Google

Compiling the Linux kernel with LLVM tools

FOSDEM ‘19

Signed-off-by: Bill Wendling <morbo@google.com>
Signed-off-by: Nick Desaulniers <ndesaulniers@google.com>

Goals

- Improve the Linux kernel
  - Additional warning coverage
  - Reduce undefined behavior
  - Static and dynamic analysis (via LLVM's scan-build, TSA, KASAN, KUBSAN, KTSAN)
  - Additional compiler research applied to one of the most important FLOSS codebases (LTO, PLO)
  - Lower switching costs of compilers
- Improve LLVM
  - Implement missing features
  - Compete with GCC
  - Source of feature requests
  - Find bugs given a larger corpus
- Build Linux distros entirely with Clang/LLVM
  - Android
  - ChromeOS
  - OpenMandriva
  - prodkernel (soon!)

History

- 2011 LLL Project
- 2012-2016 LLVMLinux
- 2016 Android Userspace 100% Clang (Google Pixel) (kernel was working at this time, but was punted to Pixel 2)
- 2017 Google Pixel 2 ships first Clang built kernel
- 2018 ChromeOS starts shipping Clang built kernels
- 2018 Google Pixel 3 enables kernel LTO & CFI
- 2019 and beyond the infinite:
    - LLD? ASM goto? Clang's integrated assembler? All of Android? Prod kernel? SCS? TSA? AutoFDO? BOLT? KernelCI and 0day bot CI integration?

Building the Linux Kernel with Clang

```
$ make CC=clang
```

```
$ make CC=clang LD=ld.lld
```

```
$ ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- make CC=clang
```

Kernel Changes Stats

$ git log --oneline --grep='[Cc]lang' | wc -l
1168
$ git log --oneline --grep=ClangBuiltLinux | wc -l
75
$ git log --oneline --grep=LLVMLinux | wc -l
81
$ git log --oneline --grep=llvmlinux | wc -l
4
$ git log --oneline --grep='[Ll][Ll][Vv][Mm]' | wc -l
349

LLVM Change Stats

$ git log --oneline --grep="[lL]inux [Kk]ernel" | wc -l
123

prodkernel: ThinLTO

- Modeled after Android's LTO implementation
- Many build system changes to handle LLVM IR files instead of ELF
  - built-in.o files become thin archives
    - Run LTO when linking vmlinux.o
  - Started with Gold linker instead of BFD
    - Still used BFD for final vmlinux link
  - LLVM tools: llvm-nm
  - ...

prodkernel: ThinLTO + LLD

- Gold linker had too many bugs
- LLD works great ... but doesn't retain linking order when using LTO
- Need to hack around this:

```
#define __define_initcall(fn, id) \
      static initcall_t __initcall_##fn##id __used \
      __attribute__((__section__(".initcall" #id ".init.." __FILE__))) = fn; \
      LTO_REFERENCE_INITCALL(__initcall_##fn##id)
```

```
SECTIONS {
      .initcallearly.init : {
            ...
            *(.initcallearly.init..init/do_mounts_rd.c) ;
            *(.initcallearly.init..init/do_mounts_initrd.c) ;
            *(.initcallearly.init..init/do_mounts_md.c) ;
            ...
```

prodkernel: ThinLTO + LLD

- Gold linker had too many bugs
- LLD works great ... but doesn't retain linking order when using LTO
- Need to hack around this

```
#define __def
      static
      __attri                                          = fn; \
      LTO_REF
```


```
SECTIONS {
      .initca
            .
            *
            *
            *(.initcallearly.init..init/do_mounts_md.c) ;
            ...
```

Some Fun Bugs

- `register long long foo asm("edx");` on 32b hosts https://github.com/ClangBuiltLinux/linux/issues/22#issuecomment-417454144
- Explicit NULL ptr derefence unencodeable on aarch64 https://bugs.llvm.org/show_bug.cgi?id=33134
- `__attribute__((gnu_inline))` https://lkml.org/lkml/2018/6/5/710
- Symbol clashes with C stdlib https://github.com/ClangBuiltLinux/linux/issues/59
- Making up your own calling convention with -fcall-saved-* and -fcall-used-* https://github.com/ClangBuiltLinux/linux/issues/25
- Clang doesn't support VLAiS, but all VLA's were removed from the kernel and -Wvla was enabled.
- Relying on code that's only valid at -O2. (Also, __attribute__((always_inline)) doesn't mean "always inline").
- -fno-remove-null-pointer-checks, -fno-strict-aliasing, -fno-strict-overflow: reasoning.
- member_address_is_nonnull()
- Stack alignment with VLAs.
- __builtin_constant_p() https://github.com/ClangBuiltLinux/linux/issues/7

__builtin_constant_p

- Linux frequently takes "slow" path if `__builtin_constant_p()` evaluates to 0
- Required for kernel hardening: `CONFIG_HARDENED_USERCOPY=y`

Bad or missing usercopy whitelist? Kernel memory exposure attempt detected from SLUB object 'task_struct'!

- GCC "honors" `__builtin_constant_p()` after inling
- Clang's inlining takes place after `__builtin_constant_p()`'s evaluation in the middle end
- Needed way to delay evaluation:
  - James Knight (jyknight@) created `@llvm.is.constant` intrinsic
  - I modified Clang's front-end with a new AST node:

    ConstantExpr: An expression that occurs in an constant context
  - Can be used for future C++ features

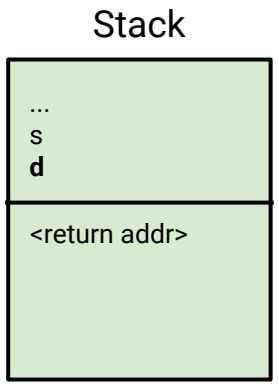
Stack Alignment with VLAs

- `sg_set_buf()` code masks the lower 12 bits of the array addresses and uses that as the page link value
- After inlining, Clang changes the mask from **`0x0FFF`** to **`0x0FF0`** because the arrays are specified as **16-byte** aligned in the LLVM IR
- However, VLAs were generated as **8-byte** aligned
  - Being off by 8 bytes causes the decrypting functions to overwrite the return address on the stack

In sg_set_buf():

```
unsigned int offset = (unsigned long) buf & ~(~((1UL << 12) - 1));  /* 0x0FFF */
```

```
static int cts_cbc_decrypt(...) {
  u8 s[bsize * 2], d[bsize * 2];
    ...
  sg_set_buf(&sgsrc[0], s + bsize, bsize);
  sg_set_buf(&sgdst[0], d, bsize);  /* 0x0FF0 after inlining */
    ...
```



Google

Links

Bug Reports: https://github.com/ClangBuiltLinux/linux/issues

Continuous Integration: https://travis-ci.com/ClangBuiltLinux/continuous-integration/builds

Previous Talks: https://github.com/ClangBuiltLinux/linux/wiki/Talks,-Presentations,-and-Communications

godbolt.org: Great for sharing reproducers

creduce: Minimize reproducers

bear: compile_commands.json for compiler flags

.<target>.o.cmd files contain the exact compiler flags

Köszi!

Dziękuję!

شكرا جزيلا!

qatlho'!

Danke!

Спасибо!

Ευχαριστώ!

Merci!

Grazie!

Takk!

ありがとう!

謝謝!

Thank you!

תודה רבה!

¡Gracias!

Me hoovercraft zit vull mè poaling'n!

Google