Technical UNIX® Users Group

January 1989 $2.50
Volume 1, Number 4

newsletter of the Technical UNIX® User Group

This month ...

Late Breaking News...
Next Meeting to be held at UNISYS
See inside for details

Editor's Note

by Darren Besler, Editor

Happy New Year! I hope everyone survived the holiday season without indulging too much. I find that the holiday season always seems to sneak a few pounds in here or there. It couldn't be all that baking, could it? I think it's the turkey! Well, at least it only comes around once a year. I will certainly have to renew my gym membership now.

Anyhow, this month we have an article/report describing the "Internet Worm". The Internet Worm is the name given to that virus that made national news coverage a little while back. This report came via the the unix-wizards electronic mailing list that I am a member off. It gets quite detailed on how the worm was spread throughout the Internet network. For those members that are more interested in this topic, or don't understand the method of attack, we may be able to set up a meeting that will discuss this topic in more detail.

Also, in this issue is an article on password aging by Susan Zuk. This is very appropriate, as it provides some additional information to last months article on userid creation, and makes a nice follow up for the article on the Internet Worm.

I would like to thank Bob Page, Susan Zuk, Gilbert Detillieux, and Ken Wilkie for their contributions to this months newsletter. See you at the next metting!

Group Information

The Technical Unix User Group meets at 7:30 pm the second Tuesday of every month, except July and August. The newsletter is mailed to all paid up members 1 week prior to the meeting. Membership dues are $20 annually and are due at the October meeting. Membership dues are accepted by mail and dues for new members will be pro-rated accordingly.

The Executive

President:	Gilbert Detillieux	261-9146
Vice President:	Vacant	
Treasurer:	Gilles Detillieux	261-9146
Secretary:	Susan Zuk	(W) 786-8483
Newletter Editor:	Darren Besler	(W) 934-5475
		(H) 254-3392
Membership Sec.:	Pat Macdonald	(W) 474-9870
Information:	Gilbert Detillieux	261-9146
	Susan Zuk	(W) 786-8483

Technical UNIX User Group
P.O. Box 130
Saint-Boniface, Manitoba
R2H 3B4

Copyright Policy and Disclaimer

ANNOUNCEMENT...

Meeting Location Change:
For January's meeting only, we will be gathering at UNISYS, 300 - 1661 Portage Avenue. In order to have an idea of the number of people to expect please RSVP if you are planning on coming out to the meeting. This can be done by phoning Susan Zuk at 786-8483 and leaving a message that you will be attending the meeting. This can be done up until 5:00 pm January 10, 1988.

President's Corner

by Gilbert Detillieux, President

A new year is upon us now! At this time of year, we are filled with hope and expectation for what may lie ahead. Will this be the year that UNIX really breaks through to become the dominant operating system? Will our group continue to grow? Will we finally see one standard UNIX? (Well, let's not get carried away with our hopes...)

With the new year also comes the time where we each sit down and look at our past accomplishments, and then set out to improve ourselves by striving to reach certain new goals in the year ahead. Yes, it's time for us to make our new year's resolutions, keep them in mind for the next few hours, and then proceed to break each one of them throughout the year.

And so, with this self-analytical attitude in mind, your humble president now sits down to itemize his resolutions, with a half-hearted intent to keep them:

1. I will learn at least one new UNIX utility every month. At that rate, I will have mastered UNIX by the time OS/2 applications start to appear.
2. I will try to say something nice about VMS this year.
3. I will write at least one column this year that won't degenerate into a lot of editorial commentary.
4. I will not make any snide remarks about IBM's role in the Open Software Foundation.
5. I promise not to gloat too much when John C. Dvorak (Author and UNIX-basher in PC Magazine and DEC Professional magazine) admits to being wrong about the future of UNIX.
6. I promise not to lie about my student status in order to buy a NeXT Workstation.

According to the latest issue of DEC Professional (Dec. 1988), the ANSI C Committee has voted the draft standard as final. Once all i's are dotted and t's crossed, all X3 voting members vote on the final documents, and the standard could be approved by ANSI in March 1989. Bless their bureaucratic hearts!

For the January meeting, we'll be back at Unisys (300-1661 Portage, between St. James St. and Route 90), to continue the system administration workshop that we had in November. That meeting's discussion was focussed primarily on user ID creation and user setup.

As system administration is a very broad topic, and can easily cover several such sessions, we will try to focus in on more specific topics at each session. At any particular session, the topic or topics will be decided by a quick vote from those present. Following is a list of some of the suggested topics to cover:

1. System tuning
   a) The sar utility (system activity reporting)
   b) Kernel parameters (clists, files, buffers, etc.)
   c) Generating a new system
   d) Error reports
2. System accounting (process, login accounting, etc.)
3. Communications
   a) Serial terminals (stty, gettydefs, terminfo)
   b) Printer setup (lp, lpadmin, custom interfaces)
   c) UUCP (use, setup, HDB-UUCP)
   d) Networks (Ethernet, token ring, X.25, Usenet)
4. Security (passwords, file permissions, etc.)

Notice to non-members: This is the last newsletter you will receive, unless you sign up at the January meeting, at the latest. If you can't make it to the meeting, you can phone our membership secretary, Pat Macdonald, or any member of the executive for that matter, or you can send a $20 cheque to the Technical UNIX User Group, at our usual mailing address.

The fortune file

This months fortune comes care of Ken Wilkie.

```
If god had meant us to be nude, He would have given us bigger hands.
```

A REPORT ON THE INTERNET WORM

Bob Page
University of Lowell
Computer Science Department

November 7, 1988

Here's the scoop on the "Internet Worm". Actually it's not a virus - a virus is a piece of code that adds itself to other programs, including operating systems. It cannot run independently, but rather requires that its "host" program be run to activate it. As such, it has a clear analog to biologic viruses — those viruses are not considered live, but they invade host cells and take them over, making them produce new viruses.

A worm is a program that can run by itself and can propagate a fully working version of itself to other machines. As such, what was loosed on the Internet was clearly a worm.

This data was collected through an emergency mailing list set up by Gene Spafford at Purdue University, for administrators of major Internet sites - some of the text is included verbatim from that list. Mail was heavy since the formation of the list; it continues to be on Monday afternoon - I get at least 2-3 messages every hour. It's possible that some of this information is incomplete, but I thought you'd like to know what I know so far.

The basic object of the worm is to get a shell on another machine so it can reproduce further. There are three ways it attacks: sendmail, fingerd, and rsh/rexec.

THE SENDMAIL ATTACK:

In the sendmail attack, the worm opens a TCP connection to another machine's sendmail (the SMTP port), invokes debug mode, and sends a RCPT TO that requests its data be piped through a shell. That data, a shell script (first-stage bootstrap) creates a temporary second-stage bootstrap file called x$$,l1.c (where '$$' is the current process ID). This is a small (40-line) C program.

The first-stage bootstrap compiles this program with the local cc and executes it with arguments giving the Internet hostid/socket/password of where it just came from. The second-stage bootstrap (the compiled C program) sucks over two object files, x$$,vax.o and x$$,sun3.o from the attacking host. It has an array for 20 file names (presumably for 20 different machines), but only two (vax and sun) were compiled in to this code. It then figures out whether it's running under BSD or SunOS and links the appropriate file against the C library to produce an executable program called /usr/tmp/sh - so it looks like the Bourne shell to anyone who looked there.

THE FINGERD ATTACK:

In the fingerd attack, it tries to infiltrate systems via a bug in fingerd, the finger daemon. Apparently this is where most of its success was (not in sendmail, as was originally reported). When fingerd is connected to, it reads its arguments from a pipe, but doesn't limit how much it reads. If it reads more than the internal 512-byte buffer allowed, it writes past the end of its stack. After the stack is a command to be executed ("/usr/ucb/finger") that actually does the work. On a VAX, the worm knew how much further from the stack it had to clobber to get to this command, which it replaced with the command "/bin/sh" (the bourne shell). So instead of the finger command being executed, a shell was started with no arguments. Since this is run in the context of the finger daemon, stdin and stdout are connected to the network socket, and all the files were sucked over just like the shell that sendmail provided.

THE RSH/REXEC ATTACK:

The third way it tried to get into systems was via the .rhosts and /etc/hosts.equiv files to determine 'trusted' hosts where it might be able to migrate to. To use the .rhosts feature, it needed to actually get into people's accounts - since the worm was not running as root (it was running as daemon) it had to figure out people's passwords. To do this, it went through the /etc/passwd file, trying to guess passwords. It tried combinations of: the username, the last, first, last+first, nick names (from the GECOS field), and a list of special "popular" passwords:

aaa
academia
aerobics
airplane
albany
albatross
albert
alex
alexander
algebra
aliases
alphabet
ama
amorphous
analog
anchor
andromache
animals
answer
anthropo-
genic
anvils
anything
aria
ariadne
arrow
arthur
athena
atmosphere
aztecs
azure
bacchus
bailey
banana
bananas
bandit
banks
barber
baritone
bass
bassoon
batman
beater
beauty
beethoven
beloved
benz
beowulf
berkeley
berliner
beryl
beverly
bicameral
bob
brenda
brian
bridget
broadway
bumbling
burgess
campanile
cantor
cardinal
carmen
carolina
caroline
cascades
castle
cat
cayuga
celtics
cerulean
change
charles
charming
charon
chester
cigar
classic
clusters
coffee
coke
collins
commrades
computer
condo
cookie
cooper
cornelius
couscous
creation
creosote
cretin
daemon
dancer
daniel
danny
dave
december
defoe
deluge
desperate
develop
dieter
digital
discovery
disney
dog
drought
duncan
eager
easier
edges
edinburgh
edwin
edwina
egghead
eiderdown
eileen
einstein
elephant
elizabeth
ellen
emerald
engine
engineer
enterprise
enzyme
ersatz
establish
estate
euclid
evelyn
extension
fairway
felicia
fender
fermat
fidelity
finite
fishers
flakes
float
flower
flowers
foolproof
football
foresight
format
forsythe
fourier
fred
friend
frighten
fun
fungible
gabriel
gardner
garfield
gauss
george
gertrude
ginger
glacier
gnu
golfer
gorgeous
gorges
gosling
gouge
graham
gryphon
guest
guitar
gumption
guntis
hacker
hamlet
handily
happening
harmony
harold
harvey
hebrides
heinlein
hello
help
herbert
hiawatha
hibernia
honey
horse
horus
hutchins
imbroglio
imperial
include
ingres
inna
innocuous
irishman
isis
japan
jessica
jester
jixian
johnny
joseph
joshua
judith
juggle
julia
kathleen
kermit
kernel
kirkland
knight
ladle
lambda
lamination
larkin
larry
lazarus
lebesgue
lee
leland
leroy
lewis
light
lisa
louis
lynne
macintosh
mack
maggot
magic
malcolm
mark
markus
marty
marvin
master
maurice
mellon
merlin
mets
michael
michelle
mike
minimum
minsky
moguls
moose
morley
mozart
nancy
napoleon
nepenthe
ness
network
newton
next
noxious
nutrition
nyquist
oceanogra-
phy
ocelot
olivetti
olivia
oracle
orca
orwell
osiris
outlaw
oxford
pacific
painless
pakistan
pam
papers
password
patricia
penguin
peoria
percolate
persimmon
persona
pete
peter
philip
phoenix
pierre
pizza
plover
plymouth
polynomial
pondering
pork
poster
praise
precious
prelude
prince
princeton
protect
protozoa
pumpkin
puneet
puppet
rabbit
rachmanin-
off
rainbow
raindrop
raleigh
random
rascal
really
rebecca
remote
rick
ripple
robotics
rochester
rolex
romano
ronald
rosebud
rosemary
roses
ruben
rules
ruth
sal
saxon
scamper
scheme
scott
scotty
secret
sensor
serenity
sharks
sharon
sheffield
sheldon
shiva
shivers
shuttle
signature
simon
simple
singer
single
smile
smiles
smooch
smother
snatch
snoopy
soap
socrates
sossina
sparrows
spit
spring
springer
squires
strangle
stratford
stuttgart
subway
success
summer
super
superstage
support
supported
surfer
suzanne
swearer
symmetry
tangerine
tape
target
tarragon
taylor
telephone
temptation
thailand
tiger
toggle
tomato
topography
tortoise
toyota
trails
trivial
trombone
tubas
tuttle
umesh
unhappy
unicorn
unknown
urchin
utility
vasant
vertigo
vicky
village
virginia
warren
water
weenie
whatnot
whiting
whitney
will
william
wil-
liamsburg
willie
winston
wisconsin
wizard
wombat
woodwind
wormwood
yaco
yang
yellowstone
yosemite
zap
zimmerman

[I wouldn't have picked some of these as "popular" passwords, but then again, I'm not a worm writer. What do I know?]

When everything else fails, it opens /usr/dict/words and tries every word in the dictionary. It is pretty successful in finding passwords, as most people don't choose them very well. Once it gets into someone's account, it looks for a .rhosts file and does an 'rsh' and/or 'rexec' to another host, it sucks over the necessary files into /usr/tmp and runs /usr/tmp/sh to start all over again.

Between these three methods of attack (sendmail, fingerd, .rhosts) it was able to spread very quickly.

THE WORM ITSELF:

The 'sh' program is the actual worm. When it starts up it clobbers its argv array so a 'ps' will not show its name. It opens all its necessary files, then unlinks (deletes) them so they can't be found (since it has them open, however, it can still access the contents). It then tries to infect as many other hosts as possible - when it sucessfully connects to one host, it forks a child to continue the infection while the parent keeps on trying new hosts.

One of the things it does before it attacks a host is connect to the telnet port and immediately close it. Thus, "telnetd: ttloop: peer died" in /usr/adm/messages means the worm attempted an attack.

The worm's role in life is to reproduce - nothing more. To do that it needs to find other hosts. It does a 'netstat -r -n' to find local routes to other hosts & networks, looks in /etc/hosts, and uses the yellow pages distributed hosts file if it's available. Any time it finds a host, it tries to infect it through one of the three methods, see above. Once it finds a local network (like 129.63.nn.nn for ulowell) it sequentially tries every address in that range.

If the system crashes or is rebooted, most system boot procedures clear /tmp and /usr/tmp as a matter of course, erasing any evidence. However, sendmail log files show mail coming in from user /dev/null for user /bin/sed, which is a tipoff that the worm entered.

Each time the worm is started, there is a 1/15 chance (it calls random()) that it sends a single byte to ernie.berkeley.edu on some magic port, apparently to act as some kind of monitoring mechanism.

THE CRACKDOWN:

Three main 'swat' teams from Berkeley, MIT and Purdue found copies of the VAX code (the .o files had all the symbols intact with somewhat meaningful names) and disassembled it into about 3000 lines of C. The BSD development team poked fun at the code, even going so far to point out bugs in the code and supplying source patches for it! They have not released the actual source code, however, and refuse to do so. That could change - there are a number of people who want to see the code.

Portions of the code appear incomplete, as if the program development was not yet finished. For example, it knows the offset needed to break the BSD fingerd, but doesn't know the correct offset for Sun's fingerd (which causes it to dump core); it also doesn't erase its tracks as cleverly as it might; and so on.

The worm uses a variable called 'pleasequit' but doesn't correctly initialize it, so some folks added a module called _worm.o to the C library, which is produced from:

```
            int pleasequit = -1;
```

the fact that this value is set to -1 will cause it to exit after one iteration.

The close scrutiny of the code also turned up comments on the programmer's style. Verbatim from someone at MIT:

> From disassembling the code, it looks like the programmer is really anally retentive about checking return codes, and, in addition, prefers to use array indexing instead of pointers to walk through arrays.

Anyone who looks at the binary will not see any embedded strings - they are XOR'ed with 81 (hex). That's how the shell commands are imbedded. The "obvious" passwords are stored with their high bit set.

Although it spreads very fast, it is somewhat slowed down by the fact that it drives the load average up on the machine - this is due to all the encryptions going on, and the large number of incoming worms from other machines.

[Initially, the fastest defense against the worm is is to create a directory called /usr/tmp/sh. The script that creates /usr/tmp/sh from one of the .o files checks to see if /usr/tmp/sh exists, but not to see if it's a directory. This fix is known as 'the condom'.]

NOW WHAT?

None of the ULowell machines were hit by the worm. When BBN staffers found their systems infected, they cut themselves off from all other hosts. Since our connection to the Internet is through BBN, we were cut off as well. Before we were cut off, I received mail about the sendmail problem and installed a patch to disable the feature the worm uses to get in through sendmail. I had made local modifications to fingerd which changed the offsets, so any attempt to scribble over the stack would probably have ended up in a core dump.

Most Internet systems running 4.3BSD or SunOS have installed the necessary patches to close the holes and have rejoined the Internet. As you would expect, there is a renewed interest in system/network security, finding and plugging holes, and speculation over what will happen to the worm's creator.

If you haven't read or watched the news, various log files have named the responsible person as Robert Morris Jr., a 23-year old doctoral student at Cornell. His father is head of the National Computer Security Center, the NSA's public effort in computer security, and has lectured widely on security aspects of UNIX.

Associates of the student claim the worm was a 'mistake' - that he intended to unleash it but it was not supposed to move so quickly or spread so much. His goal (from what I understand) was to have a program 'live' within the Internet. If the reports that he intended it to spread slowly are true, then it's possible that the bytes sent to ernie.berkeley.edu were intended to monitor the spread of the worm. Some news reports mentioned that he panicked when, via some "monitoring mechanism" he saw how fast it had propagated.

A source inside DEC reports that although the worm didn't make much progress there, it was sighted on several machines that wouldn't be on its normal propagation path, i.e. not gateways and not on the same subnet. These machines are not reachable from the outside. Morris was a summer intern at DEC in '87. He might have included names or addresses he remembered as targets for infesting hidden internal networks. Most of the DEC machines in question belong to the group he worked in.

The final word has not been written - I don't think the FBI have even met with this guy yet. It will be interesting to see what happens.

Password Aging

by Susan Zuk, Secretary

At the November system administration workshop, a method of requiring users to periodically change their passwords was mentioned. Some of the workshop participants were interested in this topic so the following is a description of how to implement the facility.

Password aging is created and controlled by the super user. It forces and also restricts changes to passwords. The procedure is handled through the *passwd* file which is found in the /etc directory. The aging information is added to the password field of each login-id definition (Each line in the file holds the information required for a user to log into their system). The following is an illustration of a *passwd* file and some samples of password aging:

```
user1:nhsk0ujsksiiwj,7/:User 1:/usr/acct/user1:/bin/sh
user2:jhdksh768hdjsh,12:User 2:/usr/acct/user2:/bin/sh
user3:hjdJJU8u5us6gh,..:User 3:/usr/acct/user3:/bin/sh
user4:d89hf7hfdd6gHs:User 4:/usr/acct/user4:/bin/sh
```

Notice the first three lines include a comma in the encrypted password field (the area after the first colon). This states that password aging has been activated. The final line, user4, is a user without password aging.

The first character following the comma states the maximum number of weeks before a change is required. The second character following the comma defines the minimum number of weeks before a change is allowed. The minimum change prevents a user from changing their password and immediately changing back to the same one.

The values of the password aging characters are found in the following table:

Password Aging Values

.	0	E	16	U	32	k	48
/	1	F	17	V	33	l	49
0	2	G	18	W	34	m	50
1	3	H	19	X	35	n	51
2	4	I	20	Y	36	o	52
3	5	J	21	Z	37	p	53
4	6	K	22	a	38	q	54
5	7	L	23	b	39	r	55
6	8	M	24	c	40	s	56
7	9	N	25	d	41	t	57
8	10	O	26	e	42	u	58
9	11	P	27	f	43	v	59
A	12	Q	28	g	44	w	60
B	13	R	29	h	45	x	61
C	14	S	30	i	46	y	62
D	15	T	31	j	47	z	63

Let's take a look at what the values mean for user1, user2, and user3. User1 shows 7/ after the encrypted password and comma. This indicates that the user can change their password after the first week up to a maximum of 9 weeks. The value 7 corresponding to 9 weeks and the value / corresponding to 1 week, as displayed in the above table.

Whenever there is a situation with the minimum value being greater than the maximum value only the super user can change the password. For example, user2 has a maximum value of 3 weeks (can change the password up to 3 weeks) but a minimum value of 4 weeks (the password is not allowed to be changed until 4 weeks have passed). The user can never reach the point where he/she can change the password.

User3 shows a special case. The two periods indicate that the user can change the password upon the first login (when first given a userid). After this first session the periods are removed and password aging does not exist.

When the user logs into the system after expiration has occurred, the system displays a message requesting for a new password. The message reads as follows:

```
Your password has expired. Choose a new one.
Changing password for user3
New password:
```

Password aging assists the system administrator in ensuring that specific security guidelines are maintained. Passwords are kept current for all system users by allowing the system administrator to specify various password time limits, or by modifying the parameters as required.

Technical UNIX® User Group

Agenda

for

Tuesday, January 10, 1989

7:30pm

UNISYS

Canadian Indemnity Building

300-1661 Portage Avenue

1. Round Table	7:30
2. Business Meeting	8:00
a) Minutes of November's Meeting	
b) Membership Secretary's Report	
c) Newsletter Report	
d) Treasurer's Report	
3. Break	8:30
4. Presented Topic	8:40
System Administration Workshop	
a) Resource Accounting	
b) System Tuning	
c) Backup/Restore Procedures	
5. Adjourn	9:30