How to package eBPF software

presented on Gentoo Linux

Jakov Smolić
Agenda

- Introduction
- eBPF Development ecosystem
- Packaging on Gentoo
- Challenges and solutions
About

- Delivering solutions based on GNU/Linux firmware
  - Focused on network edge, network switching, and CPEs
  - OpenWrt, Gentoo, Yocto, etc.
- Continuous participation in Open Source projects
  - Contributions for the Linux kernel, systemd, etc.
eBPF - Extended Berkeley Packet Filter

- Linux subsystem that can run programs in a virtualized environment
- Initially - network packet filtering
- Today - event processing framework
- Applications: observability, networking, security, tracing
- Brendan Gregg - books and videos about Linux performance and observability
eBPF Development ecosystem
eBPF development ecosystem

- bcc: Toolkit for writing eBPF programs using Python or Lua
- bpftrace: High level tracing language
- libbpf: Userspace library for loading and interacting with eBPF programs
- eBPF Go library
- ply: Lightweight dynamic tracing tool
- More info: https://ebpf.io/applications
Packaging on Gentoo
Gentoo

- Source-based Linux distribution
- Package manager - Portage
- Allows fine-grained control over the system
  - profiles
  - USE flags
Packaging on Gentoo

- ebuild - text file with build instructions
- eclass - common code used by ebuilds
- USE flag - configuration switch
Packaging example

- https://github.com/sartura/ebpf-hello-world
- CMake build system
- Blog posts
  - https://www.sartura.hr/blog/simple-ebpf-core-application
  - https://www.sartura.hr/blog/ebpf-development-and-integration-with-replica-one
Tree

  cmake
    Modules
      FindLIBBPF.cmake
      UseBPF.cmake
  CMakeLists.txt
  include
    vmlinux-arm64.h
    vmlinux-arm.h
    vmlinux-x86.h
  README.md
  src
    hello.bpf.c
    hello.c
    maps.bpf.c
    maps.c
    maps.h
# Copyright 2023 Gentoo Authors
# Distributed under the terms of the GNU General Public License v2

EAPI=8

inherit cmake git-r3 linux-info llvm

DESCRIPTION="Sample eBPF CO-RE applications"
HOMEPAGE="https://github.com/sartura/ebpf-hello-world"
EGIT_REPO_URI="https://github.com/sartura/ebpf-hello-world.git"

LICENSE="MIT"
SLOT="0"
KEYWORDS="~amd64 ~arm ~arm64 ~x86"
IUSE="+bpfobjs vmlinux"
RESTRICT="strip"
DEPEND=""
  dev-libs/libbpf:=
  virtual/libelf
"
RDEPEND="$\{DEPEND\}"
BDEPEND=""
  dev-util/bpftool
  sys-devel/clang:=[llvm_targets_BPF(+)]
"
CONFIG_CHECK="~BPF ~BPF_EVENTS ~BPF_JIT ~BPF_SYSCALL ~HAVE_EBPF_JIT ~DEBUG_INFO_BTF"
src_configure() {
    local mycmakeargs=(
        -DBPF_ARCH=$(tc-arch-kernel)
        -DINSTALL_VMLINUX=$(usex vmlinux)
    )
    cmake_src_configure
}

src_install() {
    cmake_src_install

    if ! use bpfobjs; then
        rm -r "${ED}"/usr/share || die
    fi
}

Challenges and solutions
Challenges

- Supporting different build configurations
- Compatibility with latest toolchain components
  - gcc-10  -fno-common  change
  - clang-16  -Werror=implicit-function-declaration,implicit-int  change
- Cross-compilation issues
- Heavy build and run time dependencies
  - Not suitable for embedded systems
Solutions

- Proactively testing packages
- Writing and submitting patches upstream
- Packaging lightweight applications
How to package eBPF software

presented on Gentoo Linux

jakov.smolic@sartura.hr, jsmolic@gentoo.org