Bidirectional Unicode characters in Ada source code

ADACORE SECURITY ADVISORY
Document Id: SEC.UB03-050

Version 1 - Feb 10, 2022
<table>
<thead>
<tr>
<th>Title</th>
<th>Bidirectional Unicode characters in Ada source code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Final</td>
</tr>
<tr>
<td>Author</td>
<td>Johannes Kliemann</td>
</tr>
<tr>
<td>Reviewed by</td>
<td>Alexander Senier, Romain Berrendonner</td>
</tr>
</tbody>
</table>

**Revision History**

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Feb 10, 2022</td>
<td>initial version</td>
</tr>
</tbody>
</table>
Contents

1 Preface 4
  1.1 Scope ........................................... 4
  1.2 Distribution ..................................... 4
  1.3 Contact ......................................... 4

2 Vulnerability 5
  2.1 Affected Products ................................... 5
  2.2 Severity and Impact .................................. 5
  2.3 Detailed Description .................................. 5

3 Solution 7
  3.1 Workarounds ....................................... 7
  3.2 Correction ......................................... 7
1. Preface

1.1. Scope

This document is an advisory describing the security impact of the Unicode Bidirectional Algorithm on GNAT, GNAT Studio and GNATbench. The issue is tracked under the ticket number UB03-050 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 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
- GNAT Studio 21.2
- GNATbench 21.2

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

- GNAT <= 21.2
- GNAT Studio <= 21.2
- GNATbench <= 21.2

2.2. Severity and Impact

CVSS v3.1 score: 5.9 (medium) (AV:N/AC:H/PR:N/UI:N/S:C/L/I:L/A:L/E:P/RL:T/RC:C)

Without mitigation this issue can be used to implant malicious code into a code base without the developers’ and reviewers’ knowledge. This may also include backdoors in security and safety critical software.

Exploiting this issue requires a targeted attack to a specific code base. The attack involves both technical knowledge to craft a working change for a backdoor but also social skills to disguise it sufficiently in a comment and place it successfully into a code review.

A successful attack may compromise confidentiality, integrity and availability of any entity that uses the software compiled from the affected code base.

2.3. Detailed Description

This description is based on CVE-2021-42574.

Unicode allows control sequences that instruct a text renderer to change the direction of block of text inside a line or to reorder blocks of text inside a line. While this behaviour is desired to properly display left to right, right to left languages and a mix of both it can be used to hide malicious code by making it look like a comment.

The following line of Ada code clearly does not contain a comment:

```ada
S : constant String := ("Hello World -- This is not a comment");
```

However by inserting the Unicode code point U+202E (Right-To-Left-Override) before the comment symbol and the end of the string without restoring the left to right order afterwards may cause a text renderer to display the code with a different semantics. As this is purely a display issue the compiler still treats the whole entry as a string:

```ada
S : constant String := ("Hello World <U+202E>-- This is not a comment<U+202E>");
```

A text editor or reviewing tool may display the code differently, making a human think that the second part of the line is actually a comment.

```ada
S : constant String := ("Hello World");-- This is not a comment
```

This leads to an inconsistency between the code shown by the editor or reviewing tool and the code that is actually compiled by the compiler allowing an attacker to implant malicious code by disguising it as a comment.

The Unicode characters affected by this are listed in the following table.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Code Point</th>
<th>UTF Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRE</td>
<td>U+202a</td>
<td>e2 80 aa</td>
</tr>
<tr>
<td>RLE</td>
<td>U+202b</td>
<td>e2 80 ab</td>
</tr>
<tr>
<td>LRO</td>
<td>U+202d</td>
<td>e2 80 ad</td>
</tr>
<tr>
<td>RLO</td>
<td>U+202c</td>
<td>e2 80 ae</td>
</tr>
<tr>
<td>LRI</td>
<td>U+2066</td>
<td>e2 81 a6</td>
</tr>
<tr>
<td>RLI</td>
<td>U+2067</td>
<td>e2 81 a7</td>
</tr>
<tr>
<td>FSI</td>
<td>U+2068</td>
<td>e2 81 a8</td>
</tr>
<tr>
<td>PDF</td>
<td>U+202c</td>
<td>e2 80 ac</td>
</tr>
<tr>
<td>PDI</td>
<td>U+2069</td>
<td>e2 81 a9</td>
</tr>
</tbody>
</table>
3. Solution

3.1. Workarounds

This issue can be mitigated in the following ways:

- Using an editor or code review tool that either already detects and warns if these control sequences are used or that does not render but simply display them.

- Check all sources for the presence of this character e.g. by using `grep $'\u202e'`. This can be applied to all characters listed in the description above.

3.2. Correction

This issue is worked on in GNAT Studio and GNATbench since it is a rendering issue. The fix is done by displaying a warning to the user if such a character is detected in a source file.

While technically speaking GNAT is not directly affected as it compiles the string as it is written in bytes correctly it can still be a place to mitigate this issue. With wavefront 20211005 and later GNAT will issue a warning when compiling code containing the affected Unicode characters.