Out-of-bound writes possible on `Heap` initialization and `Heap::extend`

High **phil-opp** published **GHSA-xg8p-34w2-j49j** on Sep 6

Package

linked_list_allocator (Rust)

Affected versions	Patched versions
<=0.10.1	0.10.2

Description

Impact

What kind of vulnerability is it? Who is impacted?

This vulnerability impacts all the initialization functions on the `Heap` and `LockedHeap` types, including `Heap::new`, `Heap::init`, `Heap::init_from_slice`, and `LockedHeap::new`. It also affects multiple uses of the `Heap::extend` method.

Initialization Functions

The heap initialization methods were missing a minimum size check for the given heap size argument. This could lead to ***out-of-bound writes*** when a heap was initialized with a size smaller than `3 * size_of::<usize>` because of metadata write operations.

`Heap::extend`

This vulnerability impacts three specific uses of the `Heap::extend` method:

- When calling `Heap::extend` with a size smaller than two `usize`s (e.g., 16 on `x86_64`), the size was erroneously rounded up to the minimum size, which could result in an ***out-of-bounds write***.
- Calling `Heap::extend` on an empty heap tried to construct a heap starting at address 0, which is also an ***out-of-bounds write***.

    - One specific way to trigger this accidentally is to call `Heap::new` (or a similar constructor) with a heap size that is smaller than two `usize`s. This was treated as an empty heap as well.
- Calling `Heap::extend` on a heap whose size is not a multiple of the size of two `usize`s resulted in *unaligned writes*. It also left the heap in an unexpected state, which might lead to subsequent issues. We did not find a way to exploit this undefined behavior yet (apart from DoS on platforms that fault on unaligned writes).

Patches

Has the problem been patched? What versions should users upgrade to?

We published a patch in version `0.10.2` and recommend all users to upgrade to it. This patch release includes the following changes:

- The initialization functions now panic if the given size is not large enough to store the necessary metadata. Depending on the alignment of the heap bottom pointer, the minimum size is between `2 * size_of::<usize>` and `3 * size_of::<usize>`.
- The `extend` method now panics when trying to extend an unitialized heap.
- Extend calls with a size smaller than `size_of::<usize>() * 2` are now buffered internally and not added to the list directly. The buffered region will be merged with future `extend` calls.
- The `size()` method now returns the *usable* size of the heap, which might be slightly smaller than the `top() - bottom()` difference because of alignment constraints.

Workarounds

Is there a way for users to fix or remediate the vulnerability without upgrading?

To avoid this issue, ensure that the heap is only initialized with a size larger than `3 * size_of::<usize>` and that the `Heap::extend` method is only called with sizes larger than `2 * size_of::<usize>()`. Also, ensure that the total heap size is (and stays) a multiple of `2 * size_of::<usize>()`.

For more information

If you have any questions or comments about this advisory:

- Open an issue in this repository
- Email **@phil-opp** at security@phil-opp.com

Acknowledgements

This issue was responsibly reported by Evan Richter at ForAllSecure and found with Mayhem and cargo fuzz.

Severity

High

CVE ID

CVE-2022-36086

Weaknesses

CWE-119 CWE-787

Credits

 evanrichter