System Administration Decal

Intermediate Lab #6

April 9, 2008

Introduction

This week we talked about the wonderful world of Internet mail. Unfortunately, it is rather impractical with our virtual server setup to set up your own incoming mail service. So for this lab, we'll focus on investigating the mail setup on the OCF servers and using the SMTP protocol. And then we'll be doing something totally unrelated because it's fun. Remember, work with your groups in person; it really helps everyone's understanding the more people work together. As always, send responses to joshk.decal@triplehelix.org.

I'll also be looking at your project proposals, so watch your inboxes throughout the course of the week in case I need to discuss it with your group. Once I've acked your proposal you should begin working on it.

I. Interacting with Mail Locally

In this exercise we'll poke around with the most common mail-oriented commands in the Unix world, using the OCF machines.

1. Use the 'mail' command to send yourself a message. How do you specify a subject? How do you read your mail using this command?
2. Now connect to the OCF SMTP server, mail.ocf.berkeley.edu using telnet or PuTTY, and send yourself a message. Provide a transcript of your connection.
3. On an OCF Linux box, is there an alternate way to access a SMTP-type interface without having to use telnet? Hint: look at 'sendmail' options.
4. Find a non-SMTP method of sending messages using sendmail. (Figure out how mutt does it.)
5. Play around with mutt and pine and figure out how to send and browse email messages in both. Which do you prefer, and why? (The mail client war is about 40% more intense than the editor war.)

II. Just For Fun - PGP 101

(This is probably the longest exercise marked "Just For Fun" in history, but the truth is that it's only tangentially related to email.)

PGP (Pretty Good Privacy) is an encryption system based upon several strong algorithms such as RSA and DSA. It seems irrelevant right now, but email is the most popular use of PGP right now. In this section of the lab we'll go through the basics of public key encryption.

When you create an encryption key in the PGP model, you are creating two keys: a public key that you can disseminate on your web site or on a public key server, and a private key, that you must keep secret at all times. Data encrypted by your public key is only decryptable with your private key, and vice versa.

Thus, you distribute your public key so that people can send you email for your eyes only whenever they want to, by encrypting the message with your public key. And then when you send emails, you can *sign* them by taking a hash of the message's text, and then encrypting that with your private key. A recipient can verify the hash by decrypting the signature and comparing it against the hash it made of the message text.

http://www.dewinter.com/gnupg_howto/english/GPGMiniHowto.html may be useful for this exercise. Also, you don't have to use your virtual server for this, but you are responsible for figuring out how to get gpg (the GNU implementation of PGP) on *some* machine, with or without a frontend.

1. Use gpg to generate a key. You can use the default options of creating a 1024-bit DSA key. Have each member of your group create a key for themselves. In your submission, include an ASCII export of each member's public key.
2. Use gpg to send your public key to a server (for example, keyserver.noreply.org) and download everyone else's public key into your personal keyring.
3. Find a way to set up your personal email clients (Thunderbird, etc.) to use gpg. Look for Enigmail if you're using Thunderbird. Play with the encryption and signature options while sending messages to each other.

If you meet a person who gives you their public key, you can make a claim that the person who owns that key pair is actually, legally, who they say they are. You do this by *signing* their public key (not to be confused with signing a message with your private key, totally different.) Nerds set up parties where the whole purpose is to get everyone to sign everyone else's key, so as to build a *web of trust*, that is, an interconnected network of people who have all signed each other's keys. They often take it an extra step by requiring proof of identification.

This isn't as dumb as it sounds like. Suppose you know John, whose key you have signed. John knows Eric, whose key he has signed. One day you receive a signed message from Eric. Because you have signed John's key and you trust him, you can transitively trust Eric (at least, that he is who he says he is) even if you've never met him before.

4. Sign everyone's keys in your group. Since the signing is done with your copy of your group members' public keys, you have to upload them back to the server when you're done.
5. Finally, use GPG to encrypt this entire lab submission. My public key 78446F26 can be downloaded from many keyservers. Sign it with one of your private keys too.
6. (Really, actually, just for fun) Nerds participate in all sorts of pissing contests related to key signatures. Check out http://dtype.org/keyanalyze/ and look up the latest keyanalyze report. See where I am! What does MSD mean?