GCC plugin infrastructure

Introduction

GCC plugins are loadable modules that provide extra features to the compiler [1]. They are useful for runtime instrumentation and static analysis. We can analyse, change and add further code during compilation via callbacks [2], GIMPLE [3], IPA [4] and RTL passes [5].

The GCC plugin infrastructure of the kernel supports building out-of-tree modules, cross-compilation and building in a separate directory. Plugin source files have to be compilable by a C++ compiler.

Currently the GCC plugin infrastructure supports only some architectures. Grep "select HAVE_GCC_PLUGINS" to find out which architectures support GCC plugins.

This infrastructure was ported from grsecurity [6] and PaX [7].

--

[1] https://gcc.gnu.org/onlinedocs/gccint/Plugins.html

[2] https://gcc.gnu.org/onlinedocs/gccint/Plugin-API.html#Plugin-API

[3] https://gcc.gnu.org/onlinedocs/gccint/GIMPLE.html

[4] https://gcc.gnu.org/onlinedocs/gccint/IPA.html

[5] https://gcc.gnu.org/onlinedocs/gccint/RTL.html

[6] https://grsecurity.net/

[7] https://pax.grsecurity.net/

Purpose

GCC plugins are designed to provide a place to experiment with potential compiler features that are neither in GCC nor Clang upstream. Once their utility is proven, the goal is to upstream the feature into GCC (and Clang), and then to finally remove them from the kernel once the feature is available in all supported versions of GCC.

Specifically, new plugins should implement only features that have no upstream compiler support (in either GCC or Clang).

When a feature exists in Clang but not GCC, effort should be made to bring the feature to upstream GCC (rather than just as a kernel-specific GCC plugin), so the entire ecosystem can benefit from it.

Similarly, even if a feature provided by a GCC plugin does *not* exist in Clang, but the feature is proven to be useful, effort should be spent to upstream the feature to GCC (and Clang).

After a feature is available in upstream GCC, the plugin will be made unbuildable for the corresponding GCC version (and later). Once all kernel-supported versions of GCC provide the feature, the plugin will be removed from the kernel.

Files

$(src)/scripts/gcc-plugins

This is the directory of the GCC plugins.

$(src)/scripts/gcc-plugins/gcc-common.h

This is a compatibility header for GCC plugins. It should be always included instead of individual gcc headers.

$(src)/scripts/gcc-plugins/gcc-generate-gimple-pass.h, $(src)/scripts/gcc-plugins/gcc-generate-ipa-pass.h, $(src)/scripts/gcc-plugins/gcc-generate-simple_ipa-pass.h, $(src)/scripts/gcc-plugins/gcc-generate-rtl-pass.h

These headers automatically generate the registration structures for GIMPLE, SIMPLE_IPA, IPA and RTL passes. They should be preferred to creating the structures by hand.

Usage

You must install the gcc plugin headers for your gcc version, e.g., on Ubuntu for gcc-10:

```
apt-get install gcc-10-plugin-dev
```

Or on Fedora:

```
dnf install gcc-plugin-devel
```

Enable the GCC plugin infrastructure and some plugin(s) you want to use in the kernel config:

```
CONFIG_GCC_PLUGINS=y
CONFIG_GCC_PLUGIN_LATENT_ENTROPY=y
...
```

To compile the minimum tool set including the plugin(s):

```
make scripts
```

or just run the kernel make and compile the whole kernel with the cyclomatic complexity GCC plugin.

4. How to add a new GCC plugin

The GCC plugins are in scripts/gcc-plugins/. You need to put plugin source files right under scripts/gcc-plugins/. Creating subdirectories is not supported. It must be added to scripts/gcc-plugins/Makefile, scripts/Makefile.gcc-plugins and a relevant Kconfig file.