New issue Jump to bottom

Double free in Vec::from_iter specialization when drop panics #83618

Closed **Qwaz** opened this issue on Mar 28, 2021 · 2 comments

Assignees

Labels A-collections A-destructors C-bug I-unsound P-critical T-libs

Qwaz commented on Mar 28, 2021 • edited by rustbot Contributor

rust/library/alloc/src/vec/source_iter_marker.rs
Lines 71 to 72 in 4a20eb6

```
// drop any remaining values at the tail of the source
src.drop_remaining();
```

rust/library/alloc/src/vec/into_iter.rs
Lines 88 to 93 in 4a20eb6

```
pub(super) fn drop_remaining(&mut self) {
    unsafe {
        ptr::drop_in_place(self.as_mut_slice());
    }
    self.ptr = self.end;
}
```

`SpecFromIter<T, I> for Vec<T>` calls `Vec::IntoIter::drop_remaining()`. `drop_remaining()` calls `drop_in_place()` before overwriting the pointer. As a result, dropped elements are not invalidated and dropped again under panic.

PoC:

```
#![forbid(unsafe_code)]

use std::iter::FromIterator;

#[derive(Debug)]
enum MyEnum {
    DroppedTwice(Box<i32>),
    PanicOnDrop,
}

impl Drop for MyEnum {
    fn drop(&mut self) {
        match self {
            MyEnum::DroppedTwice(_) => println!("Dropping!"),
            MyEnum::PanicOnDrop => {
                if !std::thread::panicking() {
                    panic!();
                }
            }
        }
    }
}

fn main() {
    let v = vec![MyEnum::DroppedTwice(Box::new(123)), MyEnum::PanicOnDrop];
    Vec::from_iter(v.into_iter().take(0));
}
```

Output:

```
Dropping!
thread 'main' panicked at 'explicit panic', src/main.rs:17:21
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
Dropping!
free(): double free detected in tcache 2
```

Tested with `rustc 1.51.0`. Here is a playground link to the code snippet.

👍 2 ❤️ 2

Qwaz added the C-bug label on Mar 28, 2021

jonas-schievink added A-collections A-destructors I-unsound T-libs labels on Mar 28, 2021

rustbot added the I-prioritize label on Mar 28, 2021

the8472 commented on Mar 28, 2021 Contributor

Thanks, I totally didn't consider that case!
According to #14875 panic during drop should result in the storage being leaked. That shouldn't be too difficult to achieve by forgetting the original allocation before the drop.

The question is whether it is necessary to attempt to drop the stuff moved to output vec or whether we can leak those too.

@rustbot claim

rustbot assigned **the8472** on Mar 28, 2021

the8472 mentioned this issue on Mar 28, 2021

Fix double-drop in `Vec::from_iter(vec.into_iter())` specialization when items drop during panic #83629

Closed

JohnTitor commented on Mar 29, 2021 Member

Assigning `P-critical` as discussed as part of the Prioritization Working Group procedure and removing `I-prioritize`.

JohnTitor added **P-critical** and removed I-prioritize labels on Mar 29, 2021

Dylan-DPC-zz pushed a commit to Dylan-DPC-zz/rust that referenced this issue on Apr 1, 2021

`Rollup merge of rust-lang#83629 - the8472:fix-inplace-panic-on-drop, …` … `2433ef1`

Dylan-DPC-zz pushed a commit to Dylan-DPC-zz/rust that referenced this issue on Apr 2, 2021

`Rollup merge of rust-lang#83629 - the8472:fix-inplace-panic-on-drop, …` … `e25e2b9`

bors closed this as completed in `542f441` on Apr 2, 2021

zarniwhoop73 mentioned this issue on May 2, 2021

Will 1.52.0 fix CVE-2021-31162 ? #84847

Closed

Mark-Simulacrum mentioned this issue on Sep 26, 2021

Never allow unwinding from Drop impls rust-lang/lang-team#97

Closed

the8472 mentioned this issue on Sep 10

In-place optimisation in IntoIterator can lead to memory leak #101628

Closed

Assignees

the8472

Labels

A-collections A-destructors C-bug I-unsound P-critical T-libs

Projects

None yet

Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

Fix double-drop in `Vec::from_iter(vec.into_iter())` specialization when items drop during panic
the8472/rust

5 participants