Debugging the BPF Virtual Machine

eBPF Summit

Lorenzo Fontana

October 28, 2020

Why ?

- Debugging is useful to understand how things work
- Sometimes, eBPF programs can’t even load
- I couldn’t find good resources on this, so, here I am
- I break lots of eBPF programs
- The BPF Virtual machine is not easy to understand

The approach

The BPF subsystem lives in the kernel

The kernel can be debugged using gdb

First - The environment

We need:

- A kernel image
- A root filesystem
- An eBPF program that doesn’t work
- gdb

Kernel image

```
git clone https://git.kernel.org/pub/scm/linux/kernel/git/next/linux-next.git
/source/linux

cd linux
mkdir build
make O=$PWD/build ARCH=x86_64 x86_64_defconfig
make O=$PWD/build ARCH=x86_64 menuconfig
make O=$PWD/build ARCH=x86_64 -j16
```

Remember to:

- Enable debugging symbols under Kernel Hacking -> compile options

Rootfs

```
git clone git://git.buildroot.net/buildroot /source/buildroot
cd buildroot
make menuconfig
make -j16
```

Remember to:
- Select ext4 as filesystem image
- Enable networking
- Enable the SSH daemon

Start the test VM

```
cd /source/linux
qemu-system-x86_64 -kernel build/arch/x86/boot/bzImage \
--enable-kvm \
-nic user,hostfwd=tcp::2222-:22 \
-boot c -m 2049M -hda /source/buildroot/output/images/rootfs.ext4 \
-append "root=/dev/sda rw console=ttyS0,115200 acpi=off nokaslr" \
-serial stdio -display none
```

Debug!

```
cd /source/linux
gdb build/vmlinux

(gdb) target remote localhost:1234
(gdb) bpf/syscall.c:4180
(gdb) bpf/syscall.c:796
(gdb) b bpf/syscall.c:121
(gdb) b kernel/bpf/ringbuf.c:159
```

Remember to:

- Load the eBPF program and see if the breakpoint hits

verbose(void *, const char *, ...)

- We can use it in bpf/verifier.c to do printf based debugging
- Sometimes it’s better than messing around with pointers in gdb

Thank you!