Filtering DNS message capture with tcpdump

Dave Knight <dave.knight@icann.org>
ICANN

Why use tcpdump?

- There are better ways to filter DNS message capture than tcpdump, dnscap for example, however...
  - These might not be available for your platform
  - Site policy might now allow them to be installed
  - This tcpdump method might actually be faster
- tcpdump is available pretty much everywhere

How to use tcpdump?

- libpcap expression
- this sort of thing is familiar

```
src host my.cache and dst host a.server and port 53
```

- need to match against dns wire format to match names

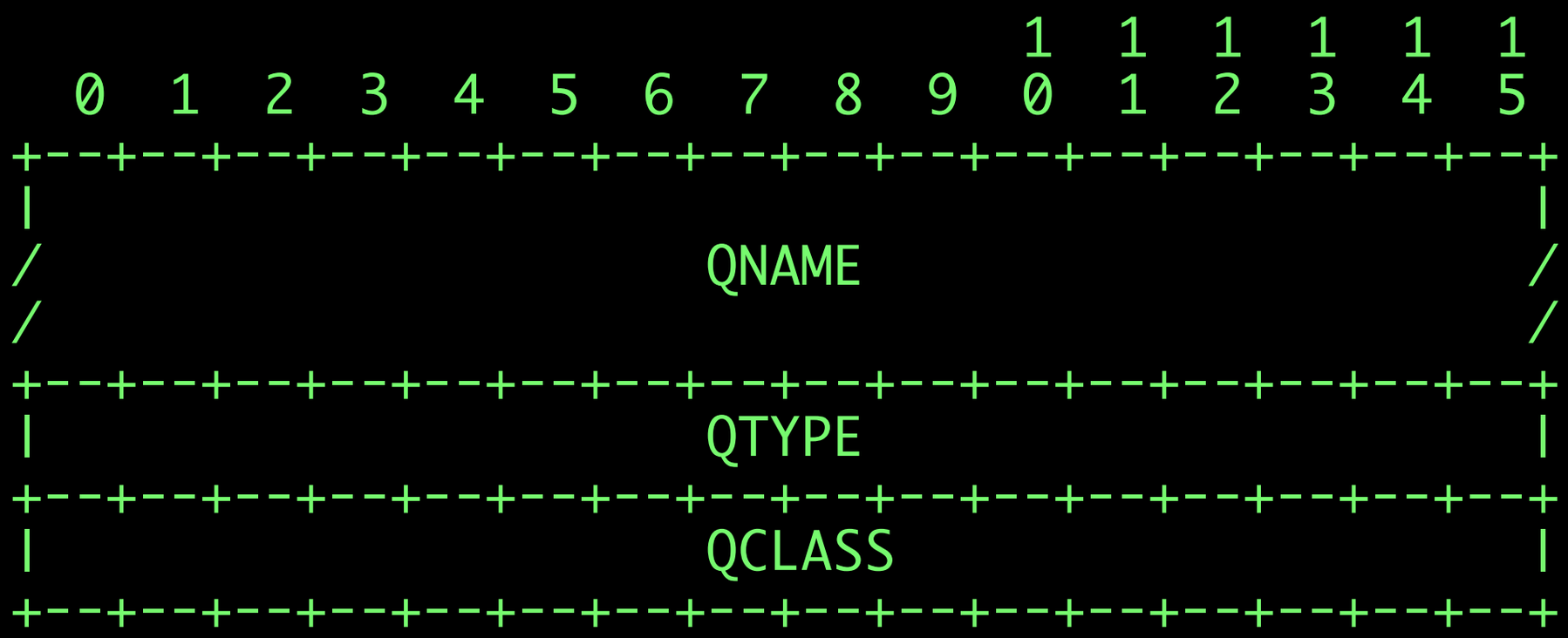
DNS message format

RFC 1035 4.1 Format

```
+----------------------+
|        Header        |
+----------------------+
|       Question       | the question for the name server
+----------------------+
|        Answer        | RRs answering the question
+----------------------+
|      Authority       | RRs pointing toward an authority
+----------------------+
|      Additional      | RRs holding additional information
+----------------------+
```

Question section

RFC 1035 4.1.2 Question section format



QNAME/QTYPE/QCLASS arpa. / NS / IN ?

```
    udp[20]          ==   4   # . Label length
and udp[21] | 0x20 ==  97   # A OR a
and udp[22] | 0x20 == 114   # R OR r
and udp[23] | 0x20 == 112   # P OR p
and udp[24] | 0x20 ==  97   # A OR a
and udp[25]          ==   0   # . Null label
and udp[26]          ==   0   # NS
and udp[27]          ==   2   # NS
and udp[28]          ==   0   # IN
and udp[29]          ==   1   # IN
```

But QNAME length is variable...

- This expression will only match exactly “arpa.”
- Want to match everything under the arpa. domain
- This means matching QNAMEs ending in /arpa./ up to 255 bytes long
- Can write pcap expression matching for /arpa./ at incrementing offsets

Match in QNAME at incrementing offsets

```
(
      udp[20]         ==   4   # Label length
    and udp[21] | 0x20 ==  97   # A OR a
    and udp[22] | 0x20 == 114   # R OR r
    and udp[23] | 0x20 == 112   # P OR p
    and udp[24] | 0x20 ==  97   # A OR a
    and udp[25]         ==   0   # Null label
) or (
      udp[21]         ==   4   # Label length
    and udp[22] | 0x20 ==  97   # A OR a
    and udp[23] | 0x20 == 114   # R OR r
    and udp[24] | 0x20 == 112   # P OR p
    and udp[25] | 0x20 ==  97   # A OR a
    and udp[26]         ==   0   # Null label
) or (
    ..
```

Only matches udp, repeat for tcp, upd6, tcp6

```
        udp[20]          ==   4   # Label length
    and udp[21] | 0x20 ==  97   # A OR a
..
        tcp[46]          ==   4   # Label length
    and tcp[47] | 0x20 ==  97   # A OR a
..
    udp
    and ip6[60]          ==   4   # Label length
    and ip6[61] | 0x20 ==  97   # A OR a
..
    tcp
    and ip6[86]          ==   4   # Label length
    and ip6[87] | 0x20 ==  97   # A OR a
..
```

Optimization

- Seperate expressions for udp, tcp, udp6, tcp6 matching for arpa./NS/IN? gives an expression slightly more than 10,000 lines long.
- Matching for the exact same thing in the udp and tcp array, can sidestep that and match directly in ip and ip6 arrays. This gets the expression down to around 7000 lines.
- Can go one step further and match in the ether array directly and make the expression smaller still. This could reduce flexibility.

script

http://dave.knig.ht/dnspcap/

```
./dnspcap --help

usage: dnspcap

    --help              show this usage message
    --src-ip addr       match source ip address
    --dst-ip addr       match destination ip address
    --src-port port     match source port
    --dst-port port     match destination port
    --qname string      match QNAME string
    --qtype QTYPE       match QTYPE
    --qclass QCLASS     match QCLASS
    --max-length int    max domain length
```

Performance comparison with dnscap

Find 50,000 matches for /in-addr.arpa/ in a dump of 50,000,000 packets from L.root-servers.net

```
$ time dnscap -6 -r dump -w dnscap.out -c 50000 -x in-addr\.arpa
real    0m0.989s
user    0m0.853s
sys     0m0.049s

$ dnspcap --qname in-addr.arpa --max-length 255 > dnspcap.filter
8614 /var/tmp/dns/dnspcap.filter

$ time tcpdump -nn -r dump -F dnspcap.filter -c 50000 -w dnspcap.out
real    0m5.606s
user    0m4.645s
sys     0m0.201s

4421226 13 Mar 12:43 dnscap.out.20110311.041603.795466
5121280 13 Mar 12:43 dnspcap.out
```

Performance comparison with dnscap

Find 5,000,000 matches for /in-addr.arpa/ in a dump of 50,000,000 packets from L.root-servers.net

```
$ dnscap -6 -r dump -w dnscap.out -c 5000000 -x in-addr\.arpa
real    1m36.813s
user    1m18.110s
sys     0m4.775s

$ dnspcap --qname in-addr.arpa --max-length 255 > dnspcap.filter
8614 /var/tmp/dns/dnspcap.filter

$ tcpdump -nn -r dump -F dnspcap.filter -c 5000000 -w dnspcap.out
real    0m54.331s
user    0m22.227s
sys     0m4.927s

442885303 13 Mar 12:45 dnscap.out.20110311.041603.795466
516972707 13 Mar 12:46 dnspcap.out
```

Notes

- Slight difference in packets captured vs dnscap. In 5,000,000 matches got around 10 messages which dnscap missed.
- Only captures matching tcp packets, the rest of the session is ignored.
- No consideration for fragments, but assume it's unlikely to see fragmented questions.

Summary

- Probably already have tcpdump installed, nothing else is required as pcap can be generated elsewhere.
- Fast for larger datasets, probably similarly less resource intensive for live capture, but haven't tested.

Acknowledgements...

- Geoff Sisson
- Shane Kerr

Questions?

dave@knig.ht