CVE-2017-7612

描述：The check\_sysv\_hash function in elflint.c in elfutils 0.168 allows remote attackers to cause a denial of service (heap-based buffer over-read and application crash) via a crafted ELF file.

软件：elfutils 0.168

源码：src/elflint.c

出现位置： 1993-2004行、2044-2055行、2288-2298行、2300-2307行

关键源码：

--- a/src/elflint.c

+++ b/src/elflint.c

@@ -1,5 +1,5 @@

/\* Pedantic checking of ELF files compliance with gABI/psABI spec.

- Copyright (C) 2001-2015 Red Hat, Inc.

+ Copyright (C) 2001-2015, 2017 Red Hat, Inc.

This file is part of elfutils.

Written by Ulrich Drepper <drepper@redhat.com>, 2001.

@@ -1993,11 +1993,14 @@ check\_sysv\_hash (Ebl \*ebl, GElf\_Shdr \*shdr, Elf\_Data \*data, int idx,

Elf32\_Word nbucket = ((Elf32\_Word \*) data->d\_buf)[0];

Elf32\_Word nchain = ((Elf32\_Word \*) data->d\_buf)[1];

- if (shdr->sh\_size < (2 + nbucket + nchain) \* shdr->sh\_entsize)

- ERROR (gettext ("\

+ if (shdr->sh\_size < (2 + nbucket + nchain) \* sizeof (Elf32\_Word))

+ {

+ ERROR (gettext ("\

section [%2d] '%s': hash table section is too small (is %ld, expected %ld)\n"),

- idx, section\_name (ebl, idx), (long int) shdr->sh\_size,

- (long int) ((2 + nbucket + nchain) \* shdr->sh\_entsize));

+ idx, section\_name (ebl, idx), (long int) shdr->sh\_size,

+ (long int) ((2 + nbucket + nchain) \* sizeof (Elf32\_Word)));

+ return;

+ }

size\_t maxidx = nchain;

@@ -2044,11 +2047,14 @@ check\_sysv\_hash64 (Ebl \*ebl, GElf\_Shdr \*shdr, Elf\_Data \*data, int idx,

Elf64\_Xword nbucket = ((Elf64\_Xword \*) data->d\_buf)[0];

Elf64\_Xword nchain = ((Elf64\_Xword \*) data->d\_buf)[1];

- if (shdr->sh\_size < (2 + nbucket + nchain) \* shdr->sh\_entsize)

- ERROR (gettext ("\

+ if (shdr->sh\_size < (2 + nbucket + nchain) \* sizeof (Elf64\_Xword))

+ {

+ ERROR (gettext ("\

section [%2d] '%s': hash table section is too small (is %ld, expected %ld)\n"),

- idx, section\_name (ebl, idx), (long int) shdr->sh\_size,

- (long int) ((2 + nbucket + nchain) \* shdr->sh\_entsize));

+ idx, section\_name (ebl, idx), (long int) shdr->sh\_size,

+ (long int) ((2 + nbucket + nchain) \* sizeof (Elf64\_Xword)));

+ return;

+ }

size\_t maxidx = nchain;

@@ -2288,10 +2294,12 @@ section [%2d] '%s': hash table not for dynamic symbol table\n"),

section [%2d] '%s': invalid sh\_link symbol table section index [%2d]\n"),

idx, section\_name (ebl, idx), shdr->sh\_link);

- if (shdr->sh\_entsize != (tag == SHT\_GNU\_HASH

+ size\_t expect\_entsize = (tag == SHT\_GNU\_HASH

? (gelf\_getclass (ebl->elf) == ELFCLASS32

? sizeof (Elf32\_Word) : 0)

- : (size\_t) ebl\_sysvhash\_entrysize (ebl)))

+ : (size\_t) ebl\_sysvhash\_entrysize (ebl));

+

+ if (shdr->sh\_entsize != expect\_entsize)

ERROR (gettext ("\

section [%2d] '%s': hash table entry size incorrect\n"),

idx, section\_name (ebl, idx));

@@ -2300,7 +2308,7 @@ section [%2d] '%s': hash table entry size incorrect\n"),

ERROR (gettext ("section [%2d] '%s': not marked to be allocated\n"),

idx, section\_name (ebl, idx));

- if (shdr->sh\_size < (tag == SHT\_GNU\_HASH ? 4 : 2) \* (shdr->sh\_entsize ?: 4))

+ if (shdr->sh\_size < (tag == SHT\_GNU\_HASH ? 4 : 2) \* (expect\_entsize ?: 4))

{

ERROR (gettext ("\

section [%2d] '%s': hash table has not even room for initial administrative entries\n"),