CISC 856, Fall '03
DNS homework assignment

Due: Nov. 6th
address questions to: Greg Forte (gforte@udel.edu)

For this assignment you may work individually or in teams of two.
Part A: Forouzan, Chapter 18, questions 1-20, 27-33.
Part B: hands-on experiments (see below).

For the following exercises you will use tcpdump (or ethereal) and dig (domain information groper; type 'man dig' for more information). You will need to work on a host on which you have access to the full functionality of tcpdump (or ethereal), specifically the ability to dump the contents of packets (i.e. NOT the acad.ece machines). This means either you must have root privileges or tcpdump must be installed setuid root. If you do not have access to any such machine, you may use random.rdms.udel.edu.

Please e-mail gforte@udel.edu to set up an account, or concerning problems with the machine.

(Note: ethereal is available on random, but you can not use ethereal to monitor the network interface directly, due to security concerns. If you wish to use ethereal, you can save tcpdump's output to a file and then import the data into ethereal).

For each exercise, run tcpdump in one window and dig in another (unless otherwise noted). Submit scripts from both windows, including all commands and output. For tcpdump, use the following options:
- -x (dump raw packets in hex format; -X will also display ASCII, but this is only marginally useful.)
- -n (prevents conversion of addresses and ports to names. IMPORTANT! if you don't use this option, tcpdump will perform DNS lookups to convert the addresses in the monitored packets to names, and these lookups will clutter up in your output.)
- -s ### (tells tcpdump to 'snarf ' (capture) ### bytes of data; use a large number, such as 9999.)
- port dns (to capture only DNS messages.)

There may be spurious DNS requests sent by system processes while you are running your experiments. You may edit these out of your output.

1. Using dig, query for the address of any hostname in the udel.edu domain. Use dig's +qr option to print out the query message as well as the response. e.g. 'dig +qr name.udel.edu'. Analyze your tcpdump and dig output, as follows:
   a) highlight the IP header in the query and the response. (tcpdump output)
   b) highlight the UDP or TCP header in the query and response. (tcpdump output)
      • which transport layer protocol was used? Is it the one you expected? Why (not)?
   c) highlight the DNS message header in the query and response. (tcpdump and dig output)
      • what flags and codes are set in the flags fields?
      • how many of each type of record (question, answer, authoritative, additional) are there?
   d) highlight the question record in the query and the response (tcpdump and dig output).
      • are they identical? If not, why not?
   e) highlight all resource records in the response (tcpdump and dig output).
      • are there any additional records? If so, what information do they provide?
   f) highlight all offset pointers in the response (tcpdump output). Indicate the locations the offset pointers point to in the message (i.e. what domain names do they represent?)
      • Offset pointers have 14 bits of usable information. Given the requirement in the DNS standard that repeated domain names always be represented by offset pointers, does this impose a size limitation on DNS messages that is smaller than the maximum transport layer data size for either UDP or TCP? If so, is this a serious deficiency? Why (not)?
2. Repeat exercise 1, this time using dig to do an inverse (pointer) lookup on the IP address of the host you looked up in exercise 1. Remember, to form the inverse domain name for an IP address you must reverse the order of the bytes and append '.in-addr.arpa.' You must also specify that you want to do a pointer query by using dig's `-t` option: e.g. 'dig +qr -t PTR z.y.x.w.in-addr.arpa' for IP address w.x.y.z. Annotate your output as in exercise 1.

Is there anything unusual or different about this query/response as compared to the ones in exercise 1 (other than the fact that the answer is a pointer record rather than an address record?)

3. For this exercise you don't need to use tcpdump. Look up the MX records for several (at least three) different domains using dig's `-t MX` option. Try to pick domains that have multiple MX records (high volume commercial e-mail providers are prime suspects). In the dig output, highlight the preference values for each mail exchange record. Which host has top priority for each domain? What do you think is meant if two hosts have equal preference?

4. Begin a zone transfer for udel.edu, using 'dig -t axfr @dns1.udel.edu udel.edu'. (Press CTRL-C shortly thereafter – no need to download the entire zone ;-) Capture the first 10 (approximately) packets of the connection with tcpdump (note that you'll first see an address lookup for dns1.udel.edu!), at least up to the first packet of the response message. You can use the `-c` option to limit the number of packets tcpdump displays before exiting. You can (and should) also drop the `-s` option for this exercise.

(Note: this will only work from hosts on UD's networks, 128.175.0.0 or 128.4.0.0).

How many answer records are there in the first DNS response message? How many bytes? Why do you think DNS requires large transfers to be handled by TCP? (hint: data reliability is one possible answer, but it's not the main reason.)