$2.50

MUUGlines

February 1998 Volume 10, Number 6

This Month's Meeting

Protecting Data Transfer in an Insecure Environment

This month's presentation is on how to protect your information as it is being transmitted across a network using Pretty Good Privacy(PGP) version 5.0 and Secure Shell(ssh) version 1.2.22. We will be looking at some of the current weaknesses of network security and how PGP and ssh address these problems. In addition, some of the new features of PGP v5.0 will be presented.

For door prizes this month, we have a copy of Red Hat 5.0, Red Hat t-shirts (size XL), Red Hat frisbees, and Red Hat bumper stickers!

Our meeting this month is Tuesday, the 10th of February. We meet at IBM Canada's offices in the TD Centre, at the corner of Portage and Main. We'll gather in the lobby on the main floor – please try to be there by about 7:15 PM. Steve Moffat will then take us up to the meeting room just before the meeting starts at 7:30. Don't be late, or you may not get in.

Parking is available either in the parkade behind the TD building, off Albert Street, or in the ground level lot just north of the TD building. Entrance to the lot is from Albert Street, behind the parkade. Either way, parking is a $1.25 flat rate for the evening. You purchase your ticket from a dispenser, so make sure you've got exact change – a loonie and a quarter, or 5 quarters.

During Last Month's Meeting...

Contributed by Doug Shewfelt

At our January meeting, Brian Pauls of the City of Winnipeg gave a presentation on TCP/IP addressing and routing. He started with a brief summary of networking concepts and history. He then described TCP/IP addresses and subnet masking, and finished off with a discussion of routing and common routing protocols.

The TCP/IP address has two parts: the network component and the host component. Unlike many protocol addresses the dividing line between the two parts is movable. This allows various classes, such as a small number of networks with a large number of hosts, or a large number of networks with a small number of hosts.

The TCP/IP address it generally shown as four numbers called "octets" separated by periods. Each octet has a range from 0 to 255. A Class A address uses the first octet for the network number and the last three for the host number, allowing for over 16 million hosts per network. The first octet of a Class A address must be in the range of 1 to 126.

The first octet of a Class B address is between 128 and 191. It uses two octets for the network component, and two octets for the host component. This allows up to 65,534 hosts per network.

The first octet of a Class C address is between 192 and 223. It uses the first three octets for the network component, and the last octet for the host component. This allows up to 254 hosts per network.

There are other less common classes. Class D is used for multicast networks, while Class E addresses are marked as experimental.

Sometimes it makes sense to take a larger network and subdivide it into a number of smaller networks. This is done by taking bits from the host component of the IP address, and interpreting them as bits belonging to the network component.

For example, the default subnet mask of a Class C address is 255.255.255.0. This means that the first three octets are used for a network address, and the last one is used for a host address, which would allow for 254 hosts per network.

However, we could use the mask 255.255.255.224. If we write this in binary we would get 11111111. 11111111. 11111111. 11100000.

Note the last octet. There are three 1s, followed by five 0s. This means that when we subnet, we will interpret three bits of the host component as part of the network component. That will divide the network that we have into 8 subnetworks of 30 hosts each. Brian offered this example:

Subnet	Hosts
200.130.45.0	200.130.45.1 to 200.130.45.30
200.130.45.32	200.130.45.33 to 200.130.45.62
200.130.45.64	200.130.45.65 to 200.130.45.94
200.130.45.96	200.130.45.97 to 200.130.45.126
200.130.45.128	200.130.45.129 to 200.130.45.158
200.130.45.160	200.130.45.161 to 200.130.45.190
200.130.45.192	200.130.45.193 to 200.130.45.222
200.130.45.224	200.130.45.225 to 200.130.45.254

A host number of 0 is reserved for the address of the network, while a host number whose bits are all 1s is the network broadcast address.

Some routers also support variable length subnet masks, so that you can divide your network into several differentsized subnetworks.

Computers that do connect with the Internet must use IP addresses that are unique across the entire Internet. However, an organization may have a number of computers that will only communicate on their internal network, and these computers do not need a registered IP address. Therefore, some ranges of addresses have been reserved as unregistered IP addresses. Packets sent to these destinations are dropped from the Internet backbone. The organization can use these addresses freely without worrying about collisions with other sites using the same address.

The ranges for unregistered addresses are:

Class A	10.x.x.x
Class B	172.16.0.0 to 172.31.0.0
Class C	192.168.1.x to 192.168.254.x

Brian then provided an overview of routing. He explained how packets of data are encapsulated and passed between routers. He described how there are different approaches to routing — for example a router can know only about its own network and the networks immediately around it, or it can have knowledge of the configuration of a broader part of the network. He described a number of common protocols in use, and the advantages and disadvantages of each.

He ended his presentation with a description of some common diagnostic tools.

More 2¢ Tips!

Send Linux Tips and Tricks to gazette@ssc.com

Followup to PostScript and VC Key Sequences (LG#23)

From: Ivan Griffin ivan.griffin@ul.ie

I just wanted to point out that some of my 2cent tips in Issue 23 of the Linux Gazettte (December, 1997) were a little funky in their appearance.

While it doesn't really matter at all with the VC key sequences, it may affect someone's understanding of the bad (imho) PostScript generated by the Microsoft PS driver.

In this, the PostScript should have been pre-formatted using the appropriate HTML tags. Basically, the line :

30000 VM?

is on its own, and not part of any other line. All that you have to do to remove this artificial restriction on viewing/converting the PostScript with ghostscript is to delete this line.

On another note, someone asked me where those key sequences come from. If you check either keyboard.c or keyb_m68k.c, you will find an array of function pointers called spec_fn_table[].

This array contains a list of functions to execute when certain key combinations are received... The key combinations listed in the 2cent tips execute the functions show_state(), show_mem() and show_regs()

You will find the source for function show_state() in /usr/src/linux/kernel/sched.c; show_mem() is in /usr/src/linux/arch/i386/mm/init.c; and show_regs() is in /usr/src/linux/arch/i386/kernel/process.c

PostScript $0.02 follow-up

From: Kyle Ferrio kbf@phy.duke.edu

In the December issue of LG, Ivan Griffin suggests using pstops from the psutils package to accomplish two-up printing, gives a helpful example for A4 paper, and points out that the command line needs to be tweaked for US letter. If you're using US letter paper, then psnup (also part of psutils) already does the job nicely with no uncomfortable thinking. It might even work for A4, but I haven't checked. The psutils are generally very handy, so folks might want to have a look. An RPM is available in /contrib at ftp.redhat.com, for instance. Be advised that there seem to be at least two very distinct packages called psutils floating around Net-space.

Yet another cheap tip.

From: Gary Johnson gjohnson@season.com

Sorry if it has been mentioned before, I thought I would throw it in the Gazette pile just in case it hasn't...

Cat-proof keyboard.

Switching to an unused virtual console is a quick way to blank the screen and disable the keyboard. To make one available try

```
setterm -clear  /dev/tty12
```

on startup. ALT F12 flips to it, or ALT CTRL F12 from X. Because there (probably) isn't a login running on that VC it doesn't do much, which can be a feature. A smart cat may still luck into a troublesome key sequence.

2 cent tip - dosemu

From: Joey Hess joey@kitenet.net

I occasionally use dosemu, mainly to run some games I can't live without, but I hate seeing the C:\ prompt. So I thought it'd be nice if there were a way to tell dosemu what dos command to run, and it would run that command on bootup. Here's a perl script that does just that. Read the comments at the top, they explain some changes you need to make on the dos side of this. The basic idea is, make a ~/dos_do.bat file, that contains the command you want to run, and use lredir to let dosemu see your home directory. Then run the batch file.

```
#!/usr/bin/perl
#
```

```
# This runs dosemu.
#
# Any parameters specified
after "—" will be passed in to
dosemu to be
# run as dos commands.
#
# Setup: add to autoexec.emu:
# lredir.com h:
linux\fs\${home}
# if exist h:\dos_do.bat call
h:\dos_do.bat
#
# GPL Copyright 1996, 1997 Joey
Hess

# Split params into dosemu
parameters and dos commands.
while ($a=shift @ARGV) {
    if ($a=~m/—/ ne undef) {
last }
    $dosemu_command_line.="$a ";
}
$dos_command_line=join('
',@ARGV);
$dos_command_line=~s/;/\r\n/g;

open (OUT,"$ENV{HOME}/
dos_do.bat") || exit print
"$ENV{HOME}/dos_do.bat:
$!";
if ($dos_command_line) {
    print OUT
"$dos_command_line\r\n"; # note
dos CR LF
    print OUT "exitemu\r\n";
}
close OUT;
system "/usr/bin/dos
$dosemu_command_line";
unlink "$ENV{HOME}/dos_do.bat";
```

Re: 2c Tip "Finding What You Want with find"

From: Mike Neuhauser mike@gams.co.at

Jon Rabone, jkr@camcon.co.uk, wrote in the December 97 issue of LG:
> In the October 97 issue, Dave Nelson suggests using
> find . -type f -exec grep "string" /dev/null {} \;
> to persuade grep to print the filenames that it finds the search
> expression in. This starts up a grep for each file, however. A
> shorter and more efficient way of doing it uses backticks:
>
> grep "string" `find . -type f`
>
> Note however, that if the find matches a large number of files you
> may exceed a command line buffer in the shell and cause it to complain.

To avoid an overflow of the command line buffer use:

find . -type f | xargs grep "string"

This may give problems if filenames contain white space (e.g. touch "test file") — to avoid use:

find . -type f -print0 | xargs -0 grep "string"

Note also that find doesn't follow symbolic links to directories per default. Using find with the option -follow does the trick (find . -follow ...).

Re: Finding What You Want with find

From: Dale K. Hawkins dhawkins@mines.edu

find . -type f -exec grep "string" /dev/null {} \;

That is how I used to run things too, but a friend showed me the xargs program. Very nice. So one could turn the above statement to something like:

find . -type f | xargs fgrep "string" /dev/null

Again, the /dev/null will force the name of the file to be printed (in the unlikely case that find only found one file name). This has the benefit of not invoking a new grep process each time.

But for a really slick (and much faster search) try this:

locate $PWD | grep "^$PWD" |xargs fgrep "string" /dev/null

This assumes that your locate database is current for the directory to be searched. It does have a problem though: it tries to grep everything, including directories!

locate $PWD | grep "^$PWD" |xargs -ifilename sh -c \
"if [-f filename]; then echo filename; fi " | \
xargs fgrep "string" /dev/null

And as an exercise for the reader: Take a look at lesspipe.sh (if it is installed; download it otherwise!) See if you can create a shell script called supercat (or something) which preprocesses the input to prevent grep'ing binary files, etc.

You gotta love UNIX and especially Linux!

-Dale K. Hawkins

Finding What You Want with find Part III

From: Axel Dietrich Axel.Dietrich@neuroinformatik.ruhr-uni-bochum.de

>In the October 97 issue, Dave Nelson suggests using
> find . -type f -exec grep "string" /dev/null {} \;
>to persuade grep to print the filenames that it finds the search
>expression in.

Besides Jon Rabone's "shorter and more efficient" version in the December 97 issue using backticks:

```
grep "string" `find . -type f`
```

the following variant can be used without the danger of exceeding a command line buffer limit:

```
find . -type f -exec grep -l
"string" {} \;
```

The "-l" switch tells grep to show the name of the file in which "string" was found. To limit such a search on selected files I use a combination of the -type and -name switches.

```
find . \( -type f -name
"*\.html" \) -exec grep -l
"string" {} \;
```

This searches in all files with the suffix "html" for the string "string" and outputs the name(s) of the file(s) in which "string" was found.

Axel

More on finding

From: Alexander Larsson alla@ lysator.liu.se

In the December 97 issue Jon Rabone wrote: This starts up a grep for each file, however. A shorter and more efficient way of doing it uses backticks:

```
grep “string” `find . -type f`
```

Note however, that if the find matches a large number of files you may exceed a command line buffer in the shell and cause it to complain. A better way would be to use:

```
find . -type f | xargs grep “string”
```

which starts up a new grep everytime the command line buffer is full.

Another way to find

From: rchandra@letter.com

In an article in the LG, it was suggested that, in order to cut down on having to fork(2)/exec(2) for each grep when you’re searching through a tree of files, you use the shell’s capability of command substitution (for the file names paramaters to the grep command) with “backquotes,” “grave accents,” “backticks,” etc. as they are commonly called (“`”). In that little tidbit, it is noted that it has the limitation of the system-wide imposed limit on number of arguments, and I possibly think there might be a length issue as well (too many total bytes). Enter xargs(1).

The job of the xargs command is to read its stdin and use the resultant strings as arguments to some command prefix (such as “grep -n somestring”), much like backquotes work. However, the xargs program is “aware of” the limitations imposed by the system, and will run the command prefix as many times as necessary to exhaust the list provided on stdin, while on each run giving the command only the maximum number of arguments and the maximum byte count (?) that an exec(2) call can handle. Thus, provided that the program named in the command prefix follows the UNIX program protocol of iterating over its non-option arguments, one can search one, hundreds, thousands, even millions of files with a line like:

```
find / -type f -print | xargs grep -n ‘where is that string?’
```

Cryptographic System

From:Emil Laurentiu emil@ interlog.com

I would appreciate a lot an answer even a short one like 'no' :) I am (desperately) searching a cryptographic system for my Linux box. I am already using TCFS but I'm not very happy with it for several reasons: it is slow, I experienced some data loss, must use the login password, cannot share encypted files with other users, NFS - increses security riscs. And the people in Italy seemed to have stoped work on this project (latest version is dated february).

February doesn't seem that old. Are you sure you're using the latest TCFS (v 2.0.1)? You can find that at: http:// pegaso.globenet.it (which is a web form leading to an HTTPS page -- so use and SSL capable browser to get there).

If you find it slow than any other decent encryption is also likely to be too slow for you. You could look at http:// www.replay.com (in the Netherlands). This has the best collection of cryptography software I've seen anywhere.

The two fs level alternatives to TCFS are CFS (Matt Blaze's work, on which TCFS was based) and userfs (which support a few different user-level filesystem types including an experimental cryptographic one.

O'REILLY RELEASES THE SECOND EDITION OF TCP/ IP NETWORK ADMINISTRATION

SEBASTOPOL, CA-TCP/IP, the preeminent communications protocol for linking together diverse computer systems, is growing in importance as the Internet and other computer networks become more central to business, government, and education. O'Reilly & Associates has released the 2nd Edition of their classic "TCP/IP Network Administration," which gives system administrators the latest information on TCP/IP, as well as thorough coverage of configuration, security, and troubleshooting.

"TCP/IP Network Administration, 2nd ed." is a complete guide to setting up and running a TCP/IP network. It starts with the fundamentals: what the protocols do and how they work, how addresses and routing are used to move data through the network, and how to set up your network connection. Beyond basic setup, this new second edition discusses the Internet routing protocols and provides a tutorial on how to configure important network services. It also includes a new section on Linux in addition to BSD and System V TCP/IP implementations.

As a User Group member, you are entitled to a 20% discount on all O'Reilly books and software when you order directly from us (800-998-9938) and mention your user group’s code [available to MUUG members from any board member – Ed.] when placing your order. O'Reilly's Catalog and Resource Center on the Web is at: http:// www.oreilly.com/

Contact Information

To contact the MUUG board for membership information or anything else, send e-mail to board@muug.mb.ca. We have a Web presence as well, at http:// www.muug.mb.ca/, where you can find all kinds of information, including details of upcoming and past meetings and presentations and references related to them. E-mail the editor at editor@muug.mb.ca.