

## objtool on arm64

Friday, 24 September 2021 07:35 (30 minutes)

objtool is heavily used on x86, but isn't currently support upstream by arm64.

In order to avoid depending on objtool to enable any kernel features for arm64 and also to avoid disabling compiler optimisations along the lines of <https://git.kernel.org/linus/3193c0836f20> when objtool cannot reconstruct the control flow, how much of its functionality is actually required on arm64 and how much of that could be directly implemented by the toolchain instead?

From:

<https://lore.kernel.org/r/YKO/di4h3XGjqu68@hirez.programming.kicks-ass.net>

some objtool features on x86 are:

- validate stack frames
- generate ORC unwind data (optional)
- validates unreachable instructions; specifically the lack thereof (optional)
- validates retpoline; or specifically the lack of indirect jump/call sites (with annotations for those few that are okay). (optional)
- validates uaccess rules; specifically no call/ret in between `__user_access_begin()` and `__user_access_end()`. (optional)
- validates noinstr annotation; *HOWEVER* we rely on objtool to NOP all `__sanitizer_cov_*` calls in `.noinstr/.entry` text sections because `__no_sanitise_cov` is 'broken' in all known compilers.
- generates `__mcount_loc` section and NOPs the `__fentry` call sites (optional)
- generates `.static_call_sites` section for `STATIC_CALL_INLINE` support
- rewrites compiler generates call/jump to the retpoline thunk to an alternative such that we can patch out the thunk with an indirect call/jmp when retpolines are disabled. (arch dependent)
- rewrites specific `jmp.d8` sites (as found through the `__jump_table` section) to `nop2`, because GAS is unable to determine if a `jmp` becomes a `jmp.d8` or `jmp.d32` and emit the right sized `nop`. (optional)

### I agree to abide by the anti-harassment policy

I agree

**Primary authors:** POIMBOEUF, Josh (Red Hat); RUTLAND, Mark (Arm Ltd); ZIJLSTRA, Peter (Intel OTC); DEACON, Will

**Presenters:** POIMBOEUF, Josh (Red Hat); RUTLAND, Mark (Arm Ltd); ZIJLSTRA, Peter (Intel OTC); DEACON, Will

**Session Classification:** Toolchains and Kernel MC

**Track Classification:** Toolchains and Kernel MC