

---

## Awesome eBPF

A curated list of awesome projects related to eBPF.

BPF, as in *Berkeley Packet Filter*, is an in-kernel virtual machine running programs passed from user space. Initially implemented on BSD, then Linux, the (now legacy) “classic BPF” or cBPF machine would be used with tools like tcpdump for filtering packets in the kernel to avoid useless copies to user space. More recently, the BPF infrastructure in Linux has been completely reworked and gave life to the “extended BPF”, or eBPF, which gained new features (safety and termination checks, JIT-compiling for programs, persistent maps, a standard library, hardware offload support, etc.) and is now used for many tasks. Processing packets at a very low level (XDP), tracing and monitoring events on the system, or enforcing access control over cgroups are but a few examples to which eBPF brings performance, programmability and flexibility.

Recently Cilium launched a great website about eBPF called [ebpf.io](https://ebpf.io). It serves a similar purpose to this list, with an introduction to eBPF and links to related projects.

Note: eBPF is an exciting piece of technology, and its ecosystem is constantly evolving. We’d love help from *you* to keep this awesome list up to date, and improve its signal-to-noise ratio in anyway we can. Please feel free to leave any feedback.

### Contents

- Reference Documentation
- Articles and Presentations
- Tutorials
- Examples
- eBPF Workflow: Tools and Utilities
- Projects Related to eBPF
- eBPF in Security
- The Code
- Development and Community
- Other Lists of Resources on eBPF
- Acknowledgement

---

## Reference Documentation

### eBPF Essentials

- [ebpf.io](#) - A gateway to discover all the basics of eBPF, including a listing of the main related projects and of community resources.
- Cilium's BPF and XDP Reference Guide - In-depth documentation about most features and aspects of eBPF.

### Kernel Documentation

- BPF Documentation - Index for BPF-related documentation coming with the Linux kernel.
- [linux/Documentation/networking/filter.rst](#) - eBPF specification (somewhat outdated; information should still be valid, but not exhaustive).
- BPF Design Q&A - Frequently Asked Questions on the decisions behind the BPF infrastructure.
- HOWTO interact with BPF subsystem - Frequently Asked Questions about contributing to eBPF development.

### Manual Pages

- [bpf\(2\)](#) - Manual page about the [bpf\(\)](#) system call, used to manage BPF programs and maps from userspace.
- [tc-bpf\(8\)](#) - Manual page about using BPF with tc, including example commands and samples of code.
- [bpf-helpers\(7\)](#) man page - Description of the in-kernel helper functions forming the BPF standard library.

### Other

- IO Visor's Unofficial eBPF spec - Summary of eBPF syntax and operation codes.
- Jesper Dangaard Brouer's documentation - Work in progress, contributions welcome.
- Emails from David Miller to the xdp-newbies mailing list:
  - [bpf.h](#) and you...
  - Contextually speaking...
  - BPF Verifier Overview
- List of BPF features per kernel version

---

## Articles and Presentations

### Generic eBPF Presentations and Articles

If you are new to eBPF, you may want to try the links described as “introductions” in this section.

- A brief introduction to XDP and eBPF - An accessible introduction providing context, history, and details about the functioning of eBPF.
- An eBPF Overview - Blog series by Adrian Ratiu, covering many aspects of the eBPF infrastructure:
  - Part 1: Introduction
  - Part 2: Machine & Bytecode
- Ferris Ellis’s blog posts about eBPF - They have a few posts about eBPF:
  - Part 1: Past, Present, and Future
  - Part 2: Syscall and Map Types
- A BPF reference guide - About BPF C and bcc Python helpers, from bcc repository.
- Making the Kernel’s Networking Data Path Programmable with BPF and XDP - A set of slides covering all the basics about eBPF and XDP (mostly for network processing).
- The BSD Packet Filter - An introduction mostly covering the tracing aspects.
- BPF: tracing and more - An introduction mostly covering the tracing aspects.
- Linux BPF Superpowers - An introduction mostly covering the tracing aspects, first part with flame graphs.
- IO Visor - Also introduces IO Visor project.
- BPF – in-kernel virtual machine - Presentation by the author of eBPF.
- Extending extended BPF - A blog post from 2014 on the development of BPF and demonstrating what can be done with it, using an example of stateful socket filtering by attaching an eBPF program to a socket.
- Greg Marsden made some documentation about eBPF:
  - A Tour of Program Types - A description of all existing hooks for BPF program types, and of their interest.
  - BPF helper functions - A review of the kernel functions that can be called from within eBPF programs.

- 
- Communicating with Userspace - How BPF communicates with userspace - BPF maps, perf events, `bpf_trace_printk`.
    - Building BPF Programs - Setting up your environment to build BPF programs.
    - The BPF Bytecode and the BPF Verifier - How does BPF ensure that programs are safe?
    - Using BPF to do Packet Transformation - One eBPF usage about packet transformation.
  - Linux Kernel Observability through eBPF - A blog post covering the basics of eBPF as well as code samples in Go on how to build and load a minimal eBPF program into the kernel.
  - eBPF - From a Programmer's Perspective - A short paper describing the fundamentals of eBPF and how to get started with writing eBPF programs.
  - Cloudflare's blog posts on eBPF - Different blog posts about networking use cases and low-level aspects of eBPF.
  - Linux Extended BPF (eBPF) Tracing Tools - An in-depth collection of information around examples of performance analysis tools using eBPF. Contains also a section at the end of the page about other resources.
  - Beginner's guide to eBPF - A set of live-coding talks and the accompanying code examples, introducing eBPF programming using a variety of libraries and program types.

## **BPF Internals**

- Daniel Borkmann has made several presentations and papers covering the internals of eBPF, in particular about its use with tc.
  - eBPF and XDP walkthrough and recent (2017) updates
  - Advanced programmability and recent updates with tc's `cls_bpf` - Details on eBPF, its use for tunneling and encapsulation, direct packet access, and more.
  - `cls_bpf`/eBPF updates since netdev 1.1 - Part of this tc workshop.
  - On getting tc classifier fully programmable with `cls_bpf` - Introduction to eBPF, including several features (map management, tail calls, verifier). The full paper is also available here.
  - Linux tc and eBPF
- IO Visor blog
- Linux Networking Explained - Linux networking internals, with a part about eBPF.

## **Kernel Tracing**

- Full-system dynamic tracing on Linux using eBPF and `bpftool` - A detailed introduction to tracing with eBPF, from listing the available trace points to running `bpftool` programs.

- 
- Meet-cute between eBPF and Kernel Tracing - Kprobes, uprobes, ftrace.
  - Linux Kernel Tracing - Systemtap, Kernelshark, trace-cmd, LTTng, perf-tool, ftrace, hist-trigger, perf, function tracer, tracepoint, kprobe/uprobe, and more.
  - Brendan Gregg's blog, and in particular Linux BPF Superpowers article.

## **XDP**

- The eXpress Data Path - A very accessible introduction to XDP, providing sample code to show how to process packets.
- All XDP details in a technical paper: The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel, by Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern and David Miller, all being essential eBPF and XDP contributors.
- Work-in-progress documentation for XDP
- BPF and XDP Reference Guide - Guide from the Cilium project.
- XDP Project overview
- eXpress Data Path (XDP) - The first presentation about XDP.
- BoF - What Can BPF Do For You?
- eXpress Data Path - Contains some benchmark results obtained with the mlx4 driver.
- Jesper Dangaard Brouer has several sets of slides describing the internals of XDP:
  - XDP – eXpress Data Path, Intro and future use-cases - Linux Kernel's fight against DPDK. Future plans (as of this writing) for XDP and comparison with DPDK.
  - Network Performance Workshop - Additional hints about XDP internals and expected evolution.
  - XDP – eXpress Data Path, Used for DDoS protection - Details and use cases about XDP, with benchmark results, and code snippets for benchmarking as well as for basic DDoS protection with eBPF/XDP (based on an IP blacklisting scheme).
  - Memory vs. Networking, Provoking and fixing memory bottlenecks - Advanced details about current memory issues faced by XDP developers.
  - XDP for the Rest of Us - How to get started with eBPF and XDP for normal humans. Also summarized by Julia Evans on her blog.
  - XDP now with REDIRECT - Update on XDP, and in particular on the redirect actions.
- XDP workshop – Introduction, experience, and future development (Video)

- 
- High Speed Packet Filtering on Linux - About packet filtering on Linux, DDoS protection, packet processing in the kernel, kernel bypass, XDP and eBPF.
  - How to drop 10 million packets per second - Cloudflare's blog post talking about their move to using XDP for packet filtering.

## **AF\_XDP**

- AF\_XDP - Kernel documentation on the AF\_XDP address family.
- Fast Packet Processing in Linux with AF\_XDP

## **bpfilter**

- Why is the kernel community replacing iptables with BPF? - A blog post by Cilium on the the motivations behind eBPF and bpfilter, with a couple examples and links to other projects using eBPF and bpfilter.
- bpfilter: Linux firewall with eBPF sauce - Slides from a talk by Quentin Monnet with a background on eBPF and comparing bpfilter to iptables.

## **BTF**

- BPF Type Format (BTF) - Kernel documentation about BTF, explaining how to use it.
- Enhancing the Linux kernel with BTF type information - A description of the work done with BTF to provide debugging information for BPF programs.

## **cBPF**

- The BSD Packet Filter: A New Architecture for User-level Packet Capture - The original paper about (classic) BPF.
- The FreeBSD manual page about BPF
- Linux' packet mmap(2), BPF, and Netsniff-NG
- tc and cls bpf: lightweight packet classifying with BPF
- Introducing Cloudflare's BPF Tools - Usage of BPF bytecode with the `xt_bpf` module for iptables.
- Libpcap filters syntax

---

## Hardware Offload

- eBPF/XDP hardware offload to SmartNICs - Hardware offload for eBPF with TC or XDP (Linux kernel 4.9+), introduced by Netronome.
- Comprehensive XDP offload—Handling the edge cases - An update on the topic above.
- hBPF - eBPF in hardware - An eBPF CPU written for FPGAs.
- OpenCSD eBPF SSD offloading - Computational Storage simulation (QEMU) platform with FUSE LFS filesystem for Zoned Namespaces NVMe SSDs using uBPF for compute kernel offloading, all in userspace.

## Tutorials

- bcc Reference Guide - Many incremental steps to start using bcc and eBPF, mostly centered on tracing and monitoring.
- bcc Python Developer Tutorial - Comes with bcc, but targets the Python bits across seventeen “lessons”.
- Building BPF applications with libbpf-bootstrap - Helps generate minimal or advanced templates to bootstrap your own applications (kernel side and user space management for maps and programs) with features like CO-RE, global variables, and ring buffer.
- How I ended up writing opensnoop in pure C using eBPF - A thorough walk-through of how to write eBPF programs, first using only bpf() syscall, and then libbpf library, with reproducible code examples.
- Linux Tracing Workshops Materials - Involves the use of several BPF tools for tracing.
- Tracing a packet journey using Linux tracepoints, perf and eBPF - Troubleshooting ping requests and replies with perf and bcc programs.
- Open NFP platform - Operated by Netronome: some tutorials for network-related eBPF use cases, including an eBPF Offload Starting Guide.
- XDP for the Rest of Us - First edition of a workshop to get started with XDP.
- XDP for the Rest of Us - Second edition, with new contents.
- Load XDP programs using the ip (iproute2) command
- XDP Hands-On Tutorial - A progressive (three levels of difficulty) tutorial to learn how to process packets with XDP.
- All your tracing are belong to BPF - A step-by-step walkthrough to integrate tracing capabilities in your C++ applications with the LLVM libraries.
- Firewalling with BPF/XDP: Examples and Deep Dive - A simple guide to build basic firewalls with TC and XDP.
- A Deep Dive into eBPF: Writing an Efficient DNS Monitoring. - A detailed explanation of methods used to capture DNS requests at the socket filter layer.

- 
- eBPF Developer Tutorial - Learn eBPF by examples - Start with eBPF basics and progress to advanced topics using 20+ hands-on tutorials and examples. Covers performance, networking, and security with libbpf and CO-RE. Available in Chinese and English.
  - Catch Performance Regressions in eBPF - A step-by-step guide to benchmarking both the client and kernel eBPF code written in Rust.

## Examples

- linux/samples/bpf/ - In the kernel tree: some sample eBPF programs.
- linux/tools/testing/selftests/bpf - In the kernel tree: Linux BPF selftests, with many eBPF programs.
- prototype-kernel/kernel/samples/bpf - Jesper Dangaard Brouer's prototype-kernel repository contains some additional examples that can be compiled outside of kernel infrastructure.
- iproute2/examples/bpf/ - Some networking programs to attach to the TC interface.
- Netronome sample network applications - Provides basic but complete examples of eBPF applications also compatible with hardware offload.
- bcc/examples - Examples coming along with the bcc tools, mostly about tracing.
- bcc/tools - These tools themselves can be seen as example use cases for BPF programs, mostly for tracing and monitoring. bcc tools have been packaged for some Linux distributions.
- MPLSiNIP sample - A heavily commented sample demonstrating how to encapsulate & decapsulate MPLS within IP. The code is commented for those new to BPF development.
- ebpf-samples - A collection of compiled (as ELF object files) samples gathered from several projects, primarily intended to serve as test cases for user space verifiers.
- ebpf-kill-example - A fully documented and tested example of an eBPF probe that logs all force-kills and prints them out in user-space.
- redbpf examples - Example programs for using RedBPF to write eBPF programs in Rust.
- XDP/TC-eBPF example - Program that uses XDP/TC-eBPF to provide statefull firewalling and socket redirection.

## eBPF Workflow: Tools and Utilities

### bcc

- bcc - Framework and set of tools - One way to handle BPF programs, in particular for tracing and monitoring. Also includes some utilities that may help inspect maps or programs on the system.
- Lua front-end for BCC - Another alternative to C, and even to most of the Python code used in bcc.



---

## iproute2

- iproute2 - Package containing tools for network management on Linux. In particular, it contains `tc`, used to manage eBPF filters and actions, and `ip`, used to manage XDP programs. Most of the code related to BPF is in `lib/bpf.c`.
- iproute2-next - The development tree, synchronised with net-next.

## LLVM

- LLVM - Contains several tools used in eBPF workflows. Snapshots of the latest versions for Ubuntu/Debian can be retrieved from [here](#).
  - clang is used to compile C to eBPF object file under the ELF format (clang v3.7.1+). The BPF backend was added with this commit.
  - llvm-objdump is used to dump the content of an object file in human-readable format, possibly with the initial C source code (llvm-objdump v4.0+).
  - llvm-mc is used to compile from LLVM intermediate representation to eBPF object file, so that one can compile from C to eBPF assembly, tinker with assembly, then compile to ELF file.

## libbpf

- libbpf - A C library used for handling BPF objects (programs and maps), and manipulating ELF object files containing them. It is shipped with the kernel and mirrored on GitHub.
- libbpf-bootstrap - Scaffolding for BPF application development with libbpf and BPF CO-RE.

## Go libraries

- cilium/ebpf - Pure-Go library to read, modify and load eBPF programs and attach them to various hooks in the Linux kernel.
- libbpfgo - eBPF library for Go, powered by libbpf.
- gobpf - Go bindings for BCC for creating eBPF programs.

## Aya

- aya - A pure Rust library for writing, loading, and managing eBPF objects, with a focus on developer experience and operability. It supports writing eBPF programs in Rust and distributing library code over crates.io to share it between eBPF programs. Aya does not depend on libbpf.

- 
- `aya-template` - Templates for writing BPF applications in Aya that can be used with `cargo generate`.
  - `Ebpfguard` - Rust library for writing Linux security policies using eBPF.

## **zbp**

- `zbp` - A pure Zig framework for writing cross platform eBPF programs, powered by `libbpf` and Zig toolchain.

## **eunomia-bpf**

- `eunomia-bpf` - A compilation framework and runtime library to build, distribute, dynamically load, and run CO-RE eBPF applications in multiple languages and WebAssembly. It supports writing eBPF kernel code only (to build simple CO-RE `libbpf` eBPF applications), writing the kernel part in both BCC and `libbpf` styles, and writing userspace in multiple languages in a WASM module and distributing it with simple JSON data or WASM OCI images. The runtime is based on `libbpf` only and provides CO-RE to BCC-style eBPF programs without depending on the LLVM library.

## **oxidebpf**

- `oxidebpf` - A pure Rust library for managing eBPF programs, designed for security use cases. The featureset is more limited than other libraries but emphasizes stability across a wide range of kernels and backwards-compatible compile-once-run-most-places.

## **bpftool and Other Tools from the Kernel Tree**

- `bpftool` - Also some other tools in the kernel tree, under `linux/tools/net/` for versions earlier than 4.15, or `linux/tools/bpf/` after that:
  - `bpftool` - A generic utility that can be used to interact with eBPF programs and maps from userspace, for example to show, dump, load, disassemble, pin programs, or to show, create, pin, update, delete maps, or to attach and detach programs to cgroups.
  - `bpf_asm` - A minimal cBPF assembler.
  - `bpf_dbg` - A small debugger for cBPF programs.
  - `bpf_jit_disasm` - A disassembler for both BPF flavors and could be highly useful for JIT debugging.

---

## User Space eBPF

- uBPF - Written in C. Contains an interpreter, a JIT compiler for x86\_64 architecture, an assembler and a disassembler.
- A generic implementation - With support for FreeBSD kernel, FreeBSD user space, Linux kernel, Linux user space and macOS user space. Used for the VALE software switch's BPF extension module.
- rbpf - Written in Rust. Interpreter for Linux, macOS and Windows, and JIT-compiler for x86\_64 under Linux.
- PREVAIL - A user space verifier for eBPF using an abstract interpretation layer, with support for loops.
- oster - Written in Go. A tool for tracing execution of Go programs by attaching eBPF to uprobes.
- wachy - A tracing profiler that aims to make eBPF uprobe-based debugging easier to use. This is done by displaying traces in a UI next to the source code and allowing interactive drilldown analysis.

## eBPF on Other Platforms

- eBPF for Windows - This project is a work-in-progress that allows using existing eBPF toolchains and APIs familiar in the Linux ecosystem to be used on top of Windows.

## Testing in Virtual Environments

- A Vagrant setup - To easily test XDP. Less useful now that generic XDP (driver-independant, mostly for testing) exists.
- bcc in a Docker container

## Projects Related to eBPF

### Networking

- P4 has some interactions with eBPF:
  - P4 on the Edge - P4 with eBPF to create high-performance programmable switches.
  - OvS Orbit episode (#11), called P4 on the Edge - Related to the former item. Audio interview of John Fastabend by Ben Pfaff, one of the core maintainers of Open vSwitch.
  - P4, EBPF and Linux TC Offload - P4 with some elements related to eBPF hardware offload on Netronome's NFP (Network Flow Processor) architecture.

- 
- Old documentation for P4 usage with eBPF - From bcc repository; deprecated by the P4\_16 backend linked below.
    - P4\_16 backend for eBPF
  - Cilium project (GitHub repository) is a technology relying on BPF and XDP to provide “fast in-kernel networking and security policy enforcement for containers based on eBPF programs generated on the fly”. Many presentations available (with overlap):
    - Cilium: Networking & Security for Containers with BPF & XDP - Also featuring a load balancer use case
    - Cilium: Networking & Security for Containers with BPF & XDP - video
    - Cilium: Fast IPv6 container Networking with BPF and XDP
    - Cilium: BPF & XDP for containers
    - OvS Orbit episode (#4) - Interview of Thomas Graf by Ben Pfaff.
    - A generic introduction to Cilium
    - A podcast interviewing Thomas Graf - Ivan Pepelnjak interviewing Thomas, October 2016, on eBPF, P4, XDP and Cilium.
  - Open vSwitch (OvS), and its related project Open Virtual Network (OVN, an open source network virtualization solution) are considering using eBPF at various level:
    - Offloading OVS Flow Processing using eBPF
    - Coupling the Flexibility of OVN with the Efficiency of IOVisor
  - Katran - A layer 4 load-balancer based on XDP, open-sourced by Facebook.
  - XDP in practice: integrating XDP in our DDoS mitigation pipeline - Protection against DDoS with XDP at Cloudflare.
  - Droplet: DDoS countermeasures powered by BPF + XDP - Protection against DDoS with XDP at Facebook.
  - DPDK has a poll-mode driver (PMD) based on AF\_XDP
  - CETH for XDP - Common Ethernet Driver Framework for faster network I/O, a technology initiated by Mellanox.
  - Suricata, an open source intrusion detection system, relies on eBPF components for its “capture bypass” features:
    - “eBPF and XDP” section of Suricata documentation
    - SEPTun-Mark-II - Extreme Performance Tuning guide - Mark II.
    - A blog post introducing the feature
    - The adventures of a Suricate in eBPF land
-

- 
- eBPF and XDP seen from the eyes of a meerkat
  - Project Calico - Calico is an open source networking and network security solution for containers, virtual machines, and native host-based workloads. Calico's eBPF data plane delivers a low latency, high throughput data plane with a rich network security policy model.
    - Enabling eBPF data plane with Calico
  - merbridge - Use eBPF to speed up your Service Mesh. Merbridge replaces iptables rules with eBPF to intercept traffic. It also combines msg\_redirect to reduce latency with a shortened datapath between sidecars and services.
  - PcapPlusPlus - An open-source C++ library for capturing, parsing and crafting network packets. It features a C++ interface for creating AF\_XDP sockets, making it easy to send and receive packets through them.

## Observability

- InKeV: In-Kernel Distributed Network Virtualization for DCN
- DEEP-mon - Helps with measuring power consumption for servers and uses eBPF programs for in-kernel aggregation of data.
- pixie - Observability for Kubernetes using eBPF. Features include protocol tracing, application profiling, and support for distributed bpftool deployments.
- SkyWalking Rover - Apache SkyWalking is an open-source Application Performance Monitoring (APM) platform specially designed for distributed systems with microservices, cloud-native and container-based (Kubernetes) architectures. SkyWalking Rover is an eBPF-based profiler and metrics collector for C, C++, Golang, and Rust applications.
- parca-agent - eBPF based always-on continuous profiler for analysis of CPU and memory usage, down to the line number and throughout time.
- rbperf - Sampling profiler and tracer for Ruby.
- Hubble - Network, service and security observability for Kubernetes using eBPF.
- Caretta - Instant Kubernetes service dependency map generated by eBPF, right to a Grafana instance.

## Security

- Falco - A cloud-native runtime security project used as a Kubernetes threat detection engine.
- Sysmon for Linux - A security monitoring tool. It depends on SysinternalsEBPF.
- Red Canary Linux Agent - Red Canary has started to incorporate eBPF to their Linux security sensor.

- 
- Tracee - A runtime security and forensics tool for Linux which uses eBPF technology to trace the system and applications at runtime, and analyze collected events to detect suspicious behavioral patterns.
  - redcanary-ebpf-sensor - A set of BPF programs that gather security relevant event data from the Linux kernel. The BPF programs are combined into a single ELF file from which individual probes can be selectively loaded, depending on the running operating system and kernel version.
  - bpfflock - Lock Linux machines - An eBPF driven security tool for locking and auditing Linux machines.
  - Tetragon - Kubernetes-aware, eBPF-based security observability and runtime enforcement.

## Tools

- ply - A small but flexible open source dynamic tracer for Linux, with features similar to the bcc tools, but with a simpler language inspired by awk and DTrace.
- bpfftrace - A tool for tracing with its own high-level tracing language. It is flexible enough to be envisioned as a Linux replacement for DTrace and SystemTap.
  - bpfftrace Cheat Sheet - Summary and cheat sheet for programming in bpfftrace. Contains information about syntax, probe types, variables and functions.
- kubectrl trace - A kubectrl plug-in for executing bpfftrace programs in a Kubernetes cluster.
- inspektor-gadget - A collection of eBPF-based tools to debug and inspect Kubernetes resources and applications.
- bpf-d - Framework for running BPF programs with rules on Linux as a daemon. Container aware.
- BPFd - A distinct BPF daemon, trying to leverage the flexibility of the bcc tools to trace and debug remote targets, and in particular devices running with Android.
- adeb - A Linux shell environment for using tracing tools on Android with BPFd.
- greggd - System daemon to compile and load eBPF programs into the kernel, and forward program output to socket for metric aggregation.
- FUSE - Considers using eBPF.
- upf-bpf - An in-kernel solution based on XDP for 5G UPF.
- redbpf - Tooling and framework to write eBPF code in Rust efficiently.
- ebpf-explorer - A web interface to explore system's maps and programs.
- ebpfmon - A TUI (terminal user interface) application for real time monitoring of eBPF programs.
- bpfman - An eBPF Manager for Linux and Kubernetes. Includes a built-in program loader that supports program cooperation for XDP and TC programs, as well as deployment of eBPF programs from OCI images.

---

## eBPF in Security

- Embrace The Red: Offensive BPF! - A series of posts around the introduction into BPF with a focus to an offensive setting, and also how its misuse can be detected. Posts include discussions on the rootkit capabilities of eBPF, or on which tracing type is needed for different use cases.
- eBPF: Block Linux Fileless Payload “Malware” Execution with BPF LSM - Blog post about how BPF can help detection and blocking fileless malware.
- Blackhat 2021: With Friends Like eBPF, Who Needs Enemies? - Talk about an eBPF rootkit and how the capabilities of eBPF could be abused. The rootkit was also the object of a talk at Defcon, eBPF, I thought we were friends !.
- ebpfkit - A rootkit that leverages multiple eBPF features to implement offensive security techniques.
- ebpfkit-monitor - An utility to statically analyze eBPF bytecode or monitor suspicious eBPF activity at runtime. It was specifically designed to detect ebpfkit.
- Bad BPF - A collection of malicious eBPF programs that make use of eBPF’s ability to read and write user data in between the usermode program and the kernel.
- TripleCross - A Linux eBPF rootkit with a backdoor, C2, library injection, execution hijacking, persistence and stealth capabilities.

## The Code

- linux/include/linux/bpf.h - with linux/include/uapi/bpf.h: definitions related to eBPF, to be used respectively in the kernel and to interface with userspace programs.
- linux/include/linux/filter.h - with linux/include/uapi/filter.h: information used to run the BPF programs themselves.
- linux/kernel/bpf/ - This directory contains most of BPF-related code. In particular, those files are worth of interest:
  - `syscall.c` - Different operations permitted by the system call, such as program loading or map management.
  - `core.c` - BPF interpreter.
  - `verifier.c` - BPF verifier.
- linux/net/core/filter.c - Functions and eBPF helpers related to networking (TC, XDP etc.); also contains the code to migrate cBPF bytecode to eBPF (all cBPF programs are translated to eBPF in recent kernels).
- linux/kernel/trace/bpf\_trace.c - Functions and eBPF helpers related to tracing and monitoring (kprobes, tracepoints, etc.).

- 
- The JIT compilers are under the directory of their respective architectures, such as file `linux/arch/x86/net/bpf_jit_comp.c` for x86. Exception is made for JIT compilers used for hardware offload, sitting in their drivers, such as `linux/drivers/net/ethernet/netronome/nfp/bpf/jit.c` for Netronome NFP.
  - `linux/net/sched/` - and in particular in files `act_bpf.c` (action) and `cls_bpf.c` (filter): code related to BPF actions and filters with TC.
  - `linux/kernel/seccomp.c`
  - `linux/net/core/dev.c` - contains the function `dev_change_xdp_fd()` that is called through a Netlink command to hook a XDP program to a device, after it has been loaded into the kernel from user space. This function in turn uses a callback from the relevant driver.

## Development and Community

- The bpf-next tree - BPF patches land in this tree. It is regularly merged into net-next, which is itself merged for each release to Linus' tree.
- Kernel documentation - About contributions to BPF.
- The netdev mailing list - Mailing list for Linux kernel networking stack development. All patches are sent there for review and inclusion.
- XDP-newbies - A mailing list specially dedicated to XDP programming (both for architecture or for asking for help).
- IO Visor mailing list - BPF is at the heart of the project, and is regularly discussed on the mailing list.
- @IOVisor Twitter account
- The XDP Collaboration Project - A GitHub repository with notes and ideas regarding the future evolutions of XDP.

## Other Lists of Resources on eBPF

- IO Visor's bcc documentation
- IO Visor's bpf-docs repository
- Dive into BPF: A List of Reading Material

## Acknowledgement

Thank you to Quentin Monnet and Daniel Borkmann for their original work on Dive into BPF: A List of Reading Material which became the basis for this list.



---

## Contributing

Contributions welcome! Read the contribution guidelines first.

## License



To the extent possible under law, zoidbergwill has waived all copyright and related or neighboring rights to this work.