Steering connections to sockets with BPF socket lookup hook

eBPF Summit

Jakub Sitnicki, Cloudflare
@jkbs0
@cloudflare

October 28-29, 2020

Who am I?

- Software Engineer at Cloudflare
  Spectrum TCP/UDP reverse proxy, Linux kernel, ...

- Contributor to Linux kernel
  networking & BPF subsystems

CLOUDFLARE

Goal

Run a TCP echo service on ports 7, 77, and 777

… using **one** TCP listening socket.

Fun?

We will need...

- ❏ VM running Linux kernel 5.9+
- ❏ bpftool 5.9+
- ❏ libbpf headers
- ❏ kernel headers

```
vm $ uname -r
5.9.1-36.vanilla.1.fc32.x86_64
vm $ bpftool version
bpftool v5.9.1
```

CLOUDFLARE

Code and instructions at

https://github.com/jsitnicki/ebpf-summit-2020

We will need... a TCP echo server

```
$ sudo dnf install nmap-ncat
$ nc -4kle /bin/cat 127.0.0.1 7777 &
[1] 1289
$ ss -4tlpn sport = 7777
State  Recv-Q Send-Q Local Address:Port Peer Address:Port Process
LISTEN 0      10         127.0.0.1:7777      0.0.0.0:*    users:(("nc",pid=1289,fd=3))
```

Netcat + /bin/cat

```
$ nc -4 127.0.0.1 7777
hello⏎
hello
^D
```

Test it!

Check open ports on VM external IP

```
vm $ ip -4 addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    inet 192.168.122.221/24 brd 192.168.122.255 scope global dynamic noprefixroute eth0
       valid_lft 2563sec preferred_lft 2563sec
```

check VM IP

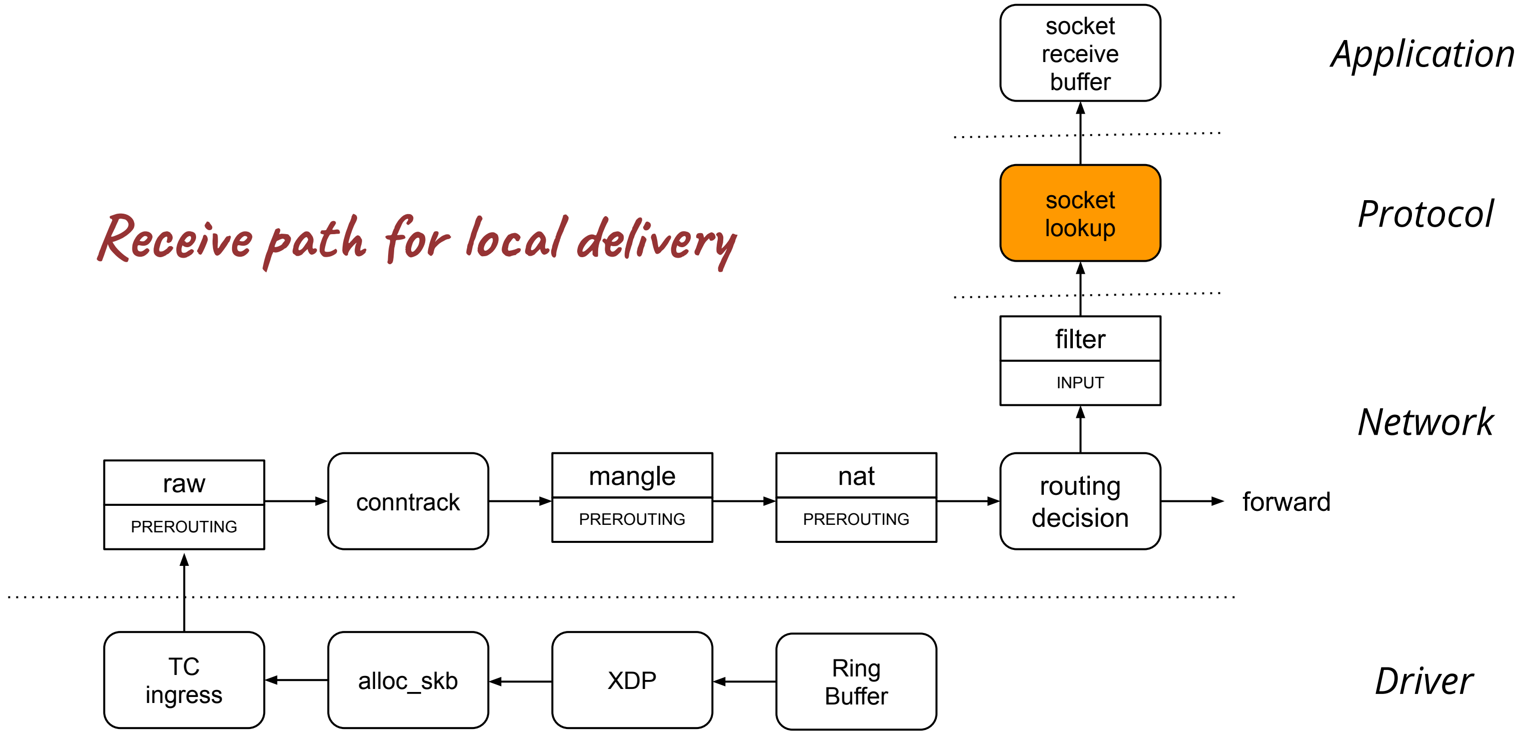
```
host $ nmap -sT -p 1-1000 192.168.122.221
…
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds
```

scan first 1000 ports

7, 77, 777 are closed

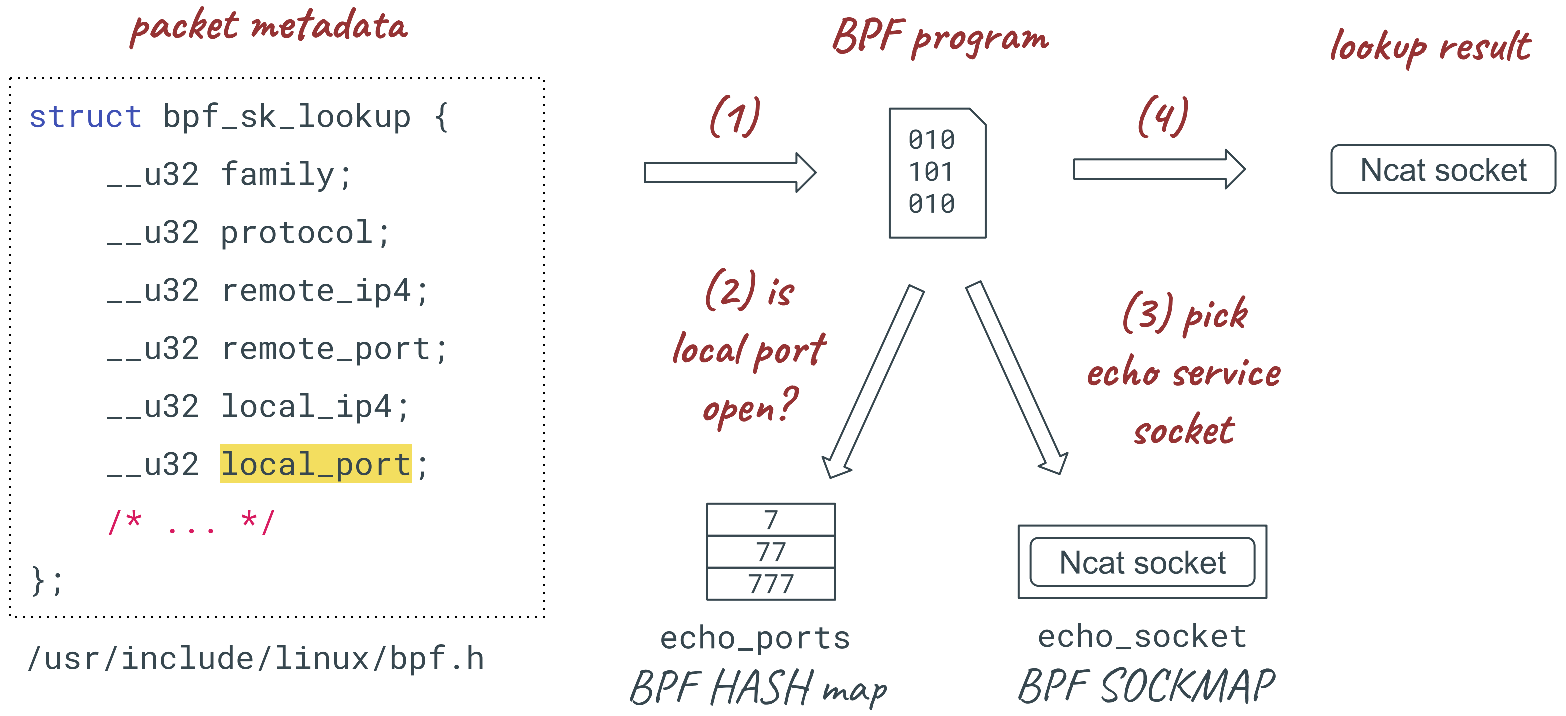
What is socket lookup?



Receive path for local delivery

Wikipedia - Packet flow in Netfilter and General Networking

Service dispatch with BPF socket lookup



```
struct bpf_sk_lookup {
    __u32 family;
    __u32 protocol;
    __u32 remote_ip4;
    __u32 remote_port;
    __u32 local_ip4;
    __u32 local_port;
    /* ... */
};
```

/usr/include/linux/bpf.h

echo_dispatch.bpf.c - BPF sk_lookup program

```
/* Declare BPF maps */

struct bpf_map_def SEC("maps") echo_ports = {
        .type               = BPF_MAP_TYPE_HASH,
        .max_entries        = 1024,
        .key_size           = sizeof(__u16),
        .value_size         = sizeof(__u8),
};


struct bpf_map_def SEC("maps") echo_socket = {
        .type               = BPF_MAP_TYPE_SOCKMAP,
        .max_entries        = 1,
        .key_size           = sizeof(__u32),
        .value_size         = sizeof(__u64),
};
```

`echo_dispatch.bpf.c` - BPF sk_lookup program

```
SEC("sk_lookup/echo_dispatch")
int echo_dispatch(struct bpf_sk_lookup *ctx)
{
    // … declarations omitted …

    port = ctx->local_port;
    open = bpf_map_lookup_elem(&echo_ports, &port);
    if (!open)
        return SK_PASS;

    sk = bpf_map_lookup_elem(&echo_socket, &zero);
    if (!sk)
        return SK_DROP;

    err = bpf_sk_assign(ctx, sk, 0);
    bpf_sk_release(sk);
    return err ? SK_DROP : SK_PASS;
}
```

is echo service configured on this port?

get echo server socket

dispatch the packet to echo server

CLOUDFLARE

Load echo_dispatch program

```
$ make echo_dispatch.bpf.o
clang -I…/linux/usr/include -I…/linux/tools/lib -g -O2 -Wall -Wextra -target bpf
-c -o echo_dispatch.bpf.o echo_dispatch.bpf.c

# bpftool prog load echo_dispatch.bpf.o /sys/fs/bpf/echo_dispatch_prog

# bpftool prog show pinned /sys/fs/bpf/echo_dispatch_prog
75: sk_lookup  name echo_dispatch  tag 77fd96f660a5d2ab  gpl
        loaded_at 2020-10-23T09:36:45+0000  uid 0
        xlated 304B  jited 197B  memlock 4096B  map_ids 28,29
        btf_id 32
```

build the prog

load & pin the prog

Pin BPF maps used by `echo_dispatch`

```
# mount -t bpf none ~vagrant/bpffs
# sudo chown vagrant.vagrant ~vagrant/bpffs

# bpftool map show id 28
28: hash  name echo_ports  flags 0x0
        key 2B  value 1B  max_entries 1024  memlock 86016B
# bpftool map pin id 28 ~vagrant/bpffs/echo_ports

# bpftool map show id 29
29: sockmap  name echo_socket  flags 0x0
        key 4B  value 8B  max_entries 1  memlock 4096B
# bpftool map pin id 29 ~vagrant/bpffs/echo_socket

# chown vagrant.vagrant ~vagrant/bpffs/{echo_ports,echo_socket}
```

mount another bpf fs

pin maps

grant access

Insert Ncat socket into **echo_socket** map

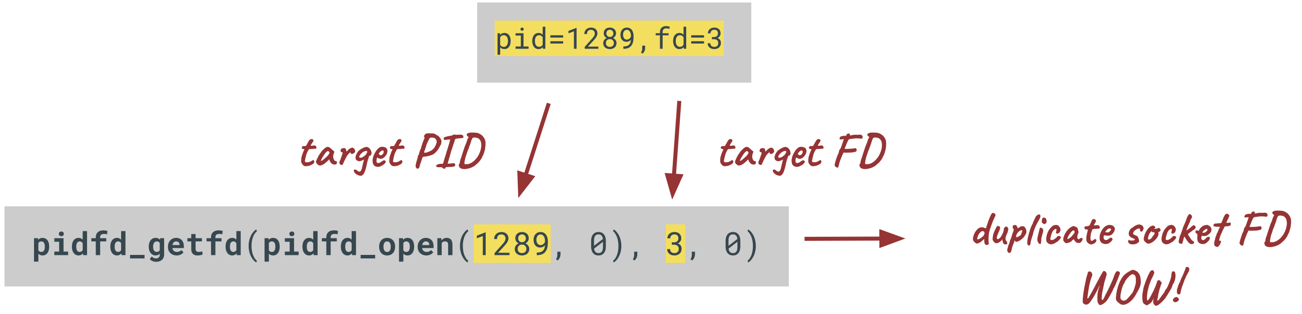
```
$ nc -4kle /bin/cat 127.0.0.1 7777 &
[1] 1289

$ ss -tlpne 'sport = 7777'
State  Recv-Q Send-Q Local Address:Port Peer Address:Port Process
LISTEN 0      10         127.0.0.1:7777      0.0.0.0:*    users:(("nc",pid=1289,fd=3))⏎
uid:1000 ino:22797 sk:1 <->
```

CLOUDFLARE

Get another socket file descriptor

1. pass FD with SCM_RIGHTS cmsg - see unix(7) man page
2. inherit FD from parent process - see systemd socket activation
3. **use pidfd_getfd() syscall - Linux 5.6+**



```
pid=1289,fd=3
```

```
pidfd_getfd(pidfd_open(1289, 0), 3, 0)
```

sockmap_update.c - Put socket FD in BPF map

```
$ ./sockmap-update
Usage: ./sockmap-update <target pid> <target fd> <pinned map path>

$ strace -e … ./sockmap-update 1289 3 $HOME/bpffs/echo_socket
pidfd_open(1289, 0)                       = 3
pidfd_getfd(3, 3, 0)                      = 4
bpf(BPF_OBJ_GET, {pathname="/home/vagrant/bpffs/echo_socket", …}, …) = 5
bpf(BPF_MAP_UPDATE_ELEM, {map_fd=5, key=0x7fff9c4e0b14, value=0x7fff9c4e0b08}, 120) = 0
+++ exited with 0 +++

$ bpftool map dump pinned $HOME/bpffs/echo_socket
key: 00 00 00 00  value: 01 00 00 00 00 00 00 00
Found 1 element
```

dup'ed socket FD

pointer to socket FD

socket cookie from ss output (sk:1)

Attach **echo_dispatch** to network namespace

```
# ./sk-lookup-attach
Usage: ./sk-lookup-attach <prog path> <link path>
# ./sk-lookup-attach /sys/fs/bpf/echo_dispatch_prog /sys/fs/bpf/echo_dispatch_link
```

```
bpf(BPF_OBJ_GET, {pathname="/sys/fs/bpf/echo_dispatch_prog", …) = 3
openat(…, "/proc/self/ns/net", …) = 4
bpf(BPF_LINK_CREATE, {link_create={prog_fd=3, target_fd=4,
                                   attach_type=BPF_SK_LOOKUP, …) = 5
bpf(BPF_OBJ_PIN, {pathname="/sys/fs/bpf/echo_dispatch_link", bpf_fd=5, …) = 0
```

syscall trace

```
# bpftool link show pinned /sys/fs/bpf/echo_dispatch_link
14: netns  prog 75
        netns_ino 4026531992  attach_type sk_lookup
$ ls -l /proc/self/ns/net
lrwxrwxrwx. 1 vagrant vagrant 0 Oct 23 13:35 /proc/self/ns/net -> 'net:[4026531992]'
```

prog attached to netns

CLOUDFLARE

Enable echo on ports 7, 77, 777

```
$ bpftool map update pinned $HOME/bpffs/echo_ports key 0x07 0x00 value 0x00
                                                        0x0007 = 7

$ bpftool map update pinned $HOME/bpffs/echo_ports key 0x4d 0x00 value 0x00
                                                        0x004d = 77

$ bpftool map update pinned $HOME/bpffs/echo_ports key 0x09 0x03 value 0x00
                                                        0x0309 = 777

$ bpftool map dump pinned $HOME/bpffs/echo_ports
key: 4d 00  value: 00
key: 07 00  value: 00
key: 09 03  value: 00                                   dump map contents
Found 3 elements
```

Re-scan open ports on VM

```
host $ nmap -sT -p 1-1024 192.168.122.221
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-24 21:56 CEST
Nmap scan report for 192.168.122.221
Host is up (0.00014s latency).
Not shown: 1020 closed ports
PORT    STATE SERVICE
7/tcp   open  echo
22/tcp  open  ssh
77/tcp  open  priv-rje
777/tcp open  multiling-http

Nmap done: 1 IP address (1 host up) scanned in 0.09 seconds
```

CLOUDFLARE

Test echo service on ports 7, 77, 777

```
$ { echo 'Hip';     sleep 0.1; } | nc -4 192.168.122.221   7 && \
> { echo 'hip';     sleep 0.1; } | nc -4 192.168.122.221  77 && \
> { echo 'hooray!'; sleep 0.1; } | nc -4 192.168.122.221 777
Hip
hip
hooray!
```

🎉

Want to use BPF socket lookup?

- Repo with code and setup instructions
  https://github.com/jsitnicki/ebpf-summit-2020
- BPF sk_lookup program kernel documentation
  https://github.com/torvalds/linux/blob/master/Documentation/bpf/prog_sk_lookup.rst
- BPF sk_lookup context object declaration
  https://github.com/torvalds/linux/blob/v5.9/include/uapi/linux/bpf.h#L4436
- bpf_sk_assign() helper documentation on bpf-helpers(7) man page
  https://man7.org/linux/man-pages/man7/bpf-helpers.7.html
- Linux kernel selftests for BPF sk_lookup program
  https://github.com/torvalds/linux/blob/v5.9/tools/testing/selftests/bpf/prog_tests/sk_lookup.c
  https://github.com/torvalds/linux/blob/v5.9/tools/testing/selftests/bpf/progs/test_sk_lookup.c
- “It’s crowded in here” blog post
  https://blog.cloudflare.com/its-crowded-in-here/
- Proof-of-concept tool for configuring BPF socket dispatch
  https://github.com/majek/inet-tool/
- “Programmable socket lookup with BPF” presentation at Linux Plumbers Conference 2019
  https://www.youtube.com/watch?v=qRDoUpqvYjY

CLOUDFLARE

Watch out for the blog post at

https://blog.cloudflare.com/

… will cover setup for UDP

Thank you!