Comprehensive Kernel Function Support

Current bpf utilizing internal kernel functions

- struct_ops maps
    - Tcp congestion control
- Bpf helpers

BPF Kfunc Id Set

```
enum btf_kfunc_type {
        BTF_KFUNC_TYPE_CHECK,
        BTF_KFUNC_TYPE_ACQUIRE,
        BTF_KFUNC_TYPE_RELEASE,
        BTF_KFUNC_TYPE_RET_NULL,
        BTF_KFUNC_TYPE_KPTR_ACQUIRE,
        BTF_KFUNC_TYPE_MAX,
};

BTF_SET_START(nf_ct_xdp_check_kfunc_ids)
BTF_ID(func, bpf_xdp_ct_lookup)
BTF_ID(func, bpf_ct_release)
BTF_SET_END(nf_ct_xdp_check_kfunc_ids)
…
BTF_SET_START(nf_ct_release_kfunc_ids)
BTF_ID(func, bpf_ct_release)
BTF_SET_END(nf_ct_release_kfunc_ids)
```

```
static const struct btf_kfunc_id_set nf_conntrack_xdp_kfunc_set = {
        .owner       = THIS_MODULE,
        .check_set    = &nf_ct_xdp_check_kfunc_ids,
        .acquire_set  = &nf_ct_acquire_kfunc_ids,
        .release_set  = &nf_ct_release_kfunc_ids,
        .ret_null_set = &nf_ct_ret_null_kfunc_ids,
};

register_btf_kfunc_id_set(BPF_PROG_TYPE_XDP,
&nf_conntrack_xdp_kfunc_set);
```

In bpf program:

```
  struct nf_conn *bpf_xdp_ct_lookup(struct xdp_md *, struct
bpf_sock_tuple *, u32, struct bpf_ct_opts *, u32) __ksym;
  …
  bpf_xdp_ct_lookup(ctx, NULL, 0, &opts_def, sizeof(opts_def));
 …
```

Use More Kernel Functions in BPF Programs

- Existing kfunc_id functions are classified with *program type* and *btf_kfunc_type* (CHECK, ACQUIRE, RELEASE) etc. with BTF_SET/ID infrastructure.
- If we want to use much more kernel functions in bpf programs, we may need to classify kernel functions with more information.
- A couple of examples:
  - int proc_cgroup_show(struct seq_file *m, struct pid_namespace *ns, struct pid *pid, struct task_struct *tsk);
  - mutex_lock(&cgroup_mutex)
  - void __insert_inode_hash(struct inode *inode, unsigned long hashval)
  - spin_lock/unlock(&inode_hash_lock); spin_lock/unlock(&inode->i_lock);
  - Lots of more EXPORT_SYMBOL functions and other functions

Use btf_decl_tag

- Close to the source
- Encoded in vmlinux BTF
- int proc_cgroup_show(struct seq_file *m, struct pid_namespace *ns, struct pid *pid, struct task_struct *tsk) __attribute__((btf_decl_tag("mutex_lock: cgroup_mutex")));
-
- void __insert_inode_hash(struct inode *inode, unsigned long hashval) __attribute__((btf_decl_tag("spin_lock:inode_hash_lock"))) __attribute__((btf_decl_tag(("spin_lock:&param1->i_lock")))
-
- Lots of more EXPORT_SYMBOL functions could be annotated and used by bpf programs
- Btf_decl_tag applied to structures, structure members, global variables, functions, function parameters.

Use btf_type_tag

- Btf_type_tag is used to annotation types.

```
truct watch_queue {
 struct rcu_head          rcu;
 // struct watch_filter _rcu *filter;
 struct watch_filter __attribute__((btf_type_tag(“rcu”))) *filter;
 …
```

- Encoded in vmlinux BTF.
- Current kernel supports btf_type_tag for __user and __percpu.

Print Opaque Kernel Data (1)

```
struct task_struct {
     …
     struct bpf_local_storage __rcu  *bpf_storage;
     …
};

struct bpf_local_storage {
    struct bpf_local_storage_data __rcu *cache[BPF_LOCAL_STORAGE_CACHE_SIZE];
    struct hlist_head list; /* List of bpf_local_storage_elem */
    ….
};
```

bpf_snprintf_btf() to dump a data structure with its contents based BTF types. But dumps stops at hish_head_list pointer contents as the actual types for list element not available.

Print Opaque Kernel Data (2)

```
struct task_struct {
     …
     struct bpf_local_storage __rcu  *bpf_storage;
     …
};

struct bpf_local_storage {
    struct bpf_local_storage_data __rcu *cache[BPF_LOCAL_STORAGE_CACHE_SIZE];
    // struct hlist_head list; /* List of bpf_local_storage_elem */
    struct hlist_head list __attribute__((btf_decl_tag("bpf_local_storage_elem…)));;
    ….
};
```

bpf_snprintf_btf() can be enhanced to print bpf_local_storage_elem contents, starting from a task_struct or bpf_local_storage.