Linux and the Clang Static Analyzer

Eduard Bachmakov

September 19th, 2013

About me

- Undergraduate student at Villanova University
- GSoC student with The Linux Foundation
  - Mentors: Behan Webster, Jan-Simon Möller

Why static analysis?

- Catch runtime errors at compile-time

Why not?

- “Compile” performance
- False positives (~4500 issues on x64)
  - retval
  - Lack of interprocedural analysis
  - unsigned int uninitialized_var(msecs);

What about sparse?

From the sparse FAQ ("Why not just use gcc?")

Gcc is big, complex, and the gcc maintainers are not interested in other uses of the gcc front-end. In fact, gcc has explicitly resisted splitting up the front and back ends and having some common intermediate language because of religious license issues - you can have multiple front ends and back ends, but they all have to be part of gcc and licensed under the GPL.

This all (in my opinion) makes gcc development harder than it should be, and makes the end result very ungainly. With "sparse", the front-end is very explicitly separated into its own independent project, and is totally independent from the users. I don't want to know what you do in the back-end, because I don't think I should know or care.

Use/Integration

- CHECK & CHECKFLAGS
- scan−build
  - don't forgt get around things like

```
# check for 'asm goto'                                                    1
ifeq ($(shell $(CONFIG_SHELL) $(srctree)/scripts/gcc-goto.sh              2
    $(CC)), y)
    KBUILD_CFLAGS += -DCC_HAVE_ASM_GOTO                                   3
endif                                                                     4
```

Results for Linux

(Examples in browser.)

Also see at `http://buildbot.llvm.linuxfoundation.org/checker/scan-build-latest`

Problems found

```
static __init void reloc_dyn(Elf32_Ehdr *ehdr, unsigned offset) {
...
case 0x6ffffe00 ... 0x6fffffff:
  dyn->d_un.d_ptr += (0xfffe000U - (unsigned long) VDSO32_PRELINK);
  break;
case 32 ... 0x60000000 -1:
case 0x6000000d ... 0x6ffff000 -1:
  if (dyn->d_tag >= 32 && (dyn->d_tag & 1) == 0)
    dyn->d_un.d_ptr += (0xfffe000U - (unsigned long) VDSO32_PRELINK);
break;
...
```

```
static __init void reloc_dyn(Elf32_Ehdr *ehdr, unsigned offset) {
...
case 0x6ffffe00 ... 0x6fffffeff:
  dyn->d_un.d_ptr += (0xfffe000U - (unsigned long) VDSO32_PRELINK);
  break;
case 32 ... 0x60000000 -1: // So ... about that ...
                           // LLVM Bug 16833
case 0x6000000d ... 0x6ffff000 -1:
  if (dyn->d_tag >= 32 && (dyn->d_tag & 1) == 0)
    dyn->d_un.d_ptr += (0xfffe000U - (unsigned long) VDSO32_PRELINK);
break;
...
```

Discussion

- Annotations/attributes
  - `http://clang-analyzer.llvm.org/annotations.html`
  - __user, __init __exit, ...
  - How to declare?

```
def NoDeref : TypeAttr {                                    1
  let Spellings = [GNU<"noderef">];                         2
}                                                           3
```

The really big question . . .

- What would you like to see for checkers?