KLP with Clang LTO Kernel

Song Liu and Yonghong Song
Kernel Toolchain @ Meta

- 5.6 or earlier kernels: GCC
- 5.12 kernels: GCC, LLVM with PGO
- 5.19 kernels: LLVM with PGO, LLVM with PGO + LTO (experimental)
- 6.4 kernels: LLVM with PGO + LTO
KLP @ Meta

- Fork of kpatch ([https://github.com/dynup/kpatch](https://github.com/dynup/kpatch))
- Changes for PGO and LTO support
- Status
  - KLP for PGO kernel in production for 2+ years
  - KLP for LTO kernels works in tests and limited deployment (~50k hosts)
No-LTO Compilation Workflow

- `foo.c` and `bar.c` compiled by `clang` to ELF files: `foo.o` and `bar.o`
- `objtool` link the ELF files: `foo.o` and `bar.o` to create `vmlinux.o`
- `ld.lld` links `vmlinux.o` and `link-vmlinux.sh` to produce `vmlinux`
No-LTO kpatch-build Workflow

Before patches

foo.c
bar.c

ELF files:
foo.o
bar.o

CDO

After patches

foo.c
bar.c

ELF files:
foo.o
bar.o
LTO Compilation Workflow

#include "foo.c"
#include "bar.c"

LLVM IR files:
- foo.o
- bar.o

Compiling with clang:
- foo.c
- bar.c

Cross-file inlining:
- foo IR
- bar IR

Linking with ld.lld:
- vmlinux.o

Objtool:
- vmlinux.o

Linking with link-vmlinux.sh:
- vmlinux
KLP Issues with LTO

- Current CDO with vmlinux.o is too expensive.
- Takes more than 10 hours cannot finish.
LTO Compilation Workflow

LLVM IR files:
- foo.o
- bar.o

Cross-file inlining
- foo IR
- bar IR

Internal obj files
- foo ELF
- bar ELF

vmlinux.o

-vmlinux.o.thinlto.o[0-9]+ => foo.o
-vmlinux.o.thinlto.o[0-9]+ => bar.o

objtool

vmlinux

link-vmlinux.sh

vmlinux.o

--lto-obj-path=vmlinux.o.thinlto.o
LTO kpatch-build Workflow

Before patches
- foo.c
- bar.c

LLVM IR files:
- foo.o
- bar.o

Cross-file inlining:
- foo IR
- bar IR

Internal obj files:
- foo ELF
- bar ELF

CDO

vmlinux.o.thinlto.o[0-9]+, vmlinux.o.thinlto.o[0-9]+, vmlinux.o.thinlto.o[0-9]+

After patches
- foo.c
- bar.c

LLVM IR files:
- foo.o
- bar.o

Cross-file inlining:
- foo IR
- bar IR

Internal obj files:
- foo ELF
- bar ELF
Details of KLP/LTO in Meta (1)

- Add CONFIG_LTO_CLANG_THIN support.
- CDO does not work at IR level. So add \texttt{--lto-obj-path} option to the linker (ld.lld). The object files will be dumped after cross-file inlining and before final linking.
Details of KLP/LTO in Meta (2)

- A special `__initcall_stub()` with CONFIG_LTO_CLANG to avoid name collisions.
- In `kpatch_mangled_strcmp()` by ignoring all digits for symbols start with `__initstub__kmod_syscall__`. 
Details of KLP/LTO in Meta (3)

- PGO is also supported in Meta kernels.
- [https://lore.kernel.org/lkml/20210111081821.3041587-1-morbo@google.com/](https://lore.kernel.org/lkml/20210111081821.3041587-1-morbo@google.com/)
- Instrumentation based, so a special training kernel is needed to collect training data
Details of KLP/LTO in Meta (4)

- Kernel changes caused KLP change.
- `init/version-timestamps.o` change due to build process change.
- Ignore section `.rel.llvm.call-graph-profile .comment` section. For example, `.comment` holds some compiler information, cross-file inline might bring more entries in `.comment` for a particular objfile.
Upstream Discussion

- [https://github.com/dynup/kpatch/issues/1320](https://github.com/dynup/kpatch/issues/1320)
- Big create-diff-object time with vmlinux.o, key related functions:
  - `find_symbol_by_index()`
  - `kpatch_create_rela_list()`
  - `kpatch_elf_open()`
- How IBT (CONFIG_X86_KERNEL_IBT and CONFIG_FINEIBT) handled by live patch?
Next Steps

- CDO vmlinux.o is suggested in https://github.com/dynup/kpatch/issues/1320 (for LTO and IBT). How and Who?