Ruby master - Bug #15536
Crash on merging specific hashes using keyword splat
01/15/2019 08:42 AM - decuplet (Nikita Shilnikov)

Status: Closed
Priority: Normal
Assignee:
Target version:
ruby -v: ruby 2.5.3p105 (2018-10-18 revision 65156) [x86_64-linux]
Backport: 2.4: UNKNOWN, 2.5: DONE, 2.6: DONE

Description
Here's a snippet that leads to a crash on ruby 2.5.3. I tried to make it as small as possible.

```ruby
1000.times do
  **{
      a1: nil,
      a2: nil,
      a3: nil,
      a4: nil,
      a5: nil,
      a6: nil,
      a7: nil,
      a8: nil,
      a9: nil
  },
  b1: nil,
  b2: nil,
  a4: nil,
  **{
      c1: nil, c2: nil
  },
  a7: nil,
  a8: nil,
  a9: nil
}
eend
```

Results in *** Error in irb': malloc(): memory corruption: 0x000055ca6c832bd0 *** (more detail in the attached file).

We came across this on ruby 2.5.3 and as far as I can tell it's no longer a problem on 2.6 but we yet to upgrade.

Associated revisions
Revision ab2547d7 - 01/15/2019 02:19 PM - mame (Yusuke Endoh)

```
st.c (rb_hash_bulk_insert_into_st_table): avoid out-of-bounds write

"hash_bulk_insert" first expands the table, but the target size was wrong: it was calculated by "num_entries + (size to buld insert)", but it was wrong when "num_entries < entries_bound", i.e., it has a deleted entry. "hash_bulk_insert" adds the given entries from entries_bound, which led to out-of-bounds write access. [Bug #15536]

As a simple fix, this commit changes the calculation to "entries_bound + size". I'm afraid if this might be inefficient, but I think it is safe anyway.
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66832 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 66832 - 01/15/2019 02:19 PM - mame (Yusuke Endoh)

```
st.c (rb_hash_bulk_insert_into_st_table): avoid out-of-bounds write

"hash_bulk_insert" first expands the table, but the target size was wrong: it was calculated by "num_entries + (size to buld insert)", but it was wrong when "num_entries < entries_bound", i.e., it has a deleted entry. "hash_bulk_insert" adds the given entries from entries_bound, which led to out-of-bounds write access. [Bug #15536]

As a simple fix, this commit changes the calculation to "entries_bound + size". I'm afraid if this might be inefficient, but I think it is safe anyway.
```

02/23/2020
entry. "hash_bulk_insert" adds the given entries from entries_bound, which led to out-of-bounds write access. [Bug #15536]

As a simple fix, this commit changes the calculation to "entries_bound + size". I'm afraid if this might be inefficient, but I think it is safe anyway.

Revision a5dae936 - 01/17/2019 10:08 PM - naruse (Yui NARUSE)
merge revision(s) 66832: [Backport #15536]

st.c (rb_hash_bulk_insert_into_st_table): avoid out-of-bounds write

"hash_bulk_insert" first expands the table, but the target size was wrong: it was calculated by "num_entries + (size to build insert)", but it was wrong when "num_entries < entries_bound", i.e., it has a deleted entry. "hash_bulk_insert" adds the given entries from entries_bound, which led to out-of-bounds write access. [Bug #15536]

As a simple fix, this commit changes the calculation to "entries_bound + size". I'm afraid if this might be inefficient, but I think it is safe anyway.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_6@66853 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 66853 - 01/17/2019 10:08 PM - naruse (Yui NARUSE)
merge revision(s) 66832: [Backport #15536]

st.c (rb_hash_bulk_insert_into_st_table): avoid out-of-bounds write

"hash_bulk_insert" first expands the table, but the target size was wrong: it was calculated by "num_entries + (size to build insert)", but it was wrong when "num_entries < entries_bound", i.e., it has a deleted entry. "hash_bulk_insert" adds the given entries from entries_bound, which led to out-of-bounds write access. [Bug #15536]

As a simple fix, this commit changes the calculation to "entries_bound + size". I'm afraid if this might be inefficient, but I think it is safe anyway.

Revision b828c95b - 03/12/2019 10:01 PM - nagachika (Tomoyuki Chikanaga)
merge revision(s) 66832: [Backport #15536]

st.c (rb_hash_bulk_insert_into_st_table): avoid out-of-bounds write

"hash_bulk_insert" first expands the table, but the target size was wrong: it was calculated by "num_entries + (size to build insert)", but it was wrong when "num_entries < entries_bound", i.e., it has a deleted entry. "hash_bulk_insert" adds the given entries from entries_bound, which led to out-of-bounds write access. [Bug #15536]

As a simple fix, this commit changes the calculation to "entries_bound + size". I'm afraid if this might be inefficient, but I think it is safe anyway.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@67236 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 67236 - 03/12/2019 10:01 PM - nagachika (Tomoyuki Chikanaga)
merge revision(s) 66832: [Backport #15536]

st.c (rb_hash_bulk_insert_into_st_table): avoid out-of-bounds write

"hash_bulk_insert" first expands the table, but the target size was wrong: it was calculated by "num_entries + (size to build insert)", but it was wrong when "num_entries < entries_bound", i.e., it has a deleted entry. "hash_bulk_insert" adds the given entries from entries_bound, which led to out-of-bounds write access. [Bug #15536]

As a simple fix, this commit changes the calculation to "entries_bound + size". I'm afraid if this might be inefficient, but I think it is safe anyway.

02/23/2020
Good catch. The following code still crashes on trunk.

```
**{
    a0: nil,
    a1: nil,
    a2: nil,
    a3: nil,
    a4: nil,
    a5: nil,
    a6: nil,
    a7: nil,
    a8: nil,
},

a0: nil,
    a1: nil,
    a2: nil,
    a3: nil,
    a4: nil,
    a5: nil,
    a6: nil,
    a7: nil,
    a8: nil,
**{
    c: nil
},

b0: nil,
    b1: nil,
    b2: nil,
    b3: nil,
    b4: nil,
    b5: nil,
    b6: nil,
    b7: nil,
    b8: nil,
    b9: nil,
    b10: nil,
    b11: nil,
    b12: nil,
    b13: nil,
    b14: nil,
    b15: nil,
    b16: nil,
    b17: nil,
    b18: nil,
    b19: nil,
    b20: nil,
    b21: nil,
}
```

Here is a patch. It might have an inefficient case, but I think it is easy to backport.

diff --git a/st.c b/st.c
index c6b3644e39..ed235c674e 100644
--- a/st.c
+++ b/st.c
@@ -2299,7 +2299,7 @@ rb_hash_bulk_insert_into_st_table
     st_table *tab = RHASH_ST_TABLE(hash);
     st_table *tab = RHASH_ST_TABLE(hash);
     st_table *tab = RHASH_ST_TABLE(hash);
-    n = tab->num_entries + size;
+    n = tab->entries_bound + size;
     st_expand_table(tab, n);
     if (UNLIKELY(tab->num_entries))
         st_insert_generic(tab, argc, argv, hash);
```

#2 - 01/15/2019 02:19 PM - mame (Yusuke Endoh)

- Status changed from Open to Closed

Applied in changeset trunkr66832.
st.c (rb_hash_bulk_insert_into_st_table): avoid out-of-bounds write

"hash_bulk_insert" first expands the table, but the target size was wrong: it was calculated by "num_entries + (size to build insert)". but it was wrong when "num_entries < entries_bound", i.e., it has a deleted entry. "hash_bulk_insert" adds the given entries from entries_bound, which led to out-of-bounds write access. [Bug #15536]

As a simple fix, this commit changes the calculation to "entries_bound + size". I'm afraid if this might be inefficient, but I think it is safe anyway.

#3 - 01/15/2019 02:20 PM - mame (Yusuke Endoh)
- Backport changed from 2.4: UNKNOWN, 2.5: UNKNOWN, 2.6: UNKNOWN to 2.4: UNKNOWN, 2.5: REQUIRED, 2.6: REQUIRED

#4 - 01/15/2019 03:34 PM - decuplet (Nikita Shilnikov)
That was fast, thank you.

#5 - 01/17/2019 10:09 PM - naruse (Yui NARUSE)
- Backport changed from 2.4: UNKNOWN, 2.5: REQUIRED, 2.6: REQUIRED to 2.4: UNKNOWN, 2.5: REQUIRED, 2.6: DONE

ruby_2_6 r66853 merged revision(s) 66832.

#6 - 03/12/2019 10:01 PM - nagachika (Tomoyuki Chikanaga)
- Backport changed from 2.4: UNKNOWN, 2.5: REQUIRED, 2.6: DONE to 2.4: UNKNOWN, 2.5: DONE, 2.6: DONE

ruby_2_5 r67236 merged revision(s) 66832.

Files
segfault.txt 30.8 KB 01/15/2019 decuplet (Nikita Shilnikov)