Linux Kernel 2.6 の IPv6 プロトコル・スタック詳解

第4回

xinetdのdaytime-udpを利用するudpv6_rcv関数の処理

赤松 徹

受信した自局宛てのIPv6パケットを第3層から第4層に処理を依頼するとき，tcpの場合はtcp_v6_rcv関数，udpの場合はudpv6_rcv関数，icmpの場合はicmpv6_rcv関数へ分岐することを第2回目で解説した．前回はicmpv6_rcv関数を紹介したので，今回はudpv6_rcv関数に焦点を当てて解説する．（筆者）

daytime-udp設定ファイルの修正

簡単なIPv6のUDPクライアント・プログラム（**リスト1**）を作成して，xinetd内に含まれるdaytime-udp（13番ポート）へサービス要求パケットを発信しました．そのパケットをカーネル内部のudpv6_rcv()関数がどのように処理するかを確認します．最初にxinetdがIPv6パケットを処理できるように，/etc/xinetd.d/daytime-udp設定ファイルを修正します．修正点は次の2か所です．

① disable = no　　動作を有効にする
② flags = IPv6　　IPv6パケットを処理する

リスト2に示す修正を行った後，xinetdをリスタートします．IPv6のUDPソケットが動作していることを，/proc/net/udp6ファイルの内容を表示して確認します（**図1**）．

```
# /etc/rc.d/init.d/xinetd restart
Stopping xinetd:     [  OK  ]
Starting xinetd:     [  OK  ]
```

リスト1　IPv6のUDPクライアント・プログラムIPv6_UDP.c

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>

int main(){
  struct sockaddr_in6 to_addr;
  socklen_t slen;
  struct hostent *to;
  char mess_from[1500], mess[]="UDP_daytime IPv6";
  int so, con, ies, len=sizeof(mess);

  so=socket(AF_INET6, SOCK_DGRAM, 0);
  if(so<0){
     printf("Cannot open socket¥n");
     exit(1);
  }
  bzero((char *)&to_addr, sizeof(to_addr));
  to_addr.sin6_family=AF_INET6;
  to_addr.sin6_port=htons(13);
  to_addr.sin6_addr=in6addr_loopback;
  slen=sizeof(to_addr);
  if(sendto(so, mess, sizeof(mess), 0, (struct sockaddr *)&to_addr, slen)<0){
     printf("Error sendto %s daytime_server¥n", to->h_name );
     close(so);
     exit(1);
  }
  recvfrom(so, mess_from, 1500, 0, (struct sockaddr *)&to_addr, &slen);
  ies=strlen(mess_from)-5;
  mess_from[ies]='¥0';
  printf("Message:%s¥n", mess_from);
  system("date ¥"+%Y %m %d %H:%M:%S¥"");
  close(so);
}
```

wrapperによるアクセス制限

xinetdをconfigureしたときにwrapperライブラリを組み込んだかを，次のlddコマンドで確認します．

```
# ldd /usr/sbin/xinetd | grep wrap
libwrap.so.0 => /usr/lib64/libwrap.so.0
                                (0x00002aaaaaabd000)
```

libwrapライブラリの表示があれば組み込まれています．その場合，動作確認できるようにアクセス制限の設定を解除する必要があります．/etc/hosts.denyですべてのアクセスを拒

リスト2　/etc/xinetd.d/daytime-udp設定ファイルの内容

```
service daytime
{
   disable       = no
   flags         = IPv6
   type          = INTERNAL UNLISTED
   id            = daytime-dgram
   socket_type   = dgram
   protocol      = udp
   user          = root
   wait          = yes
   port          = 13
}
```

```
  sl  local_address remote_address st tx_queue rx_queue tr tm->when retrnsmt   uid  timeout inode
  13: 00000000:000D 000000000:0000 07 00000000:00000000 00:00000000 00000000     0        0 4665 2 ffff81007975fc00
```

図1　/proc/net/udp6ファイルの内容

KEYWORD——UDPクライアント・プログラム，wrapperライブラリ，疑似ヘッダ，RFC2460

```
・サービス要求パケット
T.Akamatsu ipq_packet_msg_t m->data_len=65
60000000
00191140  ペイロード長：0x019=25 オクテット，次ヘッダ：0x11=17(UDP)、HOP_LIMIT：0x40
00000000000000000000000000000001  始点アドレス  ::1
00000000000000000000000000000001  終点アドレス  ::1
[ペイロード]
8003000D  始点ポート：0x8003，終点ポート：0x0D=13(daytime)
00196E1E  長さ：0x19=25 オクテット，チェックサム =0x6E1E
5544505F64617974696D652049507636  送信メッセージ：UDP_daytime IPv6
00


・時刻返信パケット
T.Akamatsu ipq_packet_msg_t m->data_len=74
60000000
00221140  ペイロード長：0x022=34 オクテット，次ヘッダ：0x11=17(UDP)，HOP_LIMIT：0x40
00000000000000000000000000000001  始点アドレス  ::1
00000000000000000000000000000001  終点アドレス  ::1
[ペイロード]
000D8003  始点ポート：0x0D=13(daytime)，終点ポート：0x8003
002216BB  長さ：0x22=34 オクテット，チェックサム =0x61BB
3038204D4159203233030362031353A33  返信メッセージ：08 MAY 2006 15:3
313A3331204A53540D0A               1:31 JST
```

図2
daytime-udp パケット情報

否しているならば，クライアント・プログラムを実行したときに，/var/log/messages ログに拒否したメッセージが記録されます．

```
xinetd[1713]: libwrap refused connection to
    daytime-dgram (libwrap=daytime) from ::1
```

そこで，/etc/hosts.allow ファイルに::1 からの daytime 接続要求のパケットを許可する行を追加します．IPv6 アドレスの前後を[]で囲むことを忘れてはいけません．これ以外にも ip6tables などでパケット・フィルタリングを設定している場合は，適切に許可して実験できる環境にします．

```
daytime:        [::1]/128
```

実行の確認

IPv6 アドレス::1 の 13 番ポート(daytime)へ UDP パケットで接続要求するプログラム(**リスト1**)を確認します．ここで，IPv4 と IPv6 の両方とも処理するプログラムにしなかったのは，IPv6 パケットのみ発信して確認したかったからです．

また，IPv6 のローカル・ホスト(::1)からパケットを送受信するので，発信パケットに「UDP_daytime IPv6」のメッセージを付加することにしました．コンパイルして実行すると，次に示すように，受信したパケット内容と現在の時刻情報を表示します．

```
# /SOFT/MyProgram/UDP/a.out
Message:08 MAY 2006 15:31:31 JST
2006 05 08 15:31:31
```

アプリケーション層の表示結果だけでは，カーネル内部の処理を確実に読み取ることはできません．そこで ip6tables の QUEUE ターゲットを使用して，IPv6 パケットをユーザ領域に

```
struct udphdr {
        __u16   source;
        __u16   dest;
        __u16   len;
        __u16   check;
};
```

source	dest
len	check

図3　UDP ヘッダ定義とヘッダ構造図

取得しました．**図2**に示す daytime-udp パケット情報の重要ポイントは次の 7 点です．

① ペイロード長：IPv6 ヘッダ長を含まない
② 次ヘッダ：17 はペイロード上には UDP 情報を運んでいる
③ HOP_LIMIT ：0x40 の値は/proc/sys/net/ipv6/conf/default/hop_limit に定義してある
④ UDP ヘッダの始点ポート：サービスを要求する側は未使用の UDP ポートを探して利用する
⑤ UDP ヘッダの終点ポート：サービスする側は 13 番ポート(daytime)
⑥ UDP ヘッダ上の len ：(UDP ヘッダ＋メッセージ)長
⑦ UDP ヘッダの後に第 7 層に渡すメッセージが続く

次ヘッダが UDP なので，ペイロード上の先頭には include/linux/udp.h ファイルで定義した UDP ヘッダ(**図3**)が設定されています．UDP ヘッダ定義を構造図に示すと，パケット情報との関連が理解しやすくなります．

udpv6_rcv 関数の処理内容

それでは net/ipv6/udp.c ファイルの 450 行目以降に書かれている udpv6_rcv()関数(**リスト3**)の処理内容を確認しましょう．

459 行目の pskb_may_pull()関数は，skb に保管している未処理データ長が UDP ヘッダ以上の長さがあるかを確認する