if you can do something about that, as he mentioned, I bet Greg would gladly accept such a patch :-)

Thanks! Hmm, perhaps I'll take that on when I learn kobjects better. :)

```
+static void device_id_check(const char *modname, const char *device_id,  
+ unsigned long size, unsigned long id_size,  
+ void *symval)
```

If you pass the Elf_Sym *sym all the way from handle_moddevtable() (which means you can get rid of the sym->st_size argument in the call chain), then it would be possible to print out the *symbol name* too here ...

That's true. I actually threw away an earlier version of this patch

Re: [PATCH] modpost: detect unterminated device id lists

that made some extensive changes to the elf parser (due to the NOBITS thing I explain below), but instead opted for the smaller version that stayed out of there.

```
for (p = symval+size-id_size; p < symval+size; p++) {  
    if (*p) {
```

is probably clearer ?

Ah, yeah, that's much nicer. I think I must have still have my ELF parser hat on when I wrote that. ;)

As I just said, printing out just the modname and device_id "type" sounds insufficient here. Note that they were sufficient before your patch, because previously, this function only checked if the device_id *type* itself was incorrectly defined. But here we're talking about a specific errant *symbol*.

That's true, yeah.

```
+ fprintf(stderr, "\n");  
+ fatal("%s: struct %s_device_id is not terminated "  
+ "with a NULL entry!\n", modname, device_id);
```

Subtle nit, but it's not really a "NULL" entry. It's an "empty object" entry, not a "NULL" pointer ... how about replacing "a NULL" with "an empty" ?

Yeah, that bugged me when I put that in too. :) Yes, "an empty" is better.

```
@@ -527,14 +542,22 @@ void handle_moddevtable(struct module *m  
Elf_Sym *sym, const char *symname)  
{  
    void *symval;  
    + char *zeros = NULL;  
  
    /* We're looking for a section relative symbol */  
    if (!sym->st_shndx || sym->st_shndx >= info->hdr->e_shnum)  
        return;  
  
    - symval = (void *)info->hdr  
    - + info->sechdrs[sym->st_shndx].sh_offset  
    - + sym->st_value;  
    + /* Handle all-NULL symbols allocated into .bss */
```

Re: [PATCH] modpost: detect unterminated device id lists

```
+ if (info->sechdrs[sym->st_shndx].sh_type & SHT_NOBITS) {  
+ zeros = calloc(1, sym->st_size);  
+ symval = zeros;  
+ }
```

Hmm, I don't quite grok this case. Care to explain?

In the ELF format, the .bss segment is initialized by the loader to all zeros, but it contains no in-file representation (since it's all zeros and would be a waste of space). Such segments are flags as "SHT_NOBITS" meaning that the loader must allocate cleared memory instead of loading the segment from the file.

In the case of modules that have a totally empty *_device_id list (?!), the compiler optimizes this into the .bss segment, since there is no need to store an all-zero-contents object in a segment that would be loaded from the file itself.

As a result, attempting to dereference such a symbol without noticing the SHT_NOBITS flag lands you somewhere in uninitialized memory. So, the above code basically side-steps the incorrect symbol location calculation and just aims it at a cleared part of memory.

As I mentioned, there was a larger patch that attempted to sort this out in the elf parser, but I didn't like it; it was big, not 100% correct, and the above approach seemed like a much less invasive change.

The cleanups you suggested, who should I send those to? Or will you (or Sam?) make them directly to the kbuild.git tree? (I've never poked at this part of the kernel source before... I'm unclear on the processes surrounding it maintainership.)

Thanks!

-Kees

--

Kees Cook

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>