第5回

トンネリング・ドライバのカーネル内部処理

赤松 徹

第1回目で6to4を利用してIPv6ネットワーク接続の実験をしたときは，設定手順を説明しただけでカーネルの内部処理については触れなかった．そこで，今回は「IPv6 over IPv4 tunneling driver」をカーネル内部で処理するsit.cに焦点を当てて解説する．（筆者）

6to4パケット送受信のネットワーク接続チェック

6to4パケットは，IPv6パケットの先頭にIPv4ヘッダを付加して発信し，既存のIPv4ネットワークを経由してリレー・ルータに届きます（IPv4ヘッダの終点アドレス）．そこで，IPv4ヘッダを削除してIPv6ネットワークへ転送するので，最終的にIPv6サーバ（IPv6ヘッダの終点アドレス）に届きます．

非常に残念ですが，KDDIが6to4リレー・ルータのサービスを停止したため，今回の実験では::192.88.99.1を利用しました．次の3組のコマンドで60.56.229.25（3c38:e519）からIPv6ネットワークに接続できます．ping6コマンドで接続チェックをしました（**図1**）．

```
# ifconfig sit0 up
# ifconfig sit0 add 2002:3c38:e519::1/16
# route -A inet6 add ::/1 dev sit0
                    gw ::192.88.99.1
```

図1で送受信したping6のパケットをQUEUEターゲットから取得しました．このとき，送受信するパケットはIPv4なので，ip6tablesでなくiptablesで処理します．QUEUEターゲットでユーザ領域に取得するプログラムを起動していなければ，パケットを送受信できなくなり，IPv4ネットワークが利用できなくなります．また全部のIPv4パケットを取得するとログのサイズが大きくなるので，-p ipv6（/etc/protocolsで41に定義してある）として，6to4パケットだけに限定してQUEUEに取り出します．

```
# iptables -I INPUT -j QUEUE -p ipv6
# iptables -I OUTPUT -j QUEUE -p ipv6
```

6to4パケットの中身

取得した6to4パケットを理解しやすいように三つの部分に分けて解説します．まず，**図2**（**a**）で示すstd05からwww.kame.net宛てのパケットに注目してください．

① IPv4ヘッダ部分のキー・ポイントは，始点アドレスがstd05のIPv4アドレスで，終点アドレスはリレー・ルータのIPv4アドレスであることです．もう一つ重要なポイントはIPプロトコルの値は41で，この値は4層で処理する情報を示すもので，UDPなら17，TCPなら6であることです．

② IPv6ヘッダ部分の始点アドレスはstd05のIPv6アドレスで，終点アドレスはwww.kame.netのIPv6アドレスです．次のhostコマンドでwww.kame.netのIPv6アドレスを確認しました．

```
# host -t AAAA www.kame.net
www.kame.net has AAAA address
        2001:200:0:8002:203:47ff:fea5:3085
```

もう一つ重要なポイントは，IPv4のパケット全長124オクテットにはIPv4ヘッダ長も含みますが，IPv6ヘッダのペイロード長はIPv6ヘッダ長を含まないことです．

③ ペイロード部分はIPv6ヘッダの次のヘッダ情報からICMPV6_ECHO_REQUESTであることがわかります．

引き続き，**図2**（**b**）で示すwww.kame.netからstd05に返信されたパケットに注目してください．この重要ポイントは，www.kame.netが返信したのはIPv6ヘッダ以降のIPv6パケットだけですが，6to4リレー・ルータがIPv6ヘッダの終点アド

```
[root@std05 linux]# ping6 -c 1 www.kame.net
PING www.kame.net(orange.kame.net) 56 data bytes
64 bytes from orange.kame.net: icmp_seq=0 ttl=52 time=462 ms

--- www.kame.net ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 462.288/462.288/462.288/0.000 ms, pipe 2
```

図1　ping6コマンドでIPv6ネットワーク接続チェックを実行した結果

KEYWORD ── トンネリング，6to4パケット送受信，6to4リレー・ルータ，ping6，トンネル・パケット

```
4500007C  IPv4，ヘッダ長5<<2=20オクテッド，
                全長0x7C=124オクテッド
00004000
4029F5AD  TTL=0x40，IPプロトコル：0x29=41(IPPROTO_IPV6)，
3C38E519  始点アドレス：60.56.229.25
(std05.kic.ac.jpのIPv4アドレス)
C0586301  終点アドレス：192.88.99.1
(トンネリング・ルータのIPv4アドレス)

60000000  IPv6ヘッダ
00403A40  ペイロード長：0x40=64オクテッド，
              次ヘッダ：0x3A=58：ICMPv6，hoplimit=0x40
20023C38E51900000000000000000001
              始点アドレス：std05.kic.ac.jpのIPv6アドレス
2001020000008002020347FFFEA53085
              終点アドレス：www.kame.netのIPv6アドレス
[ペイロード]
800056F4  タイプ：0x80=128，
          コード：0x00：ICMPV6_ECHO_REQUEST
43490000  識別子：0x4349，シーケンス番号：0x0000
43E676440000000001C40E0000000000
101112131415161718191A1B1C1D1E1F
202122232425262728292A2B2C2D2E2F
3031323334353637
```

(**a**) std05から発信した6to4パケット

```
4500007C  IPv4，ヘッダ長5<<2=20オクテッド，
                全長=0x7C=124オクテッド
C42B0000
EB29C681  TTL=0xEB，IPプロトコル：0x29=41(IPPROTO_IPV6)
C0586301  始点アドレス：192.88.99.1
                        (トンネリング・ルータのIPv4アドレス)
3C38E519  終点アドレス：60.56.229.25
                       (std05.kic.ac.jpのIPv4アドレス)

60000000  IPv6ヘッダ
00403A34  ペイロード長：0x40=64オクテッド，
              次ヘッダ：0x3A=58：ICMPv6，hoplimit=0x34
2001020000008002020347FFFEA53085
              始点アドレス：www.kame.netのIPv6アドレス
20023C38E51900000000000000000001
              終点アドレス：std05.kic.ac.jpのIPv6アドレス
[ペイロード]
810055F4  タイプ：0x81=124，
          コード：0x00：ICMPV6_ECHO_REPLAY
43490000  識別子：0x4349，シーケンス番号：0x0000
43E676440000000001C40E0000000000
101112131415161718191A1B1C1D1E1F
202122232425262728292A2B2C2D2E2F
3031323334353637
```

(**b**) www.kame.netから返信された6to4パケット

図2　6to4パケットの詳解

リスト1　ipip6_rcv()関数

```
367  static int ipip6_rcv(struct sk_buff *skb)
368  {
369      struct iphdr *iph;
370      struct ip_tunnel *tunnel;
371
372      if (!pskb_may_pull(skb, sizeof(struct ipv6hdr)))
373              goto out;
374
375      iph = skb->nh.iph;
376
377      read_lock(&ipip6_lock);
378      if ((tunnel = ipip6_tunnel_lookup(iph->saddr, iph->daddr)) != NULL) {
379              secpath_reset(skb);
380              skb->mac.raw = skb->nh.raw;
381              skb->nh.raw = skb->data;
382              memset(&(IPCB(skb)->opt), 0, sizeof(struct ip_options));
383              skb->protocol = htons(ETH_P_IPV6);
384              skb->pkt_type = PACKET_HOST;
385              tunnel->stat.rx_packets++;
386              tunnel->stat.rx_bytes += skb->len;
387              skb->dev = tunnel->dev;
388              dst_release(skb->dst);
389              skb->dst = NULL;
390              nf_reset(skb);
391              ipip6_ecn_decapsulate(iph, skb);
392              netif_rx(skb);
393              read_unlock(&ipip6_lock);
394              return 0;
395      }
396
397      icmp_send(skb, ICMP_DEST_UNREACH,
                              ICMP_PROT_UNREACH, 0);
398      kfree_skb(skb);
399      read_unlock(&ipip6_lock);
400  out:
401      return 0;
402  }
```

リスト2　PACKET_HOST以外のパケット・タイプ

```
 22  /* Packet types */
 23
 24  #define PACKET_HOST        0  /* To us*/
 25  #define PACKET_BROADCAST   1  /* To all*/
 26  #define PACKET_MULTICAST   2  /* To group*/
 27  #define PACKET_OTHERHOST   3  /* To someone else*/
 28  #define PACKET_OUTGOING    4  /* Outgoing of any type*/
 29  /* These ones are invisible by user level */
 30  #define PACKET_LOOPBACK    5  /* MC/BRD frame looped back
*/
```

レスに含まれる(3c38:e519)からIPv4アドレスを取得してIPv4ヘッダに付加する点です．

6to4パケット処理はGREトンネリング技術の応用

6to4処理は，カーネル・ソースのnet/ipv6/sit.cファイルで記述しています．その最初の部分に「6to4はGREトンネリング技術を導入したもの」と書かれています(1)．

今まで解説してきた流れから，受信処理を先に読んでいきます〔**図2**(**b**)〕．IPv4ヘッダのプロトコル情報が41の場合，4層受信処理関数はipip6_rcv()関数へ分岐してきます(**リスト1**)．

未処理部分のデータ長がipv6hdr長より短かければ壊れたパケットなので，ラベルoutへ分岐します．受信したのはIPv4パケットなので，375行目でiphに3層のIPv4ヘッダ(skb->nh.iph)の先頭位置を設定します．

378行目のipip6_tunnel_lookup()関数で自分が発信した6to4パケットかどうかを確認します．自分が発信していない場合は，397行目のicmp_send()関数でICMPパケットを返信します．

自分が発信した返信パケットの場合は，379行目から394行目の処理を実行します．ここで，IPv4ヘッダを削除してIPv6