# `invalid flag in #cgo CFLAGS`

This page describes the background for build errors like `invalid flag in #cgo CFLAGS` and what you can do about them.

CVE-2018-6574 described a potential security violation in the go tool: running `go get` downloads and builds Go code from the Internet, Go code that uses cgo can specify options to pass to the compiler, so careful use of `-fplugin` can cause `go get` to execute arbitrary code. While it is difficult to block every possible way that the compiler might be attacked, we have chosen to block the obvious ones.

As described at issue 23672, this is done by using a safelist of compiler/linker options that are permitted during `go get`, `go build`, and friends. When cgo code tries to use to pass an option that is not on the safelist, the go tool will report an error `invalid flag in #cgo CFLAGS` (or `#cgo LDFLAGS`, `pkg-config --cflags`, `pkg-config --ldflags`, and so forth).

This safelist is new in releases 1.8.7, 1.9.4, and 1.10, and all subsequent releases.

## What can I do?

If this happens to you, and the option is benign, you should do two things: 1. Set the environment variable `CGO_CFLAGS_ALLOW` (or `CGO_LDFLAGS_ALLOW`, `CGO_CXXFLAGS_ALLOW`, and so forth) to a regexp that matches the option. 2. File a bug requesting that the option be added to the safelist. Be sure to include the complete error message and, if possible, a description of the code you are building.

## Why not use an unsafe list?

Because if some new unsafe option is added to a compiler, all existing Go releases will become immediately vulnerable.

## Why not get a complete list of compiler options and safelist all of them?

Because there are hundreds of options, and there is no clear way to get a complete list. Many compiler and linker options are target dependent, and thus only reported on specific platforms or in specific configurations. The documentation is known to be incomplete.