Alignment on a tagged type may cause invalid deallocation

ADACORE SECURITY ADVISORY

Document Id: SEC.UB10-053

Version 1 - Feb 02, 2023

AdaCore

Title	Alignment on a tagged type may cause invalid deallocation	
Status	Final	
Author	Johannes Kliemann	
Reviewed by	Alexander Senier, Olivier Ramonat	

Revision History

Version	**Date**	**Comments**
1	Feb 02, 2023	initial version

Contents

1. Preface

1.1. Scope

This document is an advisory describing the security impact of an invalid deallocation caused by an alignment clause on a tagged type or one of its components that is larger than `Standard'System_Allocator_Alignment`. The issue is tracked under the ticket number UB10-053 in AdaCore's issue tracking database. This document also presents possible workarounds and mitigations for the issue.

1.2. Distribution

This advisory is made available in confidence to AdaCore customers under embargo until 2022-05-01 so that they can address the issue it describes before public availability. Thereafter, it will be available to the general public under the terms of the CC BY-ND 4.0 licence.

1.3. Contact

For questions on this document, please contact AdaCore support at product-security@adacore.com or using the standard reporting procedures if you are an AdaCore customer.

2. Vulnerability

2.1. Affected Products

The vulnerability described in this document was reported for the following product versions:

- GNAT 21.2

The vulnerability described in this document applies to the following product versions:

- GNAT < 23.0

2.2. Severity and Impact

CVSS v3.1 score: 6.7 (medium) (AV:N/AC:H/PR:N/UI:N/S:C/C:N/I:L/A:H/E:U/RL:W/RC:C)

This issue can possibly cause use-after-free bugs since the deallocated memory may still be used by the program which is unaware that this memory has been freed. This may either lead to denial of service in case of a crash but could also lead to an information leak or code execution vulnerability. An attacker may be able to reallocate or control the freed memory and thereby observe or even influence the program execution. While this issue is not always controllable a sophisticated attacker with sufficient knowledge about the program may be able to execute a targeted attack.

2.3. Detailed Description

The problem is caused by a difference in alignment for the allocation and deallocation of tagged objects. When the allocator is passed an expression of a class-wide type and the dynamic instance of that class-wide expression has larger alignment than its parent the allocator will wrongly choose the alignment of the parent type instead that of the instance. However when the object is deallocated the correct alignment is passed to the deallocation. If the specified alignment is larger than what was chosen at allocation time the resulting deallocation will be done on a different address.

The following code demonstrates the issue:

```ada
type A is tagged record
   V : Integer;
end record with
   Alignment => 16;

type B is new A with null record with
   Alignment => 32;

B_Ptr : access all A'Class := new A'Class (B'(V => 0));
```

In this example the allocator will use the alignment of `A` for the allocation of `B_Ptr` even though it points to an object of `B` which has a different alignment. When `B_Ptr` is freed, the deallocator will use the alignment of `B` and deallocate a different address than what has been allocated before.

3. Solution

3.1. Workarounds

To prevent this from happening the alignment of a controlled type should not be larger than `Standard'System_Allocator_Alignment` or else a type extension should not be more aligned than its parent type. This can be enforced statically by a `pragma Compile_Time_Error`.

3.2. Correction

The vulnerability described in this document is corrected in the following product versions:

- GNAT Pro >= 23.0

This previous versions the issue can be mitigated by either preventing super-alignment (alignment larger than `Standard'System_Allocator_Alignment`) or making sure that the entire hierarchy has the same alignment.