Open Computing ‘‘Hands-On’’: ‘‘Answers to Unix’’ Column: August 94

Doing remote computing by issuing commands through electronic mail

By Ray Swartz

Alberto Ferrari is looking for a way to run commands on a system that he can’t log into. What’s more, he needs to do it from different systems. The solution involves a mail filter shell script and a C program to add security.

The Script Is in the Mail

Question: Because I sometimes travel to places with e-mail but no online connections to the Internet, I’d like to have a way to remotely execute a script on my office system. My idea is to send the script as an e-mail message and get the results back in another e-mail message.

I know this feature would introduce security problems into the system, but I could send some form of identification along with the script. Any ideas?

Alberto Ferrari
Florianopolis, Brazil

Answer: In essence, you want to filter your mail and, if the message contains commands to execute, have the filter run the commands for you.

As you mention, you need a way to stop unauthorized people from executing commands on your system. Using passwords is dangerous because they must travel in plain text across the network; a better idea is to encrypt the mail containing the commands to run.

While always using the same key for encrypting your command messages will work, I decided to add more security by using a onetime key system; that is, a series of random numbers known to both the sender and the receiver but no one else.

The random-number generator I selected was certified as good by Donald Knuth’s seminal work, ‘‘The Art of Computer Programming, Volume 2.’’

Every time you use one of these numbers, you mark it off your list. Your system will be doing the same thing.

The information required is (1) a way to announce that a message contains commands to run, (2) the e-mail address that is to receive the commands’ output, and (3) authorization.

To identify that a message contains commands, put the words ‘‘Script to run’’ in the subject line. The message body must identify the return address with the line ''Return address:'' followed by the e-mail address that is to get the output of the enclosed commands. Authorization is granted if the message contains a Return address line after decrypting. The rest of the message contains the commands you want to run.

Because the encrypted message may contain non-ASCII characters, it must be run through uuencode before you can e-mail it. The `cmd.filter` script, is shown in Part A of the Listing. After it performs all the protocol verification, it puts command lines in a temporary file, calls the shell to run them, and then mails the output to the listed receiver (lines 33-35).

The `chkcode` program (Part B) generates encryption keys. If you use the `-l` option, it prints a list of encryption keys on the standard output. The number you want listed is placed after the `- l` option on the command line. If `chkcode` is called without arguments, it generates the current encryption key based on a counter kept in a file.

Some setup is required for these programs. First, in `cmd.filter` you need to assign the path name of the `chkcode` executable file to the variable `CHKCODE` (Part A, line 2). Note that it must be an absolute path name. Second, in `chkcode` you must change the define term `ONETIMEPADFILE` (line 7) to be the absolute path name you want to use for the counter file. You must initialize this file with a starting value (I recommend ''1'') before this system will work. Next, you should redefine the random-number generator's starting seed. Because I have published the one in `chkcode.c`, it is best to pick another one (Part B, line 8). Choose any number between one and four billion.

You must also set up a .forward file in the home directory of the account that will receive the execution messages. The `.forward` file should contain two entries: this account's name preceded by a backslash and a pipe command to the `cmd.filter` script. Part C shows the `.forward` file I used while testing this application for account ray.

The `cmd.filter` program relies on the crypt command, which may be missing from your system. If it is, you can find other encryption programs on the Internet. Another solution is the simple encryption technique of using exclusive OR operations. An example is the `mailcrypt` program shown in my June 1994 column.

Part D shows the steps required to prepare a command file for insertion into an e-mail message.

One important point about preparing your mail message. When a file is encoded with `uuencode`, you have to list the file name where `uudecode` is to place the decoded file. Because a daemon will run `cmd.filter`, you have to know the directory where `uudecode` will put the decoded script. Thus, you need to use an absolute path name when you encode the script file before mailing it. In Part D, I chose to put the script file in my home directory.

Edited by Becca Thomas / Online Editor / UnixWorld Online / beccat@wcmh.com

Go To Content   Search

Last Modified: Tuesday, 22-Aug-95 15:45:14 PDT