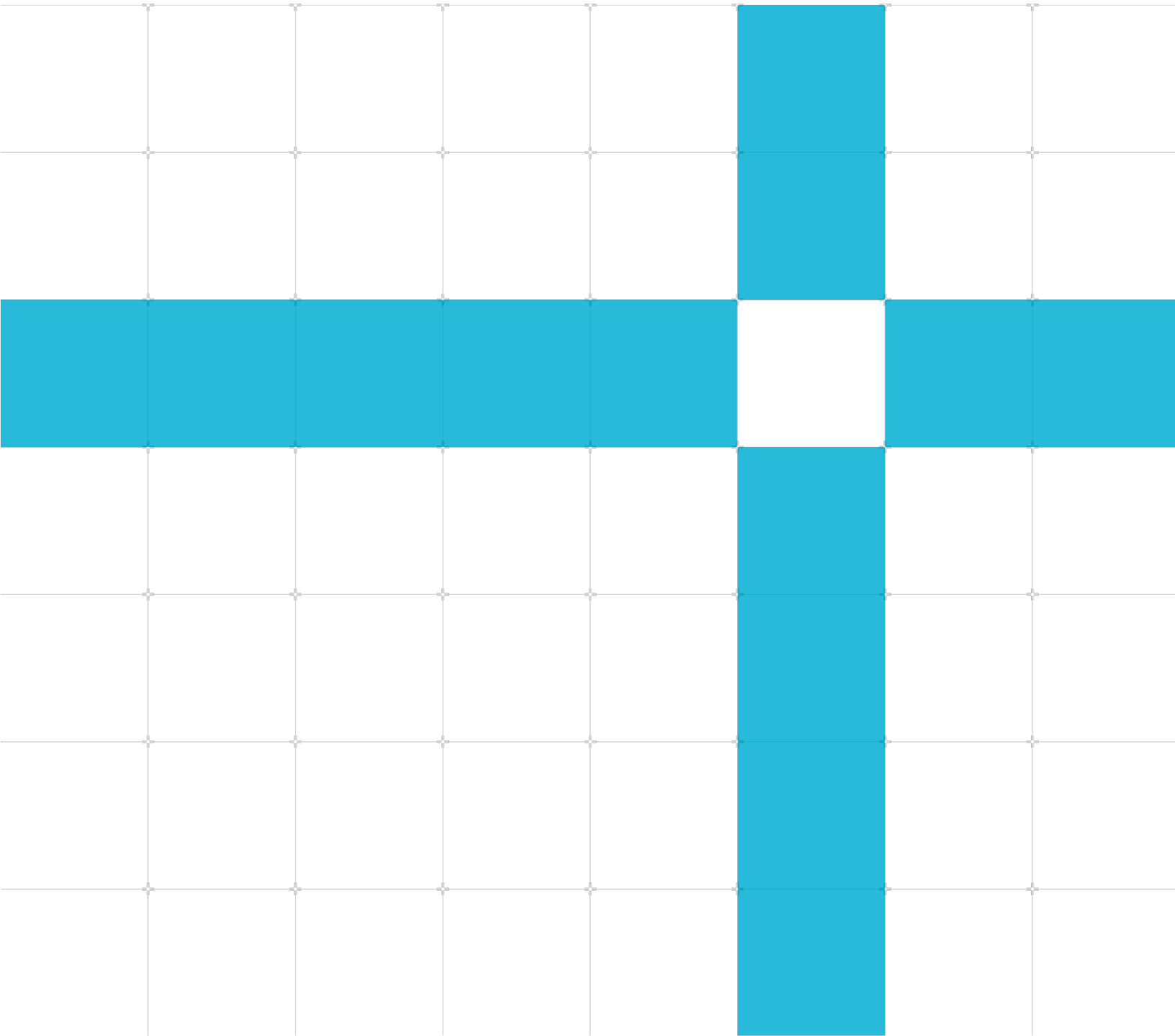
Arm Compiler for Linux

Product revision: 22.0.2

Release Note

Non-Confidential

Document ID:
107578



Arm Compiler for Linux

Release Note

Non-Confidential Proprietary Notice

Arm Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.
(LES-PRE-20349)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Arm Compiler for Linux, create a ticket on **https://support.developer.arm.com**.

To provide feedback on the document, fill the following survey:
https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

To report offensive language in this document, email **terms@arm.com**.

Contents

1 Release overview 5
1.1 Product description 5
1.2 Release status 5

2 Release contents 6
2.1 Licensing information 6
2.2 Prerequisites 6
2.3 Download the product 6
2.4 Unpack the product 7
2.5 Directory structure 7
2.6 Install the product 7
2.7 RPM and DEB files 8
2.8 Run the product 8
2.9 Examples 9
2.10 Uninstall 10
2.11 Deliverables 10
2.12 Differences from previous release 11
2.12.1 Additions and changes 11
2.12.2 Resolved issues 14
2.13 Known limitations 15

3 Support 16
3.1 OS 16

4 Release history 17

5 Conventions 18
5.1 Glossary 18

1 Release overview

The following sections describe the product that this release note describes and its quality status at time of release.

1.1 Product description

The Arm Compiler for Linux 22.0.2 suite provides a complete compiling environment for natively developing and tuning your server and HPC applications on Arm-based platforms.

The suite contains the following packages:

- Arm C/C++/Fortran Compiler 22.0.2

  Arm Compiler is a Linux user-space C/C++ and Fortran compiler tailored for scientific computing, HPC, and enterprise workloads.

- Arm Performance Libraries 22.0.2

  Arm Performance Libraries contains optimized math functions, such as linear algebra and Fast Fourier Transforms, for Arm AArch64 implementations, including those with SVE. It is compatible with Arm C/C++/Fortran Compiler 22.0.2 and GCC 11.2.0.

  Arm Performance Libraries is optimized for a number of microarchitectures. The latest information is available on the Arm Developer website:

  https://developer.arm.com/Tools%20and%20Software/Arm%20Compiler%20for%20Linux#Supported-Devices

- GCC 11.2.0

  For convenience, and to provide the optimal experience of using Arm Performance Libraries and GCC on the latest Arm server and HPC systems, a build of GCC 11.2.0 is provided. The GCC 11.2.0 build is also provided for OpenMP/libgfortran compatibility with Arm Performance Libraries.

  As a GNU tool suite, the GPL-licensed source code can be downloaded separately.

1.2 Release status

This is the 22.0.2 release of the Arm Compiler for Linux software.

These deliverables are being released under the terms of the agreement between Arm and each licensee (the "Agreement"). All planned verification and validation is complete.

The release is suitable for volume production under the terms of the Agreement.

2 Release contents

The following sections describe:

- Downloading and unpacking the product.
- The contents of this release.
- Any changes since the previous release.
- Any known issues and limitations that exist at the time of this release.

2.1 Licensing information

Use of Arm Compiler for Linux is subject to the terms and conditions of the applicable End User License Agreement (“EULA”). A copy of the EULA can be found in the 'license_terms' folder of your product installation.

You do not require a license to use this Arm Compiler for Linux package.

2.2 Prerequisites

If any of the following tools are not already installed by your Linux distribution, you must install them before installing Arm Compiler for Linux:

- Python (version 2.7 or later)

  To ensure you have the necessary version of Python for your system, install Python using the appropriate package manager for your OS.

- C Libraries:

  SLES and RHEL: glibc-devel

  Ubuntu: libc6-dev

You must have at least 2 GB of free hard disk space to both download and unpack the Arm Compiler for Linux package. You must also have an additional 6 GB of free space to install the package.

2.3 Download the product

Arm delivers the files through the Arm Developer website:

https://developer.arm.com/Tools%20and%20Software/Arm%20Compiler%20for%20Linux#Technical-Specifications

2.4 Unpack the product

To unpack the package, extract the tar file contents using a tar utility:

```
tar -xvf <package_name>.tar
```

2.5 Directory structure

Shows the top-level directory structure of this installer package, which is available after you unpack the bundle:

```
license_terms/
arm-compiler-for-linux-22.0.2*.sh
RELEASE_NOTES.txt
```

2.6 Install the product

To install Arm Compiler for Linux, navigate into the extracted package directory (<package_name>) and run the installation script as a privileged user. Pass any options to configure the installation:

```
cd path/to/<package_name>/
./arm-compiler-for-linux-22.0.2*.sh [option]...
```

Some common installation options are:

- For a headless installation and to automatically accept the EULA, use the '--accept' option.

- To install to an alternate location to the default, use the '--install-to <install_location>' option.

For a full list of supported installation options pass the '-h' or '--help' options to the installer script.

To learn more about installing Arm Compiler for Linux, see:

https://developer.arm.com/documentation/102621/latest/

The installer displays the EULA and prompts you to agree to the terms. Type 'yes' at the prompt to continue.

All the packages are unpacked to <install_location>/<package_name> with environment modulefiles available under <install_location>/modulefiles. The default installation location is /opt/arm/. Local installs have the same directory structure starting from your chosen installation root.

2.7 RPM and DEB files

The install packages contain RPM (.rpm) files, for Red Hat-based Linux distributions, or DEB (.deb) files, for Debian-based Linux distributions.

To extract the .rpm or .deb files from the installer, run the installer script with the '-s' or '--save-packages-to <directory_location>' option. If <directory_location> is not an empty directory, you also need to include the '-f' or '--force' option. The installer script requires you to accept the EULA. If you accept the EULA, the .rpm or .deb files extract to <directory_location>.

RPM files are signed by Arm's HPC GPG key. DEB files are not signed. To verify RPM files, you can download and import the Arm's HPC GPG key, and check the signatures:

1. Download the Arm HPC GPG public key from: https://developer.arm.com/-/media/files/keys/GPG-PUB-KEY-ARM-HPC-SW-TOOLS.PUB
2. Import the GPG key, run: rpm --import GPG-PUB-KEY-ARM-HPC-SW-TOOLS.PUB
3. Check the signature of an .rpm file, run: rpm -K <rpm_file>

To install an RPM file, use 'rpm -i <rpm_file>'. To install a DEB file, use 'dpkg -i <deb_file>'.

Note: Arm does not recommend that you install directly from the .rpm or .deb files. Only experienced users who are comfortable with this type of installation route should attempt to install the Arm Compiler for Linux package using this method.

2.8 Run the product

1. Load the environment module:

   For Arm C/C++/Fortran Compiler, use:

   ```
   module load acfl/22.0.2
   ```

   To also use Arm Performance Libraries, include the ``-armpl`` compiler option when linking your executable. You do not need to load the Arm Performance Libraries modulefile.

   For GCC 11.2.0 only, use:

   ```
   module load gnu/11.2.0
   ```

   For GCC 11.2.0 with Arm Performance Libraries, use:

   ```
   module load gnu/11.2.0
   module load armpl/22.0.2
   ```

2. Generate your executable binary.

   To generate an executable binary with Arm Compiler for Linux, compile your program with Arm C/C++/Fortran Compiler and specify any options ([options]), the output binary name (-o <binary>), and the input file (<input>):

   {armclang|armclang++|armflang} [options] -o <binary> <input>

   Refer to the GCC documentation to see the equivalent command syntax for the GCC compiler.

2.9 Examples

Example code is included in this suite as part of Arm Performance Libraries. This code can be found at:

<install_location>/<ARMPL_Name>*<ARMPL_Version>*/examples*

Examples that use, and do not use, SVE are included for each of Arm C/C++/Fortran Compiler and GCC.

Multiple examples directories are provided in the installation. The suffix of the directory name indicates whether the examples inside link to the 32-bit ('_lp64') or 64-bit ('_ilp64') integer variants, and sequential (no suffix indicator) or OpenMP ('_mp') multi-threaded variants, of Arm Performance Libraries.

The default set of examples in the 'examples' directory link to the sequential, 32-bit integers variant of Arm Performance Libraries.

To build the default set of examples:

* For Arm Compiler for Linux:

1. Copy the 'examples' directory somewhere writeable:
   cp -r <install_location>/armpl-22.0.2_AArch64_*arm-linux-compiler*/examples ./cd examples
2. Load the Arm Compiler for Linux environment module:
   module load acfl/22.0.2
3. Build the examples:
   make

* For GCC:

1. Copy the 'examples' directory somewhere writeable:
   cp -r <install_location>/armpl-22.0.2_AArch64_*gcc*/examples ./cd examples
2. Load the GCC environment modules:
   module load gnu/11.2.0
   module load armpl/22.0.2
3. Build the examples:
   Make

For more information about the Arm Performance Libraries examples, see:

https://developer.arm.com/documentation/102574/0100/compile-an-example

2.10 Uninstall

For convenience, this package includes an "uninstall.sh" script at:

<install_location>/arm-compiler-for-linux-22.0.2*/uninstall.sh

This script attempts to uninstall all the components supplied as part of Arm Compiler for Linux. However, if other packages outside of this product depend on the GCC component, GCC will not be uninstalled.

2.11 Deliverables

The downloaded product includes the deliverables listed in this section.

- Arm C/C+/Fortran Compiler 22.0.2

- Arm Performance Libraries 22.0.2

- GCC 11.2.0

- Release Notes (this document)

- Documentation

Arm Compiler for Linux reference guides are available in <install_location>/<package_name>/share. The guides that are in the '/share' location are also available on the Arm Developer website:

https://developer.arm.com/Tools%20and%20Software/Arm%20Compiler%20for%20Linux#Resources

The same Arm Developer web page also contains links to tutorials, installation guides, and application porting guides.

Documentation and release notes might change between product releases. For the latest documentation bundle, check the product download page.

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of this document when used with any other PDF reader.
A suitable PDF reader can be downloaded from Adobe at **http://www.adobe.com**.

2.12 Differences from previous release

Arm Compiler for Linux 22.0.2 includes various internal changes that resolve defects and improve performance.

The following subsections describe the significant differences from the previous release of Arm Compiler for Linux.

2.12.1 Additions and changes

This section describes the new features or components added, or any significant technical changes to features or components, in the 22.0.2 release.

- Arm Compiler for Linux suite 22.0.2:
  - Arm Compiler for Linux 22.0.2 includes an updated End User License Agreement (EULA) to support the shift to a free of charge distribution model. See https://community.arm.com/arm-community-blogs/b/high-performance-computing-blog/posts/arm-compilers-and-libraries-for-hpc-now-free

There are no new features or components added, or any significant technical changes to features or components, in the 22.0.1 release.

This section describes the new features or components added, or any significant technical changes to features or components, in the 22.0 release.

- Arm Compiler for Linux suite 22.0:
  - The module names have changed in Arm Compiler for Linux 22.0. For Arm C/C++/Fortran Compiler, 'arm<major-version>' has been replaced with 'acfl' in the module load command. To load Arm C/C++/Fortran Compiler, use 'module load acfl/22.0'. For GCC, the major version number has been removed from the module load command. To load GCC only, use 'module load gnu/11.2.0'. For Arm Performance Libraries, there are no longer separate libraries for SVE and non-SVE cores. To load Arm Performance Libraries, use 'module load armpl/22.0'.
  - Added support for Amazon Linux 22.
  - When using Arm Performance Libraries built for GCC, C/C++ users do not need to link to libgfortran.

- Arm C/C++/Fortran Compiler 22.0:
  - Arm Compiler for Linux 22.0 is based on Clang 13.0.0 (of the LLVM Compiler project), which provides improved performance and stability.
  - Arm Compiler for Linux 22.0 features further performance improvements for Neoverse V1-based platforms, when the '-mcpu=neoverse-v1' option is used.
  - Support has been added for the following Arm targets:
    - Armv9-A, Armv9.1-A, Armv9.2-A
    - Cortex-A510

- Use of the most suitable versions of ArmPL is no longer decided at compilation time. The most optimal implementation is detected at runtime. The consequence of this change is that "-armpl=sve" is no longer a valid compiler option.
- When using Arm Performance Libraries built for GCC, C/C++ users do not need to link to libgfortran.
- A number of binaries have been added to the llvm-bin directory in the Arm Compiler for Linux package. These are common auxilliary tools from the LLVM project associated with a clang build, each with a different function. For example, 'llvm-cov' is a tool for run time performance analysis, and 'llvm-symbolizer' is used by address sanitizer to give source location information. To use these tools, add the llvm-bin directory to the PATH environment variable in the product installation. Use of these tools is considered a community feature of Arm Compiler for Linux. For details about community features, see the compiler reference guides:
  - https://developer.arm.com/documentation/101380/2200/Supporting-reference-information/Support-level-definitions (Fortran) or
  - https://developer.arm.com/documentation/101458/2200/Supporting-reference-information/Support-level-definitions (C/C++).
- glibc has a known defect - reported as https://sourceware.org/bugzilla/show_bug.cgi?id=26798 - affecting lazy binding on AArch64 platforms. As a mitigation, Arm Compiler for Linux was using '-z now' flag at the linking stage to disable lazy binding when building dynamically linked programs. Arm Compiler for Linux 22.0 removes this mitigation, and generates programs that do not disable lazy binding. On SVE platforms affected by this glibc defect, we recommend that you set the environment variable LD_BIND_NOW to a non empty value to disable lazy binding at execution time. This defect is likely to affect the following Arm Compiler for Linux supported platforms: Ubuntu 18.04, Ubuntu 20.04, and SLES 15.

- Arm Performance Libraries 22.0.0:
  - Added support for GCC 11.2.0.
  - When using Arm Performance Libraries built for GCC, C/C++ users do not need to link to libgfortran.
  - Arm Performance Libraries are no longer packaged with separate libraries for SVE and non-SVE cores. We now provide a single library, which contains optimized versions for all supported cores, including SVE. Automatic selection of the most appropriate implementation for the microarchitecture occurs at run time.
  - Added the Arm Neoverse V1 core as a new microarchitecture target with specific tunings.
  - Improved the performance for:
    - BLAS level 1 routines: SVE optimizations for ?COPY, ?SCAL, ?AXPY
    - BLAS level 2 routines: packed and banded functionality; ?TRMV and ?TRSV for large problems
    - BLAS level 3 routines: ?TRMM and ?TRSM for large problems

    - LAPACK routines: ?EEVD (eigenvalue decomposition) for small problems; ?POTRF for multithreaded cases
- Added support for I?AMIN BLAS extension routines for all types, finding the location of the first minimum value in a vector.
- Added support for LAPACK version 3.10.0. In addition, an out of bounds bug in LAPACK ?LARRV routines (CVE-2021-4048) has been patched.
- Performance improvements in libamath, for:
    - atan, atanf (vector)
    - atan2, atan2f (scalar & vector)
    - cos, cosf (vector)
    - erfc, erfcf (vector)
    - exp, expf (vector)
    - logf (vector)
    - pow (vector)
    - sin, sinf (vector)
    - tanf (vector)

2.12.2 Resolved issues

There are no resolved issues to report in the 22.0.2 release.

Describes any technical issues that are resolved in the 22.0.1 release.

- Arm C/C++/Fortran Compiler 22.0.1:

- Fixed a bug in armclang that caused the error "fatal error: error in backend: Cannot select" when compiling code using a bfcvt instruction in an inline assembly block at -O1 and above.
- Fixed an issue that meant Arm Compiler for Linux 22.0 did not work correctly on systems with SLES 15 SP2 installed due to insufficient glibc dependency. Arm Compiler for Linux now works correctly on SLES 15 SP2 and SP3.

Describes any technical issues that are resolved in the 22.0 release.

- Arm C/C++/Fortran Compiler 22.0:

- Fixed an issue in armclang affecting the performance of loops marked with #omp pragma simd when targeting SVE-enabled platforms. Now there should not be any performance penalties when this pragma is used in these cases.
- Fixed an issue affecting armclang when invoked with -fLTO, where armclang emitted a "Wrong types for attribute" error.
- Fixed a defect in armflang triggered in the DBCSR application, where armflang would incorrectly emit an "Illegal use of symbol" error when a forward-declared pure function is used in a context that requires pure functions.
- Fixed an armflang defect affecting the CP2k application where multiple uses of ISO_FORTRAN_ENV could cause the module contents not to be available, resulting in "Symbol X has not been explicitly declared" errors.
- Fixed an issue in armflang affecting the BigDFT application where armflang would gave an incorrect warning when using privatization in combination with use statements.

2.13 Known limitations

The following subsection describes any issues that are known at the time of this release.

Open technical issues:

Describes any technical issues that are open in the 22.0.2 release.

- Arm C/C++/Fortran Compiler 22.0.2:

  - In November 2021, two general source code vulnerabilities in compilers were disclosed - CVE-2021-42574 and CVE-2021-42694. Both exploits use Unicode characters to make source code look different when viewed in a text editor to what is processed by the compiler. This exploit could be used by a malicious programmer to inject malicious code into software that looks non-malicious when reviewed or inspected by programmers responsible for the integrity of said software. Arm Compiler for Linux does not have any mitigation for source code containing these attacks. Arm recommends using static analysis tools to detect such vulnerabilities in source code before compilation. For more information see https://developer.arm.com/documentation/ka005002

3 Support

The documentation that is available for Arm Compiler for Linux can be found on the product resources page on the Arm Developer website:

https://developer.arm.com/Tools%20and%20Software/Arm%20Compiler%20for%20Linux#Resources

You can also find a subset of that documentation, available in <install_location>/<package_name>/share.

These deliverables are being released under the terms of the agreement between Arm and each licensee (the "Agreement"). All planned verification and validation is complete. The release is suitable for volume production under the terms of the Agreement.

3.1 OS

This suite is supported on the following Linux platforms:

- AArch64 RHEL 7 and 8
- AArch64 SLES 15
- AArch64 Ubuntu 18.04 and 20.04
- Amazon Linux 2022 (AL 2022)

Full information about the platforms supported by Arm Compiler for Linux is available on the Arm Developer website:

https://developer.arm.com/Tools%20and%20Software/Arm%20Compiler%20for%20Linux#Supported-Devices

4 Release history

A full release history (with release notes) for Arm Compiler for Linux is available on the Arm Developer website:

https://developer.arm.com/documentation/107578/latest

5 Conventions

The following subsections describe conventions used in Arm documents.

5.1 Glossary

The Arm Glossary is a list of terms that are used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: **https://developer.arm.com/glossary**.