Linux Cross Reference

Free Electrons

Embedded Linux Experts

• *source navigation* • diff markup • identifier search • freetext search •

Version:
2.0.40 2.2.26 2.4.37 3.6 3.7 3.8 3.9 3.10 3.11 3.12 3.13 3.14 3.15 3.16 3.17 3.18 3.19 4.0 4.1 ***4.2***

Linux/net/ipv4/ip_forward.c

```
/*
 * INET        An implementation of the TCP/IP protocol suite for the LINUX
 *             operating system.  INET is implemented using the  BSD Socket
 *             interface as the means of communication with the user level.
 *
 *             The IP forwarding functionality.
 *
 * Authors:    see ip.c
 *
 * Fixes:
 *             Many            :       Split from ip.c , see ip_input.c for
 *                                     history.
 *             Dave Gregorich  :       NULL ip_rt_put fix for multicast
 *                                     routing.
 *             Jos Vos         :       Add call_out_firewall before sending,
 *                                     use output device for accounting.
 *             Jos Vos         :       Call forward firewall after routing
 *                                     (always use output device).
 *             Mike McLagan    :       Routing by source
 */

#include <linux/types.h>
#include <linux/mm.h>
#include <linux/skbuff.h>
#include <linux/ip.h>
#include <linux/icmp.h>
#include <linux/netdevice.h>
#include <linux/slab.h>
#include <net/sock.h>
#include <net/ip.h>
#include <net/tcp.h>
#include <net/udp.h>
#include <net/icmp.h>
#include <linux/tcp.h>
#include <linux/udp.h>
#include <linux/netfilter_ipv4.h>
#include <net/checksum.h>
#include <linux/route.h>
#include <net/route.h>
#include <net/xfrm.h>

static bool ip_exceeds_mtu(const struct sk_buff *skb, unsigned int mtu)
{
```

```
44          if (skb->len <= mtu)
45                  return false;
46
47          if (unlikely((ip_hdr(skb)->frag_off & htons(IP_DF)) == 0))
48                  return false;
49
50          /* original fragment exceeds mtu and DF is set */
51          if (unlikely(IPCB(skb)->frag_max_size > mtu))
52                  return true;
53
54          if (skb->ignore_df)
55                  return false;
56
57          if (skb_is_gso(skb) && skb_gso_network_seglen(skb) <= mtu)
58                  return false;
59
60          return true;
61  }
62
63
64  static int ip_forward_finish(struct sock *sk, struct sk_buff *skb)
65  {
66          struct ip_options *opt  = &(IPCB(skb)->opt);
67
68          IP_INC_STATS_BH(dev_net(skb_dst(skb)->dev), IPSTATS_MIB_OUTFORWDATAGRAMS);
69          IP_ADD_STATS_BH(dev_net(skb_dst(skb)->dev), IPSTATS_MIB_OUTOCTETS, skb->len);
70
71          if (unlikely(opt->optlen))
72                  ip_forward_options(skb);
73
74          skb_sender_cpu_clear(skb);
75          return dst_output_sk(sk, skb);
76  }
77
78  int ip_forward(struct sk_buff *skb)
79  {
80          u32 mtu;
81          struct iphdr *iph;      /* Our header */
82          struct rtable *rt;      /* Route we use */
83          struct ip_options *opt  = &(IPCB(skb)->opt);
84
85          /* that should never happen */
86          if (skb->pkt_type != PACKET_HOST)
87                  goto drop;
88
89          if (unlikely(skb->sk))
90                  goto drop;
91
92          if (skb_warn_if_lro(skb))
93                  goto drop;
94
95          if (!xfrm4_policy_check(NULL, XFRM_POLICY_FWD, skb))
96                  goto drop;
97
98          if (IPCB(skb)->opt.router_alert && ip_call_ra_chain(skb))
99                  return NET_RX_SUCCESS;
100
101         skb_forward_csum(skb);
102
103         /*
104          *      According to the RFC, we must first decrease the TTL field. If
105          *      that reaches zero, we must reply an ICMP control message telling
106          *      that the packet's lifetime expired.
107          */
108         if (ip_hdr(skb)->ttl <= 1)
```

```
                goto too_many_hops;

        if (!xfrm4_route_forward(skb))
                goto drop;

        rt = skb_rtable(skb);

        if (opt->is_strictroute && rt->rt_uses_gateway)
                goto sr_failed;

        IPCB(skb)->flags |= IPSKB_FORWARDED;
        mtu = ip_dst_mtu_maybe_forward(&rt->dst, true);
        if (ip_exceeds_mtu(skb, mtu)) {
                IP_INC_STATS(dev_net(rt->dst.dev), IPSTATS_MIB_FRAGFAILS);
                icmp_send(skb, ICMP_DEST_UNREACH, ICMP_FRAG_NEEDED,
                          htonl(mtu));
                goto drop;
        }

        /* We are about to mangle packet. Copy it! */
        if (skb_cow(skb, LL_RESERVED_SPACE(rt->dst.dev)+rt->dst.header_len))
                goto drop;
        iph = ip_hdr(skb);

        /* Decrease ttl after skb cow done */
        ip_decrease_ttl(iph);

        /*
         *      We now generate an ICMP HOST REDIRECT giving the route
         *      we calculated.
         */
        if (IPCB(skb)->flags & IPSKB_DOREDIRECT && !opt->srr &&
            !skb_sec_path(skb))
                ip_rt_send_redirect(skb);

        skb->priority = rt_tos2priority(iph->tos);

        return NF_HOOK(NFPROTO_IPV4, NF_INET_FORWARD, NULL, skb,
                       skb->dev, rt->dst.dev, ip_forward_finish);

sr_failed:
        /*
         *      Strict routing permits no gatewaying
         */
         icmp_send(skb, ICMP_DEST_UNREACH, ICMP_SR_FAILED, 0);
         goto drop;

too_many_hops:
        /* Tell the sender its packet died... */
        IP_INC_STATS_BH(dev_net(skb_dst(skb)->dev), IPSTATS_MIB_INHDRERRORS);
        icmp_send(skb, ICMP_TIME_EXCEEDED, ICMP_EXC_TTL, 0);
drop:
        kfree_skb(skb);
        return NET_RX_DROP;
}

```

- Docs
- Community
- Company
- Blog