Skip to site navigation (Press enter)

[PATCH net-next v3 1/2] bpf: remove struct bpf_prog_type_list

Johannes Berg Mon, 10 Apr 2017 05:44:54 -0700

From: Johannes Berg <johannes.b...@intel.com>

There's no need to have struct bpf_prog_type_list since
it just contains a list_head, the type, and the ops
pointer. Since the types are densely packed and not
actually dynamically registered, it's much easier and
smaller to have an array of type->ops pointer. Also
initialize this array statically to remove code needed
to initialize it.

In order to save duplicating the list, move it to a new
header file and include it in the places needing it.

Signed-off-by: Johannes Berg <johannes.b...@intel.com>

```
 include/linux/bpf.h       | 16 ++++------
 include/linux/bpf_types.h | 18 ++++++++++
 kernel/bpf/syscall.c      | 27 +++++++----------
 kernel/trace/bpf_trace.c  | 30 ++++-------------
 net/core/filter.c         | 75 +++++++-----------------------------------
 5 files changed, 44 insertions(+), 122 deletions(-)
 create mode 100644 include/linux/bpf_types.h

diff --git a/include/linux/bpf.h b/include/linux/bpf.h
index bbb513da5075..07fc02bb38e4 100644
--- a/include/linux/bpf.h
+++ b/include/linux/bpf.h
@@ -173,12 +173,6 @@ struct bpf_verifier_ops {
                      union bpf_attr __user *uattr);
 };

-struct bpf_prog_type_list {
-       struct list_head list_node;
-       const struct bpf_verifier_ops *ops;
-       enum bpf_prog_type type;
-};
-
 struct bpf_prog_aux {
        atomic_t refcnt;
        u32 used_map_cnt;
@@ -243,7 +237,11 @@ int bpf_prog_test_run_skb(struct bpf_prog *prog, const
```

```
 union bpf_attr *kattr,
  #ifdef CONFIG_BPF_SYSCALL
  DECLARE_PER_CPU(int, bpf_prog_active);

 -void bpf_register_prog_type(struct bpf_prog_type_list *tl);
 +#define BPF_PROG_TYPE(_id, _ops) \
 +       extern const struct bpf_verifier_ops _ops;
 +#include <linux/bpf_types.h>
 +#undef BPF_PROG_TYPE
 +
  void bpf_register_map_type(struct bpf_map_type_list *tl);

  struct bpf_prog *bpf_prog_get(u32 ufd);
 @@ -306,10 +304,6 @@ static inline void bpf_long_memcpy(void *dst, const void
 *src, u32 size)
  /* verify correctness of eBPF program */
  int bpf_check(struct bpf_prog **fp, union bpf_attr *attr);
  #else
 -static inline void bpf_register_prog_type(struct bpf_prog_type_list *tl)
 -{
 -}
 -
  static inline struct bpf_prog *bpf_prog_get(u32 ufd)
  {
         return ERR_PTR(-EOPNOTSUPP);
 diff --git a/include/linux/bpf_types.h b/include/linux/bpf_types.h
 new file mode 100644
 index 000000000000..68b0a9811216
 --- /dev/null
 +++ b/include/linux/bpf_types.h
 @@ -0,0 +1,18 @@
 +/* internal file - do not include directly */
 +
 +#ifdef CONFIG_NET
 +BPF_PROG_TYPE(BPF_PROG_TYPE_SOCKET_FILTER, sk_filter_prog_ops)
 +BPF_PROG_TYPE(BPF_PROG_TYPE_SCHED_CLS, tc_cls_act_prog_ops)
 +BPF_PROG_TYPE(BPF_PROG_TYPE_SCHED_ACT, tc_cls_act_prog_ops)
 +BPF_PROG_TYPE(BPF_PROG_TYPE_XDP, xdp_prog_ops)
 +BPF_PROG_TYPE(BPF_PROG_TYPE_CGROUP_SKB, cg_skb_prog_ops)
 +BPF_PROG_TYPE(BPF_PROG_TYPE_CGROUP_SOCK, cg_sock_prog_ops)
 +BPF_PROG_TYPE(BPF_PROG_TYPE_LWT_IN, lwt_inout_prog_ops)
 +BPF_PROG_TYPE(BPF_PROG_TYPE_LWT_OUT, lwt_inout_prog_ops)
 +BPF_PROG_TYPE(BPF_PROG_TYPE_LWT_XMIT, lwt_xmit_prog_ops)
 +#endif
 +#ifdef CONFIG_BPF_EVENTS
 +BPF_PROG_TYPE(BPF_PROG_TYPE_KPROBE, kprobe_prog_ops)
 +BPF_PROG_TYPE(BPF_PROG_TYPE_TRACEPOINT, tracepoint_prog_ops)
 +BPF_PROG_TYPE(BPF_PROG_TYPE_PERF_EVENT, perf_event_prog_ops)
 +#endif
 diff --git a/kernel/bpf/syscall.c b/kernel/bpf/syscall.c
 index ab0cf4c43690..ea55691cbf5e 100644
 --- a/kernel/bpf/syscall.c
 +++ b/kernel/bpf/syscall.c
 @@ -573,26 +573,21 @@ static int map_get_next_key(union bpf_attr *attr)
         return err;
```

```
 }

-static LIST_HEAD(bpf_prog_types);
+static const struct bpf_verifier_ops * const bpf_prog_types[] = {
+#define BPF_PROG_TYPE(_id, _ops) \
+       [_id] = &_ops,
+#include <linux/bpf_types.h>
+#undef BPF_PROG_TYPE
+};

 static int find_prog_type(enum bpf_prog_type type, struct bpf_prog *prog)
 {
-       struct bpf_prog_type_list *tl;
-
-       list_for_each_entry(tl, &bpf_prog_types, list_node) {
-               if (tl->type == type) {
-                       prog->aux->ops = tl->ops;
-                       prog->type = type;
-                       return 0;
-               }
-       }
-
-       return -EINVAL;
-}
+       if (type >= ARRAY_SIZE(bpf_prog_types) || !bpf_prog_types[type])
+               return -EINVAL;

-void bpf_register_prog_type(struct bpf_prog_type_list *tl)
-{
-       list_add(&tl->list_node, &bpf_prog_types);
+       prog->aux->ops = bpf_prog_types[type];
+       prog->type = type;
+       return 0;
 }

 /* drop refcnt on maps used by eBPF program and free auxilary data */
diff --git a/kernel/trace/bpf_trace.c b/kernel/trace/bpf_trace.c
index cee9802cf3e0..8a4efac28710 100644
--- a/kernel/trace/bpf_trace.c
+++ b/kernel/trace/bpf_trace.c
@@ -501,16 +501,11 @@ static bool kprobe_prog_is_valid_access(int off, int
size, enum bpf_access_type
        return true;
 }

-static const struct bpf_verifier_ops kprobe_prog_ops = {
+const struct bpf_verifier_ops kprobe_prog_ops = {
        .get_func_proto  = kprobe_prog_func_proto,
        .is_valid_access = kprobe_prog_is_valid_access,
 };

-static struct bpf_prog_type_list kprobe_tl __ro_after_init = {
-       .ops    = &kprobe_prog_ops,
-       .type   = BPF_PROG_TYPE_KPROBE,
-};
```

```
-
 BPF_CALL_5(bpf_perf_event_output_tp, void *, tp_buff, struct bpf_map *, map,
            u64, flags, void *, data, u64, size)
 {
@@ -584,16 +579,11 @@ static bool tp_prog_is_valid_access(int off, int size,
enum bpf_access_type type
        return true;
 }

-static const struct bpf_verifier_ops tracepoint_prog_ops = {
+const struct bpf_verifier_ops tracepoint_prog_ops = {
        .get_func_proto  = tp_prog_func_proto,
        .is_valid_access = tp_prog_is_valid_access,
 };

-static struct bpf_prog_type_list tracepoint_tl __ro_after_init = {
-       .ops    = &tracepoint_prog_ops,
-       .type   = BPF_PROG_TYPE_TRACEPOINT,
-};
-
 static bool pe_prog_is_valid_access(int off, int size, enum bpf_access_type
type,
                                    enum bpf_reg_type *reg_type)
 {
@@ -642,22 +632,8 @@ static u32 pe_prog_convert_ctx_access(enum bpf_access_type
type,
        return insn - insn_buf;
 }

-static const struct bpf_verifier_ops perf_event_prog_ops = {
+const struct bpf_verifier_ops perf_event_prog_ops = {
        .get_func_proto         = tp_prog_func_proto,
        .is_valid_access        = pe_prog_is_valid_access,
        .convert_ctx_access     = pe_prog_convert_ctx_access,
 };
-
-static struct bpf_prog_type_list perf_event_tl __ro_after_init = {
-       .ops    = &perf_event_prog_ops,
-       .type   = BPF_PROG_TYPE_PERF_EVENT,
-};
-
-static int __init register_kprobe_prog_ops(void)
-{
-       bpf_register_prog_type(&kprobe_tl);
-       bpf_register_prog_type(&tracepoint_tl);
-       bpf_register_prog_type(&perf_event_tl);
-       return 0;
-}
-late_initcall(register_kprobe_prog_ops);
diff --git a/net/core/filter.c b/net/core/filter.c
index 15e9a81ffebe..bbe0cf415105 100644
--- a/net/core/filter.c
+++ b/net/core/filter.c
@@ -3298,13 +3298,13 @@ static u32 xdp_convert_ctx_access(enum bpf_access_type
type,
```

```
        return insn - insn_buf;
 }

-static const struct bpf_verifier_ops sk_filter_ops = {
+const struct bpf_verifier_ops sk_filter_prog_ops = {
        .get_func_proto         = sk_filter_func_proto,
        .is_valid_access        = sk_filter_is_valid_access,
        .convert_ctx_access     = bpf_convert_ctx_access,
 };

-static const struct bpf_verifier_ops tc_cls_act_ops = {
+const struct bpf_verifier_ops tc_cls_act_prog_ops = {
        .get_func_proto         = tc_cls_act_func_proto,
        .is_valid_access        = tc_cls_act_is_valid_access,
        .convert_ctx_access     = tc_cls_act_convert_ctx_access,
@@ -3312,28 +3312,28 @@ static const struct bpf_verifier_ops tc_cls_act_ops = {
        .test_run               = bpf_prog_test_run_skb,
 };

-static const struct bpf_verifier_ops xdp_ops = {
+const struct bpf_verifier_ops xdp_prog_ops = {
        .get_func_proto         = xdp_func_proto,
        .is_valid_access        = xdp_is_valid_access,
        .convert_ctx_access     = xdp_convert_ctx_access,
        .test_run               = bpf_prog_test_run_xdp,
 };

-static const struct bpf_verifier_ops cg_skb_ops = {
+const struct bpf_verifier_ops cg_skb_prog_ops = {
        .get_func_proto         = cg_skb_func_proto,
        .is_valid_access        = sk_filter_is_valid_access,
        .convert_ctx_access     = bpf_convert_ctx_access,
        .test_run               = bpf_prog_test_run_skb,
 };

-static const struct bpf_verifier_ops lwt_inout_ops = {
+const struct bpf_verifier_ops lwt_inout_prog_ops = {
        .get_func_proto         = lwt_inout_func_proto,
        .is_valid_access        = lwt_is_valid_access,
        .convert_ctx_access     = bpf_convert_ctx_access,
        .test_run               = bpf_prog_test_run_skb,
 };

-static const struct bpf_verifier_ops lwt_xmit_ops = {
+const struct bpf_verifier_ops lwt_xmit_prog_ops = {
        .get_func_proto         = lwt_xmit_func_proto,
        .is_valid_access        = lwt_is_valid_access,
        .convert_ctx_access     = bpf_convert_ctx_access,
@@ -3341,73 +3341,12 @@ static const struct bpf_verifier_ops lwt_xmit_ops = {
        .test_run               = bpf_prog_test_run_skb,
 };

-static const struct bpf_verifier_ops cg_sock_ops = {
+const struct bpf_verifier_ops cg_sock_prog_ops = {
        .get_func_proto         = bpf_base_func_proto,
```

```
        .is_valid_access        = sock_filter_is_valid_access,
        .convert_ctx_access     = sock_filter_convert_ctx_access,
 };

-static struct bpf_prog_type_list sk_filter_type __ro_after_init = {
-       .ops    = &sk_filter_ops,
-       .type   = BPF_PROG_TYPE_SOCKET_FILTER,
-};
-
-static struct bpf_prog_type_list sched_cls_type __ro_after_init = {
-       .ops    = &tc_cls_act_ops,
-       .type   = BPF_PROG_TYPE_SCHED_CLS,
-};
-
-static struct bpf_prog_type_list sched_act_type __ro_after_init = {
-       .ops    = &tc_cls_act_ops,
-       .type   = BPF_PROG_TYPE_SCHED_ACT,
-};
-
-static struct bpf_prog_type_list xdp_type __ro_after_init = {
-       .ops    = &xdp_ops,
-       .type   = BPF_PROG_TYPE_XDP,
-};
-
-static struct bpf_prog_type_list cg_skb_type __ro_after_init = {
-       .ops    = &cg_skb_ops,
-       .type   = BPF_PROG_TYPE_CGROUP_SKB,
-};
-
-static struct bpf_prog_type_list lwt_in_type __ro_after_init = {
-       .ops    = &lwt_inout_ops,
-       .type   = BPF_PROG_TYPE_LWT_IN,
-};
-
-static struct bpf_prog_type_list lwt_out_type __ro_after_init = {
-       .ops    = &lwt_inout_ops,
-       .type   = BPF_PROG_TYPE_LWT_OUT,
-};
-
-static struct bpf_prog_type_list lwt_xmit_type __ro_after_init = {
-       .ops    = &lwt_xmit_ops,
-       .type   = BPF_PROG_TYPE_LWT_XMIT,
-};
-
-static struct bpf_prog_type_list cg_sock_type __ro_after_init = {
-       .ops    = &cg_sock_ops,
-       .type   = BPF_PROG_TYPE_CGROUP_SOCK
-};
-
-static int __init register_sk_filter_ops(void)
-{
-       bpf_register_prog_type(&sk_filter_type);
-       bpf_register_prog_type(&sched_cls_type);
-       bpf_register_prog_type(&sched_act_type);
-       bpf_register_prog_type(&xdp_type);
```

```
-       bpf_register_prog_type(&cg_skb_type);
-       bpf_register_prog_type(&cg_sock_type);
-       bpf_register_prog_type(&lwt_in_type);
-       bpf_register_prog_type(&lwt_out_type);
-       bpf_register_prog_type(&lwt_xmit_type);
-
-       return 0;
-}
-late_initcall(register_sk_filter_ops);
-
 int sk_detach_filter(struct sock *sk)
 {
        int ret = -ENOENT;
--
2.11.0
```


- Previous message
- View by thread
- View by date
- Next message

- [PATCH net-next v3 0/2] bpf: prog/map type reductions Johannes Berg
- - [PATCH net-next v3 1/2] bpf: remove struct bpf_pro... Johannes Berg
    - - Re: [PATCH net-next v3 1/2] bpf: remove struct... Alexei Starovoitov
      - Re: [PATCH net-next v3 1/2] bpf: remove struct... Daniel Borkmann
    - [PATCH net-next v3 2/2] bpf: remove struct bpf_map... Johannes Berg
    - - Re: [PATCH net-next v3 2/2] bpf: remove struct... Alexei Starovoitov
      - Re: [PATCH net-next v3 2/2] bpf: remove struct... Daniel Borkmann
      - Re: [PATCH net-next v3 2/2] bpf: remove struct... Johannes Berg

Reply via email to

Johannes Berg