

Lab Assignment #1

Description

This lab assignment will help you get ready for Homework #1, which we will post soon. It should take less than one hour, and is graded on the scale of credit or no-credit. You may work in groups of 2.

Please write your answers and related explanations (if necessary) in a Lab1.txt file, with the names and student IDs of both group partners, and submit it using the submit script at /usr/class/cs193i/bin/submit. Look at the file /usr/class/cs193i/bin/README-submit for instructions on using the script.

This lab assignment is due Wednesday, June 30, 2004 at 11:59pm.

Useful Networking Programs

In this section, we will play with a couple of useful networking programs. Log into a leland machine, such as elaine, saga, epic, etc...

nslookup – skim the man pages of nslookup by typing:

```
man nslookup
```

at the prompt. Basically, nslookup allows you to find the host name of any computer that's on the Internet, provided that you know the IP address. Here are a few IP addresses to play with:

```
128.12.132.29  
127.0.0.1  
66.94.230.42
```

Note that nslookup will tell you which DNS server responded to your request. For some information on the DNS system, see:

<http://en.wikipedia.org/wiki/DNS>

An interesting note is that IP address 127.0.0.1 is reserved for the machine that you are running the program on (aka localhost). Any TCP/IP packet with this “loopback” destination address will be routed by the OS back to the originating machine. The network card will never see this packet.

ping – given a host name, or an IP address, it will tell you whether the host is up and running and connected to the Internet (or rather, connected to your computer). Skim the man pages quickly. To stop the ping program (and any other UNIX program that you run) type `Control-C`.

What ping does is send out a special packet called an Internet Control Message Protocol (ICMP) Echo Request packet. When a machine receives this echo request, it responds with an echo reply packet, placing the original echo request packet into the data field of the echo reply.

The Time To Live (TTL) field is interesting. Ping packets usually start out with a TTL of 255 (normal IP packets have TTL set to 60). As the packet traverses the network, the TTL field gets decreased by one by each router it goes through. When the TTL drops to 0, the packet is discarded by the router. This prevents packets from living forever on the Internet, bouncing back and forth, because their destination computer is off, or disconnected from the network.

We can use the TTL to figure out approximately how many router hops the packet has gone through, in this case:

$$\# \text{ router hops} = 255 - (\text{TTL of Echo Reply})$$

Play around with a few hosts to see how many hops your packets go through.

tracert – this program finds the path that packets take to a destination computer. Try typing:

```
tracert www.berkeley.edu  
tracert www.stanford.edu
```

Tracert works by setting the TTL of a packet to 1, and the Destination IP address to the ultimate destination (www.stanford.edu, in the second example). When the packet hits the first router, the router will decrement TTL, see a 0, and respond with an ICMP notification that the packet has been expired. Then, tracert will set TTL to 2, and send a second packet out. Then, TTL is set to 3, and so on... In this manner, you can map the route to the destination IP address.

Question 1: What is the name of the DNS server that responds to your nslookup requests?

Question 2: For the ping program, what does the time field mean—is it the single-trip latency, or round-trip latency?

Question 3: Find a live host with a high ping. What is the host's name/IP Address? What is the ping? This probably means that the host computer is far away from you, or in a hard to reach location.

RFCs – Documenting the Internet

An RFC (Request for Comment) document is one of a series of numbered documents that describe Internet standards or contain information about the Internet. Skim briefly the Wikipedia entry on RFCs:

<http://en.wikipedia.org/wiki/RFC>

Pay attention, in particular, to the section titled: *List of the most important RFCs*, and the section describing joke RFCs published on April 1 of every year (=). Here are a few more useful sites where you can find the RFCs.

<http://www.faqs.org/rfcs/rfc-index.html>
<http://www.rfc-editor.org/>
<http://www.rfc-editor.org/rfcxx00.html>

See if you can find the RFC that describes all you need to know about the concept of a Uniform Resource Locator (URL). Also, look for the RFC on Dynamic Host Configuration Protocol. Notice that RFCs are numbered sequentially, and that many times, a newer RFC will replace, and make obsolete, an older one.

In Homework #1, we will implement a client to read email from a server that understands the Post Office Protocol – Version 3. See if you can find the RFC on POP3. You may want to read this RFC once, in preparation for Homework #1.

Question 4: Who are the authors (editors) of the RFC on URLs? What is the date? Why do the authors say that URLs should never contain spaces?

Authentication

This is a topic we will discuss in future classes, but I want to get your feet wet a little here. Basically, authentication is a process by which you prove to someone else *that you are who you say you are*. First, you can check out Wikipedia once again: <http://en.wikipedia.org/wiki/Authentication>. Make sure you read about the three general ways to authenticate yourself to someone else.

In Homework #1, your POP3 client will need to submit a password to the mail server. The problem with the standard POP3 way of doing things is that the password is transmitted “in the clear,” meaning that the password is not encrypted when it is put into a packet and shipped onto the network wire. From what we know so far about Ethernet and TCP/IP, this could potentially be a bad thing. Someone could potentially “sniff” the packets on your network, and collect your user name and password (see About.com’s article on Packet Sniffing: <http://netsecurity.about.com/cs/hackertools/a/aa121403.htm>).

Thus, we might want to look into using some way to encrypt our password before we send it across the wire. Check out the MD5 hash function (see <http://en.wikipedia.org/wiki/Md5>). On the UNIX prompt, you can try out the md5sum function. Read the man pages of md5sum to see how it works. Note that in interactive mode, md5sum requires an End-of-File (EOF) character to stop receiving input and compute the hash. Insert an EOF by pressing Control-D (possibly twice).

So, one could imagine that you can compute the md5sum of your password, send it across the network, and then the server can compare your result with the md5sum of the password that it has previously stored for you. In that way, your password will never be seen “in the clear” on the network.

Question 5: Of the three classes of authentication methods as described in the Wikipedia article, which one does the POP3 method fall under?

Question 6: Computing the md5sum of a password and then sending it across the network to the server prevents the password from being seen on the network. However, it *still* does not prevent someone who is sniffing on the network wires to break into your account. Why not? Can you think of a better solution?