Packet processing by Express Data Path XDP

Muhammad Zohaib Aslam Khan
Saifullah Khan

July 2022

Contents

1 Abstract

Today networks have been grown very large and growing complexity of network service have required ever-higher packet processing rates, so many programmable tool-kits for packet processing have been introduced. Mostly are using kernel bypass technique, such tool-kits generally bypass the operating system. It speeds up the process but there are many drawbacks, for example these cannot be implemented within existing kernel, in worst scenario there can be separate environment for processing where operating system's familiar tools cannot be used. This result in increased system complexity and blur security boundaries.
To avoid this, XDP (express data path) has been used that is actually part of main Linux kernel.It can be implemented (programmed) in kernel. Applications that are attached to kernel are written in high level language C, converted into byte format which kernel verifies and attach to it.

2 Introduction

XDP is last layer of Linux kernel stack that is present at lowest level of software level. It is present only in RX path, inside the network device drivers, allowing us to analyse packets at earliest stage as possible.XDP interpret packets even before OS allocate memory in network stack (skb).
It is also possible for some NICs which can afford (NICs drivers directly support offload XDP program on device), XDP can be offloaded. This make processing more faster. XDP take opposite approach to kernel bypass, instead of moving network hardware control out of the kernel, the performance of packet processing moves in the kernel. This retain both advantage of security and no distance between network hardware and packet processing program.
XDP program is attached in kernel by eBPF an extended version of BPF or Berkeley Packet Filter. eBPF emerged as efficient network filter from 1992. The eBPF XDP type enable eBPF program to inspect packet early into stack. There are many actions performed by XDP.
Before diving into XDP, look for eBPF Programe how it help.

3 eBPF

eBPF is an instruction set and execution environment in the Linux Kernel.it is composed of series of component to compile, verify and and execute the source code.

3.1 Work flow

An eBPF program is written in a high-level language (restricted C). The clang compiler convert/compile it to ELF object. Then this ELF object is loaded into kernel by different helper functions, during this process versifier verifies the program so it doesn't harm the kernel.The program can be offloaded to hardware or executed by processor.

3.2 Compiler

LLVM allow us to develop ebpf code in c and generate executable format file using clang. Also, the BPF Compiler Collection (BCC) project enables extra abstractions over the standard C code to facilitate writing and interacting with eBPF programs, as well as libbpf, the main upstream library for user space interaction with eBPF.
There are some restrictions in compiling of ebpf program
• eBPF can only use a subset of C language libraries. For example, the printf() function is not available for use;
• Non-static global variables are not allowed
• Only bounded loops are allowed.
• Stack space is limited to 512 by

3.3 Verifier

To ensure the integrity of working kernel, verifier verifies the ELF object before attaching to the kernel. There are two options, verifier can pass or reject the program to do so verifier consider many aspects.
Among other things, the verifier checks if a program is larger than allowed (current limit is 106 instructions), whether or not the program ends, if the memory addresses are within the memory range allowed for the program, and how deep the execution path is. It is called after the code has been compiled and during the process of loading the program into the data plane.

3.4 Maps

eBPF maps offer a two-way data structure for transferring data in and out of kernel-space. Maps are the only way for an eBPF program to communicate with other eBPF program invocations and/or user-space.

3.4.1 Map-Creation

Maps are created by invoking the bpf syscall with the BPF-MAP-CREATE argument. One can also make use the SEC attribute discussed earlier to automatically create.

3.4.2 In-Kernel-Aggregation

An interesting property of eBPF maps is the in-kernel aggregation. If you, for example, want to compute the minimum or maximum value, you can determine this value in the eBPF program and thus not stream all values to user-space.

This significantly decreases the overhead compared to systems that transfer all samples to user-space for processing.

3.4.3 Map-types

Different map types:-
1:- BPF_MAP_TYPE_CGROUP_ARRAY
2:- BPF_MAP_TYPE_UNSPEC
3:- BPF_MAP_TYPE_HASH
4:- BPF_MAP_TYPE_ARRAY
5:- BPF_MAP_TYPE_PROG_ARRAY
6:- BPF_MAP_TYPE_PERF_EVENT_ARRAY
7:- BPF_MAP_TYPE_PERCPU_HASH
8:- BPF_MAP_TYPE_PERCPU_ARRAY
9:- BPF_MAP_TYPE_STACK_TRACE

4 XDP

XDP program being the part of ebpf program is attached at early stage of RX path. At this stage, programs are capable of taking quick decisions about incoming packets and also performing arbitrary modifications on them, avoiding additional overhead imposed by processing inside the kernel.
This renders the XDP as the best hook in terms of performance speed for applications such as mitigation of DDoS attacks. After processing a packet, an XDP program returns an action, which represents the final verdict regarding what should be done to the packet after program exit.

4.1 XDP Actions

The action is specified as a program return code, which is stored at register r0 right before the eBPF programe exists.

4.2 XDP-PASS

XDP-PASS mean that XDP program pass packet to the normal network stack. we can do modification in data and then pass it to network stack.

4.3 XDP-DROP

XDP DROP is the simplest action which instruct device driver to drop the packet. Dropping pocket simply mean recycle packet back into RX ring.

4.4 XDP-TX

TX action result in bouncing back packet at NIC where it come from. This is combined with modification in the packet. we can swap destination and source addresses in the packet's headers.

4.5 XDP-ABORTED

This is not a XDP program action but this return code is something eBPF program return when get any error, e.g division by zero.
This results in the dropping of packet, but is different from XDP-DROP.

4.6 XDP_ REDIRECT

XDP-REDIRECT allow XDP programe to pass packet to another NIC, any cpu and AF-XDP socket for passing to user space processing. unlike other return codes this required addition argument that is target.