Perlでネットワーク管理

正規表現を利用した設定管理
バイナリデータの読み解き方

台場 圭一

私が何者かというと…

- ちょっと前まではキャリアの研究所でのんびりと
- 今は小さなISPのエンジニア
- Perlを使い始めたのはLAN上のマシン群を制御するため
- 私にとってのPerlはネットワーク用万能ナイフ

最近の持ちネタ

- ネットワーク機器の設定管理をなんとかしないと
- パケット解析までしないと原因がわからない
  トラブル頻発
- プロトコル種別ごとにトラヒック量を測定する
  必要性増大

機器設定(簡単編)

- 例えば某スイッチのアクセス制御は,
  - create access-list deny102_32 udp dest 10.2.0.0/16 ip-port 32 source any ip-port any deny ports any precedence 10
  - create access-list deny102_23 tcp dest 10.2.0.0/16 ip-port 23 source any ip-port any deny ports any precedence 20
  - create access-list allow10_23 tcp dest 10.0.0.0/8 ip-port 23 source any ip-port any permit qosprofile qp4 ports any precedence 30
  - create access-list allow102 ip dest 10.2.0.0/16 source 0.0.0.0/0 permit qosprofile qp3 ports any precedence 40
- これがやろうとしているのは
  - Deny UDP port 32 and TCP port 23 traffic to the 10.2.XX network.
  - All other TCP port 23 traffic destined for other 10.X.X.X networks is permitted using QoS profile Qp4.
  - All remaining traffic to 10.2.0.0 uses QoS profile Qp3.

設定データを抽出するには

- 定義を使ってそのまま正規表現を作る
  - create access-list <name> ip
    destination [<dest_ipaddress>/<mask> | any]
    source [<src_ipaddress>/<src_mask> | any] [permit {<qosprofile>} | deny]
    ports [<portlist> | any] {precedence <prec_number>}
- が，あんまり使ってもらえない．
  もう少しひねらないと
- (? { code }) 万能な正規表現: 大崎さん曰く
  - Perl 正規表現雑技(http://www.din.or.jp/~ohzaki/regex.html)

ACL抽出コード(1/2)

```
#!perl
#       acl extracter
#
use re 'eval';
use strict;

my $digit       = q{[0-9]};
my $upalpha     = q{[A-Z]};
my $lowalpha    = q{[a-z]};
my $ubar        = q{_};
my $schar       = q{[:,]};
my $alpha       = qq{(?:$lowalpha|$upalpha)};
my $alphanum    = qq{(?:$alpha|$digit)};
my $alphanumub = qq{(?:$alphanum|$ubar)};
my $alphanumsc = qq{(?:$alphanum|$schar)};
my $HEAD        = qr{(?:create¥s+access-list)};
my $PROTO       = qq{(?:ip|icmp|udp|tcp)};
my $name        = qr{$alphanumub+};
my $ANY         = q{any};
my $DST         = q{destination};
my $SRC         = q{source};
my $net         =
qr{(?:¥d{1,3}¥.¥d{1,3}¥.¥d{1,3}¥.¥d{1,3}/¥d{1,2})};
my $target      = qq{(?:$net|$ANY)};
my $TYPE        = q{type};
my $type        = qr{$digit+};
my $CODE        = qr{code};
my $code        = qr{$digit+};
my $IPPRT       = q{ip-port};
my $ipprt       = qr{(?:any|$digit+)};
my $PRTS        = q{ports};
my $prt         = qr{(?:any|$alphanumsc+)};
my $ACT         = q{(?:permit|deny)};
my $PREC        = q{precedence};
my $prec        = qr{$digit+};
my ($file,$flag,%rule,%rules);
sub usage();
sub hcopy($$);

$file = shift || usage();
open IN, "<$file";
```

ACL抽出コード(2/E)

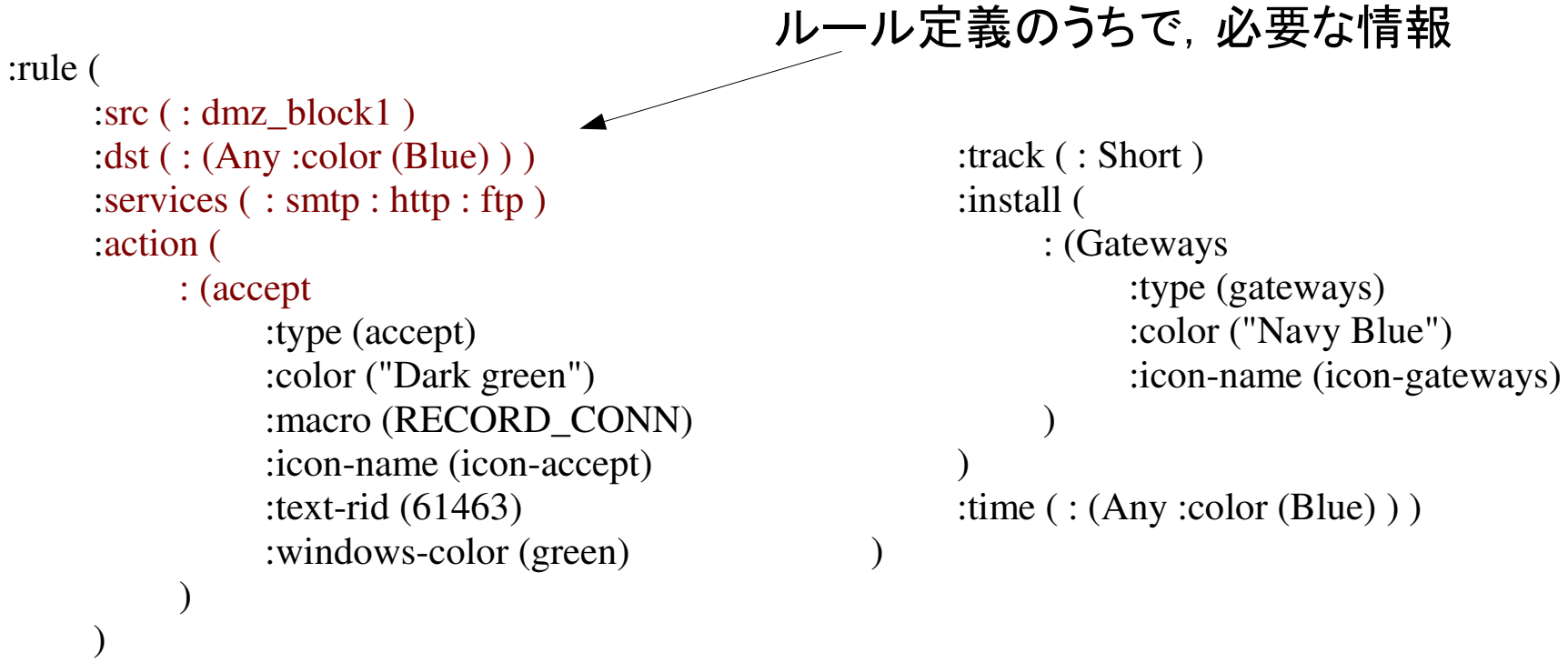
```
while (<IN>){
  %rule = ();
  $flag = m<
    $HEAD¥s*
    ($name)¥s*  (?{ $rule{NAME}  = $^N })
    ($PROTO)¥s* (?{ $rule{PROTO} = $^N })
    $DST¥s*
    ($target)¥s*(?{ $rule{DADDR} = $^N })
    (?:$IPPRT¥s*
    ($ipprt)¥s* (?{ $rule{DPTR}  = $^N }))?
    $SRC¥s*
    ($target)¥s*(?{ $rule{SADDR} = $^N })
    (?:$IPPRT¥s*
      ($ipprt)¥s* (?{ $rule{SPTR}  = $^N }))?
    (?:$TYPE¥s*
      ($type)¥s*  (?{ $rule{TYPE}  = $^N })
      $CODE¥s*
      ($code)¥s*  (?{ $rule{CODE}  = $^N }))?
    ($ACT)¥s*   (?{ $rule{ACT}   = $^N })
    (?:$PRTS¥s*
      ($prt)¥s*   (?{ $rule{PRT}   = $^N })
      $PREC¥s*
      ($prec)*    (?{ $rule{PREC}  = $^N }))?
  >x;
  if ($flag){ hcopy (¥%rule, ¥%rules) }
}
```

```
  map {printf "%d,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s¥n",
    $rules{$_}{PREC},
    $rules{$_}{NAME},
    $rules{$_}{ACT},
    $rules{$_}{PROTO},
    $rules{$_}{TYPE},
    $rules{$_}{CODE},
    $rules{$_}{SADDR},
    $rules{$_}{SPTR},
    $rules{$_}{DADDR},
    $rules{$_}{DPTR},
    $rules{$_}{PRT}}(sort {$a <=> $b} keys %rules);

sub hcopy($$)
{
  my ($fref,$tref) = @_;
  map{$$tref{$$fref{PREC}}{$_} = $$fref{$_}}(sort keys %$fref);
}

sub usage()
{
  print "USAGE: $0 FILE¥n";
  exit 1;
}
```

機器設定（複雑編）

- 某ファイアウォールの設定はGUIで行うが，ルールがテキストで保存されていた

ルール定義のうちで，必要な情報



ルールを抽出するには

- 入れ子を許した括弧内にマッチさせればよい
- これも正規表現雑技にネタがある

```
use strict;
my ($openclose,$str);
$openclose = qr/\([^()]*(?:(??{$openclose})[^()]*)*\)/;
while (<DATA>){
	/^([^#]+)(?{ $str = $^N })/;
	while ($str =~ /($openclose)/g) {
		print $1, "\n";
	}
}
__END__
(a)		# (a)
((a)		# (a)
(((a)(b))(d))	# (((a)(b))(d))
```

ルールを抜き出すコード(1/2)

```
#!perl
# crex.pl
# checkpoint's rule extractor
#
use strict;

my ($in,$out,$str,$line,@ele,$i);
my ($openclose,$head,$rule,$src,$dst,$ser,$act,$dis);
my (@rule,@dis,@src,@dst,@ser,@act);

$openclose = qr/¥([^()]*(?:(??{$openclose})[^()]*)*¥)/;
$head      = qr/¥((¥S+)/;
$rule      = qr/rule ¥(([^()]*(?:(??{$openclose})[^()]*)*)¥)/;
$src       = qr/src ¥(([^()]*(?:(??{$openclose})[^()]*)*)¥)/;
$dst       = qr/dst ¥(([^()]*(?:(??{$openclose})[^()]*)*)¥)/;
$ser       = qr/services ¥(([^()]*(?:(??{$openclose})[^()]*)*)¥)/;
$act       = qr/action ¥(([^()]*(?:(??{$openclose})[^()]*)*)¥)/;
$dis       = qr/disabled ¥(([^()]*(?:(??{$openclose})[^()]*)*)¥)/;

sub extract($$);
sub output($$$$$$);
sub usage();

$in  = shift || usage();
$out = shift || 'fw_rule.csv';
open IN, "<$in";
open OUT,">$out";

{
  local $/ = undef; $str = <IN>;
  $str =~ s/¥s+/ /g;
}

$i = 1;
while ($str =~ /$rule/g){
  $line = $1;
  @rule = ($i++);
  @dis = extract($line,$dis);
  @dis = map{$_ = 'Disabled' if ($_ eq 'true')}@dis;
  @src = extract($line,$src);
  @dst = extract($line,$dst);
  @ser = extract($line,$ser);
  @act = extract($line,$act);
  output(¥@rule,¥@dis,¥@src,¥@dst,¥@ser,¥@act);
}
```

ルールを抜き出すコード(2/E)

CSVファイルとして出力するために，行と列を置換

```
sub extract($$)
{
  my $text = shift;
  my $tmp  = shift;
  my ($line,$ele,$fla,@ret);
  if ($text =~ /$tmp/){
    $ele = $1;
    if ($ele =~ /$head/){
      $fla = $1;
      @ret = ($1);
      return @ret;
    }else{
      $fla = $1;
      @ret = grep{$_ ne ':' && $_ ne ''}(split / /, $fla);
      return @ret;
    }
  }
}
```

エントリが複数ある時用のトリック

```
sub output($$$$$$)
{
  my ($ruleref,$disref,$srcref,$dstref,$serref,$actref) = @_;
  my ($max,$i,$ref);
  $max = (sort {$b <=> $a} (
    scalar @$ruleref, scalar @$disref,
    scalar @$srcref,  scalar @$dstref,
    scalar @$serref))[0];
  for $i (0 .. $max-1){
    for $ref ($ruleref,$disref,$srcref,$dstref,$serref,$actref){
      printf OUT ("%s, ", (defined $$ref[$i])? $$ref[$i] : '')
    }
    print OUT "¥n";
  }
}

sub usage()
{
  print "$0 IN_FILE [OUT_FILE]¥n";
  exit;
}
```

流れるパケットが見えたなら

- ユーザがやりとりするDNSパケットはUDPでだけ流れているわけではない．512Bを超えるデータはTCPを使う．
- そんな，AやらPTRレコードやらで512Bも使うようなデータなんてないと思って止めていたら・・・
- あるとき，大量のパケットドロップが発生．あるサイトのAレコードがものすごい数あった．
- 気がついたのはTruncateBitが立っていたから

UDPパケットの構成

Ethernet numbers of INTERNET (RFC1340)

DST	SRC		Data	CRC
6	6	2	46 – 1500	4

,0800　DoD IP

0								1							
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Version				IHL				Type of Service				Total Length			
Identification												Flags	Fragment Offset		
Time to Live								Protocol				Header Checksum			
Source Address															
Destination Address															
Options													Padding		

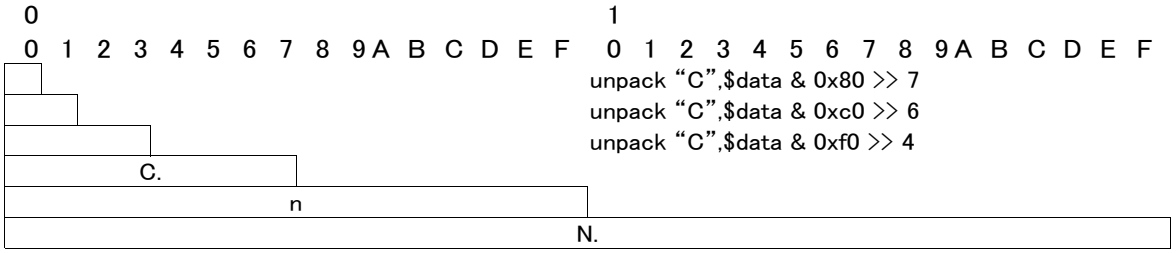
0								1							
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Source Port												Destination Port			
Length												Checksum			
data octets ...															

DNSパケットヘッダの構成

0 1 2 3 4 5 6 7 8 9A B C D E F							
ID							
QR	Opcode	AA	TC	RD	RA	Z	RCODE
QDCOUNT							
ANCOUNT							
NSCOUNT							
ARCOUNT							

ID	16bit識別子
QR	メッセージが質問(0)か回答(1)かを示す
OPCODE	質問の種類，標準的な質問(0)，逆問合せ(1)，サーバ状態問合せ(2)
AA	権威をもった正式な回答かどうか
TC	伝送可能な最大長よりもメッセージが長くなった際に，メッセージを切り落
RD	再帰回答の要望
RA	再帰回答可能
Z	予約語
RCODE	回答コード，エラーなし(0)，フォーマットエラー(1)，サーバ上に問題(2)，
	参照されたドメインが存在しない(3)，未実装の問合せ(4)，
	回答拒否(5)
QDCOUNT	質問部の項目数
ANCOUNT	回答部の項目数
NSCOUNT	権威を持った正式な項目数
ARCOUNT	追加の項目数

pack/unpackの使い方



h	A hex string (low nybble first)
H.	A hex string (high nybble first)
c	A signed char value (8 bits)
C.	An unsigned char value (8 bits)
n	An unsigned short in network (big-endian) order (16 bits)
N.	An unsigned long in network (big-endian) order (32 bits)

BPFの使い方

- /dev/fd? というファイルをオープンしておく
  ? の部分はOSの設定に依存

```
immediate = 1;
ioctl(fd,BIOCIMMEDIATE,&immediate);
(void)strncpy(ifr.ifr_name,"pcn0",4);
ifr.ifr_name[4] = '¥0';
ioctl(fd, BIOCSETIF, (caddr_t) & ifr);
ioctl(fd, BIOCPROMISC);
ioctl(fd, BIOCSETF, (caddr_t) & filter);
ioctl(fd, BIOCGBLEN, (caddr_t) & bufsize);
buf = (u_char *) malloc ((unsigned) bufsize);
cc = read(fd,(char *) buf,bufsize);
```

限界までパケットをためずに処理する

VMWare のネットワークインタフェース

/dev/pcn0 を fd に関連づける

プロミスキャスモードを設定

フィルタを設定

取得したデータのサイズを把握

DNSパケット用のBPFフィルタ

```
static struct bpf_insn insns[] = {
  BPF_STMT(BPF_LD  +BPF_H   +BPF_ABS, 12),
  BPF_JUMP(BPF_JMP +BPF_JEQ +BPF_K,   ETHERTYPE_IP, 0, 8),

  BPF_STMT(BPF_LD  +BPF_H   +BPF_ABS, 20),
  BPF_JUMP(BPF_JMP +BPF_JSET+BPF_K,   0x1FFF, 6, 0),

  BPF_STMT(BPF_LD  +BPF_B   +BPF_ABS, 23),
  BPF_JUMP(BPF_JMP +BPF_JEQ +BPF_K,   0x11, 0, 4),

  BPF_STMT(BPF_LDX +BPF_B   +BPF_MSH, 14),
  BPF_STMT(BPF_LD  +BPF_H   +BPF_IND, 14),
  BPF_JUMP(BPF_JMP +BPF_JEQ +BPF_K,   0x0035, 0, 1),

  BPF_STMT(BPF_RET +BPF_K,            (u_int)-1),
  BPF_STMT(BPF_RET +BPF_K,            0),
};
```

先頭から12Byte（RFC894 の場合）
ETHERTYPE がIP (0080) か？
A <- P[K : 2]

先頭から14+6Byte
IP Fragment offset が立っているか？
pc += (A & K) ? jt : jf

先頭から14+9 Byte
IP Protocol Type が UDP (17) か？
pc += (A == K) ? jt : jf

先頭から14 + IPHeaderLen + 0 Byte
Source Port が 53 か？
X <- 4 * (P[K:1] & 0xf)
A <- P[X+K : 2]

受諾

破棄

IHL

BPFを使うコード(1/4)

```
#!/usr/bin/perl
#
use Inline C;
use NetPacket::Ethernet qw(:strip);
use NetPacket::IP;
use NetPacket::UDP;
use Net::DNS::Packet;
use strict;

my ($fd,$rin,$rout,@data,$pac);
my ($ip,$udp,$pac,$dns);

if(($fd = prepare()) < 0){
  print "Can't setup BPF\n";
  exit (1);
}

$rin = '';
vec($rin,$fd,1) = 1;

print "Ready!\n";

while(1){
  select($rout = $rin, undef, undef, undef);
  @data = getdata($fd);
  for $pac (@data){
    $ip  = NetPacket::IP->decode(eth_strip($pac));
    $udp = NetPacket::UDP->decode($ip->{data});
    $pac = $udp->{data};
    $dns = Net::DNS::Packet->new(\$pac);
    printf "%s:%s -> %s:%s\n%s\n----------------------\n",
      $ip->{src_ip},
      $udp->{src_port},
      $ip->{dest_ip},
      $udp->{dest_port},
      $dns->print
  }
}
__END__
__C__
```

BPFを使うコード(2/4)

```
#include <sys/types.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <net/bpf.h>
#include <net/if.h>
#include <net/ethertypes.h>
#include <stdio.h>
#include <fcntl.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#define  bhp ((struct bpf_hdr *)bp)

static int    bpf_open(void);

int prepare()
{
  int fd,immediate;
  struct ifreq ifr;
```

```
static struct bpf_insn insns[] = {
  BPF_STMT(BPF_LD  +BPF_H   +BPF_ABS, 12),
  BPF_JUMP(BPF_JMP +BPF_JEQ +BPF_K,   ETHERTYPE_IP, 0, 8),

  BPF_STMT(BPF_LD  +BPF_H   +BPF_ABS, 20),
  BPF_JUMP(BPF_JMP +BPF_JSET+BPF_K,   0x1FFF, 6, 0),

  BPF_STMT(BPF_LD  +BPF_B   +BPF_ABS, 23),
  BPF_JUMP(BPF_JMP +BPF_JEQ +BPF_K,   0x11, 0, 4),

  BPF_STMT(BPF_LDX +BPF_B   +BPF_MSH, 14),
  BPF_STMT(BPF_LD  +BPF_H   +BPF_IND, 14),
  BPF_JUMP(BPF_JMP +BPF_JEQ +BPF_K,   0x0035, 0, 1),

  BPF_STMT(BPF_RET +BPF_K,          (u_int)-1),
  BPF_STMT(BPF_RET +BPF_K,          0),
};

static struct bpf_program filter = {
  sizeof insns / sizeof(insns[0]),
  insns
};
```

BPFを使うコード(3/4)

```
 fd = bpf_open();
 immediate = 1;
 if(ioctl(fd,BIOCIMMEDIATE,&immediate) < 0){
   fprintf(stderr,"BIOCIMMEDIATE: %s\n",strerror(errno));
   return (-1);
 }
 (void)strncpy(ifr.ifr_name,"pcn0",4);
 ifr.ifr_name[4] = '\0';
 if (ioctl(fd, BIOCSETIF, (caddr_t) & ifr) < 0){
   fprintf(stderr,"BIOCSETIF: %s\n",strerror(errno));
   return (-2);
 }
 if (ioctl(fd, BIOCPROMISC) < 0){
   fprintf(stderr,"BIOCPROMISC: %s\n",strerror(errno));
   return (-3);
 }
 if (ioctl(fd, BIOCSETF, (caddr_t) & filter) < 0){
   fprintf(stderr,"BIOCSETF: %s\n",strerror(errno));
   return (-4);
 }
 return fd;
}
```

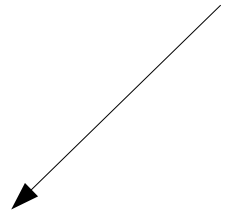
```
void getdata(int fd)
{
 int i,cc;
 u_char *buf, *bp, *ep;
 int bufsize;
 Inline_Stack_Vars;

 if (ioctl(fd, BIOCGBLEN, (caddr_t) & bufsize) < 0){
   fprintf(stderr,"BIOCGBLEN: %s\n",strerror(errno));
 }

 if((buf = (u_char *) malloc ((unsigned) bufsize)) == 0){
   fprintf(stderr,"malloc: %s",strerror(errno));
 }

 cc = read(fd,(char *) buf,bufsize);
 bp = buf;
 ep = bp + cc;
```

関数の最初で宣言

BPFを使うコード(4/E)

```
  Inline_Stack_Reset;

  while (bp < ep){
    int caplen,hdrlen;

    caplen = bhp->bh_caplen;
    hdrlen = bhp->bh_hdrlen;

    Inline_Stack_Push(sv_2mortal(newSVpvn(bp+hdrlen,caplen)));

    bp += BPF_WORDALIGN(hdrlen + caplen);
  }
  Inline_Stack_Done;
}
```

Perlでの文字列を作ってスタックに積む



これを動かすと

```
./dns.pl
Ready!
;; HEADER SECTION
;; id = 44135
;; qr = 1    opcode = QUERY    aa = 1    tc = 0    rd = 1
;; ra = 1    ad = 0    cd = 0    rcode  = NOERROR
;; qdcount = 1  ancount = 2  nscount = 3  arcount = 2

;; QUESTION SECTION (1 record)
;; www.perl.com.        IN      A

;; ANSWER SECTION (2 records)
www.perl.com.   3600    IN      A       208.201.239.56
www.perl.com.   3600    IN      A       208.201.239.8

;; AUTHORITY SECTION (3 records)
perl.com.       86400   IN      NS      ns2.xor.com.
perl.com.       86400   IN      NS      ns.songline.com.
perl.com.       86400   IN      NS      ns1.sonic.net.

;; ADDITIONAL SECTION (2 records)
ns.songline.com.        86400   IN      A       208.201.239.31
ns1.sonic.net.  86400   IN      A       208.201.224.11
172.16.100.11:53 -> 172.16.102.52:65490
1
----------------------
```

おわりに

- BPFスクリプトはNetBSD上で作成しました
- まだまだ作成途中です
- あやしい？ネタを仕込み中
- ご質問等ございましたらどうぞ