GNU bug report logs - #78507
[Security] Heap Buffer Overflow in GNU Coreutils sort (CWE-122)

Previous Next

Package: coreutils;

Reported by: Med Maatallah <hotelsmaatallahrecemail <at> gmail.com>
Date: Tue, 20 May 2025 11:47:02 UTC
Severity: normal
Done: Pádraig Brady <P <at> draigBrady.com>
Bug is archived. No further changes may be made.

To add a comment to this bug, you must first **unarchive** it, by sending
a message to control AT debbugs.gnu.org, with *unarchive 78507* in the body.
You can then email your comments to 78507 AT debbugs.gnu.org in the normal way.

Toggle the display of automated, internal messages from the tracker.

View this report as an **mbox folder**, **status mbox**, **maintainer mbox**

Message #5 received at submit <at> debbugs.gnu.org (**full text**, **mbox**):

From: Med Maatallah <hotelsmaatallahrecemail <at> gmail.com>
To: bug-coreutils <at> gnu.org
Subject: [Security] Heap Buffer Overflow in GNU Coreutils sort (CWE-122)
Date: Tue, 20 May 2025 10:31:35 +0100

[**Message part 1** (text/plain, inline)]

```
Dear GNU Coreutils Maintainers,

I am reporting a heap buffer overflow vulnerability (CWE-122) I've
discovered in the GNU Coreutils sort utility. This issue affects the
traditional key specification syntax processing and leads to an
out-of-bounds read.
Vulnerability Details

The vulnerability occurs when the traditional key specification syntax (
+POS1[.C1][OPTS]) is used with UINTMAX_MAX as the character position value.
The begfield() function in src/sort.c performs unsafe pointer arithmetic
that leads to integer wraparound, resulting in a pointer that points one
byte before the start of an allocated heap buffer.

The vulnerable code is in the begfield() function in src/sort.c:


static char *begfield (struct line const *line, struct keyfield const *key){
```

```
  char *ptr = line->text, *lim = ptr + line->length - 1;
  size_t sword = key->sword;
  size_t schar = key->schar;

  /* The leading field separator itself is included in a field when -t
    is absent.  */

  if (tab != TAB_DEFAULT)
    while (ptr < lim && sword--)
      {
        while (ptr < lim && *ptr != tab)
          ++ptr;
        if (ptr < lim)
          ++ptr;
      }
  else
    while (ptr < lim && sword--)
      {
        while (ptr < lim && blanks[to_uchar (*ptr)])
          ++ptr;
        while (ptr < lim && !blanks[to_uchar (*ptr)])
          ++ptr;
      }

  /* If we're ignoring leading blanks when computing the Start     of
the field, skip past them here.  */
  if (key->skipsblanks)
    while (ptr < lim && blanks[to_uchar (*ptr)])
      ++ptr;

  /* Advance PTR by SCHAR (if possible), but no further than LIM. */
  ptr = MIN (lim, ptr + schar);

  return ptr;}
```

The issue lies in the expression ptr + schar when schar is set to UINTMAX_MAX (18446744073709551615 on 64-bit systems). This triggers integer wraparound due to size_t arithmetic, causing the calculation to effectively become ptr - 1. As a result, the function returns a pointer that's one byte before the start of the allocated buffer.

The vulnerability is exploitable when:

1. A user passes the key specification in traditional format (+0.18446744073709551615R)
2. During command-line parsing in main(), this sets key->schar to UINTMAX_MAX
3. In fillbuf(), the begfield() function is called to precompute key positions
4. The underflow occurs during the line key pointer calculation
5. The function returns a pointer before the buffer start

6. This invalid pointer is later passed through the call chain:
   - keycompare() function assigns the pointer to texta
   - When using -R (random sort), it calls compare_random()
   - compare_random() calls xstrxfrm() with the invalid pointer
   - xstrxfrm() calls strxfrm() on the out-of-bounds address
   - strxfrm() attempts to read the byte before the buffer, triggering the overflow

Technical Impact

This is a heap buffer overflow (read) that accesses memory one byte before an allocated buffer. The vulnerability could lead to program crashes and potentially information disclosure depending on the memory layout.
Proof of Concept

The vulnerability can be reliably reproduced with this simple test case:

bash

```
# Create a test file with any contentecho -e "aa\nbb" > poc_input.txt
# Execute vulnerable command (traditional key format + random sort option)
./sort +0.18446744073709551615R poc_input.txt
```

When compiled with AddressSanitizer, this command produces the following error:

[image: image.png]

The ASan output clearly shows that the issue is a READ one byte before a 672-byte heap-allocated region. The call stack confirms the path from begfield() through keycompare() and compare_random() to strxfrm().
Proposed Fix

A proper fix would involve checking for integer overflow before performing the pointer arithmetic in begfield(). Here's a suggested fix:

c

```
/* Inside begfield() *//* Advance PTR by SCHAR (if possible), but no
further than LIM. */if (schar > 0) {
  /* Check if adding schar would overflow or wrap negatively */
  if (SIZE_MAX - (uintptr_t)ptr < schar) {
    /* If it would overflow, safest is to set to end of current segment */
    ptr = lim;
  } else {
    ptr = MIN(lim, ptr + schar);
  }} else {
  /* Original behavior for schar == 0 */
  ptr = MIN(lim, ptr + schar);}
```

This fix guards against the integer overflow by checking if ptr + schar

```
would exceed the maximum representable size_t value, which indicates a
wraparound would occur.
Affected Versions

This vulnerability affects GNU Coreutils through at least version, and
potentially earlier versions. I've confirmed the issue in the current
development code.
CVE Request

I would like to request a CVE for this vulnerability.

Thank you for your attention to this matter.

Sincerely, Mohamed Maatallah (@Zephkek)
```

[**Message part 2** (text/html, inline)]

[**image.png** (image/png, inline)]

Message #8 received at 78507 <at> debbugs.gnu.org (**full text**, **mbox**):

From: Pádraig Brady <P <at> draigBrady.com>
To: Med Maatallah <hotelsmaatallahrecemail <at> gmail.com>, 78507 <at> debbugs.gnu.org
Subject: Re: bug#78507: [Security] Heap Buffer Overflow in GNU Coreutils sort (CWE-122)
Date: Tue, 20 May 2025 16:15:26 +0100

```
On 20/05/2025 10:31, Med Maatallah wrote:
> Dear GNU Coreutils Maintainers,
>
> I am reporting a heap buffer overflow vulnerability (CWE-122) I've
> discovered in the GNU Coreutils sort utility. This issue affects the
> traditional key specification syntax processing and leads to an
> out-of-bounds read.
> Vulnerability Details
>
> The vulnerability occurs when the traditional key specification syntax (
> +POS1[.C1][OPTS]) is used with UINTMAX_MAX as the character position value.
> The begfield() function in src/sort.c performs unsafe pointer arithmetic
> that leads to integer wraparound, resulting in a pointer that points one
> byte before the start of an allocated heap buffer.
>
> The vulnerable code is in the begfield() function in src/sort.c:
>
>
> static char *begfield (struct line const *line, struct keyfield const *key){
>    char *ptr = line->text, *lim = ptr + line->length - 1;
>    size_t sword = key->sword;
>    size_t schar = key->schar;
>
>    /* The leading field separator itself is included in a field when -t
```

```
>      is absent.  */
>
>    if (tab != TAB_DEFAULT)
>      while (ptr < lim && sword--)
>        {
>          while (ptr < lim && *ptr != tab)
>            ++ptr;
>          if (ptr < lim)
>            ++ptr;
>        }
>    else
>      while (ptr < lim && sword--)
>        {
>          while (ptr < lim && blanks[to_uchar (*ptr)])
>            ++ptr;
>          while (ptr < lim && !blanks[to_uchar (*ptr)])
>            ++ptr;
>        }
>
>    /* If we're ignoring leading blanks when computing the Start     of
> the field, skip past them here.  */
>    if (key->skipsblanks)
>      while (ptr < lim && blanks[to_uchar (*ptr)])
>        ++ptr;
>
>    /* Advance PTR by SCHAR (if possible), but no further than LIM. */
>    ptr = MIN (lim, ptr + schar);
>
>    return ptr;}
>
> The issue lies in the expression ptr + schar when schar is set to
> UINTMAX_MAX (18446744073709551615 on 64-bit systems). This triggers integer
> wraparound due to size_t arithmetic, causing the calculation to effectively
> become ptr - 1. As a result, the function returns a pointer that's one byte
> before the start of the allocated buffer.
>
> The vulnerability is exploitable when:
>
>     1. A user passes the key specification in traditional format (
>     +0.18446744073709551615R)
>     2. During command-line parsing in main(), this sets key->schar to
>     UINTMAX_MAX
>     3. In fillbuf(), the begfield() function is called to precompute key
>     positions
>     4. The underflow occurs during the line key pointer calculation
>     5. The function returns a pointer before the buffer start
>     6. This invalid pointer is later passed through the call chain:
>        - keycompare() function assigns the pointer to texta
>        - When using -R (random sort), it calls compare_random()
>        - compare_random() calls xstrxfrm() with the invalid pointer
>        - xstrxfrm() calls strxfrm() on the out-of-bounds address
```

> - strxfrm() attempts to read the byte before the buffer, triggering
> the overflow
>
> Technical Impact
>
> This is a heap buffer overflow (read) that accesses memory one byte before
> an allocated buffer. The vulnerability could lead to program crashes and
> potentially information disclosure depending on the memory layout.
> Proof of Concept
>
> The vulnerability can be reliably reproduced with this simple test case:
>
> bash
>
> # Create a test file with any contentecho -e "aa\nbb" > poc_input.txt
> # Execute vulnerable command (traditional key format + random sort option)
> ./sort +0.18446744073709551615R poc_input.txt
>
> When compiled with AddressSanitizer, this command produces the following
> error:
>
> [image: image.png]
>
> The ASan output clearly shows that the issue is a READ one byte before a
> 672-byte heap-allocated region. The call stack confirms the path from
> begfield() through keycompare() and compare_random() to strxfrm().
> Proposed Fix
>
> A proper fix would involve checking for integer overflow before performing
> the pointer arithmetic in begfield(). Here's a suggested fix:
>
> c
>
> /* Inside begfield() *//* Advance PTR by SCHAR (if possible), but no
> further than LIM. */if (schar > 0) {
> /* Check if adding schar would overflow or wrap negatively */
> if (SIZE_MAX - (uintptr_t)ptr < schar) {
> /* If it would overflow, safest is to set to end of current segment */
> ptr = lim;
> } else {
> ptr = MIN(lim, ptr + schar);
> }} else {
> /* Original behavior for schar == 0 */
> ptr = MIN(lim, ptr + schar);}
>
> This fix guards against the integer overflow by checking if ptr + schar
> would exceed the maximum representable size_t value, which indicates a
> wraparound would occur.
> Affected Versions
>
> This vulnerability affects GNU Coreutils through at least version, and

```
> potentially earlier versions. I've confirmed the issue in the current
> development code.
> CVE Request
>
> I would like to request a CVE for this vulnerability.
>
> Thank you for your attention to this matter.
>
> Sincerely, Mohamed Maatallah (@Zephkek)

Indeed. I introduced this in coreutils 7.2 (2009).
One can repro on Fedora for e.g. with:

_POSIX2_VERSION=200809 LC_ALL=C valgrind sort +0.18446744073709551615R poc_input.txt
==984625== Memcheck, a memory error detector
==984625== Using Valgrind-3.24.0 and LibVEX; rerun with -h for copyright info
==984625== Command: sort +0.18446744073709551615R poc_input.txt
==984625==
==984625== Invalid read of size 1

Going back to the more verbose code from coreutils 7.1 avoids the issue.
I'll test a bit more here and post a full patch in a while.

thanks!
Pádraig
```

Message #11 received at 78507 <at> debbugs.gnu.org (**full text**, **mbox**):

From: Pádraig Brady <P <at> draigBrady.com>
To: Med Maatallah <hotelsmaatallahrecemail <at> gmail.com>, 78507 <at> debbugs.gnu.org
Subject: Re: bug#78507: [Security] Heap Buffer Overflow in GNU Coreutils sort (CWE-122)
Date: Tue, 20 May 2025 18:15:48 +0100

[**Message part 1** (text/plain, inline)]

```
On 20/05/2025 16:15, Pádraig Brady wrote:
> Indeed. I introduced this in coreutils 7.2 (2009).
> One can repro on Fedora for e.g. with:
>
> _POSIX2_VERSION=200809 LC_ALL=C valgrind sort +0.18446744073709551615R poc_input.txt
> ==984625== Memcheck, a memory error detector
> ==984625== Using Valgrind-3.24.0 and LibVEX; rerun with -h for copyright info
> ==984625== Command: sort +0.18446744073709551615R poc_input.txt
> ==984625==
> ==984625== Invalid read of size 1
>
> Going back to the more verbose code from coreutils 7.1 avoids the issue.
```

> I'll test a bit more here and post a full patch in a while.

The attached patch addresses the issue here,
and includes a test verified to trigger with ASAN or valgrind available.
I'll push this later.

thanks,
Pádraig

[**sort-under-read.patch** (text/x-patch, attachment)]

Message #14 received at 78507 <at> debbugs.gnu.org (**full text**, **mbox**):

From: Paul Eggert <eggert <at> cs.ucla.edu>
To: Pádraig Brady <P <at> draigBrady.com>,
Med Maatallah <hotelsmaatallahrecemail <at> gmail.com>, 78507 <at> debbugs.gnu.org
Subject: Re: bug#78507: [Security] Heap Buffer Overflow in GNU Coreutils sort
(CWE-122)
Date: Tue, 20 May 2025 11:24:04 -0700

On 2025-05-20 10:15, Pádraig Brady wrote:

> The attached patch addresses the issue here,
> and includes a test verified to trigger with ASAN or valgrind available.

Thanks. A nit: the patch doesn't include the change to NEWS.

Message #19 received at 78507-done <at> debbugs.gnu.org (**full text**, **mbox**):

From: Pádraig Brady <P <at> draigBrady.com>
To: Paul Eggert <eggert <at> cs.ucla.edu>,
Med Maatallah <hotelsmaatallahrecemail <at> gmail.com>, 78507-done <at> debbugs.gnu.org
Subject: Re: bug#78507: [Security] Heap Buffer Overflow in GNU Coreutils sort
(CWE-122)
Date: Tue, 20 May 2025 19:37:25 +0100

On 20/05/2025 19:24, Paul Eggert wrote:
> On 2025-05-20 10:15, Pádraig Brady wrote:
>
>> The attached patch addresses the issue here,
>> and includes a test verified to trigger with ASAN or valgrind available.
>
> Thanks. A nit: the patch doesn't include the change to NEWS.

Good spot.
Fixed and pushed.
Marking this as done.

thanks,
Pádraig

---

bug archived. Request was from Debbugs Internal Request <help-debbugs <at> gnu.org> to internal_control <at> debbugs.gnu.org. (Wed, 18 Jun 2025 11:24:08 GMT) **Full text** and **rfc822 format** available.

This bug report was last modified 1 year and 5 days ago.

Previous **Next**
