Memory corruption in GNAT.Sockets.Get_Host_By_Name

ADACORE SECURITY ADVISORY

Document Id: SEC.UB16-046

Version 1 - Feb 02, 2023

AdaCore

Title	Memory corruption in GNAT.Sockets.Get_Host_By_Name	
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 a memory corruption bug in GNAT.Sockets. The issue is tracked under the ticket number UB16-046 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 2023-05-01 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 Pro 7.4.2

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

- GNAT Pro <= 21.2

2.2. Severity and Impact

CVSS v3.1 score: 8.0 (high) (AV:N/AC:H/PR:N/UI:N/S:C/C:L/I:H/A:H/E:P/RL:O/RC:C)

This issue can possibly cause a stack overflow when resolving hostnames longer than 64 bytes.

2.3. Detailed Description

In the GNAT.Sockets function Get_Host_By_Name a stack overflow can be caused if a hostname longer than 64 byte is resolved. To successfully trigger the problem, the hostname must be resolvable through either the local hosts file or a DNS server. When it is resolved it is copied into a variant record which is constrained by a specific length type. This type has a maximum value of 64 bytes. Passing a string that is longer leads to a stack overflow that can cause an erroneous memory access. Depending on the memory layout an attacker may be able to modify the stack by tricking the application into resolving a malicious domain.

3. Solution

3.1. Workarounds

As a workaround the runtime can be compiled with assertions enabled. This does not prevent the application from crashing but it terminates it properly before the memory corruption happens and therefore prevents it from going into an undefined state.

3.2. Correction

The problem has been fixed by increasing the maximum size of a hostname to NI_MAXHOST as defined in netdb.h.

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

- GNAT Pro >= 22.1

4. Appendix

4.1. CVSS Score Justification

Metric	Justification
AV:N	Both triggering to resolve a specific DNS name as providing a resolver for this can be done over the network.
AC:H	While causing a crash only requires a long DNS hostname, impacting the integrity of the application without causing a crash requires a good knowledge of the applications memory layout.
PR:N	The highest privileges required are the privileges needed to cause the application to resolve a hostname. It is safe to assume that no privileges are needed for that.
UI:N	It is safe to assume that an application will resolve hostnames without any user interaction.
S:C	If the attacked application runs in a privileged mode, an attacker may be able to circumvent access restrictions.
C:L	Leaking memory from the application is only possible if an attacker can abuse code inside the application to do so. Reading memory directly is not possible.
I:H	An attacker can modify memory and may be able to circumvent access restrictions.
A:H	The attack can easily be used to cause an exception via a stack or buffer overflow, leading to a crash.
E:P	A proof of concept has been created to validate the issue.
RL:O	A patch fixing this issue has already been merged.
RC:C	The report has been confirmed in the analysis done for this advisory.