Per-arg kfunc flags

Enabling kfuncs to specify flags per-arg, in addition to per-kfunc

David Vernet
Linux kernel engineer

Meta

Agenda

01 kfunc background

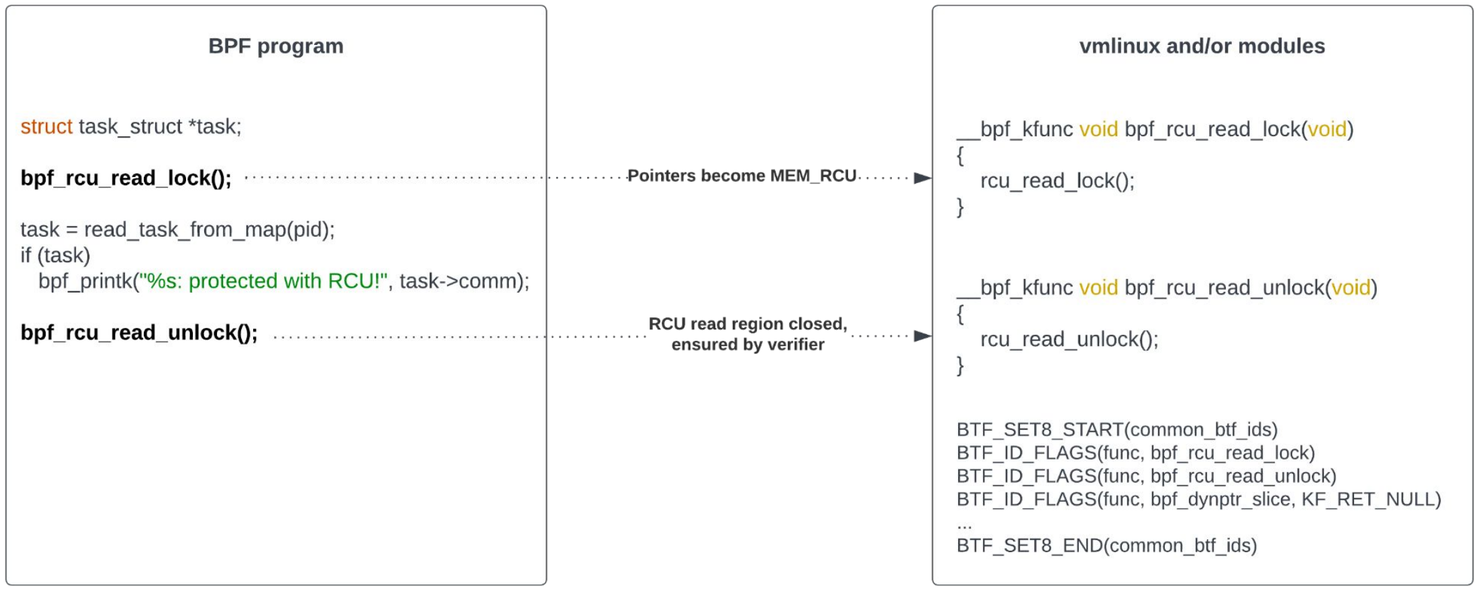
02 Why is this useful?

03 How do we do it?

01 kfunc background

kfuncs are the “modern” version of BPF helpers

- Allow BPF programs to call into the main kernel
- Hard dependency on BTF
  - Defined in BTF sets
- No ABI stability guarantees. See https://docs.kernel.org/bpf/kfuncs.html#kfunc-lifecycle-expectations for more information



kfunc properties can be statically defined with flags

- All flags are static, and defined for the whole kfunc
  - **KF_ACQUIRE**: kfunc acquires a reference to the first argument of the kfunc
  - **KF_RELEASE**: kfunc releases a reference to the first argument of the kfunc
  - **KF_RET_NULL**: kfunc may return NULL (i.e. the return type's pointer will include `PTR_MAYBE_NULL` modifier)
  - **KF_TRUSTED_ARGS**: kfunc takes only trusted pointer arguments (deprecated, will eventually be removed)
  - **KF_SLEEPABLE**: kfunc may sleep
  - **KF_DESTRUCTIVE**: kfunc may perform destructive actions – used with graph-type data structures
  - **KF_RCU**: Pointer arguments, when not NULL, are guaranteed to be valid in an RCU read region
  - **KF_ITER_NEW**: kfunc is a BPF iterator constructor
  - **KF_ITER_NEXT**: kfunc implements BPF iterator next() method
  - **KF_ITER_DESTROY**: kfunc implements a BPF iterator destructor
- Some flags make sense for whole kfunc (e.g. `KF_RET_NULL`, `KF_SLEEPABLE`, etc), others are really specific to individual arguments (`KF_ACQUIRE`, `KF_RELEASE`, `KF_RCU`, etc)
- kfunc flags defined in a single `.long` following the BTF ID of the kfunc itself

02 Why is this useful?

Better UX and future proofing

- Not immediately pressing, but has come up before
  - https://lore.kernel.org/bpf/20221217082506.1570898-2-davemarchevsky@fb.com/
  - Being hacked around with annotations like `__sz` and `__k`
- Current API requires internal knowledge of the verifier
  - kfunc authors should ideally need to know very little, if anything, about verifier internals
  - Why does arg 0 correspond to “acquire” arg or “release” arg? Purely an implementation detail.
  - Thankfully everything is very well documented at the moment, but we can do better
- Helpers are now tentatively frozen. Need to get kfuncs to feature parity
  - This is supported for helpers, should be supported for kfuncs (luckily not blocking anything yet)

03 How do we do it?

How do we do this with BTF?

- The single `.long` following the BTF ID of the kfunc is fine for kfunc global flags, doesn't work for per-arg
- In general, not sure how this would work with BTF sets
  - BTF set for each kfunc, where index `n` are the flags for arg `n`?
- More generally, the API for defining kfuncs is arguably clunky and error prone
  - Something closer to EXPORT_SYMBOL_GPL would be ideal. Hide all of the complexity behind a macro
    - Not sure how this would work with register_btf_kfunc_id_set() expecting a BTF set.
    - Seems like we'll require some post-processing ELF magic?


Meta