BPF programmable listen socket lookup

Marek Majkowski, Jakub Sitnicki, Lorenz Bauer

Heavy user of AnyIP

```
$ ip -4 route show table local|grep '/'|wc -l
107

$ ip -6 route show table local|grep '/'|wc -l
50
```

bind(0.0.0.0) doesn't scale

$ ss -tuln src 0.0.0.0/32 or src ::/128 |wc -l
235

+ ~50 internal services

#1 Sharing port between apps

* udp/53 for 1.0.0.0/24 goes to resolver
* udp/53 for 162.159.0.0/16 goes to auth

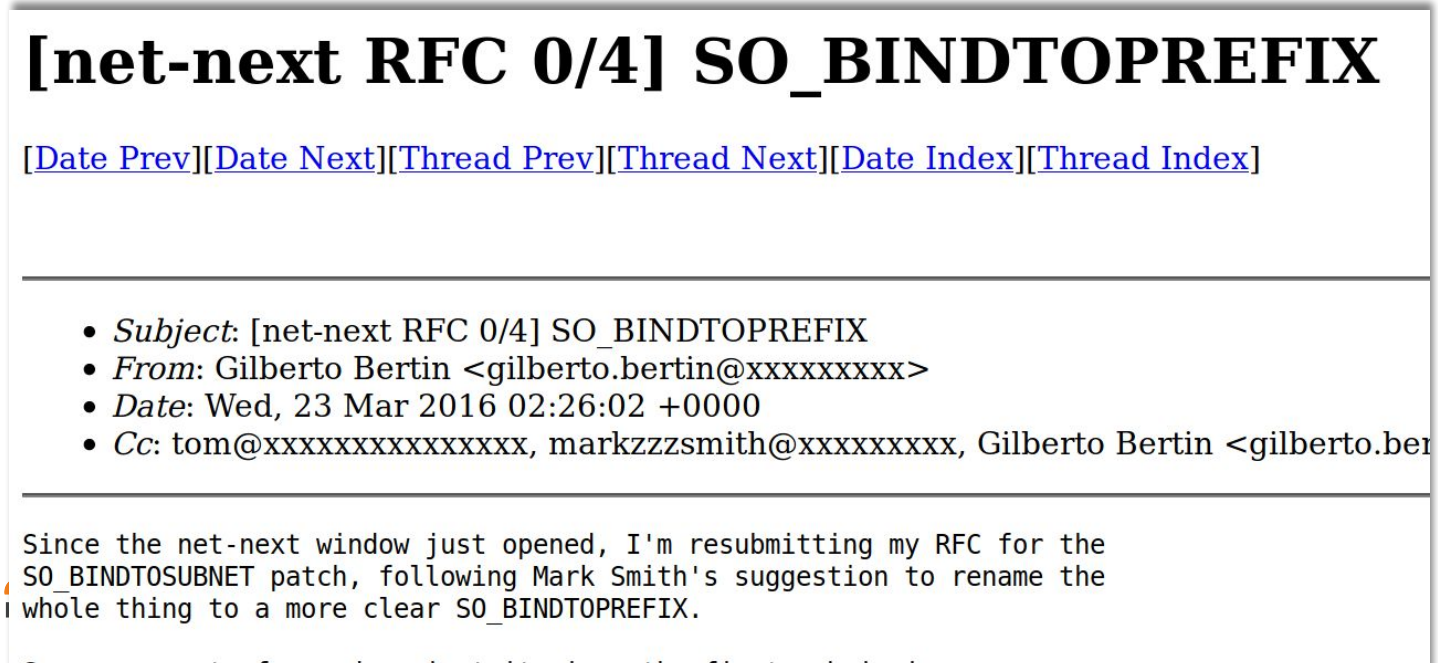
* tcp/80 0.0.0.0/0 to http-protocols
* tcp/80 172.65.128.0/24 to TCP-proxy

Dozen alternatives

- macvlan
- vrf
- BINDTODEVICE dummy
- net-ns

Say hello to SO_BINDTOPREFIX

https://www.spinics.net/lists/netdev/msg370789.html



[net-next RFC 0/4] SO_BINDTOPREFIX

[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

- *Subject*: [net-next RFC 0/4] SO_BINDTOPREFIX
- *From*: Gilberto Bertin <gilberto.bertin@xxxxxxxxx>
- *Date*: Wed, 23 Mar 2016 02:26:02 +0000
- *Cc*: tom@xxxxxxxxxxxxxxx, markzzzsmith@xxxxxxxxx, Gilberto Bertin <gilberto.ber

---

Since the net-next window just opened, I'm resubmitting my RFC for the
SO_BINDTOSUBNET patch, following Mark Smith's suggestion to rename the
whole thing to a more clear SO_BINDTOPREFIX.

Say goodbye to SO_BINDTOPREFIX

https://marc.info/?l=linux-netdev&m=145926190805592&w=2

Please do not add such monolithic option.

BPF is absolutely the way to go here, as it allows for whatever user specified tweaks, like a list of destination subnetwork, or/and a list of source network, or the date/time of the day, or port knocking without netfilter, or ... you name it.

Simply add an option to load a BPF filter on a socket, used to vary the various compute_score() functions.

No hard coded knowledge in the kernel, but a generic interface.

#2 Binding to all ports

- For our TCP-proxy product we need all 65k TCP ports
- Solved with TPROXY
- https://blog.cloudflare.com/how-we-built-spectrum/

Abusing Linux's firewall: the hack that allowed us to build Spectrum

12 Apr 2018 by Marek Majkowski.

Share Like 198 Tweet

Today we are introducing Spectrum: a new Cloudflare feature that brings DDoS protection, load balancing, and content acceleration to any TCP-based protocol.

CLOUDFLARE

TPROXY to save the world

	Single IP	**Many subnets**
Single Port	bind()	SO_BINDTOPREFIX*
Many Ports	TPROXY	TPROXY

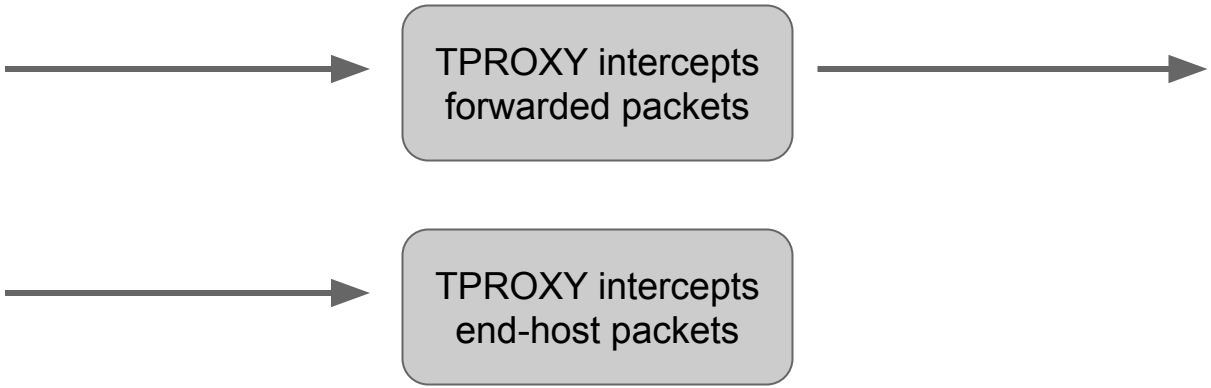
CLOUDFLARE

The hack spreads

- Replace SO_BINDTOPREFIX with TPROXY?
- mmproxy hack
  - https://blog.cloudflare.com/mmproxy-creative-way-of-preserving-client-ips-in-spectrum/
- tun/tap L3/L7 hack

TPROXY gotchas - not designed for this

- doing socket dispatch in firewall is insane



CLOUDFLARE

TPROXY gotchas - iptables

- hard to reason about

```
-t mangle -A PREROUTING -p tcp -m set --match-set paset/v4/h:n dst \
    -j TPROXY --on-port 2345 --on-ip 127.0.0.1 --tproxy-mark 0x1


-t mangle -A PREROUTING -p udp -m set --match-set paset/v4/h:n dst -m socket \
    -j MARK --set-xmark 0x1
-t mangle -A PREROUTING -p udp -m set --match-set paset/v4/h:n dst -m mark --mark 0x0 \
    -j TPROXY --on-port 2345 --on-ip 127.0.0.1 --tproxy-mark 0x1
```

TPROXY gotchas - IP_TRANSPARENT

IP_TRANSPARENT requires CAP_NET_ADMIN

(seccomp-bpf guarding socket()!)

Problem for UDP

TPROXY gotchas - reverse routing

```
$ ping 172.65.128.8
PING 172.65.128.8 (172.65.128.8) 56(84) bytes of data.
64 bytes from 172.65.128.8: icmp_seq=1 ttl=64 time=0.047 ms

$ nc -v 172.65.128.8 80
nc: connect to 172.65.128.8 port 80 (tcp) failed: Connection timed out

$ ip route get 172.65.128.8
local 172.65.128.8 dev lo table local src 172.65.128.0
    cache <local>
```

CLOUDFLARE

TPROXY gotchas - XDP sk_lookup can't find sk

In XDP we need to find sk (local socket?)

sk_lookup works fine for established, but gets confused on syn cookies

sk_lookup doesn't see TPROXY iptables!

https://www.mail-archive.com/netdev@vger.kernel.org/msg297742.html

http://vger.kernel.org/bpfconf2019.html#session-7

ACK on syn cookies is interesting

tcp_synq_no_recent_overflow() -> socket

ipv4.sysctl_tcp_syncookies -> namespace

TPROXY gotchas - lock contention

```
>  - TPROXY takes a reference to the listening socket on dispatch, which
>    raises lock contention concerns.

FWIW this could be avoided in similar way as to how we handle noref dsts.

The only reason we need to take the reference at the moment is because
once skb leaves the TPROXY target hook, the skb could leave rcu
protection as well at some point (nfqueue for example).

Maybe its even enough to move reference taking to nfqueue and add
'noref' destructor, that would allow skb_steal_sock to propagate
refcounted value in __inet_lookup_skb.

So, at least for this part I don't see a technical reason why this
has to grab a reference for listener socket.
```

BPF programmable listen socket lookup to the rescue

[RFC bpf-next 0/7] Programming socket lookup with BPF

[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

- *Subject*: [RFC bpf-next 0/7] Programming socket lookup with BPF
- *From*: Jakub Sitnicki <jakub@xxxxxxxxxxxxxx>
- *Date*: Tue, 18 Jun 2019 15:00:43 +0200
- *Cc*: kernel-team@xxxxxxxxxxxxxx

```
We have been exploring an idea of making listening socket
lookup (inet_lookup) programmable with BPF.

Why? At last Netdev Marek talked [1] about two limitations of bind() API
we're hitting when running services on our edge servers:
```

CLOUDFLARE

__inet_lookup()

1. __inet_lookup_established - (srcip, srcport, dstip, dstport)
2. __inet_lookup_listener - (dstip, dstport)
3. __inet_lookup_listener - (INADDR_ANY, dstport)

1. __inet_lookup_established - (srcip, srcport, dstip, dstport)
2. **(dstip2, dstport2) = inet_lookup_run_bpf()**
3. __inet_lookup_listener - (**dstip2, dstport2**)
4. __inet_lookup_listener - (INADDR_ANY, **dstport2**)

CLOUDFLARE

```
+++ b/net/ipv4/inet_hashtables.c
@@ -300,24 +300,27 @@ struct sock *__inet_lookup_listener(struct net *net,
                                    const int dif, const int sdif)
 {
        struct inet_listen_hashbucket *ilb2;
+       unsigned short hnum2 = hnum;
        struct sock *result = NULL;
+       __be32 daddr2 = daddr;
        unsigned int hash2;

-       hash2 = ipv4_portaddr_hash(net, daddr, hnum);
+       inet_lookup_run_bpf(net, saddr, sport, &daddr2, &hnum2);
+       hash2 = ipv4_portaddr_hash(net, daddr2, hnum2);
        ilb2 = inet_lhash2_bucket(hashinfo, hash2);

        result = inet_lhash2_lookup(net, ilb2, skb, doff,
-                                   saddr, sport, daddr, hnum,
+                                   saddr, sport, daddr2, hnum2,
                                    dif, sdif);
        if (result)
                goto done;

        /* Lookup lhash2 with INADDR_ANY */
-       hash2 = ipv4_portaddr_hash(net, htonl(INADDR_ANY), hnum);
+       hash2 = ipv4_portaddr_hash(net, htonl(INADDR_ANY), hnum2);
        ilb2 = inet_lhash2_bucket(hashinfo, hash2);

        result = inet_lhash2_lookup(net, ilb2, skb, doff,
```

New BPF hook

Attach point BPF_INET_LOOKUP; Per network-namespace; lacking skb

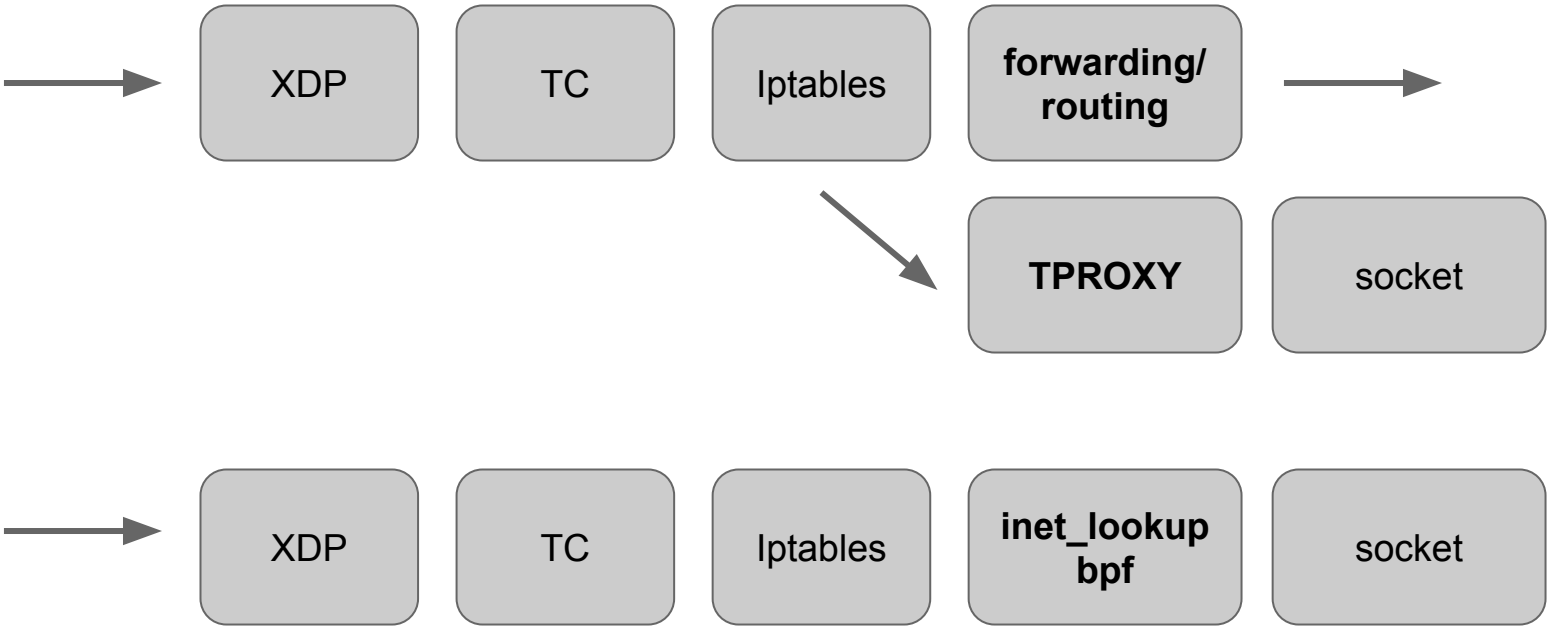
```
+struct bpf_inet_lookup {
+       __u32 family;
+       __u32 remote_ip4;       /* Allows 1,2,4-byte read but no write.
+                                * Stored in network byte order.
+                                */
+       __u32 local_ip4;        /* Allows 1,2,4-byte read and 4-byte write.
+                                * Stored in network byte order.
+                                */
+       __u32 remote_ip6[4];    /* Allows 1,2,4-byte read but no write.
+                                * Stored in network byte order.
+                                */
+       __u32 local_ip6[4];     /* Allows 1,2,4-byte read and 4-byte write.
+                                * Stored in network byte order.
+                                */
+       __u32 remote_port;      /* Allows 4-byte read but no write.
```

CLOUDFLARE

Open questions

- UDP is not symmetric with TCP at the moment
- Performance hit, especially for UDP?
- More fields - MARK (for Cilium)

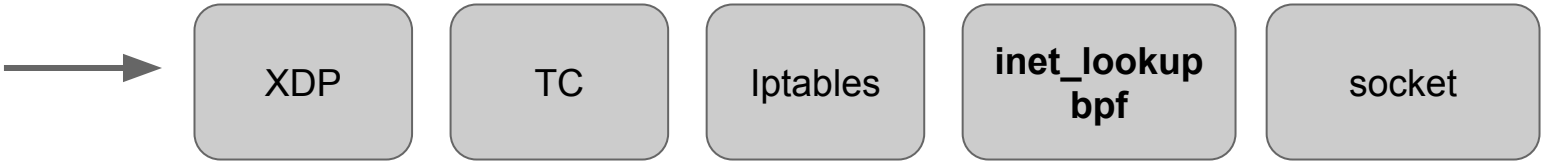
Why not sk_assign()? - two use cases



CLOUDFLARE

Why not sk_assign()?

- Fault domain, application conf - not routing feature



XDPd
* L4Drop
* L4LB

__inet_lookup() ordering

* security model (untrusted user binding)
* upgrade path hard (remove 0.0.0.0:443 bind)

1. __inet_lookup_established - (srcip, srcport, dstip, dstport)
2. __inet_lookup_listener - (dstip, dstport)
3. __inet_lookup_listener - (INADDR_ANY, dstport)
4. **(dstip2, dstport2) = inet_lookup_run_bpf()**
5. __inet_lookup_listener - (**dstip2, dstport2**)

CLOUDFLARE