BPF licensing

Background

- Classic BPF was BSD licensed

"BPF" was originally introduced as BSD Packet Filter in http://www.tcpdump.org/papers/bpf-usenix93.pdf. The corresponding instruction set and its implementation came from BSD with BSD license. That original instruction set is now known as "classic BPF".

However an instruction set is a specification for machine-language interaction, similar to a programming language. It is not a code. Therefore, the application of a BSD license may be misleading in a certain context, as the instruction set may enjoy no copyright protection.

- eBPF (extended BPF) instruction set continues to be BSD

In 2014, the classic BPF instruction set was significantly extended. We typically refer to this instruction set as eBPF to disambiguate it from cBPF. The eBPF instruction set is still BSD licensed.

Implementations of eBPF

Using the eBPF instruction set requires implementing code in both kernel space and user space.

In Linux Kernel

The reference implementations of the eBPF interpreter and various just-in-time compilers are part of Linux and are GPLv2 licensed. The implementation of eBPF helper functions is also GPLv2 licensed. Interpreters, JITs, helpers, and verifiers are called eBPF runtime.

In User Space

There are also implementations of eBPF runtime (interpreter, JITs, helper functions) under Apache2 (https://github.com/iovisor/ubpf), MIT (https://github.com/qmonnet/rbpf), and BSD (https://github.com/DPDK/dpdk/blob/main/lib/librte_bpf).

In HW

The HW can choose to execute eBPF instruction natively and provide eBPF runtime in HW or via the use of implementing firmware with a proprietary license.

In other operating systems

Other kernels or user space implementations of eBPF instruction set and runtime can have proprietary licenses.

Using BPF programs in the Linux kernel

Linux Kernel (while being GPLv2) allows linking of proprietary kernel modules under these rules: Documentation/process/license-rules.rst

When a kernel module is loaded, the linux kernel checks which functions it intends to use. If any function is marked as "GPL only," the corresponding module or program has to have GPL compatible license.

Loading BPF program into the Linux kernel is similar to loading a kernel module. BPF is loaded at run time and not statically linked to the Linux kernel. BPF program loading follows the same license checking rules as kernel modules. BPF programs can be proprietary if they don't use "GPL only" BPF helper functions.

Further, some BPF program types - Linux Security Modules (LSM) and TCP Congestion Control (struct_ops), as of Aug 2021 - are required to be GPL compatible even if they don't use "GPL only" helper functions directly. The registration step of LSM and TCP congestion control modules of the Linux kernel is done through EXPORT_SYMBOL_GPL kernel functions. In that sense LSM and struct_ops BPF programs are implicitly calling "GPL only" functions. The same restriction applies to BPF programs that call kernel functions directly via unstable interface also known as "kfunc".

Packaging BPF programs with user space applications

Generally, proprietary-licensed applications and GPL licensed BPF programs written for the Linux kernel in the same package can co-exist because they are separate executable processes. This applies to both cBPF and eBPF programs.