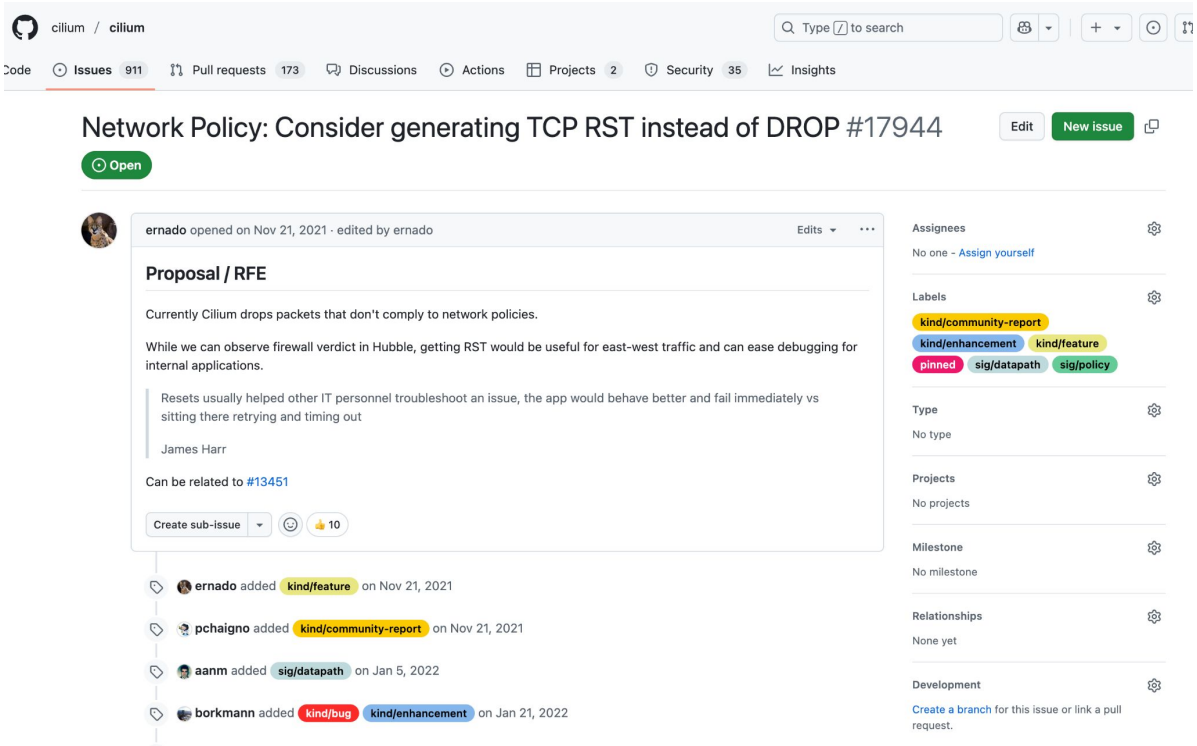
kfunc for TCP reset

John Fastabend, Mahé Tardy

Use case

Cilium and Tetragon could use the ability to TCP RST instead of just drop for more awareness to the source of the traffic.

Tetragon main use case is to send RST to Pods inside the cluster sending data out.



https://github.com/cilium/cilium/issues/17944

Consider generating TCP RST / ICMP destination unreachable in response to stale IP #13451

Edit New issue

Closed #18505

joestringer opened on Oct 7, 2020

Currently, when a source application caches a destination IP to connect to and the destination endpoint is removed, if the application continues to send traffic, it gets dropped with the reason "Stale or unroutable IP". The source application never discovers that it is connecting to the wrong IP, because the traffic is dropped and no response is sent.

It may be more graceful for the source application if we were to instead reply with either TCP RST or an ICMP type 3 code 1 ("Host Unreachable") message.

Create sub-issue 7 1

joestringer added kind/feature on Oct 7, 2020

stale on Dec 11, 2020 · Hidden as outdated

stale added stale on Dec 11, 2020

pchaigno added pinned and removed stale on Dec 11, 2020

pchaigno closed this as completed on Feb 13, 2021

Assignees

lbernail

Labels

kind/feature pinned sig/datapath

Type

No type

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

Add unreachable route for pod IP on deletion

16 ■■■■■ pkg/datapath/linux/routing/routing.go

```
@@ -209,6 +209,22 @@ func Delete(ip net.IP, compat bool) error {
209 209                 scopedLog.WithField(logfields.Rule, egress).Debug("Deleted egress rule")
210 210         }
211 211
212 +       if option.Config.EnableUnreachableRoutes {
213 +               // Replace route to old IP with an unreachable route. This will
214 +               //   - trigger ICMP error messages for clients attempting to connect to the stale IP
215 +               //   - avoid hitting rp_filter and getting Martian packet warning
216 +               // When the IP is reused, the unreachable route will be replaced to target the new pod veth
217 +               // In CRD-based IPAM, when an IP is unassigned from the CiliumNode, we delete this route
218 +               // to avoid blackholing traffic to this IP if it gets reassigned to another node
219 +               if err := netlink.RouteReplace(&netlink.Route{
220 +                       Dst:   &ipWithMask,
221 +                       Table: route.MainTable,
222 +                       Type:  unix.RTN_UNREACHABLE,
223 +               }); err != nil {
224 +                       return fmt.Errorf("unable to add unreachable route for ip %s: %w", ipWithMask.String(), err)
225 +               }
226 +       }
227 +
228         return nil
229 }
230
```

Netfilter reject target with `tcp-reset` option

11.14. REJECT target

The **REJECT** target works basically the same as the **DROP** target, but it also sends back an error message to the host sending the packet that was blocked. The **REJECT** target is as of today only valid in the INPUT, FORWARD and OUTPUT chains or their sub chains. After all, these would be the only chains in which it would make any sense to put this target. Note that all chains that use the **REJECT** target may only be called by the INPUT, FORWARD, and OUTPUT chains, else they won't work. There is currently only one option which controls the nature of how this target works, though this may in turn take a huge set of variables. Most of them are fairly easy to understand, if you have a basic knowledge of TCP/IP.

Table 11-10. REJECT target

Option	**--reject-with**
Example	**iptables -A FORWARD -p TCP --dport 22 -j REJECT --reject-with tcp-reset**
Explanation	This option tells the **REJECT** target what response to send to the host that sent the packet that we are rejecting. Once we get a packet that matches a rule in which we have specified this target, our host will first of all send the associated reply, and the packet will then be dropped dead, just as the **DROP** target would drop it. The following reject types are currently valid: `icmp-net-unreachable`, `icmp-host-unreachable`, `icmp-port-unreachable`, `icmp-proto-unreachable`, `icmp-net-prohibited` and `icmp-host-prohibited`. The default error message is to send a **port-unreachable** to the host. All of the above are ICMP error messages and may be set as you wish. You can find further information on their various purposes in the appendix *ICMP types*. Finally, there is one more option called **tcp-reset**, which may only be used together with the TCP protocol. The **tcp-reset** option will tell **REJECT** to send a TCP RST packet in reply to the sending host. TCP RST packets are used to close open TCP connections gracefully. For more information about the TCP RST read *RFC 793 - Transmission Control Protocol*. As stated in the **iptables** man page, this is mainly useful for blocking ident probes which frequently occur when sending mail to broken mail hosts, that won't otherwise accept your mail.

Netfilter reject target with `tcp-reset` option

```
30  static unsigned int
31  reject_tg(struct sk_buff *skb, const struct xt_action_param *par)
32  {
33          const struct ipt_reject_info *reject = par->targinfo;
34          int hook = xt_hooknum(par);
35
36          switch (reject->with) {
37          case IPT_ICMP_NET_UNREACHABLE:
38                  nf_send_unreach(skb, ICMP_NET_UNREACH, hook);
39                  break;
40          case IPT_ICMP_HOST_UNREACHABLE:
41                  nf_send_unreach(skb, ICMP_HOST_UNREACH, hook);
42                  break;
43          case IPT_ICMP_PROT_UNREACHABLE:
44                  nf_send_unreach(skb, ICMP_PROT_UNREACH, hook);
45                  break;
46          case IPT_ICMP_PORT_UNREACHABLE:
47                  nf_send_unreach(skb, ICMP_PORT_UNREACH, hook);
48                  break;
49          case IPT_ICMP_NET_PROHIBITED:
50                  nf_send_unreach(skb, ICMP_NET_ANO, hook);
51                  break;
52          case IPT_ICMP_HOST_PROHIBITED:
53                  nf_send_unreach(skb, ICMP_HOST_ANO, hook);
54                  break;
55          case IPT_ICMP_ADMIN_PROHIBITED:
56                  nf_send_unreach(skb, ICMP_PKT_FILTERED, hook);
57                  break;
58          case IPT_TCP_RESET:
59                  nf_send_reset(xt_net(par), par->state->sk, skb, hook);
60                  break;
61          case IPT_ICMP_ECHOREPLY:
62                  /* Doesn't happen. */
63                  break;
64          }
65
66          return NF_DROP;
67  }
```

Reuse bpf_sock_destroy?

- Aditi Ghag added the **bpf_sock_destroy** kfunc for socket load balancing and enforcing policies on existing connection use cases.
- This kfunc can only be used in iterators.
- This function can only be called from BPF contexts that have acquired the socket lock.

Could this be extended to support cgroup_skb prog types (and more)?

```
From: Aditi Ghag <aditi.ghag@isovalent.com>
To: bpf@vger.kernel.org
Cc: kafai@fb.com, sdf@google.com, aditi.ghag@isovalent.com
Subject: [PATCH v9 bpf-next 0/9] bpf: Add socket destroy capability
Date: Fri, 19 May 2023 22:51:48 +0000      [thread overview]
Message-ID: <20230519225157.760788-1-aditi.ghag@isovalent.com> (raw)

This patch set adds the capability to destroy sockets in BPF. We plan to
use the capability in Cilium to force client sockets to reconnect when
their remote load-balancing backends are deleted. The other use case is
on-the-fly policy enforcement where existing socket connections
prevented by policies need to be terminated.

The use cases, and more details around
the selected approach were presented at LPC 2022 -
https://lpc.events/event/16/contributions/1358/.
RFC discussion -
https://lore.kernel.org/netdev/CABG=zsBEh-P4NXk23eBJw7eajB5YJeRS7oPXnTAzs=yob4E
MoQ@mail.gmail.com/T/#u.
v8 patch series -
https://lore.kernel.org/bpf/20230517175359.527917-1-aditi.ghag@isovalent.com/

[...]

Aditi Ghag (9):
  bpf: tcp: Avoid taking fast sock lock in iterator
  udp: seq_file: Helper function to match socket attributes
  bpf: udp: Encapsulate logic to get udp table
  udp: seq_file: Remove bpf_seq_afinfo from udp_iter_state
  bpf: udp: Implement batching for sockets iterator
  bpf: Add kfunc filter function to 'struct btf_kfunc_id_set'
  bpf: Add bpf_sock_destroy kfunc
  selftests/bpf: Add helper to get port using getsockname
  selftests/bpf: Test bpf_sock_destroy
```

https://lore.kernel.org/all/20230519225157.760788-1-aditi.ghag@isovalent.com/

Discussions

- Should we support ICMP responses?
- Make sure to include limits to prevent DDoS.
- TCP RST direction for existing connections?