# Start with XDP


Hangbin Liu <haliu@redhat.com>
Software Engineering

# Agenda:

- What’s & Why XDP
- Write a small program
- Extend the program
- Use BPF maps
- Load the BPF program with libbpf

# XDP(eXpress Data Path):

- Easy to use
- Fast

```
$ cat xdp_drop.c
#include <linux/bpf.h>
#include <bpf/bpf_helpers.h>

SEC("xdp_drop")
int xdp_drop_prog(struct xdp_md *ctx)
{
        return XDP_DROP;
}

char _license[] SEC("license") = "GPL";
```

```
$ sudo dnf install clang llvm gcc kernel-headers
$ sudo dnf install libbpf-devel libxdp-devel xdp-tools bpftool
$ clang -O2 -g -Wall -target bpf -c xdp_drop.c -o xdp_drop.o
$ llvm-objdump -S -no-show-raw-insn xdp_drop.o

xdp_drop:       file format ELF64-BPF


Disassembly of section xdp_drop:

0000000000000000 xdp_drop_prog:
;       return XDP_DROP;
       0:       r0 = 1
       1:       exit
```

# | 2.1: Build eBPF/XDP object

```
$ llvm-objdump -h xdp_drop.o

xdp_drop:       file format ELF64-BPF

Sections:
Idx Name            Size     VMA              Type
  0                 00000000 0000000000000000
  1 .strtab         000000ad 0000000000000000
  2 .text           00000000 0000000000000000 TEXT
  3 xdp_drop        00000010 0000000000000000 TEXT
  4 license         00000004 0000000000000000 DATA
  5 .debug_str      00000125 0000000000000000
  6 .debug_abbrev   000000ba 0000000000000000
  7 .debug_info     00000114 0000000000000000
  8 .rel.debug_info 000001c0 0000000000000000
  9 .BTF            000001df 0000000000000000
 10 .rel.BTF        00000010 0000000000000000
 11 .BTF.ext        00000050 0000000000000000
 12 .rel.BTF.ext    00000020 0000000000000000
 13 .eh_frame       00000030 0000000000000000 DATA
 14 .rel.eh_frame   00000010 0000000000000000
 15 .debug_line     00000084 0000000000000000
 16 .rel.debug_line 00000010 0000000000000000
 17 .llvm_addrsig   00000002 0000000000000000
 18 .symtab         000002d0 0000000000000000
```

## | 2.2: Load XDP object via ip link

```
# Warn: dont load it on default interface!
$ sudo ip link set veth1 xdpgeneric obj xdp_drop.o sec xdp_drop
$ sudo ip link show veth1
6: veth1@veth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 xdpgeneric qdisc noqueue ...
    link/ether f6:03:7f:e7:27:bb brd ff:ff:ff:ff:ff:ff
    prog/xdp id 1 tag 57cd311f2e27366b jited

$ sudo bpftool prog show
1: xdp  tag 57cd311f2e27366b  gpl
        loaded_at 2021-02-04T04:12:00-0500  uid 0
        xlated 16B  jited 40B  memlock 4096B

$ sudo xdp-loader status
CURRENT XDP PROGRAM STATUS:

Interface        Prio  Program name     Mode     ID   Tag               Chain actions
--------------------------------------------------------------------------------------
lo               <no XDP program>
ens3             <no XDP program>
veth1                                   skb      1    57cd311f2e27366b

$ sudo ip link set veth1 xdpgeneric off
```

# | 2.2: Load bpf object via xdp-loader(recommend)

```
# Warn: dont load it on default interface!
$ sudo xdp-loader load -m skb -s xdp_drop veth1 xdp_drop.o
$ sudo xdp-loader status
CURRENT XDP PROGRAM STATUS:

Interface        Prio  Program name      Mode     ID   Tag               Chain
actions
--------------------------------------------------------------------------------
lo               <no XDP program>
ens3             <no XDP program>
veth1                  xdp_dispatcher    skb      15   d51e469e988d81da
 =>              50    xdp_drop_prog              20   57cd311f2e27366b XDP_PASS

$ sudo ip link show veth1
4: veth1@if3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 xdpgeneric qdisc noqueue ...
    link/ether ba:4d:98:21:3b:b3 brd ff:ff:ff:ff:ff:ff link-netns ns
    prog/xdp id 15 tag d51e469e988d81da jited

$ sudo bpftool prog show
15: xdp  name xdp_dispatcher  tag d51e469e988d81da  gpl
        loaded_at 2021-01-13T03:24:43-0500  uid 0
        xlated 616B  jited 638B  memlock 4096B  map_ids 8
        btf_id 12
20: ext  name xdp_drop_prog  tag 57cd311f2e27366b  gpl
        loaded_at 2021-01-13T03:24:43-0500  uid 0
        xlated 16B  jited 40B  memlock 4096B
        btf_id 16

$ sudo xdp-loader unload -a veth1
```

```
/* user accessible metadata for XDP packet hook
 * new fields must be added to the end of this structure
 */
struct xdp_md {
        __u32 data;
        __u32 data_end;
        __u32 data_meta;
        /* Below access go through struct xdp_rxq_info */
        __u32 ingress_ifindex; /* rxq->dev->ifindex */
        __u32 rx_queue_index;  /* rxq->queue_index  */

        __u32 egress_ifindex;  /* txq->dev->ifindex */
};
```

# | 3: Extend the program, Drop specific packets

```c
#include <linux/bpf.h>
#include <bpf/bpf_helpers.h>
#include <linux/if_ether.h>
#include <arpa/inet.h>


SEC("xdp_drop")
int xdp_drop_prog(struct xdp_md *ctx)
{
        void *data_end = (void *)(long)ctx->data_end;
        void *data = (void *)(long)ctx->data;
        struct ethhdr *eth = data;
        __u16 h_proto;

        if (data + sizeof(struct ethhdr) > data_end)
                return XDP_DROP;

        h_proto = eth->h_proto;

        if (h_proto == htons(ETH_P_IPV6))
                return XDP_DROP;

        return XDP_PASS;
}

char _license[] SEC("license") = "GPL";
```

```
struct {
        __uint(type, BPF_MAP_TYPE_PERCPU_ARRAY);
        __type(key, __u32);
        __type(value, long);
        __uint(max_entries, 1);
} rxcnt SEC(".maps");

enum bpf_map_type {
        BPF_MAP_TYPE_UNSPEC,
        BPF_MAP_TYPE_HASH,
        BPF_MAP_TYPE_ARRAY,
        BPF_MAP_TYPE_PROG_ARRAY,
        BPF_MAP_TYPE_PERF_EVENT_ARRAY,
        BPF_MAP_TYPE_PERCPU_HASH,
        BPF_MAP_TYPE_PERCPU_ARRAY,
[...]
        BPF_MAP_TYPE_TASK_STORAGE,
};
```

# | 4: Use BPF map: count the processed packets

```c
#include <linux/bpf.h>
#include <bpf/bpf_helpers.h>
#include <linux/if_ether.h>
#include <arpa/inet.h>

struct {
        __uint(type, BPF_MAP_TYPE_PERCPU_ARRAY);
        __type(key, __u32);
        __type(value, long);
        __uint(max_entries, 1);
} rxcnt SEC(".maps");

SEC("xdp_drop_ipv6")
int xdp_drop_ipv6_prog(struct xdp_md *ctx)
{
        void *data_end = (void *)(long)ctx->data_end;
        void *data = (void *)(long)ctx->data;
        struct ethhdr *eth = data;
        __u16 h_proto;
        __u32 key = 0;
        long *value;

        if (data + sizeof(struct ethhdr) > data_end)
                return XDP_DROP;

        h_proto = eth->h_proto;

        if (h_proto == htons(ETH_P_IPV6)) {
                value = bpf_map_lookup_elem(&rxcnt, &key);
                if (value)
                        *value += 1;
                return XDP_DROP;
        }

        return XDP_PASS;
}

char _license[] SEC("license") = "GPL";
```

# | 4.1: Load and show map counts

```
$ sudo xdp-loader load -m skb -s xdp_drop_ipv6 veth1 xdp_drop_ipv6_count.o

<receive some IPv6 packets>

$ sudo bpftool map show
bpftool map show
13: percpu_array  name rxcnt  flags 0x0
        key 4B  value 8B  max_entries 1  memlock 4096B
        btf_id 20
19: array  name xdp_disp.rodata  flags 0x480
        key 4B  value 84B  max_entries 1  memlock 8192B
        btf_id 28  frozen
# bpftool map dump id 13
[{
        "key": 0,
        "values": [{
                "cpu": 0,
                "value": 13
            },{
                "cpu": 1,
                "value": 7
            },{
                "cpu": 2,
                "value": 0
            },{
                "cpu": 3,
                "value": 0
            }
        ]
    }
]
```

```c
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <linux/if_link.h>
#include <signal.h>
#include <net/if.h>
#include <assert.h>

/* In this example we use libbpf-devel and libxdp-devel */
#include <bpf/bpf.h>
#include <bpf/libbpf.h>
#include <xdp/libxdp.h>

/* We define the following variables global */
static int ifindex;
struct xdp_program *prog = NULL;

/* This function will remove XDP from the link when program exit */
static void int_exit(int sig)
{
    xdp_program__close(prog);
    exit(0);
}
```

# | 5: Load the BPF program with libbpf/libxdp

```c
/* This function will count the per CPU number and print out total
 * dropped packets number and PPS(packets per second).
 */
static void poll_stats(int map_fd, int interval)
{
    int ncpus = libbpf_num_possible_cpus();
    if (ncpus < 0) {
        printf("Error get possible cpus\n");
        return;
    }
    long values[ncpus], prev[ncpus], total_pkts;
    int i, key = 0;

    memset(prev, 0, sizeof(prev));

    while (1) {
        long sum = 0;

        sleep(interval);
        assert(bpf_map_lookup_elem(map_fd, &key, values) == 0);
        for (i = 0; i < ncpus; i++)
            sum += (values[i] - prev[i]);
        if (sum) {
            total_pkts += sum;
            printf("total dropped %10llu, %10llu pkt/s\n",
                   total_pkts, sum / interval);
        }
        memcpy(prev, values, sizeof(values));
    }
}
```

```c
int main(int argc, char *argv[])
{
    int prog_fd, map_fd, ret;
    struct bpf_object *bpf_obj;

    if (argc != 2) {
        printf("Usage: %s IFNAME\n", argv[0]);
        return 1;
    }

    ifindex = if_nametoindex(argv[1]);
    if (!ifindex) {
        printf("get ifindex from interface name failed\n");
        return 1;
    }

    /* load XDP object by libxdp */
    prog = xdp_program__open_file("xdp_drop_ipv6_count.o", "xdp_drop_ipv6", NULL);
    if (!prog) {
        printf("Error, load xdp prog failed\n");
        return 1;
    }
```

# | 5: Load the BPF program with libbpf/libxdp

```
    /* attach XDP program to interface with skb mode
     * Please set ulimit if you got an -EPERM error.
     */
    ret = xdp_program__attach(prog, ifindex, XDP_MODE_SKB, 0);
    if (ret) {
        printf("Error, Set xdp fd on %d failed\n", ifindex);
        return ret;
    }

    /* Find the map fd from bpf object */
    bpf_obj = xdp_program__bpf_obj(prog);
    map_fd = bpf_object__find_map_fd_by_name(bpf_obj, "rxcnt");
    if (map_fd < 0) {
        printf("Error, get map fd from bpf obj failed\n");
        return map_fd;
    }

    /* Remove attached program when program is interrupted or killed */
    signal(SIGINT, int_exit);
    signal(SIGTERM, int_exit);

    poll_stats(map_fd, 2);

    return 0;
}
```

```
# Warn: dont load it on default interface!
$ sudo ulimit -l unlimited
$ gcc xdp_drop_ipv6_count_user.c -o xdp_drop_ipv6_count -lbpf -lxdp
$ sudo ./xdp_drop_ipv6_count veth1
total dropped          2,          1 pkt/s
total dropped        129,         63 pkt/s
total dropped        311,         91 pkt/s
total dropped        492,         90 pkt/s
total dropped        674,         91 pkt/s
total dropped        856,         91 pkt/s
total dropped       1038,         91 pkt/s
^C
```

All the example code could be find in
https://github.com/liuhangbin/xdp_examples

# Q & A

# Thanks!