`cf-protection`

This option enables control-flow enforcement technology (CET) on x86; a more detailed description of CET is available [here](#). Similar to `clang` , this flag takes one of the following values:

- `none` - Disable CET completely (this is the default).
- `branch` - Enable indirect branch tracking (`IBT`).
- `return` - Enable shadow stack (`SHSTK`).
- `full` - Enable both `branch` and `return` .

This flag only applies to the LLVM backend: it sets the `cf-protection-branch` and `cf-protection-return` flags on LLVM modules. Note, however, that all compiled modules linked together must have the flags set for the compiled output to be CET-enabled. Currently, Rust's standard library does not ship with CET enabled by default, so you may need to rebuild all standard modules with a `cargo` command like:

```
$ RUSTFLAGS="-Z cf-protection=full" RUSTC="rustc-custom" cargo +nightly build -Z
build-std --target x86_64-unknown-linux-gnu
```

Detection

An ELF binary is CET-enabled if it has the `IBT` and `SHSTK` tags, e.g.:

```
$ readelf -a target/x86_64-unknown-linux-gnu/debug/example | grep feature:
      Properties: x86 feature: IBT, SHSTK
```

Troubleshooting

To display modules that are not CET enabled, examine the linker errors available when `cet-report` is enabled:

```
$ RUSTC_LOG=rustc_codegen_ssa::back::link=info rustc-custom -v -Z cf-protection=full
-C link-arg="-Wl,-z,cet-report=warning" -o example example.rs
...
/usr/bin/ld: /.../build/x86_64-unknown-linux-gnu/stage1/lib/rustlib/x86_64-unknown-
linux-gnu/lib/libstd-d73f7266be14cb8b.rlib(std-d73f7266be14cb8b.std.f7443020-
cgu.12.rcgu.o): warning: missing IBT and SHSTK properties
```