AUUGN

The Journal of AUUG Inc.
Volume 24 • Number 2
June 2003

Features:

Microsoftens	10
OpenBSD 3.3 Released	11
Linux System' Administrator's Security Guide (Part 1)	15
The Roadman for FreBSD 5-stable	20
Mail and Dynamic IP	24
Intrusion Detection System (Part 1)	25
FreeBSD 5.1 Release Process	40
SCO-vs.-IBM: the Open Source Initiative Position Paper on the Complaint	41
AUUG 2003 Systems Administration Symposium Photo Gallery	57
NetBSD 1.6.1 CD	58

News:

Public Notices	4
AUUG: Corporate Members	9
AUUG Conference 2003 Invitation	59
AUUG Membership Renewal	61
AUUG: Chapter Meetings and Contact Details	63

Regulars:

President's Column	3
My Home Network	4

AUUG Inc

ISSN 1035-7521

Print post approved by Australia Post - PP2391500002

Editorial

Con Zymaris auugn@auug.org.au

I imagine I'm not alone in expressing fond memories of both Carl Sagan and Stephen Jay Gould. Both of these gentleman were not only practicing scientists but also exemplary communicators of science and of the technical, complex and beautiful cosmos that we inhabit.

Through his combination of vision and chutzpah, Sagan caused us to pause for a moment and consider the majesty of star-stuff, of galaxies and of the human discourse which ensued over millennia in trying to reveal their secrets. Gould focused on that most amazing and subversive of ideas, Darwin's evolution through natural selection, on timescales of geological significance and on the misguidedness of Creationism. Together, they offered the laity the most infectiously interesting, mind expanding glimpse into what true science, thoughtful reasoning and nature in *toto* have to offer.

Gould once quipped that he and Sagan (who were essentially contemporaries, both growing up in the same New York burrough) were destined for Biology and Astronomy in that order, as he was short, and Sagan was tall; one looked downwards and the other up. It was this practice of folding seemingly trivial human stories and interests into big-picture hard science, which made both writers so compelling. It's what led to many non-scientists beginning to understand, perhaps for the first time, the processes by which science progresses. The randomness, the contingencies, the haphazardness complexity of it all. It's part of the wonder and it's what holds our interest. It inculcated science in hundreds of millions. And it's what we in the technical sphere of the computing realm need to learn and begin practicing.

It seems to me that many non-computer-scientists are rushing to define how we technologists should go about plying the science of computers in the 21st century. What we should and shouldn't be able to do. What becomes legal and illegal. What the tenets of proof shall be in disputed issues. What standards should be adopted. What platforms should be preferred. In all, complex issues made even more opaque for non-technologists through lack of training, lack of experience and lack of an overall *gut feel* for the technologies themselves and how they will begin to impact the future of humanity.

Why are groups like AUUG's members well placed to help bring about an increased enlightenment amongst the IT laity? To be frank, it all boils down to having the kind of mind that just gets '*it*'. Further, this change of life that is happening, this alteration in our future living patterns brought about by the proliferation of technology, is something that our group has lived through already. I extend to you a challenge: few are better placed to reach out and enlighten. Do so, for our future's sake.

Cheers, Con

Contribution Deadlines for AUUGN in 2003

Volume 24 • Number 3 – September 2003: **August 30th, 2003**

Volume 24 • Number 4 – December 2003: **November 15th, 2003**

AUUG Incorporated gratefully acknowledges the support of its corporate sponsor:

AUUGN Editorial Committee

The AUUGN Editorial Committee can be reached by sending email to: auugn@auug.org.au

Or to the following address:
AUUGN Editor

Editor:
Con Zymaris
PO Box 7071
Baulkham Hills BC NSW 2153

Sub-Editors:
Frank Crawford, Mark White

Contributors:
This issue would not have happened without the transcription and editorial efforts of Gary R. Schmidt" <grschmidt@acm.org>, Rik Harris <rik@kawaja.net>, Raymond Smith <zzrasmit@uqconnect.net>, David Lloyd <lloy0076@adam.com.au>, Peter Sandilands <peter@sandilands.vu>, Stephen Jenkin <sjenkin@pcug.org.au>, Cameron Strom <camerons@statscout.com>

Public Relations and Marketing:
Elizabeth Carroll

AUUGN Submission Guidelines

Submission guidelines for AUUGN contributions can be obtained from the AUUG World Wide Web site at:

www.auug.org.au

Alternately, send email to the above correspondence address, requesting a copy.

AUUGN Back Issues

A variety of back issues of AUUGN are still available. For price and availability please contact the AUUG Secretariat, or write to:
AUUG Inc.
Back Issues Department
PO Box 7071
Baulkham Hills BC NSW 2153
Telephone: 02 8824 9511

Conference Proceedings
A limited number of copies of the Conference Proceedings from previous AUUG Conferences are still available. Contact the AUUG Secretariat for details.

Mailing Lists

Enquiries regarding the purchase of the AUUGN mailing list should be directed to the AUUG Secretariat.

President's Column

Greg Lehey <Greg.Lehey@auug.org.au>

Over the decades, UNIX has had a number of brushes with lawyers. AUUG has been spared most problems, but our sister organization, USENIX, has had its share. As you probably know, the USENIX journal *;login:* was originally called *UNIX NEWS*. In June 1977, the group which was to become USENIX were told by an AT&T lawyer that they were not allowed to use the word UNIX, a trademark of Western Electric. This was the reason for the name change, and also for the choice of the name *USENIX* when the group finally got a name. For similar reasons, AUUG is called AUUG, which does not stand for "Australian UNIX Users Group". Well, not officially, anyway: call it what you will.

AT&T's lawyers considered even the name USENIX to be a dilution of the name UNIX. More importantly, of course, they strongly regulated the use of the UNIX code, which was initially available only to universities. When, in the early 1990s, the Computer Sciences Research Group of the University of California in Berkeley started separating their own work from "AT&T UNIX" for the purposes of distributing the network stack, AT&T lawyers were involved, but initially did not see any problems.

That changed, of course, when BSDI took this code, filled in the missing bits and started selling it as a complete operating system. Suddenly USL, the owner of the UNIX code base, knew that BSD/386 contained AT&T code. At the time it was 15 year old code from the Seventh or Sixth Editions of UNIX, but that didn't make any difference. They sued BSDI, and the case took two years before BSDI and Novell, the owner of the UNIX code base, settled out of court.

We thought all this stuff was history. But no, recently it started all over again. In March, SCO, the owner of the UNIX code base, has decided that IBM has been putting UNIX source code into Linux, and they're suing them for $1,000,000,000 US. Since then they have decided that Linux itself is full of code stolen from SCO's own UNIX System V.

My first reaction when I heard this was of utter disbelief. Is this the same company that, in January last year, released the "ancient UNIX" source code under a BSD license? Well, no, that company was called Caldera, and a lot of their good people are no longer there. But yes, Caldera has traded as SCO since in the middle of last year, and they recently formally changed their name to SCO.

There are so many reasons to find the lawsuit completely ludicrous. I've summarized some of them at http://www.lemis.com/grog/sco.html, and I'm still doing so. At the end of May, AUUG made its standpoint clear with a press release, now on our web site at http://www.auug.org.au/publications/press/SCO-stance.html. The following day revealed the most obvious reason why it's ridiculous: Novell issued the text of a letter to SCO which shot the latter down in flames: http://www.novell.com/news/press/archive/2003/05/pr03033.html. Apart from asserting their ownership of UNIX System V (SCO is only a licensee), they pointed out that they love Linux, and that SCO was only spreading FUD. That doesn't quite close the case, though: even if SCO folds in the near future, which I personally consider very possible, the accusation stands: does Linux really contain restricted UNIX code? Since the free release of ancient UNIX, it can contain code derived from that source if it wants.

I consider it pretty unlikely that there's any restricted UNIX code in Linux. For most of the code, it simply wouldn't be worth the trouble. In the case of BSD/386 ten years ago, code was found that looked pretty similar to the Seventh Edition code; it had obviously grown out of that code. But that was an omission, not a deliberate theft, and it was in a kernel that looks like UNIX. Nobody would have gone to the trouble to copy it from the old UNIX sources; it would have been easier to write it again from scratch.

In the case of Linux, there are more obvious reasons: the structure of the Linux kernel is very different from the UNIX kernel, and it would be quite difficult to import UNIX code. More importantly, though, the Linux source code is available to anybody who is interested. That includes a lot of people who have access to the UNIX source code. Any significant import would be immediately apparent.

But what if the code wandered in the other direction? Until SCO releases evidence of these breaches, it would be more plausible to assume that SCO included Linux code in UnixWare 7, for example in the Linux compatibility code. This would be one place where it would make sense to copy code, and in proprietary code there's nobody to see that it has been copied. I've seen copied open source code in another version of UNIX, and I'm sure many other of our members also have: it does happen. There's no proof, not even an accusation, that SCO have done that, but it certainly would explain why they are reticent to show the places where they allege it has happened. This is one more reason why they must show which code has been copied.

Public Notices

Upcoming Conferences & Events

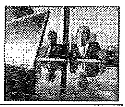
SAGE-AU
Systems Administration Conference
August 4 - 8, 2003
Hobart

AUUG 2003
September 3-5, 2003
Sydney
(More information later in this issue!)

BSDCon
September 8-12, 2003
San Mateo, CA

LISA
17th Systems Administration Conference
October 26-31, 2003
San Diego, CA

ALLOCPSA



Online Professional Services Automation

allocPSA is a suite of integrated applications (a Professional Services Automation suite) designed for services-based organisations. It enables services personnel to become more productive and profitable by improve their efficiency through increased utilisation and productive time, better planning and through integrated knowledge management.

allocPSA is a complete organisation-wide business solution that attempts to integrate all practice groups and functions in a professional services company into a single computer system. allocPSA consists of numerous software modules for business areas such as resource planning, project management, time and expenses, integration with existing invoicing, book-keeping and payroll systems, faults, messages, announcements, reminders and knowledge management, collaboration, services supply chain, human resources and staff skilling and management of cost-centres.

allocPSA offers total integration between the modules, as well as an open architecture and integration with your existing processes and software. allocPSA is an online, web-based suite, and as such, can be deployed to dozens of users with zero client-side installation, within your Local Area Network, Wide Area Network or as an extranet application to authorised personnel and partners from anywhere on the Internet

allocPSA is supplied as a self-contained server appliance, which contains the operating system platform, SQL server, web server and core allocPSA applications.

allocPSA has been designed to offer your firm the ultimate in flexible deployment integration and ongoing control. To that end, the complete source code for allocPSA is supplied under the GPL Open Source licence. You are then able to extend, modify or maintain this code, should you so wish, or hire 3rd parties (Cybersource, or others) to undertake this work for you.

The fee schedule for allocPSA includes a pre-purchased quanta of installation, customisation and support services. A fully-functioning live demo is available upon request.

Web: http://www.cyber.com.au/cyber/product/allocPSA/
Phone: +61 3 9621 2377 Mail:info@cyber.com.au *Cybersource*

My Home Network (June 2003)

By: Frank Crawford <frank@crawford.emu.id.au>

[Editor's note: Frank's column was held over from last month due to time constraints.]

A new year, a new column, but old, old problems. If you have read my recent columns, you would know that I've been upgrading my main systems over the last few months. Generally, this has gone smoothly, but, for the first time in my home network, I've had a disk fail and managed to lose data. To make matters worse, most, if not all, the data was recoverable, if I had sufficient space.

Going into more details, the problem occurred just after Christmas, when I attempted to defragment the Windows XP partition on my fastest system (2.56GHz Pentium 4, 533MHz FSB, 512MB PC2700 SDRAM, 80GB 100MB/sec UDMA Disk, etc). After leaving it to run overnight (it usually takes an hour or two), it was still only 30% of the way through, and not progressing! At that time, I decided to stop it and examine the system a bit more, as my first inclination was that it was a problem with the new system. Unfortunately, nothing worked, not STOP, not abort out of the Task Manager, nothing. Quite obviously, something was wrong. Anyway, after pushing the big "red-button" (or silver in this case), Windows would not come up, due to missing drivers, DLLs, etc. Time to do a system recovery. :-(I've actually had to do this a few times, as Windows XP has a number of issues with changing the IDE controller, which happens when you upgrade the motherboard. While this does reinstall the system drivers, it doesn't affect user files or programs. As I found out later, it also does such "handy" things as remove various system recovery files, and other system saved images (but more on this later).

So out with the install disks, go through the install process, wait the required 40 mins, answer the same questions, etc, etc, and the system should be right. Wrong, after all this the system failed to display anything on the monitor. In addition, it seemed to have problems with remote access. Hmmm, I think, maybe there is something more seriously wrong here. (In case you haven't realised it, my original thought was that the defrag had moved something important, and I'd interrupted it at just the wrong time.) So, before anything else, I decided to do a backup of the Windows partition. This is however, easier said than done. The disk is 80GB, the Windows XP partition is 40GB, and even more importantly, it is an NTFS partition, and I didn't have 40GB spare anywhere else. In fact, the data on the disk only took about 20GB, and if I tried I might be able to find space for that. So out I come with Norton Ghost, and try to dump the data to another 20GB disk. At this point the real problem comes out, as part way through Ghost complains about read errors on the disk! The disk has somehow developed some bad blocks, and in the middle of some crucial Windows files, at that.

Well, now I decide to really start looking at what is going on, and my first step is to reboot into Linux, which is also on this system. So, being from the old school, and doing the simplest things first, I just ran a `dd' across the disk, and sure enough, the Linux kernel spits out disk error messages in the Windows partition. Now, why didn't Windows do this??? Anyway, back to reports. Disks for some time have had some form of self test capability, called SMART, and one of the recent items in the kernel-utils package is `smartd' and `smartctl', to monitor and report on the disk. So, kicking that off (and it did have some issues) it agreed that there were problems with the disk.

Now, I haven't ever had any disk problems with my home network, although at work I had problems some time back. To correct this I just mapped out the bad blocks and continued on. Worst case would have been to do a low-level format of the disk. Now, all this was for SCSI disks, but I wouldn't expect that much difference for IDE. I couldn't have been more wrong. Firstly, for retail IDE disks, there is no capability for bad-block mapping, no low-level format utilities, no support of any type for handling media errors. The only option is to return the disk under warranty. Even this is a bit of an issue, as recently Seagate has changed the warranty period from 3 years to 1 year for new disks.

As the disk was nearly 9 months old, I had to return it directly to Seagate, after obtaining an RMA number. To do this I had to download the SeaTools Disk Diagnostics from Seagate's site (http://download.seagate.com/seatools/), run it and generate a diagnostic report. There are a number of different versions of this, including Windows, Linux and stand-alone (i.e. DOS) versions. I chose the stand-alone version, so I could check all my Seagate disks (an advantage of standardising). The procedure was basically, download, run the self-extracting executable to create a bootable floppy, boot floppy and run the test. While there are a number of test levels, in this case the simplest, which is reading the SMART results from the disk, verified the fault and generated a failure report to go with the RMA. To obtain an RMA, I connected to http://www.seagate.com/support/service and selected "Drive Return Procedure" item 3 (RMA). Once this was done, I was in a position to return the disk, but I was told that it could take up to six weeks for a replacement disk to be returned (Seagate return an equivalent disk, and then repair the faulty disk which returns to their spares). In the end, it only took three weeks, and I could track it via Seagate's Website.

Now, given that I would lose all the data on the disk I determined to copy everything I could. I borrowed a 40GB disk and had an old 20GB disk around, so I set about trying to squeeze things onto the 60GB of space. Given that I prefer to only have a single Windows partition (now running NTFS) it was obvious that it would take most of the 40GB disk. On the other hand, Linux spreads across multiple partitions and disks happily, so it can use some of the 40GB disk and the 20GB disk.

To copy the Windows XP partition, all I could really use was Norton Ghost, and while it had problems earlier with the bad blocks, there are options to continue past them. So firstly, copy the partition to a new disk. Next, copy the Linux partitions to the new disks, swap disks and boot. Now, while Ghost can copy Linux ext2/ext3 partitions, it does seem to have some issues. I much prefer to use dump/restore to create a new partition. In fact, I've done this so often that I've written a little script to do the copy.

```
#! /bin/sh

FROMDEV=hda
TODEV=hdb
FSTYPE=ext3
MNTDIR=/mnt/mnt

[ -z "$1" ] && set -- `sed -n -e
'/'$FSTYPE'/s/^\/dev\/'$FROMDEV'\([0-9]*\)
.*$/\1/p' /etc/fstab`

for i in "$@"
do
    fs=`awk "/^$FROMDEV$i / { print \\\$2 }"
/etc/fstab`
    echo "Creating /dev/$TODEV$i"
    mkfs -t $FSTYPE /dev/$TODEV$i
    tune2fs -i 6m -c `expr 20 + $i` -L "$fs"
/dev/$TODEV$i
    echo "Copying $fs"
    mount -t $FSTYPE /dev/$TODEV$i $MNTDIR$fs
    dump -0f - /dev/$FROMDEV$i | (cd $MNTDIR$fs ;
restore -rf -)
    rm -f $MNTDIR$fs/restoresymtable
done

# Now update the grub boot block configuration
grub_shell=/sbin/grub
grubdir=/boot/grub
grub_prefix=$grubdir

force_lba=""

$grub_shell --batch <<EOF
device (hd0) /dev/$TODEV
root (hd0,0)
setup $force_lba --stage2=$grubdir/stage2 --
prefix=$grub_prefix (hd0)
quit
EOF

exit 0
```

Now this script does some things the way I want, in particular setting the forced check of the disk to once every 20+ mounts or 6 months. More importantly, `grub', the boot loader of choice for Red Hat 8.0, is a royal pain to install on anything other than the current boot disk. The parts of the scripts that relate to `grub' come from some hunting on the grub maintainers mailing list (see Google to find it) and I'm not 100% sure that it works fully. If you want to get into more fun with `grub', try it with mirrored system disks. You need something like this repeated multiple times, and the standard script, `grub-install', doesn't know what to do with "md" devices. I should add that I did modify this script a little for my copy, as I was going from hda to partitions on both hdb and hdd, but I will leave that as an exercise for the reader.

Anyway, after all this, Linux was copied and running on the new disks, Windows was copied, but still broken on the new disks and I was ready to ship off the faulty disk.

Now back to Windows XP not working. It was obvious that something was wrong with the system, and with everything built into the system registry it is not an easy thing to fix. Just as much of a problem was that the filesystem for my Windows XP system is now NTFS, which is not writable by anything except Windows. (Well there is experimental write support in Linux, but it looks very, very dangerous.) I did this both because I believe that NTFS is a far superior file system to FAT, and because FAT32 filesystems isn't supported for a size greater than 32GB (even though it can be done).

There were a couple of steps I took to try and rectify the problem. Firstly I installed the Windows XP Recovery Console on the disk, which allowed me to access the system to both read and write files. I ended up finding where Windows stored the Registry Files (in "%SystemRoot%/System32/Config"), and where it stored backup copies ("%SystemRoot%/Repair"). However, I also discovered that there were few tools to edit it, and that the saved versions were overwritten by the Recovery Install I'd done right back at the start.

One feature of the Recovery Console (not to be confused with the Recovery Install) is that it can list, enable and disable system devices and services, so you can try and recover from problems with bad devices, etc. So, with a bit of paper and a lot of time, I stopped most items that looked like they could be problems. After a little of this I decided to see if there really was a problem with the hardware, and installed a second copy of Windows XP on the same disk but in a different directory. This worked fine, and everything was okay in this implementation, except that most of the programs I wanted to use, were not available. To go any further with this installation I would have to reinstall everything!

This new installation did give me one additional capability, it made it much easier to copy files in and out of the disk. It also gave me a working registry to compare the old system with. The only thing was, I didn't have any tools to examine the broken registry with, or so I thought!

Tools to do offline editing of the Windows Registry appear to be very rare, and most things I found that mentioned it cost money and yet didn't seem to do what I wanted. However, as it turned out, there was one utility I already had that would do just what I wanted, and yet it took me two weeks to find it.

Recently I downloaded an emergency package for Linux call `mkCDrec', which allows you to create an emergency recovery CD for your Linux system. It includes the ability to back up the system, perform remote restores and other tasks. The package can be obtained from http://mkcdrec.ota.be/, and is currently up to version 0.7.1 (released March 2003), although I only have 0.6.7 at present. The mkCDrec project does not try to do everything itself, but rather makes use of other tools to do the real work.

Now in addition to the basic system there is a additional package of useful utilities that can be downloaded to supplement the package. This includes partition editors (`parted'), salvage tools (`recover', `e2salvage', etc), memory testers and, most important for me, an offline NT password editor, `chntpw'. In fact `chntpw' allows far more than just changing passwords, it can be used to make offline changes to Windows Registry files. Obviously, since Microsoft don't supply any documentation about the registry structure, most of the details have been found through trial and error, and as such, it should be used at your own risk. The home for this utility seems to be
http://home.eunet.no/~pnordahl/ntpasswd/
and also seems to have a bootable disk for accessing NTFS filesystems.

However, in my case this wasn't necessary, as I copied the registry files and worked on them on the Linux system, rather than trying anything directly on the Windows partition. Unfortunately, while I could look at everything, I could find nothing that seemed to be wrong.

Ultimately, after a few weeks of poking and prodding and trying different things, none of which worked, I decided to give up and reinstall from a very old image I had, from well before the upgrade to Windows XP. I then followed this by an upgrade to Windows XP and all the relevant software (or at least those I could remember :-)).

At this point you would think I would be out of the woods, but no. My replacement disk had arrived back and I went to install it (after checking that it was good ;-)). Again Linux copied fine, and again Windows XP was another matter.

Now, this time I was sure something strange was going on, since everything was running on the other disk, and yet things just seemed to hang up with the new disk. After playing around a little bit more, I stumbled on the issue, at least for this copy. For some reason, even though I followed all the restrictions in copying, Windows had decided that the new disk was drive D: and a number of the drivers not surprisingly referred to drive C:. After a week of searching and reading, on the Symantec site I tracked down an option to clear the drive signature that Windows XP (and 2000) puts on each disk, when it copies the partition. By doing this, it forces Windows to "reinstall" the disk, and correctly identify it as a new C: drive. Of course the obvious question after this second problem is, "was this the same problem that caused me to reinstall?". Unfortunately, I can't answer that, as I no longer had a copy of the original system. The problems were a little different, but that may just have been different manifestations of the same problem.

So, after a month or more of installation, reinstallation, changing and exploring, what did I learn. Firstly, Unix/Linux systems are much simpler to debug and diagnose problems. Under Windows, most critical information is either in binary files, with no easy tools to process in an emergency, or very basic debug output is written to files in whatever fashion the original programming team deemed fit. For Unix most initial data is in ASCII files and in standard locations. Even more, because it is so common, it is a fairly standardised format.

Secondly, even at the most basic level, most tools are available for use, and work as they do when the system is fully configured. As well, the lack of reliance on a GUI makes much more information available and written to the console during boot and running. In fact the continuing messages written to the console is something that Windows can't compete with, during debugging. I should note that Windows XP has the capability of writing debugging information to one of the serial ports, but this requires far more support for it.

The next thing I should have learnt, but haven't is the value of backups. In fact, as a professional System Administrator, I know the value of backups, and generally ensure that I can restore just about everything, but for my home systems, I have some major financial constraints. As I said at the start, this is the first disk failure I have suffered, and at the same time, I have a lot of storage available through out the network. Disk sizes are growing rapidly, but backup media is not. At home I have a QIC tape drive capable of backing up around 250-425MB (uncompressed) per tape and a CD writer that again can take about 700MB. At the same time I have over 250GB of disk storage available to me.

Assuming that I am only using 50% of the disk space I would need over 150 CDs to back it all up. Even moving past the number of CDs, an issue comes up with the software. While I can use dump/restore or some other related product for Linux, I have to find something that will work for Windows. Not an easy ask. And then after that we would get to the Macintosh's I still have around the place.

To be honest, I will probably look at a DVD writer, which would reduce the disks needed, but still only by a factor of 4 or 5, and still at a cost of around $100-$200 for media (not counting the cost of the writer). In fact I might consider a major shift in my backup strategy and purchase a large slow disk, e.g. >140GB and use it for full backups and then possibly doing incrementals to CDs or DVDs.

Finally, I really do need to investigate more tools to handle such disaster situations. I certainly intend to look further into mkCDrec, as it looks very useful, and would fit the strategy I have, but I'm also interested in others.

So, what do others do for their home network? What other disasters have you had and had to recover from. Let me know, or better still, write it up and let everyone know.

FOR SALE: HP A Class Servers

I have 5 servers as follows these are all new

HP9000 series A400 rack mount UNIX servers with dual RISC processor capability. The Hewlett Packard HP 9000 A-Class A400 and A500 servers are compact 2U rack mount, entry level 1- or 2-way processor platforms tuned for 24x7 Internet business applications. A-class servers are ideal for deploying as low cost Internet applications, firewalls, Web-hosting, and branch office application solutions. Features 1-way RISK processor configurations, optional second CPU. Available Base 256MB main memory upgradeable to 2GB (A400) 2 internal disks, maximum internal disk capacity 30GB, (max 72 GIGA BYTE)13TB external disk capacity UltraSCSI, Ultra2 SCSI, 100Base-T LAN, RS-232 ports for console and UPS connections, 10Base-T support LAN, and web-based console Supported Operating Systems HP-UX 11 (64-bit), Linux

Price $3,300.00 Rec Retail $6500.00
If you buy without HDD then $2800.0 each
Available 5 units

Please email khalid: cisco@myphone.com.au
Phone 02-9713-7999 Fax 02-9705-2255

MY PHONE PTY LTD
ACN: 063-150-041
15 McCullouch St
RUSSELL LEA NSW 2046

EducationaLinux 2004

Author: *David Lloyd <lloy0076@adam.com.au>*

The Open Source Education Conference

Introducing the Conference

On January 12 and 13 (2004), the University of Adelaide in South Australia will be buzzing with a group of dedicated open source professionals and students to discuss one seemingly simple topic:

Open Source in Education

This conference presents an opportunity for anyone working in or a part of the education system, be that as a teacher, lecturer or student, to get together, share ideas and network with other like-minded individuals using or promoting open source in education.

We can already confirm an on-site visit to the Adelaide Institute of TAFE, a presentation by Kathryn Moyle, Ph.D. from the Department of Education and Children's Services and Matthew Geddes who, amongst other things, has managed a private school network of more than 300 computers.

Please visit our web-pages at:

- http://educationalinux.adam.com.au

Registration, Venue and Logistics

This conference will be held as a mini-conference of Linux Conference Australia 2004. Registering for L.C.A. 2004 will give you the opportunity to attend the EducationaLinux 2004 conference.

Most of the conference will be held at the University of Adelaide however there will an on-site tour and presentation given by the Adelaide Institute of TAFE. It is important that you RSVP to David Lloyd (lloy0076@adam.com.au) as a light luncheon will be provided as part of the on-site tour.

Accommodation at the St Mark's University College may be organised. For more information and to register for the L.C.A. 2004, visit:

- http://lca2004.linux.org.au/

Call for Papers!

There are still opportunities for you to present a paper related to the conference's aims. EducationaLinux 2004 is seeking papers that:

- demonstrate the successful use of open source in an educational setting
- demonstrate techniques that use open source products as educational tools
- demonstrate the cost (or lack of cost) or using open source products in comparison to other products

Your paper may discuss the use of open source tools by students or demonstrate their use in a school's (or educational institution's) administrative area.

Paper presentations should aim for 30 minutes with 10 minutes question time. Other formats, such as a workshop or led discussion, may be available.

To submit a paper, e-mail:

- David Lloyd <lloy0076@adam.com.au>

We'd Like to Thank...

EducationaLinux would like to thank the speakers who have already volunteered their time for the conference, Adam Internet for providing our web-hosting and Recall Design for developing our excellent web-pages.

Linux.conf.au 2004

Linux Australia proudly announces that preparation for Australia's National Technical Linux Conference, Linux.Conf.Au 2004 (http://lca2004.linux.org.au) is well underway to be held in Adelaide in January 2004 - organised by LinuxSA (http://www.linuxsa.org.au).

Linux.conf.au is a national conference held under the auspices of Linux Australia Inc (http://www.linux.org.au/). It has a very high standing in the international community for being very technically focused, yet having a relaxed conference schedule. Along with Linux Congress (http://www.linux-kongress.org) and the Ottawa Linux Symposium (http://www.linuxsymposium.org/2003/), Linux.Conf.Au is one of the premier grass-roots, volunteer-run, Linux and Open Source conferences run anywhere in the world today.

Speakers

A number of well-known identities in the Open Source / Free Software communities have already given an indication to speak at the conference, continuing Linux.Conf.Au's strong tradition of attracting quality speakers. The speakers that have provided an initial indication of attendance are[1]:

- Havoc Pennington (GNOME, Red Hat)
- Keith Packard (XFree86)
- Rasmus Lerdorf (PHP)
- Jon "maddog" Hall (Linux International)
- Bdale Garbee (Debian)
- Andrew Tridgell (Samba)
- Paul "Rusty" Russell (Linux Kernel)
- ...and there's more to still be announced!!!!

To supplement these invited speakers, a Call for Papers (CFP) will be made RealSoonNow, to invite submissions from others in the Open Source / Free Software community to be involved in this event.

Mini-confs

Expanding the main conference are our "pre-conferences", known as a collection of "mini-confs". These are informal sessions where conference attendees can meet prior to the main conference to talk on topics of common interests. Currently there are 5 planned mini-confs: Debian, Linux Audio, EducationaLinux, IPv6 and Python. Please see http://lca2004.linux.org.au/miniconf.cgi for further details.

Partners' Programme

In an attempt to make attendance at all of Linux.Conf.Au's events easier for conference delegates, we have introduced a Partners' Programme to provide activities for partners and their families. Currently we are conducting a survey for potential partner programme attendees to fill in to help the conference in its organisation of this event. If you believe your partner may make use of such a facility, could you please have your partner fill in our survey at http://lca2004.linux.org.au/partners.cgi

More Information

The latest news about Linux.Conf.Au 2004 can be found at our website: http://lca2004.linux.org.au

In addition, a low-volume, no-spam email list has been made available so conference related announcements can be communicated. Please subscribe to this service at http://lists.linux.org.au/listinfo/lca-announce

[1] These speakers in good faith have indicated that they will speak at the conference. Some changes in the line-up of speakers may occur closer to the event due to unforseen circumstances.

AUUG Corporate Members

as at 1st June 2003

- ac3
- ANSTO
- ANU
- Apple Computer Australia Pty Ltd
- Aust Centre for Remote Sensing
- Australian Bureau of Statistics
- Australian Taxation Office
- Bradken
- British Aerospace Australia
- Bureau of Meteorology
- Cape Grim B.A.P.S.
- Computing Services, Dept Premier & Cabinet
- Corinthian Industries (Holdings) Pty Ltd
- Cray Australia
- CSIRO Manufacturing Science and Technology
- CSIRO Telecommunications & Industrial Physics
- Curtin University of Technology
- Cybersource Pty Ltd
- Deakin University
- Department of Land & Water Conservation
- Energex
- Everything Linux & Linux Help
- Fulcrum Consulting Group
- IBM
- Land and Property Information, NSW
- LPINSW
- Macquarie University
- Multibase WebAustralis Pty Ltd
- National Australia Bank
- NSW Public Works & Services, Information Services
- Peter Harding & Associates Pty Ltd
- Rinbina Pty Ltd
- St. Vincent's Private Hospital
- Sun Microsystems Australia
- Tellurian Pty Ltd
- The University of Western Australia
- Thiess Pty Ltd
- Uni of NSW - Computer Science & Engineering
- UNITAB Limited
- University of New England
- University of New South Wales
- University of Sydney
- University of Technology, Sydney
- Workcover Queensland

Microsoftens

Author: Eben Moglen

[Note: Eben Moglen is professor of law at Columbia University Law School. He serves without fee as General Counsel of the Free Software Foundation. You can read more of his writing at http://moglen.law.columbia.edu/]

There's plenty of uneasiness in Redmond, Washington, these days. Microsoft has begun to internalize the recognition that the next, and quite probably final, period of its existence will be dominated by competition with free software. That competition presents challenges the monopoly has never faced before, and already it has become necessary, at what Microsoft hopes is still an early stage in the confrontation, to take steps that no other competitor has ever had the power to force.

Competing with free software is problematic for Microsoft for many reasons. There's no company to acquire, in the first place, in order to incorporate or suppress attractive competing products - a strategy that the monopoly has pursued so often and so successfully in the past. Because free software is continually modified and improved by all its users, there's no 'evolutionary dead end' argument with which to scare customers: someone choosing to use free software is never going to be left with an unserviceable product whose maker has gone out of business, leaving the code 'orphaned' in the face of constantly shifting technology. Microsoft's implicit message to its customers has been 'We're always going to exist; our competitors, whose products you are considering, won't last forever.' But technically sophisticated corporate and governmental users now realize that the free software codebase will last indefinitely, capable of renewal and replacement for as long as its users need it. No matter how long Microsoft lasts, free software will last longer.

As I have said in this space recently, one of the hottest fields of competition at the moment is also the largest aggregate software market on earth: the world's governments. More than sixty-five countries and dozens or hundreds of political subdivisions are actively considering legislation or regulations favoring the use of free software for government computing. National governments have begun actively considering use of free software for all the reasons (price, flexibility, power of modification - in other words, freedom) that other corporate and individual users also identify. But they have at least one additional reason for adopting free software: they suspect that Microsoft has done favors for the US intelligence community, embedding 'back doors' in Windows that permit US listeners to monitor encrypted communications generated using the Windows Cryptographic Applications Program Interface (CAPI). Last month, in an unprecedented move, Microsoft announced that it will release Windows source code for review by foreign governments, to help them evaluate whether Windows is a good public purchase. In addition, Microsoft will allow foreign governments to connect their own CAPI to Windows, supposedly in order to eliminate any risk that US intelligence services have embedded spy code in the company's products.

Unprecedented as the program to show source to foreign governments and permit limited modification was, Microsoft took an even more extraordinary step in its recent filings with the US Securities and Exchange Commission. The SEC requires publicly-traded companies to file quarterly statements indicating any major changes in position since the publication of their annual reports, and disclosing any new or additional risks to their profitability. Microsoft now states that "the popularization of the Open Source movement continues to pose a significant challenge to the Company's business model, including recent efforts by proponents of the Open Source model to convince governments worldwide to mandate the use of Open Source software in their purchase and deployment of software products. To the extent the Open Source model gains increasing market acceptance, sales of the Company's products may decline, the Company may have to reduce the prices it charges for its products, and revenues and operating margins may consequently decline."

Naturally Microsoft has always found it desirable to emphasize in its SEC filings that it faces market competition: its antitrust difficulties render it eager to discuss the highly competitive nature of the software market at every opportunity. But this decision to acknowledge free software as a fundamental challenge to the Microsoft 'business model,' along with the quite specific acknowledgement that prices of its products will have to be cut, represents a watershed moment in the monopoly's history. The global investment community now has the best possible authority for taking a second look at its unquestioned belief in Microsoft's durability: the official statements of the company itself.

These new statements, of course, like all other Microsoft pronouncements on our movement, eschew two words: 'free software.' The company still cannot acknowledge the fundamental challenge to its way of doing business, which is users' need for freedom: the freedom to understand, fix, improve, and share the software that they depend on. Microsoft likes to refer to 'open source,' which sounds like something Microsoft could do itself. Its so-called 'shared source' initiatives, including the most recent one for foreign governments, are designed to mislead in precisely this way. But what Microsoft cannot do, what it will die trying to prevent us from doing, is to make software free. 'Freedom' is the word the monopolist cannot use, which is why, at the beginning of the end of the Microsoft Era, Free Software Matters.

OpenBSD 3.3 Released

Author: Todd C. Miller <*Todd.Miller@courtesan.com*>

OpenBSD 3.3 has been released and is now available from select ftp mirrors. We believe this is our best release ever with cool new features including ProPolice stack protection in the compiler (`http://www.trl.ibm.com/projects/security/ssp/`), queue support for pf (traffic shaping), an hppa (pa-risc) port, W^X for some platforms (sparc, sparc64, alpha and hppa), privilege separation in the X server and xconsole, fewer setuid binaries, a new spam deferral daemon and much, much more.

Full release announcement follows:

OpenBSD 3.3 RELEASED

May 1, 2003.

We are pleased to announce the official release of OpenBSD 3.3. This is our 13th release on CD-ROM (and 14th via FTP). We remain proud of OpenBSD's record of seven years with only a single remote hole in the default install. As in our previous releases, 3.3 provides significant improvements, including new features, in nearly all areas of the system:

Ever-improving security
`http://www.OpenBSD.org/security.html`

- Integration of the ProPolice stack protection technology, by Hiroaki Etoh, into the system compiler. This protection is enabled by default. With this change, function prologues are modified to rearrange the stack: a random canary is placed before the return address, and buffer variables are moved closer to the canary so that regular variables are below, and harder to smash. The function epilogue then checks if the canary is still intact. If it is not, the process is terminated. This change makes it very hard for an attacker to modify the return address used when returning from a function.

- W^X (pronounced: "W xor X") on architectures capable of pure execute-bit support in the MMU (sparc, sparc64, alpha, hppa). This is a fine-grained memory permissions layout, ensuring that memory which can be written to by application programs can not be executable at the same time and vice versa. This raises the bar on potential buffer overflows and other attacks: as a result, an attacker is unable to write code anywhere in memory where it can be executed. (NOTE: i386 and powerpc do not support W^X in 3.3; however, 3.3-current already supports it on i386, and both these processors are expected to support this change in 3.4.)

- Further reduction of the number of setuid and setgid binaries and more use of chroot(2) throughout the system. While some programs are still setuid or setgid, most allocate a resource and then quickly revoke privilege.

- The X window server and xconsole now use privilege separation, for better security. Also, xterm has been modified to do privilege revocation. xdm runs as a special user and group, to further constrain what might go wrong.
- RSA blinding is now on by default in OpenSSL.
- Many occurrences of strcpy(), strcat() and sprintf() have been changed to strlcpy(), strlcat(), and snprintf() or asprintf().

- A process will now have the P_SUGIDEXEC flag set if any of the real, effective, or saved uid/gid are mismatched. Previously only the real and effective uid/gid were checked.

- ptrace(2) is now disabled for processes with the P_SUGIDEXEC flag set.

Improved hardware support
(`http://www.OpenBSD.org/plat.html`)

- The xl(4), sis(4) and vr(4) ethernet drivers are more robust.

- The ahc(4) and bktr(4) drivers now work on macppc.

- Vlan tagging now works properly in the ti(4) driver.

- The cac(4) driver is now more stable.

- Media handling has been improved in the hme(4) driver.

- Bugs have been fixed in the gem(4) driver to make it more stable on the sparc64 platform.

- Several fixes for the ami(4) driver.

- New LZS compression support for the hifn(4) driver.

- Support for new IDE controllers from Promise, VIA, NVIDIA and ServerWorks.

- siop(4) driver improvements.

- The sparc64 platform is now supported on several more models and is much more stable.

Major improvements in the pf packet filter, including:

- pf now supports altq-style queueing via the new "queue" directive. Packets are assigned to queues based on filter rules. This allows for very flexible queue settings, including quality of service bandwidth shaping.

- New support for "anchors," which allows the use of sub-rulesets which can be loaded and modified independently.

- A new "table" facility provides a mechanism for increasing the performance and flexibility of rules with large numbers of source or destination addresses.

- Address pools, redirect/NAT to multiple addresses and thus load balancing.
- The scrub option 'no-df' has been changed to better handle fragments with DF set, such as those sent by Linux NFS.
- There is a new 'random-id' option for the scrub rules. This randomizes outbound IP IDs and helps defeat NAT detection.
- TCP state inspection is now RFC 763 compliant; we now send a reset when presented with SYN-cookie schemes that send out-of-window ACKs during the TCP handshake.
- TCP window scaling support.
- Full support for CIDR addresses.
- Early checksum verification return on invalid packets.
- The configuration language has been made much more flexible.
- Large rulesets now load much more quickly.

New subsystems included with 3.3

- spamd, a spam deferral daemon, can be used to tie up resources on a spammer's machine. spamd uses the new pf(4) table facility to redirect connections from a blacklist such as SPEWS or DIPS.
- OpenBSD 3.3 includes the hppa port for HP PA-RISC machines. This should be considered a work in progress; users are advised to install the most recent snapshot instead of the formal 3.3 hppa release.

Many other bugs fixed
(http://www.OpenBSD.org/plus33.html)

The "ports" tree is greatly improved
(http://www.OpenBSD.org/ports.html)

- The 3.3 CD-ROMs ship with many pre-built packages for the common architectures. The FTP site contains hundreds more packages (for the important architectures) which we could not fit onto the CD-ROMs (or which had prohibitive licenses).

Many subsystems improved and updated since the last release:

- Free86 4.2.1 (+ patches).
- gcc 2.95.3 (+ patches and ProPolice).
- Sendmail 8.12.9.
- Apache 1.3.27 and mod_ssl 2.8.12, DSO support (+ patches).
- OpenSSL 0.9.7beta3 (+ patches).
- Stable version of KAME IPv6.
- OpenSSH 3.6 (now 100% compliant with the secsh drafts).
- Bind 9.2.2 (+ patches).
- Perl 5.8.0
- Sudo 1.6.7.
- Latest ISC cron (+ patches and atrun integration).
- Improved vlan(4) robustness.
- FreeBSD emulation now recognizes newer FreeBSD ELF binaries.
- Significant improvements to the pthread library.
- Many, many man page improvements.

If you'd like to see a list of what has changed between OpenBSD 3.2 and 3.3, look at

    http://www.OpenBSD.org/plus33.html

Even though the list is a summary of the most important changes made to OpenBSD, it still is a very very long list.

SECURITY AND ERRATA

We provide patches for known security threats and other important issues discovered after each CD release. As usual, between the creation of the OpenBSD 3.3 FTP/CD-ROM binaries and the actual 3.3 release date, our team found and fixed some new reliability problems (note: most are minor, and in subsystems that are not enabled by default). Our continued research into security means we will find new security problems -- and we always provide patches as soon as possible. Therefore, we advise regular visits to

    http://www.OpenBSD.org/security.html
and
    http://www.OpenBSD.org/errata.html

Security patch announcements are sent to the security-announce@OpenBSD.org mailing list. For information on OpenBSD mailing lists, please see:

    http://www.OpenBSD.org/mail.html

CD-ROM SALES

OpenBSD 3.3 is also available on CD-ROM. The 3-CD set costs $40USD (EUR 45) and is available via mail order and from a number of contacts around the world. The set includes a colorful booklet which carefully explains the installation of OpenBSD. A new set of cute little stickers are also included (sorry, but our FTP mirror sites do not support STP, the Sticker Transfer Protocol). As an added bonus, the second CD contains an exclusive audio track, "Puff the Barbarian." Lyrics for the song may be found at:

```
http://www.OpenBSD.org/lyrics.html#33
```

Profits from CD sales are the primary income source for the OpenBSD project -- in essence selling these CD-ROM units ensures that OpenBSD will continue to make another release six months from now.

The OpenBSD 3.3 CD-ROMs are bootable on the following four platforms:

- i386
- macppc
- sparc
- sparc64 (UltraSPARC)

(Other platforms must boot from floppy, network, or other method).

For more information on ordering CD-ROMs, see:

```
http://www.OpenBSD.org/orders.html
```

The above web page lists a number of places where OpenBSD CD-ROMs can be purchased from. For our default mail order, go directly to:

```
https://https.OpenBSD.org/cgi-bin/order
```

or, for European orders:

```
https://https.OpenBSD.org/cgi-bin/order.eu
```

All of our developers strongly urge you to buy a CD-ROM and support our future efforts. Additionally, donations to the project are highly appreciated, as described in more detail at:

```
http://www.OpenBSD.org/goals.html#funding
```

T-SHIRT SALES

The project continues to expand its funding base by selling t-shirts and polo shirts. And our users like them too. We have a variety of shirts available, with the new and old designs, from our web ordering system at:

```
https://https.OpenBSD.org/cgi-bin/order
```

and for Europe:

```
https://https.OpenBSD.org/cgi-bin/order.eu
```

The OpenBSD 3.3 and OpenSSH t-shirts are available now!

FTP INSTALLS

If you choose not to buy an OpenBSD CD-ROM, OpenBSD can be easily installed via FTP. Typically you need a single small piece of boot media (e.g., a boot floppy) and then the rest of the files can be installed from a number of locations, including directly off the Internet. Follow this simple set of instructions to ensure that you find all of the documentation you will need while performing an install via FTP. With the CD-ROMs, the necessary documentation is easier to find.

1) Read either of the following two files for a list of ftp mirrors which provide OpenBSD, then choose one near you:

   http://www.OpenBSD.org/ftp.html
   ftp://ftp.OpenBSD.org/pub/OpenBSD/3.3/ftplist

   As of May 1, 2003, the following ftp sites have the 3.3 release:

   ftp://ftp.ca.openbsd.org/pub/OpenBSD/3.3/ Alberta, Canada
   ftp://ftp.usa.openbsd.org/pub/OpenBSD/3.3/ Boulder, CO, USA
   ftp://ftp7.usa.openbsd.org/pub/os/OpenBSD/3.3/ West Lafayette, IN, USA
   ftp://openbsd.wiretapped.net/pub/OpenBSD/3.3/ Sydney, Australia
   ftp://ftp.kd85.com/pub/OpenBSD/3.3/ Ghent, Belgium
   ftp://ftp.calyx.nl/pub/OpenBSD/3.3/ Amsterdam, Netherlands
   ftp://ftp.se.openbsd.org/pub/OpenBSD/3.3/ Stockholm, Sweden
   ftp://ftp.linux.org.tr/pub/OpenBSD/3.3/ Turkey

   Other mirrors will take a day or two to update.

2) Connect to that ftp mirror site and go into the directory pub/OpenBSD/3.3/ which contains these files and directories. This is a list of what you will see:

```
ANNOUNCEMENT   ftplist       ports.tar.gz
Changelogs/    hp300/        root.mail
HARDWARE       hppa/         sparc/
PACKAGES       i386/         sparc64/
PORTS          mac68k/       src.tar.gz
README         macppc/       srcsys.tar.gz
XF4.tar.gz     mvme68k/      tools/
alpha/         packages/     vax/
```

3) It is quite likely that you will want at LEAST the following files which apply to all the architectures OpenBSD supports.

README	generic README
HARDWARE	list of hardware we support
PORTS	description of our "ports" tree
PACKAGES	description of pre-compiled packages
root.mail	a copy of root's mail at initial login. (This is really worthwhile reading).

4) Read the README file. It is short, and a quick read will make sure you understand what else you need to fetch.

5) Next, go into the directory that applies to your architecture, for example, i386. This is a list of what you will see:

```
CKSUM           base33.tgz      game33.tgz
INSTALL.ata     bsd             index.txt
INSTALL.chs     bsd.rd          man33.tgz
INSTALL.dbr     cd33.iso        misc33.tgz
INSTALL.i386    cdrom33.fs      xbase33.tgz
INSTALL.linux   comp33.tgz      xfont33.tgz
INSTALL.mbr     etc33.tgz       xserv33.tgz
INSTALL.os2br   floppy33.fs     xshare33.tgz
INSTALL.pt      floppyB33.fs
MD5             floppyC33.fs
```

If you are new to OpenBSD, fetch _at least_ the file INSTALL.i386 and the appropriate floppy*.fs or cd33.iso file. Consult the INSTALL.i386 file if you don't know which of the floppy images you need (or simply fetch all of them).

6) If you are an expert, follow the instructions in the file called README; otherwise, use the more complete instructions in the file called INSTALL.i386. INSTALL.i386 may tell you that you need to fetch other files.

7) Just in case, take a peek at:

```
http://www.OpenBSD.org/errata.html
```

This is the page where we talk about the mistakes we made while creating the 3.3 release, or the significant bugs we fixed post-release which we think our users should have fixes for. Patches and workarounds are clearly described there.

Note: If you end up needing to write a raw floppy using Windows, you can use "fdimage.exe" located in the pub/OpenBSD/3.3/tools directory to do so.

XFREE86 FOR MOST ARCHITECTURES

XFree86 has been integrated more closely into the system. This release contains XFree86 4.2.1. Most of our architectures ship with XFree86, including sparc, sparc64 and macppc. During installation, you can install XFree86 quite easily. Be sure to try out xdm(1) and see how we have customized it for OpenBSD.

On the i386 platform a few older X servers are included from Xfree86 3.3.6. These can be used for cards that are not supported by Xfree86 4.2.1 or where XFree86 4.2.1 support is buggy. Please read the `/usr/X11R6/README` file for post-installation information.

PORTS TREE

The OpenBSD ports tree contains automated instructions for building third party software. The software has been verified to build and run on the various OpenBSD architectures. The 3.3 ports collection, including many of the distribution files, is included on the 3-CD set. Please see the PORTS file for more information.

Note: some of the most popular ports, e.g., the Apache web server and several X applications, come standard with OpenBSD. Also, many popular ports have been pre-compiled for those who do not desire to build their own binaries (see PACKAGES, below).

BINARY PACKAGES WE PROVIDE

A large number of binary packages are provided. Please see the PACKAGES file

```
ftp://ftp.OpenBSD.org/pub/OpenBSD/PACKAGES
```

for more details.

SYSTEM SOURCE CODE

The CD-ROMs contain source code for all the subsystems explained above, and the README

```
ftp://ftp.OpenBSD.org/pub/OpenBSD/README
```

file explains how to deal with these source files. For those who are doing an FTP install, the source code for all four subsystems can be found in the `pub/OpenBSD/3.3/` directory:

```
XF4.tar.gz              src.tar.gz
ports.tar.gz            srcsys.tar.gz
```

THANKS

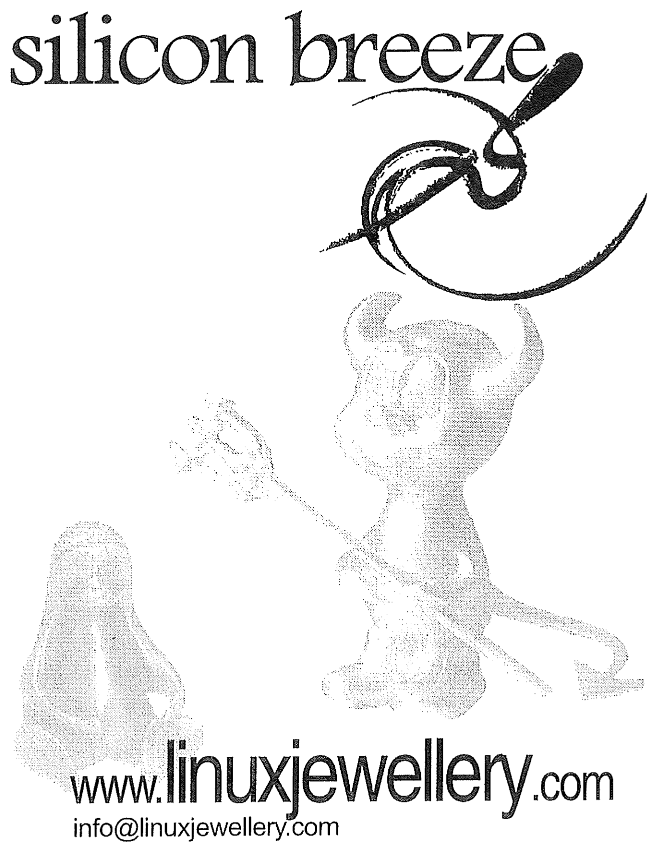
OpenBSD 3.3 includes artwork and CD artistic layout by Ty Semaka, who also wrote the lyrics and arranged an audio track on the OpenBSD 3.3 CD set. Ports tree and package building by Christian Weisgerber, David Lebel and Peter Valchev. System builds by Theo de Raadt, Henning Brauer, Todd Fries and Miod Vallat. ISO-9660 filesystem layout by Theo de Raadt.

We would like to thank all of the people who sent in bug reports, bug fixes, donation cheques, and hardware that we use. We would also like to thank those who pre-ordered the 3.3 CD-ROM or bought our previous CD-ROMs. Those who did not support us financially have still helped us with our goal of improving the quality of the software.

Our developers are:

Aaron Campbell, Alexander Yurchenko, Angelos D. Keromytis, Anil Madhavapeddy, Artur Grabowski, Ben Lindstrom, Bjorn Sandell, Bob Beck, Brad Smith, Brandon Creighton, Brian Caswell, Brian Somers, Bruno Rohee, Camiel Dobbelaar, Cedric Berger, Chad Loder, Chris Cappuccio, Christian Weisgerber, Constantine Sapuntzakis, Dale Rahn, Damien Couderc, Damien Miller, Dan Harnett, Daniel Hartmeier, David B Terrell, David Krause, David Lebel, David Leonard, Dug Song, Eric Jackson, Federico G. Schwindt, Grigoriy Orlov, Hakan Olsson, Hans Insulander, Heikki Korpela, Henning Brauer, Henric Jungheim, Hiroaki Etoh, Horacio Menezo Ganau, Hugh Graham, Ian

Darwin, Jakob Schlyter, Jan-Uwe Finck, Jason Ish, Jason McIntyre, Jason Peel, Jason Wright, Jean-Baptiste Marchand, Jean-Jacques Bernard-Gundol, Jim Rees, Joshua Stein, Jun-ichiro itojun Hagino, Kenjiro Cho, Kenneth R Westerback, Kevin Lo, Kevin Steves, Kjell Wooding, Louis Bertrand, Marc Espie, Marc Matteo, Marco S Hyman, Marcus Watts, Margarida Sequeira, Mark Grimes, Markus Friedl, Mats O Jansson, Matt Behrens, Matt Smart, Matthew Jacob, Matthieu Herrb, Michael Shalayeff, Michael T. Stolarchuk, Mike Frantzen, Mike Pechkin, Miod Vallat, Nathan Binkert, Nick Holland, Niels Provos, Niklas Hallqvist, Nikolay Sturm, Nils Nordman, Oleg Safiullin, Paul Janzen, Peter Galbavy, Peter Stromberg, Peter Valchev, Philipp Buehler, Reinhard J. Sammer, Ryan Thomas McBride, Shell Hin-lik Hung, Steve Murphree, Ted Unangst, Theo de Raadt, Thierry Deval, Thomas Nordin, Thorsten Lockert, Tobias Weingartner, Todd C. Miller, Todd T. Fries, Vincent Labrecque, Wilbern Cobb, Wim Vandeputte.



The Linux System Administrator's Security Guide

Author: Kurt Seifried <*kurt@seifried.org*>

Editors Notes and License

This is a serialization of Kurt Seifried's Linux System Administrator's Security Guide. Each AUUGN edition will contain two to three sections (depending on space) from the guide. It is released under the terms of the Open Content License. The editors have removed the preface to the guide in its entirety.

Introduction to Computer Security

What is Security?

Security is risk management...(unknown) *A computer is secure if you can depend on it and it's [sic] software to behave as you expect.* (Practical UNIX and Internet Security)

The principal objective of computer security is to protect and assure the confidentiality, integrity, and availability of automated information systems and the data they contain.
(http://csrc.nist.gov/publications/secpubs/cslaw.txt)

There are numerous definitions for "computer security", and most of them are correct. Essentially computer security means enforcement of usage policies, this must be done since people and software have flaws that can result in accidents, but also because someone may want to steal your information, use your resources inappropriately or simply deny you the use of your resources.

Security Policy

A security policy is an expression of your organizations security goals. Security goals will vary greatly by organization, for example a software vendor is typically more concerned about the confidentiality and integrity of their source code anything else, while a pornographic website is probably more concerned about processing credit cards online. There is an immense amount of security technology available, from software and hardware to specific techniques and implementations. You cannot properly implement the technology without a solid idea of what your goals are. Do home users connecting to the office need to use VPN software? If your security policy states that all file and print transfers must either be encrypted or sent across a "trusted" network then the answer is probably yes. What authentication methods are acceptable for services that can be reached from the Internet? Do you require strong passwords, tokens, biometric authentication, or some combination? The data you collect from remote sites, is it more important that this data is collected, or is it more important that this data remain secret? Data that is remote weather telemetry is probably not as sensitive as credit card numbers.

Some security policies will need to be exceptionally broad and general, for example the security policy for JANET, the academic backbone network for England states:

> **Monitoring and Enforcement**
>
> 19. It follows from the policy of cascaded responsibility backed up by written site agreements, that there must be some method for UKERNA to enforce the possible disconnection envisaged by the AUP, and to provide full access and assistance to law enforcement agencies where necessary.
>
> 20. The JANET-CERT has therefore been given the responsibility (in conjunction with UKERNA) to
>
> - Monitor use of the network, as far as is possible while respecting privacy, either in response to information about a specific threat, or generally because of a perceived situation
> - Require a primary site, through its nominated contact, to rectify any omission in its duty of responsibility
> - Where a site is unable or unwilling to co-operate, report the issue to UKERNA and initiate the procedure for achieving an emergency disconnection
> - Obtain evidence and pass on information as necessary in order to assist an investigation by a law enforcement agency
> - Provide support and co-ordination for investigations into breaches of security

On the other hand a security policy can be fine grained:

> All internal email must be encrypted with PGP or GnuPG using a key of at least 1024 bits that is signed by an authorized signing entity

Generally speaking the more detailed and technology oriented a security policy is the harder it will be to follow and keep up to date. The actually technical details of implementing a security policy should be separated from it. Keeping a separate set of best practices or an actually "implementation of security policy" document is a better idea then rolling it all into the security policy.

Acceptable Use Policy

Another component of computer security is an AUP (Acceptable Use Policy). This is a document that sates what a user of your resources may or may not do with them. Typically it is part of a contract and is signed at the time when the services are purchased. Many acceptable use policies generally forbid actions that are illegal, potentially annoying to other people (and hence likely to cause problems for the provider in the form of complaints) or are controversial (such as pornography). Other standard clauses include "this notice may change at any time without warning" and "we can terminate your service (or fire you) at any time without warning if you violate this policy. Generally speaking the majority of AUP's are reasonable and will not be a problem for normal users. These are a good compliment to a security policy as they set out in concrete terms what a user is or is not allowed to do, and as they are often part of a contract it allows a provider to enforce their security policy when a user attempts to violate it or has violated it.

Acceptable Use Policy

Privacy policies are interesting in that they are supposed to prevent an organization 9typically a company) from violating some aspects of a user's security (specifically the confidentiality of their information). Unfortunately the majority of privacy policies contain clauses like "this policy may change at any time without warning" or are simply discarded when a company decides to profit off of a user's information (such as name, address, credit card details, purchase history, etc.).

Security is a Process

You only need to make one mistake or leave one flaw available for an attacker to get in. This of course means that most sites will eventually be broken into. Witness the effects of Code Red, an Nimda, both of which were hugely successful exploiting well known and long solved vulnerabilities in the Microsoft IIS server. Regularity apply patches (after testing them). Regularly scan your network for open ports. Regularly scan your servers with intrusion testing software such as Nessus. Audit file permissions and make sure there are no unexpected setuid or setgid binaries.

Defense in depth

All technical security measures will eventually fail or be vulnerable to an attacker. This is why you must have multiple layers of protection. Firewalls are not enough, you may accidentally forget a rule or leave yourself exposed while reinitializing a ruleset, use tcp_wrappers to restrict access to services as well where possible. Restrict which users can access these services, especially if only a few users need access. Encrypt traffic where possible so that attackers cannot easily snoop usernames and passwords or hijack sessions. Since security measures will fail you also need a strong audit and logging process so that you can later find out what went wrong and how bad it is.

Technical Problems

These are just a handful of thousands of specific technical problems facing security administrators.

Network Connectivity

One of the biggest security challenges is the increase in network connectivity. If you have a machine that is not connected to any other machines an attacker will generally need to gain physical access. This of course greatly narrows down the number of attackers. However with everything connected to the Internet there are over 100 million people that can potentially get into your machine.

Insecure defaults

This is one of the problems that has caused no end of security problems since day one. Vendors typically ship their operating systems with insecure defaults (i.e. finger, telnet, ftp, etc.) meaning that administrators must expend a lot of effort to close security problems before they can even start to pro-actively secure their systems and networks.

Legitimate access vs.s a break in
Because you must grant legitimate users access to resources there is always the potential for attackers to gain access. Attacker can guess authentication credentials (i.e. weak passwords), steal them (password sniffing), exploit flaws in the server itself and so on.
Minimizing access and privilege
When possible restrict access. If you do not need to run fingerd turn it off and remove it. An attacker cannot exploit fingerd if it isn't present. Keep users off of servers if possible, if they need shell accounts for some reason setup a separate system and partition it off from the rest of your network. Lock down workstations where possible, set BIOS passwords, secure the boot sequence, do not give them administrative access.

(This section was sponsored by Vigilante http://www.viigilante.com)

Physical and Console Security

While the majority of random and remote attacks come in over the network physical and console security are important factors. In a perfect world every machine would be physically secure with access to the console (i.e. keyboard, reset switch and monitor) tightly controlled. Unfortunately this is not a perfect world it is rare to find a physically secure machine outside of a server room.
Physical security
Remember that an attacker may not want to break into your desktop machine or network, they may be looking for a quick way to make $200, and stealing a desktop computer is one way to do that. All systems should be securely fastened to something with a cable system, or locked in an equipment cage if possible. Case locks should be used when possible to slow attackers down (thefts of ram and CPU's are becoming increasingly popular). Some systems, like Apple G4's, when cable locked cannot be opened, if you need machines for public areas features like this are ideal. For secure rooms make sure that the walls go beyond the false ceiling, and below the raised floor, large vents should also be covered with bars if possible. Monitoring the room with CCTV and possibly requiring a remote person to "buzz" you in (in addition to a lock, keypad or other access control device) is recommended.

With enough time and physical access an attacker will be able to gain access to the system, several methods of attack include:

- Rebooting the system from other media such as floppy disk, CD-ROM, external SCSI drives and so on
- Removing the case, and removing the BIOS battery to get around any BIOS restrictions
- Using a default BIOS password to gain access to the BIOS
- Rebooting the system and passing boot arguments to LILO
- Installing physical monitoring devices such as KeyGhost
- Stealing the system's disk(s)
- Unplug the server, or turn the power bar off (a very effective DoS), if done several times this can lead to filesystem corruption.

These are just a few of many possible attacks. The primary goal is to stop them where possible, and failing that slow them down so that hopefully someone will notice the attacker tearing apart a system in someone's office. The installation of monitoring devices is becoming especially worrisome, as they are now available for purchase online for less then $100. An attacker can easily log all keystrokes for a long time period before the attacker comes back to retrieve them. Periodic physical inspections (in teams of two) of user machines for items like KeyGhost, modems are so on are a good idea.

Gaining access to office buildings is often trivial. While working for the government there was no access control to the building itself from 8 A.M. until 5 P.M. (after 5 P.M. a security guard forced you to sign in). Worse yet the building had a back entrance that was not monitored. If you were to enter at 4:30 P.M., hide in a bathroom for an hour or two (crouched on top of a toilet reading the latest Linux Journal) you could easily spend several hours fiddling with desktop systems and leave at your convenience without being seen by anyone. Cleaning staff also never questioned me when I stayed late (and conversely I never questioned them), you should train staff to approach people they do not recognize and ask them politely if they need assistance or are lost. While access to buildings cannot often be controlled effectively (to many entrances / different tenants / etc.) you can control access to your floor, a locked door with a CCTV monitoring it after hours is often a good deterrent.

"Practical Unix and Internet Security" from O'Reilly covers physical problems as well and is definitely worth buying.

Console security
With physical access to most machines you can simply reboot the system and ask it nicely to launch into single user mode, or tell it to use /bin/sh for init. You can enter the BIOS and tell the machine to boot from a floppy, doing a quick end run around most security. Alternatively you can simply enter the bios, disable the IDE controllers, put a password on the BIOS, rendering the machine largely unusable.
BIOS / Open Firmware security
Assuming the attacker does not steal the entire machine the first thing that they will usually try is to reboot the system and either boot from a floppy disk (or other removable media). If they can do this then any standard file protection is useless, the attacker declares themselves to be root, mounts the filesystem and gains complete access to it.

To secure a x86 BIOS you typically enter it by hitting "delete" or a function key during the boot process, the actual name and location of the BIOS password varies significantly, look for "security" or "maintenance". There are usually different levels of password protections, on some motherboards you can disable the keyboard until a password is typed in (the BIOS intercepts and blocks input until it sees the password entered on the keyboard), on others it only prevents accessing the BIOS. Generally speaking you want to block access to the BIOS, and lock the boot sequence to the first internal storage device (i.e. the first IDE disk or SCSI).

Even if you do everything right there are still some ways for an attacker to subvert the boot process. Many older BIOS's have universal passwords, generally speaking this practice has declined with modern systems, but you may wish to inquire with the vendor. Another potential problem to be aware of is that many add-on IDE and SCSI controller cards have their own BIOS, from which you can check the status of attached devices, choose a boot device, and in some cases format attached media. Many high-end network cards also allow you to control the boot sequence, letting you boot from the network instead of a local disk. This is why physical security is critical for servers. Other techniques include disabling the floppy drive so that attackers or insiders cannot copy information to floppy and take it outside. You may also wish to disable the serial ports in users machines so that they cannot attach modems, most modern computers use PS/2 keyboard and mice, so there is very little reason for a serial port in any case (plus they eat up IRQ's). Same goes for the parallel port, allowing users to print in a fashion that bypasses your network, or giving them the chance to attach an external CD-ROM burner or harddrive can decrease security greatly. As you can see this is an extension of the policy of least privilege and can decrease risks considerably, as well as making network maintenance easier (less IRQ conflicts, etc.). There are of course programs to get the BIOS password from a computer, one is available http://www.cgsecurity.org/, it is available for DOS and Linux.

If you decide to secure the BIOS's on systems you should audit them once in a while if possible, simply reboot the machine and try to boot off of a floppy disk or get into the BIOS. If you can then you know someone has changed settings on the system, and while there may be a simple explanation (a careless technician for example) it may also be your first warning that an attack has occurred. There are several programs for Linux that allow an attacker with root access to gain the BIOS password, while this is a rather moot point it does bear mentioning (if an attacker has gained root access they can already do pretty much anything).

To secure a Sparc or UltraSparc boot prom send a break during boot-up, hit stop-a, and you should be presented with the ok> prompt. Setting your password is a simple matter of using the password command and typing the password in twice. You will then want to set the security-mode, using "setenv" from the default of none to command at the very least, and full if you are security conscious (warning, you will need the password to boot the machine).

```
ok
ok password
ok New password (only first 8 chars are used):*****
ok Retype new password: *****
ok
ok setenv security-mode full
ok
```

You can also set "security-mode" to "command" which will require the password to access the open firmware but is less strict then "full". Do not lose this password, especially if you set the security-mode to full, as you may need to replace the PROM to recover the system. You can wipe the password by setting "security-mode" to "none".

Unfortunately if you are using Apple hardware you cannot secure the boot process in any meaningful manner. While booting if the user holds down the command-option-P-R keys it will wipe any settings that exist, there is no way to avoid this. About the only security related option you can set is whether the machine automatically reboots or not, this is useful for servers to prevent a remote attacker from changing the kernel for example (which require a system reboot). Hold down the command-option-O-F keys to access the OpenFirmware and from there you need to:

```
go> setenv auto-boot? False
```

LILO Security

LILO is the Linux boot loader, it handles all the nitty gritty tasks of getting the kernel into memory and bootstrapping them machine into something that resembles a useful computing device. LILO handles many things, from telling the kernel to look for multiple network cards to how much memory there is (assuming Linux won't detect how much your system has). There are several options you can pass to LILO to bypass most any account security or console security.

Booting Linux into single user mode with (assuming the label is "linux"):

```
boot: linux single
```

This will dump you into a root level command prompt on many systems. From there you can simply add a new users with UID 0, format the disks or copy Trojan'ed binaries off a floppy disk. One way many vendors deal with this is by using sulogin, single user login, which prompts for root's password before letting you on. This will prevent people from accessing single user mode and getting directly to a root prompt however there is a another attack. You can specify which program will be used to init the system, instead of the default init you can use a command shell:

```
boot: linux single init=/bin/sh
```

Which will present you with a root prompt. The best way to defeat this and the single mode problem are to secure LILO and prevent the passing of boot arguments without a password. Many people argue that this somehow inhibits normal use of the system, however there should be no need normally for the user booting the system to pass LILO arguments (if an argument is needed at boot time it should be put into lilo.conf permanently). Generally speaking the only time you will need to pass LILO arguments is when something on the system breaks, at which point it's probably wise to send out a technician who knows the password to fix the system.

To secure LILO you must set a password, and use the restricted keyword. You can also set security on a per image basis, or on LILO as a whole. If you set security on a per image basis you will be able to tell LILO which image you wish to boot without needing a password, but you will not be able to pass the kernel arguments such as "single". If you set security on LILO as a whole then you will only be able to boot to the default image, specifying a different image will require the password.

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
message=/boot/message
linear
default=linux

password=thisisapassword
restricted

image=/boot/vmlinuz-2.2.18
        label=linux
        read-only
        root=/dev/hda1

image=/boot/vmlinuz-2.2.17
        label=linux-old
        read-only
        root=/dev/hda1
```

This can be cumbersome, however it does prevent attackers from booting a different image.

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
message=/boot/message
linear
default=linux

image=/boot/vmlinuz-2.2.18
        label=linux
        read-only
        root=/dev/hda1
        password=thisisapassword
        restricted

image=/boot/vmlinuz-2.2.17
        label=linux-old
        read-only
        root=/dev/hda1
        password=thisisapassword
        restricted
```

The above example will prevent attackers from messing with the "linux" and "linux-old" image, however if they need to boot the old kernel (because a driver is broken, or SMP support doesn't work quite right) then they will not need the password to do so.

Depending on the level of security you want you can either restrict and password protect all of LILO, or just the individual images. In any event if you are deploying workstations or servers you should use one to prevent users from breaking into systems in less than 10 seconds. While things like sulogin will prevent a user from getting a root prompt if they enter single user mode the attacker can always use "init=/bin/sh".

One minor security measure you can take to secure the lilo.conf file is to set it immutable, using the "chattr" command. To set the file immutable simply run the following command as root:

```
chattr +i /sbin/lilo.conf
```

And this will prevent any changes (accidental or otherwise) to the lilo.conf file. If you wish to modify the lilo.conf file you will need to unset the immutable flag run the following command as root:

```
chattr -i /sbin/lilo.conf
```

only the root user has access to the immutable flag.

Grub security

Grub is the default boot loader in Debian, Mandrake and possibly Red Hat 7.2. More on this to come.

[Ed's Note: Debian Woody (current release) uses Lilo. Mandrake > 8 uses Grub. RedHat > 8 gives you a choice but will default to Grub. Mandrake and Redhat give you an opportunity to enter a Grub password in their installers.]

Rebooting the system

If possible you should make rebooting the system more difficult, by default almost all Linux distributions have ctrl-alt-del enabled so that it reboots the machines. However, unlike Windows, this is not necessary. In Linux you can easily control the behavior of ctrl-alt-del, simply by editing the /etc/inittab file:

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

You may wish to disable it entirely (simply comment it out with a #) or modify the command issued. Minimally you can create a bash script such as:

```
#!/bin/bash
#
# This script emails a message to root and then
shuts
# down the system after a 5 second delay
#
/bin/date | /bin/mail -s "ctrl-alt-del used" root
/bin/sleep 5s
/sbin/shutdown -t3 -r now
```

Now every time someone hits ctrl-alt-del you will have an email message in root's account logging it. You can also send that email to another system (i.e. to root@example.org). If you do this you may wish to introduce a small delay between the mail command and the shutdown command to ensure the email gets out before the mail system is turned off. Of course an attacker can always hit the reset switch, power button, unplug the system, or trip the breakers for an entire floor of computers. If the system is in a locked cabinet however this is quite a bit more conspicuous then simply using a three finger salute (ctrl-alt-del) to reboot the system.

Summary

An attacker with physical access to the console, or worse yet the hardware itself has many potential avenues of attack they can pursue. Worse yet if their goal is to simply deny use of service, or damage the software and hardware doing so is trivial. Flipping the power off and on during the boot process will often result in drive corruption, or they can simply pour a glass of water into the power supply. Locking servers up in secure rooms is critical, a competitor can easily afford to pay a janitor several thousand dollars to turn a server off, and pour pop into the power supply, resulting in a dead server when staff turns it back on. Workstations are typically more vulnerable as they are in accessible areas, but hopefully they are interchangeable with another workstation, with all data being stored on servers.

To be continued...

This article is re-printed with permission. The originals can be found at:

The Roadmap for FreeBSD 5-Stable

Author: Scott Long <scottl@FreeBSD.org>

FreeBSD team's Scott Long lays out a roadmap for FreeBSD-5 stable in this informative email. He says that although the latest release of FreeBSD 5 marks a major milestone in FreeBSD's history, there are significant improvements necessary in the areas of SMP, kernel lockdown, performance, network driver stability, ACPI and much more. He also presents a tentative schedule for the rest of the year for FreeBSD 5.1 and 5.2 releases. Thanks, mezz, our moderator for the newstip!

```
-----------------------------------------------
Date: Thu, 13 Feb 2003 16:36:05 -0800 (PST)
From: Scott Long <scottl@FreeBSD.org>
To: arch@freebsd.org
Subject: 5-STABLE Roadmap
```

All,

Thanks to the hard work of everyone, FreeBSD 5.0 became a reality and is working better than most even hoped. However, there is still a lot of work to be done before we can create the RELENG_5/5-STABLE branch and declare success. Below is a document that I have drafted with the input and review of the Release Engineering Team, the Technical Review Board, and the Core Team that defines what needs to be done in order to reach 5-STABLE. I'm happy to take further input into this, and I will also mark it up and make it available online.

1. Introduction and background

After nearly three years of work, FreeBSD 5.0 was released in January of 2003. Features like the GEOM block layer, Mandatory Access Controls, ACPI, sparc64 and ia64 platform support, and UFS snapshots, background filesystem checks, and 64-bit inode sizes make it an exciting operating system for both desktop and production users. However, some important features are not complete. The foundations for fine-grained locking and preemption in the kernel exist, but much more work is left to be done. Work on Kernel Schedulable Entities, also known as Scheduler Activations, has been ongoing but needs a push to realize its benefit. Performance compared to FreeBSD 4.x has declined and must be restored and surpassed.

This is somewhat similar to the situation that FreeBSD faced in the 3.x series. Work on 3-CURRENT trudged along seemingly forever, and finally a cry was made to 'just ship it' and clean up later. This decision resulted in the 3.0 and 3.1 releases being very unsatisfying for most, and it wasn't until 3.2 that the series was considered 'stable'. To make matters worse, the RELENG_3 branch was created along with the 3.0 release, and the HEAD branch was allowed to advance immediately towards 4-CURRENT. This resulted in a quick divergence between HEAD and RELENG_3, making maintenance of the RELENG_3 branch very difficult. FreeBSD 2.2.8 was left for quite a while as the last production-quality version of FreeBSD.

Our intent is to avoid repeating that scenario with FreeBSD 5.x. Delaying the RELENG_5 branch until it is stable and production quality will ensure that it stays maintainable and provides a compelling reason to upgrade from 4.x, To do this, we must identify the current areas of weakness and set clear goals for resolving them. This document contains what we as the release engineering team feel are the milestones and issues that must be resolved for the RELENG_5 branch. It does not dictate every aspect of FreeBSD development, and we welcome further input. Nothing that follows is meant to be a sleight against any person or group, or to trivialize any work that has been done. There are some significant issues, though, that need decisive and unbiased action.

2. Major issues

The state of SMPng and kernel lockdown is the biggest concern for 5.x. To date, few major systems have come out from under the kernel-wide mutex known as 'Giant'. The SMP status page at `http://www.FreeBSD.org/smp` provides a comprehensive breakdown of the overall SMPng status. Status specific to SMPng progress in deivce drivers can be found at at `http://www.FreeBSD.org/projects/busdma`.

In summary:

- VM - the kmem_malloc(M_NOWAIT) path no longer needs Giant held. The kmem_malloc(M_WAITOK) path is in progress and is expected to be finished in the coming weeks. Other facets of the VM system, like the vfs interface, buffer/cache, etc, are largely untouched.
- GEOM - The GEOM block layer was designed to run free of Giant, but at this time no block drivers can run without Giant. Additionally, it has the potential to suffer performance loss due to its upcall/downcall data paths happening in kernel threads. Lightweight context switches might help this.
- Network - Work is in progress to lock the TCP and UDP portions of the stack. This also includes locking the routing tree, ARP code, and ifaddr and inet data structures. RawIP, IPv6, Appletalk, etc, have not been touched. Locking the socket layer is in progress but is largely untested. None of the hardware drivers have been locked.
- VFS - Initial pre-cleanup started.
- buffer/cache - Initial work complete.
- Proc - Work on locking the proc structure was ongoing for a while but seems to have stalled.
- CAM - No significant work has occurred on the CAM SCSI layer.
- Newbus - some work has started on locking down the device_t structure.
- Pipes - complete with the exception of VM-related optimizations.
- File descriptors - complete.
- Process accounting - jails, credentials, MAC labels, and scheduler are out from under Giant.
- MAC Framework - complete
- Timekeeping - complete
- kernel encryption - crypto drivers and core crypto framework are Giant-free. KAME IPsec and FAST IPSec have not been locked.
- Sound subsystem - complete
- kernel preemption - preemption for interrupt threads is enabled. However, contention due to Giant covering much of the kernel and most of the device driver interrupt routines causes excessive context switches and might actually be hurting performance. Work is underway to explore ways to make preemption be conditional.

Another issue with SMPng is interrupt latency. The overhead of doing a complete context switch to a kernel interrupt thread is high and shows noticeable latency. Work is ongoing to implement lazy context switching on all platforms. Fine grained locking of drivers will also help this, as will converting drivers to be as efficient as possible in their interrupt routines.

Next, the state of KSE must resolved for RELENG_5. Work on it has slowed noticeably in the past 6 months but appears to be picking up again. There are a number of issues that must be addressed:

- Signal delivery to threads is not defined. Signals are delivered to the process, but which thread actually receives it is random.
- There is confusion over whether upcalls are generated on every system call or when a thread blocks. The former is highly undesirable and needs to be investigated.
- The userland threading library, currently called libkse, is incomplete and has not been used for any significant threaded application.
- KSE has the potential to uncover latent race conditions and create new ones. An audit needs to be performed to ensure that no obvious problems exist.

According to the release schedule below, KSE kernel and userland components must be functionality complete by June 2003 in order to be included in the RELENG_5 branch. For security and stability reasons, if KSE cannot be finished in time then, by default, all KSE-specific syscalls should be modified to return ENOSYS and all other KSE-specific interfaces disabled. Deprecating KSE from RELENG_5 but keeping it in the HEAD branch will pose problems in porting bugfixes and features between the two branches, so every effort should be made to finish it on time.

3. Goals for 5-STABLE

The goals for the RELENG_5 branch point are:

- All subsystems and interfaces must be mature enough to be maintainable for improvements and bug fixes
- equal or better stability from FreeBSD 4.8.
- no functional regressions from 4.8. It is important to make sure that users do not avoid upgrading to 5.x because of lost functionality.
- performance on par with FreeBSD 4.8 for most common operations. Both UP and SMP configurations should be evaluated. SMP has the potential to perform much better than 4.x, though for the purposes of creating the RELENG_5 branch, comparable performance between the two should be acceptable.

It is unrealistic to expect that the SMPng project will be fully complete by RELENG_5, or that performance will be significantly better than 4.x. However, focusing on a subset of the outstanding tasks will give enough benefit for the branch to be viable and maintainable. To break it down:

- ABI/API/Infrastructure stability - Enough infrastructure must be in place and stable to allow fixes from HEAD to easily and safely be merged into RELENG_5. Also, we must draw a line as to what subsystems are to be locked down when we go into 5-STABLE.

- SMPng
- VM - Most codepaths, others than the ones that interact with VFS, should be Giant-free for RELENG_5.
- Network - Taking the network stack out from under Giant poses the risk of uncovering latent bugs and races. Locking it down but not removing Giant imposes further performance penalties. A decision on whether to continue with locking the network layers, and whether they should be free from Giant for RELENG_5 should be made no later than March 15. If the decision is made to allow the locking to go forward, the IPv4, UDP, and TCP layers should be free of Giant. IPv6 and the socket layers would be nice to have also, though it should be investigated whether they can be safely locked down in 5.x after the RELENG_5 branch. If the decision is to keep the network stack under Giant for the branch, then an investigation should be made to determine if the present locking work can be reverted and deferred to 6-CURRENT.
  Having a Giant-free path from the the TCP/IP layers to the hardware should be investigated as it could allow significant performance gains in the network benchmarks. If this can be achieved then the hardware interface layer needs to allow for drivers to incrementally become free of Giant. Locking down at least two Ethernet drivers would be highly desirable. If the semantics are too complex to have the stack free of Giant but not the hardware drivers, investigation should be done into making it configurable.
  Lesser-used network stacks like netatlk, netipx, etc, should not break while this work is going on. However, locking them is not a high priority.
- GEOM - At least 2 block drivers should be locked in order to demonstrate that others can also be locked without changing the interface to GEOM. The ATA driver is a good candidate for this, though caution should be taken as it is also extremely high-profile and any problems with it will affect nearly all users of FreeBSD.
- Lazy context switching - sparc64 is the only platform that performs lazy context switching when entering the kernel. The performance gains promised by this are significant enough to require that it be implemented for all other Tier 1 platforms.
- KSE - The kernel side of KSE must be functionally complete and have undergone a security audit. libkse must be complete enough to demonstrate a real-world application running correctly on it using the standard POSIX Threads API. Examples would be apache 2.0, squid, and/or mozilla. A functional regression test suite is also a requirement for RELENG_5 and should test signal delivery, scheduling, performance, and process security/credentials for both KSE and non-KSE processes. KSE kernel and userland components must also reach the same level of functionality for all Tier-1 platforms in both UP and SMP configurations. The definition of 'Tier-1 platforms' can be found at `http://www.freebsd.org/doc/en_US.IS...uide/archs.html`.
- busdma interface and drivers - architectures like PAE/i386 and sparc64 which don't have a direct mapping between host memory address space and expansion bus address space require the elimination for vtophys() and friends. The busdma interface was created to handle exactly this problem, but many drivers do not use it yet. The busdma project at `http://www.FreeBSD.org/projects/busdma/index.html` tracks the progress of this and should be used to determine which drivers must be converted for RELENG_5 and which can be left behind. Also, there has been talk by several developers and the original author to give the busdma interface a minor overhaul. If this is to happen, it needs to happen before RELENG_5. Otherwise, differences between the old and new API will make driver maintenance difficult.
- PCI resource allocation - PC2003 compliance requires that x86 systems no longer configure PCI devices from the system BIOS, leaving this task solely to the OS. FreeBSD must gain the ability to manage and allocate PCI memory resources on its own. Implementing this should take into account cardbus, PCI-HotPlug, and laptop dockstation requirements. This feature will become increasingly critical through the lifetime of RELENG_5, and therefore is a requirement for the RELENG_5 branch.

Most performance gains hinge on the progress of SMPng Areas that should be concentrated on are:

- Storage I/O - I/O performance suffers from two problems, too many expensive context switches, and too much work being done in interrupt threads. Specifically, it takes 3 context switches for most drivers to get from the hardware completion interrupt to unblocking the user process: one for the interrupt thread, one for the GEOM g_up thread, and one to get back to the user thread. Drivers that attempt to be efficient and quick in their interrupt handlers (as all should be) usually also schedule a taskqueue, which adds a context switch in between the interrupt thread and the g_up thread and brings the total up to 4. Two things need to be done to attack this:
  - make all drivers defer most of their processing out of their interrupt thread. Significant performance gains have been shown recently in the aac(4) driver by making its interrupt handler be 'INTR_MPSAFE' and moving all processing to a taskqueue.
  - investigate eliminating the taskqueue context switch by adding a callback to the g_up thread that allows a driver to do its interrupt processing there instead of in the taskqueue.
- Network - Network drivers suffer from the interrupt latency previously mentioned as well as from the network stack being partially locked down but not free from Giant. Possible strategies for addressing this are described in the previous section.

- Other locking - XXX ?
- Benchmarks and performance testing - Having a source of reliable and useful benchmarks is essential to identifying performance problems and guarding against performance regressions. A 'performance team' that is made up of people and resources for formulating, developing, and executing benchmark tests should be put into place soon.

Comparisons should be made against both FreeBSD 4.x and Linux 2.4.x. Tests to consider are:

- the classic 'worldstone'
- webstone - `/usr/ports/www/webstone`
- Fstress - `http://www.cs.duke.edu/ari/fstress`
- ApacheBench - `/usr/ports/www/p5-ApacheBench`
- netperf - `/usr/ports/benchmarks/netperf`

Features:

- ACPI - Intel's ACPI power management and device configuration subsystem has become an integral part of FreeBSD's x86 and ia64 device configuration model. However, many bugs exist in Intel's vendor code, our OS-specific code, and motherboard BIOSes, causing many ACPI-enabled systems to fail to boot, misdetect drivers, and/or have many other problems. Fixing these problems seems to be an uphill battle and is often times causing a poor first-impression of FreeBSD 5.0. Most x86 systems can function with ACPI disabled, and logic should be added to the bootloader and sysinstall to allow users to easily and intuitively turn it off. Turning off ACPI by default is prone to problems also as many newer systems rely on it to provide correct interrupt routing information. Also, a centralized resource should be created to track ACPI problems and solutions. Linux uses the same Intel vendor sources as FreeBSD, so we should investigate how they have handled some of the known problems.
- NEWCARD/OLDCARD - The NEWCARD subsystem was made the default for 5.0. Unfortunately, it contains no support for non-Cardbus bridges and falls victim to interrupt routine problems on some laptops. The classic 16-bit bridge support, OLDCARD, still exists and can be compiled in, but this is highly inconvenient for users of older laptops. If OLDCARD cannot be completely deprecated for RELENG_5, then provisions must be made to allow users to easily install an OLDCARD-enabled kernel. Documentation should be written to help trasition users from OLDCARD to NEWCARD and from 'pccardd' to 'devd'. The power management and 'dumpcis' functionality of pccardc(1) needs to be brought forward to work with NEWCARD, along with the ability to load CIS quirk entries. Most of this functionality can be integrated into devd and devctl.
- New scheduler framework - The new scheduler framework is in place, and users can select between the classic 44bsd scheduler and the new ULE scheduler. A scheduler that demonstrates processor affinity, HyperThreading and KSE awareness, and no regressions in performance or interactivity characteristics must be available for RELENG_5.
- sparc64 local console - neither syscons nor vt work on sparc64, leaving it with only serial and 'fake' OFW console support. This is a major support hole for what is a Tier 1 platform. Whether syscons can be shoe-horned in or wscons be adopted from NetBSD is up for debate. However, sparc64 must have local console support for RELENG_5. Having this will also allow the XFree86 server to run, which is also a requirement for RELENG_5.
- gcc/toolchain - gcc 3.3 might be available in time for RELENG_5 and might offer some attractive benefits, but also likely to introduce ABI incompatibility with prior gcc versions. ABI compatibility should be locked down for the RELENG_5 branch.
  There has also been a request to move `/usr/include/g++` to `/usr/include/g++-v3` to be more compliant with the stock behavior of gcc. This should be investigated for RELENG_5 also.
- gdb - gdb from the base system should work for sparc64. It should also understand KSE thread semantics, assuming that KSE is included in the RELENG_5 branch. gdb 5.3 is available and there are reports that it should address the sparc64 issue.
- disklabel(8) regressions - The biggest casualty of the introduction of GEOM appears to be the disklabel utility. The '-r' option gives unpredictable results in most cases now and should be removed or fixed. Work is planned for a new unified interface for modifying labels and slices, however this should not preclude disklabel from being fixed.

Documentation:

- The manual pages, Handbook, and FAQ should be free from content specific to FreeBSD 4.x, i.e. all text should be equally applicable to FreeBSD 5.x. The installation section of the handbook needs the most work in this area.
- The release documentation needs to be complete and accurate for all Tier 1 architectures. The hardware notes and installation guides need specific attention.
- If FreeBSD 5.1 is not the branch point for RELENG_5 then the Early Adopters Guide needs to be updated. This document should then be removed just before the release closest to the RELENG_5 branch point.

4. Schedule

If branching RELENG_5 at the 5.1 release is paramount, 5.1 will probably need to move out by at least 3 months. The schedule would be:

- Jun 30, 2003 - KSE and SMPng feature freeze
- Aug 4, 2003 - 5.1-BETA, general code freeze
- Aug 18, 2003 - 5.1-RC1, RELENG_5 and

RELENG_5_1 branched
- Aug 25, 2003 - 5.1-RC2
- Sept 1, 2003 - 5.1-RELEASE

Taking an incremental approach might be more beneficial. Releasing 5.1 in time for USENIX ATC 2003 will provide a wide audience for productive feedback and will keep FreeBSD visible. In this scenario, 5.1 should offer a significant improvement over 5.0 in terms of bug fixes and performance. Lockdowns and improvements to the storage subsystem and scheduler should be expected, the NEWCARD/OLDCARD issues should be addressed, and all known bugs and regressions from the 5.0 errata list should be fixed. KSE and other SMPng tasks that cannot finish in time for 5.1 should also not reduce the stability of the release. The schedule for this would be:
- May 5, 2003 - 5.1-BETA, general code freeze
- May 19, 2003 - 5.1-RC1, RELENG_5_1 branched
- May 27, 2003 - 5.1-RC2
- Jun 2, 2003 - 5.1-RELEASE
- Jun 30, 2003 - KSE and SMPng feature freeze
- Sept 1, 2003 - 5.2-BETA, general code freeze
- Sept 15, 2003 - 5.2-RC1, RELENG_5 and RELENG_5_2 branched
- Sept 22, 2003 - 5.2-RC2
- Sept 29, 2003 - 5.2-RELEASE

5. Post RELENG_5 direction

As with all -STABLE development streams, the focus should be bug fixes and incremental improvements. Just like normal, everything should be vetted through the HEAD branch first and committed to RELENG_5 with caution. As before, new device drivers, incremental features, etc, will be welcome in the branch once they have been proven in HEAD.

Further SMPng lockdowns will be divided into two categories, driver and subsystem. The only subsystem that will be sufficiently locked down for RELENG_5 will be GEOM, so incrementally locking down device drivers under it is a worthy goal for the branch. Full subsystem lockdowns will have to be fully tested and proven in HEAD before consideration will be given to merging them into RELENG_5.

This article is re-printed with permission. The originals can be found at:

http://www.bsdforums.org/forums/showthread.php?threadid=6892

Mail and Dynamic IP

Author: Peter Chubb <peter@chubb.wattle.id.au>

It's possible to run a mail server on a dynamic IP address, (as Frank Crawford reported doing in `My Home Network' several auugn's ago) but it's usually a bad idea. The problem is that if your IP address changes while someone's sending you email (for example, your aDSL connection drops, and while it's down someone else grabs your address) the email is delivered to the machine that your old address was allocated to --- and bounces.

In addition, my current ISP blocks port access to the SMTP port, so there's no point in running a mailer daemon directly accessible from the internet.

Living with the problem

The simplest thing to do is just to live with the problem. Advertise the email address you got from your ISP, and use IMAP or whatever to read your email.

However, this defeats the purpose of having your own domain, and means that when you change your ISP, or your ISP changes the addresses on you, you have to tell everyone the new domain name.

Third Party Hosting Services

There are any number of third party email hosting services --- Hotmail, Yahoo! Mail, etc., etc. Most will not host your entire domain's email for free, although some are fairly cheap.

The one I've found to be best is fastmail.fm an Australian company whose entire business is handling outsourced email. They'll do low volume mail handling for free; and for slightly more will host your entire email requirements.

Using MailForward and ZoneEdit

If your email requirements are small (less than 200M per year, only a few addresses), and you host your DNS with zoneedit.com, then you can set up a poor-man's MTA as follows:

- Set up exim or whatever behind your firewall, to do local deliveries as you wish.

- Set up a free IMAP account (for example with fastmail.fm for each local address. The names can be random gibberish, as end users are never going to see them.

- Set up a MailForward for each user at zoneedit.com to point each to a different free IMAP account.

- Set up fetchmail to poll the IMAP accounts at 15 minute intervals and give the result to the local smtp server.

The fetchmail configuration file then looks like this (assuming two users, and use of fastmail.fm {http://www.fastmail.fm}):

```
     poll fastmail.fm with proto IMAP and options
no dns interval 4
        localdomains YourDomain.id.au
      user 'abc123' there with password 'PASS
WORD' is 'user1' here
```

```
        options no rewrite fetchall
    user 'xyz890' there with password
'PASSWORD' is 'user2' here
        options no rewrite fetchall
```

You could of course configure fetchmail in multidrop mode, and deliver all email for your domain to one email account (the one provided by your ISP, perhaps). And it's a good idea to do this as a catchall.

```
      pop-server.vic.bigpond.net.au aka
mail.dnsvr.com with proto POP3
      and options
        no dns
        envelope 3 "Received"
        localdomains YourDomain.id.au
        user 'pchubb' there with password
'PassWord'
        to pchubb@bigpond.net.au=peterc * here
        options  pass8bits stripcr no rewrite
```

The problem with this is that in forwarding the email, the envelope address is lost. The only clue as to the destination of the message is then in the To: and CC: headers, and in the trail of Received: headers. There's no guarantee that anything useful will be available from that lot; in particular, if more than one local user subscribes to a mailing list, then it'll be delivered to at most one local user (the headers will be the same in both cases, and there's no clue that two different users should receive the email).

USING ODMR

If you can find/beg/buy a mailhosting service that does ODMR (e.g., mailhost.com.au or mailkeep.com) then you can handle all your own email almost as if port 25 were not blocked.

ODMR (On-Demand Mail Relay, RFC 2645) is a protocol designed for dial-up and dynamic-IP hosts. A host with a static IP acts as your mail relay, collecting email on your behalf. At regular intervals, your system polls the host, and if there is any email for you, turns the connection around and allows the remote host to talk SMTP using the same TCP connexion. A challenge-response password system is used for authentication. Thus there is no requirement to have access to port 25 on a static IP address on your system.

Just set up MX records in your DNS to point to the ODMR host, and then use fetchmail to pull all the email down every 15 minutes or so and pipe it into your local MTA.

The main advantage of ODMR is that the envelope addresses aren't lost, so as new accounts and aliases are created locally, one doesn't have to set up new MailForwards and IMAP accounts elsewhere.

The main disadvantage is that spam filtering is a little harder; you can't set up a tarpit or bounce detected spam straight away, because all email is relayed via your mail host.

My home domains now all use mailhost.com.au and ODMR for mail, and ZoneEdit.com for DNS. I'm still looking for a free web host that can do PHP, CGI and MySQL.

Mailhost.com.au isn't yet up for commercial access; I've been told that that's coming soon. In the meantime, their beta program is open; contact Gavin Carr (gavin@openfusion.com.au) for details.

IDS - Intrusion Detection System, Part I

Author: Klaus Müller <socma@gmx.net>
Translated to English by: Hubert Kaißer <hubert@faveve.uni-stuttgart.de>

INTRODUCTION

Before I start just a few explanations so that no misconceptions arise: IDS stands for Intrusion Detection System. In the course of the text I also relate to IDS as a system that recognizes intrusions and starts counteractive measures (whereas to initiate counteractive measures is an optional feature). In the beginning, I will at first give an overview about the different kinds of IDS to be able to address different tactics afterwards. In the last part I will discuss several programs like Snort, bofh, ... and my project COLOID.

In advance, I give some annotations for terms which I will not explain directly in the text:

- false positive = an alarm that an attack has taken place whereas this was not the case
- false negative = the IDS does not detect and filter the attack

IDS OVERVIEW

The idea of this chapter is to present different kinds of Intrusion Detection Systems and at the same time their advantages and disadvantages. But first, some words about the meaning and intention of Intrusion Detection Systems. An IDS sort of observes activities on the particular host or network. With several methods which I will present here, it tries to find out if the security of the host is threatened (if there is an "attack") to initiate counteractive measures afterwards. Because, in the majority of cases, log files are created and it is one of the main issues to analyze them and react on the indication of an intrusion or an illegal attempt of a user to augment his rights.

Basically, there are the following three areas:

- Information Sources
- Response

- Analysis

Response and Analysis is covered again more precisely later, the different kinds of "Information Sources" right now. Because of the extensive possibilities to attack a PC or a network there are different kinds of IDS (of course, they can be combined with each other) which basically differ in the points where they control and what they control (Information Sources).

Host - Based Intrusion Detection

Host-Based Intrusion Detection Systems analyze information on a single host. As they normally do not look at the whole network traffic but "only" at the activities on the same host they are able to make more precise statements about the kind of attack. In addition, you can see directly the impact on the particular PC. E.g that a specific user started an attack successfully. Host-based IDSs use mostly two different sources to provide information about activities: system logs and operating system audit trails. Both have advantages and disadvantages as operating system audit trails are more exact, detailed and can give better information about activities while system logs mostly deliver only the most important information and are therefore considerably smaller. System logs can be controlled and analyzed better because of their size.

Advantages:
- you "see" the impact of an attack and can react better on it
- you can recognize Trojan horses etc. better as the available information/possibilities are very extended
- you can detect attacks which cannot be detected by Network based IDS because traffic is often encrypted
- you can observe activities on your host exactly

Disadvantages:
- they are not good in recognizing scans
- they are more vulnerable to DoS attacks
- the analysis of operating system audit trails is very time-consuming because of its size.
- they stress the host's CPU performance (partly) immensely

Examples:
- Tripwire
  http://www.tripwire.com/products/index.cfml
- SWATCH
  http://freshmeat.net/redir/swatch/10125/url_homepage/swatch
- DragonSquire
  http://www.enterasys.com/ids/squire/
- Tiger
  http://freshmeat.net/redir/tiger-audit/30581/url_homepage/tiger
- Security Manager
  http://www.netiq.com/products/sm/default.asp

Network Intrusion Detection (NIDS)

The main task of a Network IDS is the analysis and interpretation of packets transferred over the network. Signatures are used to examine packets for "conspicuous" content like e.g. /etc/passwd. We will discuss this in the course of the article. Mostly, so called sensors (often single hosts) are used which do nothing except analyze traffic and if necessary start an alarm. As this is their only task it is possible to secure those sensors better. In this context there exists often the so called "Stealth Mode", i.e., the sensors act "invisible" so that it is more difficult for the attacker to localize or attack them.

"Stealth mode can be used for network sensors to make the promiscuous mode network interface card (NIC) invisible because it simply doesn't have an IP address in this mode. This is achieved by using a separate NIC in each network sensor machine to communicate with the console over, usually, a physically isolated secure network. Stealth mode makes it more difficult for a hacker to attack the network sensor itself."

This paragraph (taken from the description for RealSecure) clarifies again what a sensor in Stealth Mode is. The sensor switches into promiscuous mode (simply put: the mode in which the network device reads the whole network traffic) and has no own IP. With this it should be made as difficult as possible for the attacker to localize the sensor. By the way, this is the mode used by packet sniffers like tcpdump...

Basically, you can use sensors in following areas:

Within the firewall (simplified):

```
---------------------------------
|        Internet               |
---------------------------------
                |
                |
        -----------------------------------
        |            Firewall             |
        -----------------------------------
                        |
                 -------------
                 |  Sensor   |
                 -------------
                        |
                 -----------------
                 |      LAN      |
                 -----------------
```

This diagram shows only one possible solution and shall only clarify that the sensors are not between DMZ and firewall. If the expression DMZ is unknown to you: it stands for DeMilitarized Zone and is an area which is secured from all sides.

If sensors are within the firewall it is easier for them to decide if a firewall was eventually configured improperly and in addition you get to know if a potential attack came through the firewall or not. According to experience, sensors create less false positives as they act "less inconspicuously" and therefore traffic is less (whereby the probability of a false alarm drops).

Sensors placed within a firewall serve as intrusion detection. Should your sensor fulfill this task you should then put it within the firewall ...

Outside the firewall (simplified):

```
---------------------------------
|           Internet            |
---------------------------------
              |
  -----------------------------
  |          Sensor           |
  -----------------------------
              |
  -----------------------------
  |         Firewall          |
  -----------------------------
              |
        -----------------
        |      LAN      |
        -----------------
```

Sensors are often placed outside the external firewall as shown in the diagram. The reason for this is that the sensor can receive and analyze the whole traffic from the internet. If you place the sensor here it is not ensured that really all attacks are filtered and detected, e.g., TCP attacks. In this case, you would try to detect the attack by using so called signatures (more about this in the part about signatures). Nevertheless, this placement is preferred by many experts because there is the advantage to log and analyze the attacks (to the firewall...), thereby, the admin can see where he should change the firewall configuration.

Sensors placed outside the firewall serve as attack detection (different to placing in within the firewall!). If your sensors should detect these attacks you should put them outside the firewall...

– Outside and within the firewall

Actually, this variant connects both previously mentioned variants. But, the danger is that you configure the sensors and/or the firewall wrongly, i.e., you cannot simply add the advantages of both variants to this variant. These are not the only possibilities to placing sensors, you can, of course, place them somewhere else but the above mentioned places are the most commonly used.

Advantages:
- the sensors can be secured well as they "only" observe traffic
- you can detect scans better - on the basis of signatures... you can "filter" traffic (actually, we will show later that this is not always the case)

Disadvantages:
- the probability of so called false negatives (attacks are not detected as attacks) is high as it is difficult to control the whole network
- mostly, they have to operate on encrypted packets where analysis of packets is complicated
- as a difference to host-based IDS they do not see the impacts of an attack

Examples:
- NetRanger [http://www.cisco.com]
- Dragon [http://www.securitywizards.com]
- NFR [http://www.nfr.net]
- Snort [http://www.snort.org]
- DTK [http://all.net/dtk/dtk.html]
- ISS RealSecure [http://www.uh.edu/infotech/software/unix/realsecure/index.html]

Though I distinguish between HIDS and NIDS the difference becomes smaller and smaller as in the meanwhile HIDS, too, have the basic functionality of NIDS. Known ID Systems like ISS RealSecure call themselves not only NIDS but "host-based and network-based IDS". In the future, the difference between both systems will become even less clear (so that both systems "grow together" more and more).

Network Node Intrusion Detection (NNIDS)

Basically, this new type (NNIDS) works like typical NIDS, i.e., you take packets from network traffic and analyze them. But it only concerns packets which are addressed to the network node (this is where the name comes from). Another difference between NNIDS and NIDS is that NIDS run in promiscuous mode while NNIDS does not run in promiscuous mode. As not every packet is analyzed the performance of the system will not suffer to much, such systems run very fast as a rule.

Application Based Intrusion Detection

Application Based IDS's are a subgroup of host-based IDS but I mention them here separately. It monitors the interaction between user and program whereby mainly additional log files are created to provide information about the activities. As you operate directly between user and program you can very easily "filter" conspicuous behaviour. You can visualize an ABIDS as in the following diagram:

```
------------------
|  Application   |
------------------
        |
------------------
|      IDS       |
------------------
        |
------------------
|      User      |
------------------
```

Advantages:

- you work at unencrypted level, in opposition to, e.g., Network Based IDS, whereby analysis is more feasible
- you can detect and prevent conspicuous commands which the user wants to use on/with the program

Disadvantages:

- no security and few possibilities to detect, e.g., Trojan horses (as you do not act in kernel space)
- the log files created by this kind of IDS are easy attack targets for "attackers" and not so secure like, e.g., operating system audit trails

Stack Based Intrusion Detection

Also a new kind of IDS is the so called Stack Based Intrusion Detection System (SBIDS). But momentarily, I do not have enough information which would allow me to brief you about this type of IDS. In a future version of this paper the description will certainly follow...

Honeypot

The word Honeypot is used in a lot of different contexts. To avoid confusion I will stick to the definition from the SANS Institute's Intrusion Deception FAQ: "Honeypots are programs that simulate one or more network services that you designate on your computer's ports. An attacker assumes you're running vulnerable services that can be used to break into the machine. A honeypot can be used to log access attempts to those ports including the attacker's keystrokes. This could give you advanced warning of a more concerted attack". In some cases, a honeypot is simply a "box". From the outside it appears vulnerable, while it logs traffic and also analyzes it. Thus, because honeypots appear vulnerable and no connections should be created every connection to the honeypot is seen as suspicious

```
---------------------------
|      Internet           |
---------------------------
            |
---------------------------
|     ext. Firewall       |
---------------------------
            |
---------------------------
|     Honeypot            |      <--- in DMZ
---------------------------
            |
---------------------------
|     int. Firewall       |
---------------------------
```

So, the internal network does not have to be exposed because a honeypot is normally placed in the DMZ (DeMilitarized Zone). The major task of a honeypot consists mostly in analyzing traffic, e.g., when certain processes were started, which files were changed..., so that you can create a quite good profile of potential attackers.

But not all honeypots are alike so you can distinguish between three "variants" which enhance security of the system in different ways:

Prevention: prevent an attack detection: notify that there was an attack (possibly a localization of the attacker, what kind of attack, which services are concerned...?) Reaction: the (counter)action, detection alone does not serve anything if you do not take steps against it. Padded Cell Systems offers special possibilities which concern counteractive measures. So, reaction implies what you do / the IDS does when you react to an attack.

Later, we will clarify the different response options of IDS again... In addition, honeypots can be classified in two bigger categories, research and production honeypots. Production honeypots are to help lessen potential risks in a network while research honeypots serve to collect and inquire information about the attackers.

Prevention:
Honeypots are surely not the most appropriate systems to avert attacks. As you will see later at Padded Cell Systems, a wrong programmed and wrong configured honeypot can even facilitate attacks whereby this covers the whole subject of IDS.

Detection:
Seemingly, that is the biggest advantage of honeypots as they can be excellent in detecting attacks. In this context, they can above all analyze and interpret system logs. The placement of the honeypot plays a decisive role, an admin can only benefit by honeypots if the are configured and placed properly. Often, they are placed between important servers to detect eventual scans of the whole network or in the proximity of one important server to detect illegal access with the help of port redirection (e.g., if someone tries to access the server at certain ports he will be redirected to the honeypot, as this access should not be allowed there should be a warning).

Reaction:
In the part about Responses (read beneath), you can see which possibilities a honeypot has to react on events. When using/creating a honeypot you should keep in mind that the host should look most attractive to an attacker, i.e., it should not be too difficult to attack the host. Basically, there should be few changes from the default configuration as too many changes only seem conspicuous. Nevertheless, you should not forget the actual idea, i.e., control traffic, log activities, ... One approach of which I have heard several times is to place the honeypot in its own subnet behind a firewall. This has normally alarm capabilities and so can display warnings. As logs are sought-after targets of attackers you should not log them on the host itself but send it to another server. Sometimes, sniffers are used to search traffic for certain signs and "log" traffic. If you collected enough information you should analyze the logs and look for weaknesses of the network, respectively rectify them. Because of the amount of information it should be easier to close security holes and take action against attackers. But you should consider that honeypots are not completely legal, respectively you should be attentive when using honeypots. The problem is that a honeypot can be interpreted as an "invitation to attack" and if you realize that the host was attacked (and you do not initiate counteractive

measures) that could be interpreted additionally as an act of gross negligence. Some judiciary argumentation against honeypots sound banal but you should check the law at your place. If someone attacks another network from your network and creates damage you will (probably) pay for it for already mentioned reasons.

Further inquiries found that the application of honeypots is, e.g., in Europe is no problem (if you find a law which contradicts this, write to me, please. My search did not find such a law...).

Padded Cell Systems

These systems normally work together with one of the other systems. If the used IDS notifies about an attack the respective attacker is forwarded to a "padded cell host". As soon as they are in this padded cell host they cannot make any damage as the whole environment is simulated, a "bogus" environment. It is important that this environment is as realistically presented as possible, respectively the attacker has to think that the attack was successful (sort of). There is a possibility to log, analyze and follow every activity of the attacker. The problem with using padded cell systems is that it is possible that it is illegal to employ them in a particular country (as eventually such counter activities of the IDS can be seen as "attack", though they are only meant to collect information about the real attacker).

Advantages:

- once in a padded cell host, attackers cannot do any more damage as it is only a "bogus" environment
- the admin can follow/log the activities of the attacker directly to get more information about the attack and the target of the attack and so is able to initiate counteractive measures more easily

Disadvantages:

- eventually, usage of padded cell systems is not legal (the same as honeypots, in Europe this should be no problem)
- the implementation of such a system is very difficult and requires some knowledge as the whole environment has to be simulated correctly. If the admin makes a little mistake somewhere this system could by all means open additional security holes

Types of attack

Before you develop or use an IDS you should specify the potential dangers, also which attacks you expect. Despite the different possibilities attacks and their target can be defined in four categories:

1) Confidentiality
The consequences of the attack are that the mutual trust to a certain user changed (mostly to his favor ;), e.g., that you do not have to authenticate anymore for a certain program... Such circumstances are still abused these days, e.g., if there is a mutual trust between two hosts, i.e., the user of one host can log in another without password (or else). If an attacker accomplishes a comparable effect with his attack it is sometimes very difficult to discover such "abused" mutual trust.

2) Integrity
If the user changes important configuration and system files, replaces existing binaries by his own... Often, existing binaries (like, e.g., /bin/login) are replaced by own binaries. There, the respective user only has to give a fixed password (in the source) and gets (mostly) root privileges. Such attacks serve to expand ones own privileges, e.g., by means of installing a login backdoor. In fact, there are tools like Tripwire but not everyone uses it. In addition, there are often devastating errors in configuration and use which make Tripwire an extra risk in such cases.

3) Availability
By this, the reachability of the system is affected. This could cause the ban of certain users to log in at all or that you can only log in at certain times... The aim is mostly to work "undisturbed" and cannot be discovered by any user.

4) Control
E.g, if the attack is for the purpose of "overtaking" the system that has control over files, programs... When this happens, you should consider that the attacker gets also all other previously listed options. If he has total control over the system he can change, restrict... what he wants.

Before I get to the different types of attack (DoS, DDoS, Scans, ...) I will make just a little excursion to the world of Integrity Checkers. Tools like Tripwire are part of host based IDS, the issue of an Integrity Checker is to check the integrity of diverse files and start an alarm if you detect changes on a file.

1) Creation of a database
The first step after the installation of an Integrity Checker like Tripwire is the creation of a database. This step should (must) be made in a situation at which the condition of the system is not compromised. If you create the database at a later time (and maybe the attacker has replaced the existing binaries) the use of an Integrity Checker would not work as it should. In such a case, the replaced binaries would be considered as "originals" and if the admin replaces these binaries by the actual "originals", an alarm would be started. Most of the tools offer extensive possibilities to specify files/directories of which you want to create checksums. We simply generate a fingerprint of the system. (Tripwire calls this Database Initialization Mode)

2) Check of Integrity
After the admin has a database (the fingerprint) of the system he can check his system when he

wants to. This is done simply by comparing the checksums in the database with the currently available files on the disk. Tripwire offers more possibilities. Simply read the manpage for Tripwire or associated documentation.

Those two steps are actually found in most Integrity Checkers. Tripwire offers the possibility to update the database (when having installed new tools...) or to update the policy file, test the email notification system...

Which errors can be made when using Integrity Checkers ?

The first and grossest error an admin can commit is to create an MD5 hashsum database when existing binaries have been replaced by an attacker already. If an admin assumes that an attack was successful and afterwards creates the hashsum database of this "compromised" system, the binary, replaced by the attacker (e.g., login backdoor), would be considered as the "real" binary. You should create the database as soon as possible after installation. One other big error when using Integrity Checkers (like, e.g., Tripwire) is to leave the hashsum database on hard disk. On first sight, it seems absolutely normal to leave the database on hard disk but if you have to leave it there you should at least make sure that the partition/the medium is read-only. You should make sure that no one can write-access our database. If the attacker could write the database he could change it as he wants. If you worked with Tripwire before you surely know that you can adjust in the configuration file of which files the integrity should be checked. But what if the attacker can read/write this configuration file? He could change search paths so that the directory with the changed binaries would not be scanned at all. It is best not to leave the configuration on hard disk and put it onto a read-only medium. Some will think that it is too much trouble to put the files on a read-only medium, actually, you should consider some more time exposure for more security (an interesting aspect with user-friendliness is that many security holes are/were opened because programs are made user-friendly. In our case it means that you do get less user-friendliness but a higher level of security). Tripwire (and other integrity checkers) are surely capable programs which can enhance security and prevent potential attackers from successfully attacking. But the best tool is of no use if the other security settings of the system are not okay. So, do not demand of integrity checkers that they do all your work.

A newer type of integrity checkers are the so called Realtime Integrity Checkers. In contrast to "normal" integrity checkers they check file integrity in realtime. Here is a short scheme:

1) Here too, the first step is the creation of a hashsum database
2) Before someone can run a program the hashsum of the file is procuced. If the value does not accord to the one in the database the binary was replaced. If this is the case, execution is prohibited.

This concept is only one possible for realtime integrity checkers (at least I arranged this concept for me). As a difference to integrity checkers you do not rely on the admin checking files regularly, you try to check the correctness of the binary before execution and not let it execute if necessary...

Besides the above mentioned categories (integrity, control, ...) attacks can be differentiated otherwise, of course, e.g., how to attack, respectively with what means. Also, here are naturally many possibilities, but the most common attacks against IDSs are:

– Scans
  These days, scans are the common "attacks", the problem here is that the IDS should not produce too many false positives. Mostly, scans serve to get (further) information about a host/network (e.g., to start subsequent attacks). What information an attacker can get about your own network you can see with the following scan result (nmap - only a small example which does not at all show all possibilities of nmap):

```
....
Host (192.168.0.0) seems to be a subnet broadcast
address (returned 1 extra pings).
Skipping host. Interesting ports on
playground.yuma.net 192.168.0.1):
Port        State     Protocol     Service
22          open      tcp          ssh
111         open      tcp          sunrpc
635         open      tcp          unknown
1024        open      tcp          unknown
2049        open      tcp              nfs

TCP Sequence Prediction: Class = random positive
increments
              Difficulty=3916950 (Good luck!)
Remote operating system guess:
     Linux 2.1.122 - 2.1.132; 2.2.0-pre1 - 2.2.2

Interesting ports on vectra.yuma.net (192.168.0.5):
Port        State     Protocol     Service
 13         open      tcp          daytime
 21         open      tcp          ftp
 22         open      tcp          ssh
 23         open      tcp          telnet
 37         open      tcp          time
 79         open      tcp          finger
 111        open      tcp          sunrpc
 113        open      tcp          auth
 513        open      tcp          login
 514        open      tcp          shell

TCP Sequence Prediction: Class = random positive
increments
              Difficulty = 17719 (Worthy challenge)
Remote operating system guess:
     OpenBSD 2.2. - 2.3

Nmap run completed -- 256 IP addresses (2 hosts up)
scanned in 6 seconds
```

Mainly, you can find out the following:
- which operating system?
- which versions of certain programs run
- which services run that you can "attack" eventually
- which ports are open

◦ ...

Using many of the most different scan techniques together result in a mass of information which allows the attacker to initiate his attack. Though, I will not describe/mention all scan techniques, I will mention some of the often used ones (if there are questions referring to protocols... look at the respective RFCs):

Ping scanning:

Ping scans are used to find out which hosts are online. For this, you send the host (or the hosts) an ICMP datagram of type 8 (i.e., echo request) and wait for an ICMP answer datagram of type 0 (i.e., echo reply). Sometimes, you do not send only an echo request but additionally an ACK as sometimes ICMP is blocked. If there is an RST answer the host is online.

Nmap parameter: -sP

TCP Scanning (Vanilla) :

With TCP scanning you (mostly) try to connect() on all ports after you did the three-way-handshake, i.e., you made a connection to the ports (the connections to the ports were successful) the reply value of connect() is checked. For the attacker the reply value informs if the used port (or the ports) is open or closed. The aim of a TCP scan is to find out if ports are open/closed.

Nmap parameter: -sT

The following nmap output is of one of my hosts, directly after a default installation. I deliberately made no changes (in the configuration)

```
[Socma]$ nmap -sT localhost

Starting nmap V. 2.54BETA36 (
www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in
state: closed)
Port       State      Service
21/tcp     open       ftp
23/tcp     open       telnet
80/tcp     open       http
111/tcp    open       sunrpc
113/tcp    open       auth
6000/tcp   open       X11

Nmap run completed -- 1 IP address (1 host up)
scanned in 1 second
```

A part of a tcpdump trace (of this scan):

```
02:10:15.804704 Diablo > Diablo: icmp: echo request
    4500 001c 2db8 0000 3501 5a27 7f00 0001
    7f00 0001 0800 fc95 fb69 0000
02:10:15.814704 Diablo > Diablo: icmp: echo reply
(DF)
    4500 001c 0000 4000 ff01 7dde 7f00 0001
    7f00 0001 0000 0496 fb69 0000
02:10:15.814704 Diablo.58725 > Diablo.http: . ack
110306597 win 3072
    4500 0028 d223 0000 2a06 c0aa 7f00 0001
    7f00 0001 e565 0050 ad48 0003 0693 2525
    5010 0c00 e718 0000
02:10:15.814704 Diablo.http > Diablo.58725: R
110306597:110306597(0)
         win 0 (DF)
    4500 0028 0000 4000 ff06 7dcd 7f00 0001
    7f00 0001 0050 e565 0693 2525 0000 0000
    5004 0000 a070 0000
02:10:16.114704 Diablo.1727 > Diablo.821: S
196002918:196002918(0)
win 32767 <mss 16396,sackOK,timestamp 213509[|tcp]>
(DF)
    4500 003c 8663 4000 4006 b656 7f00 0001
    7f00 0001 06bf 0335 0bae c466 0000 0000
    a002 7fff 739c 0000 0204 400c 0402 080a
    0003 4205
02:10:16.114704 Diablo.821 > Diablo.1727: R 0:0(0)
ack 196002919
         win 0 (DF)
    4500 0028 0000 4000 ff06 7dcd 7f00 0001
    7f00 0001 0335 06bf 0000 0000 0bae c467
    5014 0000 d7c4 0000
02:10:16.114704 Diablo.1728 > Diablo.440: S
195504823:195504823(0)
win 32767 <mss 16396,sackOK,timestamp 213509[|tcp]>
(DF)
    4500 003c 68b2 4000 4006 d407 7f00 0001
    7f00 0001 06c0 01b8 0ba7 2ab7 0000 0000
    a002 7fff 0ecf 0000 0204 400c 0402 080a
    0003 4205
```

UDP scanning:

The intention of UDP scans are analog to TCP scans as you want to find out open UDP ports. A scan runs differently as UDP is a connectionless protocol (TCP is connection-oriented). UDP scanning "uses" ICMP, if you send to the actual ports a 0 byte UDP packet to wait for an ICMP "answer". If there is a notify that the port is unreachable ("Port unreachable"/code value 3) this means that the port is closed. If the respective admin, e.g., in his /etc/inetd.conf, deactivated certain services and you try to send a packet to the respective port this would result in the error message "Port Unreachable"...

Nmap parameter: -s

```
[Socma]$ nmap -sU localhost

Starting nmap V. 2.54BETA36 (
www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1459 ports scanned but not shown below are in
state: closed)
Port       State      Service
111/udp    open       sunrpc

Nmap run completed -- 1 IP address (1 host up)
scanned in 4 seconds
```

The associated tcpdump trace:

```
10:41:55.954397 Diablo > Diablo: icmp: echo request
    4500 001c cc8f 0000 2801 c84f 7f00 0001
    7f00 0001 0800 8471 738e 0000
10:41:55.954397 Diablo > Diablo: icmp: echo reply
(DF)
    4500 001c 0000 4000 ff01 7dde 7f00 0001
    7f00 0001 0000 8c71 738e 0000
10:41:55.964397 Diablo.63793 > Diablo.http: . ack
994287972 win 2048
    4500 0028 79e3 0000 2506 1deb 7f00 0001
    7f00 0001 f931 0050 06d8 0003 3b43 a164
    5010 0800 cccd 0000
10:41:55.964397 Diablo.http > Diablo.63793: R
994287972:994287972(0)
     win 0 (DF)
    4500 0028 0000 4000 ff06 7dcd 7f00 0001
    7f00 0001 0050 f931 3b43 a164 0000 0000
    5004 0000 dbb4 0000
10:41:56.274397 Diablo.63773 > Diablo.15: udp 0
```

```
     4500 001c 8a0b 0000 3011 02c4 7f00 0001
     7f00 0001 f91d 000f 0008 08af
10:41:56.274397 Diablo > Diablo: icmp: Diablo udp
port 15
      unreachable (DF) [tos 0xc0]
     45c0 0038 0000 4000 ff01 7d02 7f00 0001
     7f00 0001 0303 fb18 0000 0000 4500 001c
     8a0b 0000 3011 02c4 7f00 0001 7f00 0001
     f91d 000f
10:41:56.274397 Diablo.63773 > Diablo.1459: udp 0
     4500 001c 6c2f 0000 3011 20a0 7f00 0001
     7f00 0001 f91d 05b3 0008 030b
10:41:56.274397 Diablo > Diablo: icmp: Diablo udp
port 1459
      unreachable (DF) [tos 0xc0]
     45c0 0038 0100 4000 ff01 7c02 7f00 0001
     7f00 0001 0303 fb18 0000 0000 4500 001c
     6c2f 0000 3011 20a0 7f00 0001 7f00 0001
     f91d 05b3
```

Another variant of a UDP scan (UDPrecvfrom() and write() scan) consists in scanning every port twice. The just now mentioned method uses ICMP with "Port Unreachable", but only root receives this message. If you scan a closed port twice you get, after the second scan: "Error 13 : Try Again"...

ACK scanning:

With sending an ACK packet to a port of a firewall you find out which ports are filtered and which are not. If you receive an RST answer it means that the referring port is "unguarded", respectively is not filtered, else you get an ICMP error message. So, you do not find out which ports are open but you get more precise information about the firewall (and its configuration).

Nmap parameter : -sA

```
[Socma]$ nmap -sA localhost

Starting nmap V. 2.54BETA36 (
www.insecure.org/nmap/ )
All 1558 scanned ports on Diablo (127.0.0.1) are:
UNfiltered

Nmap run completed -- 1 IP address (1 host up)
scanned in 6 seconds
```

The tcpdump trace:

```
10:45:51.864397 Diablo > Diablo: icmp: echo request
     4500 001c 1617 0000 3901 6dc8 7f00 0001
     7f00 0001 0800 113d e6c2 0000
10:45:51.864397 Diablo > Diablo: icmp: echo reply
(DF)
     4500 001c 0000 4000 ff01 7dde 7f00 0001
     7f00 0001 0000 193d e6c2 0000
10:45:51.864397 Diablo.53119 > Diablo.http: . ack
2682022466 win 3072
     4500 0028 0dda 0000 3206 7cf4 7f00 0001
     7f00 0001 cf7f 0050 0650 0003 9fdc 6a42
     5010 0c00 c590 0000
10:45:51.864397 Diablo.http > Diablo.53119: R
2682022466:2682022466(0)
     win 0 (DF)
     4500 0028 0000 4000 ff06 7dcd 7f00 0001
     7f00 0001 0050 cf7f 9fdc 6a42 0000 0000
     5004 0000 d7ef 0000
10:45:52.164397 Diablo.53099 > Diablo.14: . ack
2457451592 win 3072
     4500 0028 218d 0000 3206 6941 7f00 0001
     7f00 0001 cf6b 000e 5938 4710 9279 bc48
     5010 0c00 e74d 0000
10:45:52.164397 Diablo.14 > Diablo.53099: R
2457451592:2457451592(0)
      win 0 (DF)
     4500 0028 0000 4000 ff06 7dcd 7f00 0001
     7f00 0001 000e cf6b 9279 bc48 0000 0000
     5004 0000 93a2 0000
10:45:52.164397 Diablo.53099 > Diablo.imap3: . ack
2457451592 win 3072
     4500 0028 a075 0000 3206 ea58 7f00 0001
     7f00 0001 cf6b 00dc 5938 4710 9279 bc48
     5010 0c00 e67f 0000
```

Stealth scanning (NULL, XMAS, FIN, SYN, ...):

With stealth scanning you "abuse" the three-way-handshake. I present the three-way-handshake shortly:

```
------------------          SYN          -------------
| Host    A      |  ------------>  | Host    B |
------------------                       -------------

------------------        ACK/SYN        -------------
| Host    A      |  <-------------| Host    B |
------------------                       -------------

------------------          ACK          -------------
| Host    A      |  ------------>  | Host    B |
------------------                       -------------
```

The problem of TCP scans is that they are very conspicuous (as every time it does a three-way-handshake). With stealth scanning the following happens instead:

```
------------------          SYN          -------------
| Host    A      |  ------------>  | Host    B |
------------------                       -------------

------------------        ACK/SYN        -------------
| Host    A      |  <-------------| Host    B |
------------------                       -------------
```

This diagram seems to look like the three-way-handshake but with a basic difference: The shown diagram would have no connection between A and B, respectively Host B would think the connection exists, though the connection doesn't exist until A sends a further ACK to B (it is also called a "half-open" port...) . The above shown SYN scan implies that the port of the target host is open (because of the ACK/SYN), if it were closed you would receive a RST/ACK back.

Nmap parameter : -sS

```
[Socma]$ nmap -sS localhost

Starting nmap V. 2.54BETA36 (
www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in
state: closed)
Port        State       Service
21/tcp      open        ftp
23/tcp      open        telnet
80/tcp      open        http
111/tcp     open        sunrpc
113/tcp     open        auth
6000/tcp    open        X11

Nmap run completed -- 1 IP address (1 host up)
scanned in 3 seconds
```

Tcpdump trace:

```
10:47:41.674397 Diablo > Diablo: icmp: echo request
     4500 001c 8f08 0000 3501 f8d6 7f00 0001
     7f00 0001 0800 99a9 5e56 0000
10:47:41.674397 Diablo > Diablo: icmp: echo reply
(DF)
```

```
    4500 001c 0000 4000 ff01 7dde 7f00 0001
    7f00 0001 0000 a1a9 5e56 0000
10:47:41.674397 Diablo.58038 > Diablo.http: . ack
1561498752 win 3072
    4500 0028 afe5 0000 3206 dae8 7f00 0001
    7f00 0001 e2b6 0050 82b0 0003 5d12 9480
    5010 0c00 4e85 0000
10:47:41.674397 Diablo.http > Diablo.58038: R
1561498752:1561498752(0)
     win 0 (DF)
    4500 0028 0000 4000 ff06 7dcd 7f00 0001
    7f00 0001 0050 e2b6 5d12 9480 0000 0000
    5004 0000 dd44 0000
10:47:41.984397 Diablo.58018 > Diablo.1488: S
2803535203:2803535203(0)
     win 3072
    4500 0028 a4f5 0000 3206 e5d8 7f00 0001
    7f00 0001 e2a2 05d0 a71a 8d63 0000 0000
    5002 0c00 88ef 0000
10:47:41.984397 Diablo.1488 > Diablo.58018: R
0:0(0) ack 2803535204
     win 0 (DF)
    4500 0028 0000 4000 ff06 7dcd 7f00 0001
    7f00 0001 05d0 e2a2 0000 0000 a71a 8d64
    5014 0000 94dc 0000
```

Now, other scans join the game: FIN scanning, NULL scanning and XMAS scanning. FIN scanning only sends a FIN message to the "target host", though no connection exists between them. At a closed port RST is sent back, else nothing happens.

Nmap parameter : -sF

```
[Socma]$ nmap -sF localhost

Starting nmap V. 2.54BETA36 (
www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in
state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up)
scanned in 6 seconds
```

Tcpdump trace:

```
10:48:28.704397 Diablo > Diablo: icmp: echo request
    4500 001c b29d 0000 3401 d641 7f00 0001
    7f00 0001 0800 a1a7 5658 0000
10:48:28.704397 Diablo > Diablo: icmp: echo reply
(DF)
    4500 001c 0000 4000 ff01 7dde 7f00 0001
    7f00 0001 0000 a9a7 5658 0000
10:48:28.704397 Diablo.52201 > Diablo.http: . ack
2873378189 win 4096
    4500 0028 cbeb 0000 2b06 c5e2 7f00 0001
    7f00 0001 cbe9 0050 9020 0003 ab44 458d
    5010 1000 54a3 0000
10:48:28.704397 Diablo.http > Diablo.52201: R
2873378189:2873378189(0)
     win 0 (DF)
    4500 0028 0000 4000 ff06 7dcd 7f00 0001
    7f00 0001 0050 cbe9 ab44 458d 0000 0000
    5004 0000 f4d2 0000
10:48:29.004397 Diablo.52181 > Diablo.233: F
0:0(0) win 4096
    4500 0028 10c6 0000 2b06 8108 7f00 0001
    7f00 0001 cbd5 00e9 0000 0000 0000 0000
    5001 1000 d522 0000
10:48:29.004397 Diablo.233 > Diablo.52181: R
0:0(0) ack 1 win 0 (DF)
    4500 0028 0000 4000 ff06 7dcd 7f00 0001
    7f00 0001 00e9 cbd5 0000 0000 0000 0001
    5014 0000 e50e 0000
```

NULL and XMAS scans are of special interest (above all with practical implementation of protocol anomaly detection). It is called XMAS scan because all flags are set: SYN, ACK, FIN, URG, PUSH. As with FIN scanning you send back RST if the port is closed.

Nmap parameter : -sX

```
[Socma]$ nmap -sX localhost

Starting nmap V. 2.54BETA36 (
www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in
state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up)
scanned in 5 seconds
```

Tcpdump trace:

```
10:44:24.004397 Diablo > Diablo: icmp: echo request
    4500 001c ffcc 0000 2a01 9312 7f00 0001
    7f00 0001 0800 103d e7c2 0000
10:44:24.004397 Diablo > Diablo: icmp: echo reply
(DF)
    4500 001c 0000 4000 ff01 7dde 7f00 0001
    7f00 0001 0000 183d e7c2 0000
10:44:24.004397 Diablo.36398 > Diablo.http: . ack
718216305 win 2048
    4500 0028 2e28 0000 2906 65a6 7f00 0001
    7f00 0001 8e2e 0050 9220 0003 2acf 1c71
    5010 0800 41f0 0000
10:44:24.004397 Diablo.http > Diablo.36398: R
718216305:718216305(0)
     win 0 (DF)
    4500 0028 0000 4000 ff06 7dcd 7f00 0001
    7f00 0001 0050 8e2e 2acf 1c71 0000 0000
    5004 0000 dc1f 0000
10:44:24.304397 Diablo.36378 > Diablo.1996: FP
0:0(0) win 2048 urg 0
    4500 0028 7651 0000 2906 1d7d 7f00 0001
    7f00 0001 8e1a 07cc 0000 0000 0000 0000
    5029 0800 13d3 0000
10:44:24.304397 Diablo.1996 > Diablo.36378: R
0:0(0) ack 1 win 0 (DF)
    4500 0028 0000 4000 ff06 7dcd 7f00 0001
    7f00 0001 07cc 8e1a 0000 0000 0000 0001
    5014 0000 1be7 0000
```

The other possibility, called NULL scan, means that no flag is set, if the port is closed an RST is sent back.]

Nmap parameter : -sN

```
[Socma]$ nmap -sN localhost

Starting nmap V. 2.54BETA36 (
www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in
state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
```

```
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up)
scanned in 5 seconds
```

Tcpdump trace:

```
10:43:37.594397 Diablo > Diablo: icmp: echo request
     4500 001c 2ecf 0000 2c01 6210 7f00 0001
     7f00 0001 0800 8f87 6878 0000
10:43:37.594397 Diablo > Diablo: icmp: echo reply
(DF)
     4500 001c 0000 4000 ff01 7dde 7f00 0001
     7f00 0001 0000 9787 6878 0000
10:43:37.604397 Diablo.34607 > Diablo.http: . ack
1932747046 win 4096
     4500 0028 ee0f 0000 3706 97be 7f00 0001
     7f00 0001 872f 0050 5b20 0003 7333 6126
     5010 1000 ead5 0000
10:43:37.604397 Diablo.http > Diablo.34607: R
1932747046:1932747046(0)
      win 0 (DF)
     4500 0028 0000 4000 ff06 7dcd 7f00 0001
     7f00 0001 0050 872f 7333 6126 0000 0000
     5004 0000 5605 0000
10:43:37.904397 Diablo.34587 > Diablo.408: . win
4096
     4500 0028 e3bb 0000 3706 a212 7f00 0001
     7f00 0001 871b 0198 0000 0000 0000 0000
     5000 1000 192f 0000
10:43:37.904397 Diablo.408 > Diablo.34587: R
0:0(0) ack 0 win 0 (DF)
     4500 0028 0000 4000 ff06 7dcd 7f00 0001
     7f00 0001 0198 871b 0000 0000 0000 0000
     5014 0000 291b 0000
```

You do not need the complete three-way-handshake, thus, stealth scanning (like that mentioned) is less conspicuous than TCP scanning. IDS should detect those abnormalities in all cases (XMAS and NULL)...

FTP bounce:

With some ftpds the PORT command can be abused to establish an arbitrary connection from the ftp server to another machine. But first, a little overview how this "normally" happens. First, the client makes a connection to the ftp server (port 21), the ftp server "creates" a second connection back to the client (to be able to send back data). For this second connection you use the PORT command. The interesting thing here is that the command contains the IP and the port (which is to be opened) of the client. Subsequently, the server creates a connection, where source port 20 and the destination port are the ports specified by the PORT command. The point of attack is the PORT command with which you can manipulate the port of the (supposed) client to connect to the victim instead of our host. After the IP and port were manipulated you can initialize the actual traffic by "list" or "get". Now, you check the answer of ftp, because if we get a "425: Can't build data connection: Connection refused" this specified port is closed. If we receive "150 : File status okay about to open data connection" or "226: Closing data connection. Requested file action successful (for example, file transfer or file abort)" instead as an answer we know that the specified port is open.

Nmap parameter: -b

Fragmented packets:

This method uses the fragmentation of an IP packet by TCP. Normally, fragmentation occurs when the datagrams are bigger than the possible size, this size limitation is called MTU (Maximum Transmission Unit). Fragmented datagrams are put together at the end of a node. This behaviour can be abused. Not every IDS/firewall works with fragmentation, i.e., there are sometimes errors with fragmented packets. Instead of sending our packet, normally, we subdivide it (in fragments). These contain "common" data like source IP, destination IP, source port... Now, it is possible that the running firewall/IDS has problems with putting together fragments. These problems can manifest themselves in different ways, either it comes to a crash of the whole system or the packet goes through. Our packet could possibly get through because the putting together was erroneous and the packet was falsely specified "harmless". Sometimes, not all fragments are checked properly, i.e., only a certain fragment is checked so that our packet gets through again. With this technique you can scan less conspicuous as the traffic could be marked "harmless" and would not start an alarm. On the other hand this theory depends on the IDS (firewall) having problems with processing and putting together fragments.

Reverse Ident Scanning:

First, a section from RFC 1413, the RFC for Identification Protocol: "The Identification Protocol (a.k.a., "ident", a.k.a., "the Ident Protocol") provides a means to determine the identity of a user of a particular TCP connection. Given a TCP port number pair, it returns a character string which identifies the owner of that connection on the server's system." Reverse Ident Scanning uses identd to ask for the owner of the running process. This technique serves above all to find daemons which run as root in order to attack precisely these daemons.

```
----------       --------------     ----------
| YOU    | <---> | Dumb Host  | <--->| TARGET |
----------       --------------     ----------
```

Here Dumb Host should have as little traffic as possible. The reason for this will be more clear in the end. Why is Dumb Host used, why do we need one at all? Ok, this question leads to the actual attack and with this also to the explanation why a dumb host is necessary. In order that you can find out if a port of "TARGET" is open or closed you should examine the IP ID field of Dumb Host. For this, Dumb Host is sent a packet (echo request) and with its reply you can read the ID field, respectively the value of the ID field. Subsequently, you can send TARGET a packet where the source address is that of Dumb Host. The answer of TARGET is then received by our dumb host. If he receives a SYN/ACK of TARGET it means that the port is open. As an answer our dumb host will then send a packet were RST is flagged. If the dumb host receives a RST/ACK of the target machine it sends no answer to TARGET. To find out which answer Dumb Host got from TARGET we send Dumb Host another ping. If the port of the target was open and sent back RST the IP ID field of Dumb Host will be incremented, with port closed and nothing happens. By reading the new IP ID value of Dumb Host you can detect if the ports were open or closed. Now, it is hopefully clearer why we use a dumb host, i.e., a host with little traffic. If there is too much traffic on the host it would be more difficult to specify which ports are open (at TARGET), the probability of getting it wrong would be higher at higher traffic.

Fingerprint OS detection:
Fingerprint OS detection aims to detect the operating on the server. Most new scanners deliver not only, e.g., "Linux" or "Solaris" for an answer but a specification of versions (of the particular operating system). For this, you make a "fingerprint" (profile) of the operating system. These days, you cannot trust banners of telnet, ftp (see above) as they can be changed and manipulated. If the attacker finds out the exact version of the target host he can build scripts, exploits, ... for it and continue with his attack. This technique exploits the fact that operating systems differ in small details (what an insight ;). As there are a lot of documents on this technique I will only give a short overview over the most important test possibilities:

- FIN test: If a host receives a FIN packet at an open port it should rather not answer (RFC 793) but there are also exceptions, e.g., with MS Windows, BSDI, CISCO, MPS, ... which send a RESET for an answer.

- ACK sequential numbers: When sending FIN|PSH|URG to a closed port the sequential number of the ACK packet is set to the own sequential number mostly. But here also Windows makes an exception ;). Windows sends back the own sequential number +1 ...

- BOGUS flag test: If an undefined TCP Flag is used in the TCP header (in a SYN packet), Linux hosts < 2.0.35 adopt these flag settings. Other OSs reset when receiving a SYN+BOGUS packet...

- TCP initial window: Most operating systems use (almost) constant window sizes (of the reply packets). E.g., AIX delivers 0x3F25, MS NT5, OpenBSD and FreeBSD use 0x402E....

- Don't fragment bit test: Some operating systems differ in setting this bit in some packets or not. Thus, you can additionally differentiate which OS is available...

- TCP options test: The basis of this test is simply told: You send an inquiry to the particular host, set diverse options and look at the answer if there are also these options set. Those options contained in the answer are supported... As the case may be with the operating system (and version) certain options are in principal not supported, others are. Thus, the used operating system can be specified more exactly.

Nmap parameter : -O

There are many teste not discussed here but there are enough documents on the net about fingerprint OS detection. Fyodor (NMAP) has written a small paper about this. This part should only give a small overview about widespread scan techniques. For someone who works with protocol anomaly detection there were surely several starting points (certain flag combinations which you have to consider as anormal...).

Other ICMP related stuff

Aside from the mentioned echo reply/request ICMP supports further message types with which you can collect further information about the network (so-called NON - ECHO – REQUESTS):

ICMP Time Stamp Request / Reply (RFC 792):
The actual meaning of Time Stamp Requests (type 13) is to get the time settings of a remote system. If the remote host receives a Time Stamp Request it sends back a Time Stamp Reply (type 14). First, the structure of a time stamp (relative to RFC):

```
------------------------------------------------
|     Type  |      Code       |    Checksum     |
------------------------------------------------
|     Representer      |       Sequence         |
------------------------------------------------
|             Originate Timestamp               |
------------------------------------------------
|               Receive Stamp                   |
------------------------------------------------
|              Transmit Timestamp               |
------------------------------------------------
```

Before I get to the use of a Time Stamp Request for an attacker I tell you some basics about the time stamp. For us, only the last three "fields" are of importance. Here, the respective section in the RFC:

```
"The Originate Timestamp is the time the sender
last touched the  message before sending it, the
Receive Timestamp is the time the echoer first
touched it on receipt, and the Transmit
Timestamp is  the time the echoer last touched
the message on sending it."
```

As said in the RFC the sent back timestamp is the count of milliseconds since midnight UT (GMT).

Of what use is a time stamp request/reply for us?

If you are sent back a time stamp reply you would know that the host is reachable and on the other hand, with the Originate Stamp and the Receive Stamp you can estimate the load of the network (the difference is, of course, dependant on cables used, cards, ...).

At last, a tcpdump trace:

```
11:38:37.898253 Diablo > Diablo: icmp: time stamp
query id 53763 seq 64548
 (ttl 254, id 13170, len 40)
```

```
    4500 0028 3372 0000 fe01 8b60 7f00 0001
    7f00 0001 0d00 61fb d203 fc24 0211 c0ca
    0000 0000 0000 0000

 11:38:37.898253 Diablo > Diablo: icmp: time stamp
reply id 53763 seq 64548 :
 org 0x211c0ca recv 0x211c0ca xmit 0x211c0ca (DF)
(ttl 255, id 0, len 40)
    4500 0028 0000 4000 ff01 7dd2 7f00 0001
    7f00 0001 0e00 db43 d203 fc24 0211 c0ca
    0211 c0ca 0211 c0ca
```

ICMP Information Request / Reply (RFC 792): "This message may be sent with the source network in the IP header source and destination address fields zero (which means "this" network). The replying IP module should send the reply with the addresses fully specified. This message is a way for a host to find out the number of the network it is on. "

So, an Information Request (type 15) has the meaning to get the network number of the host which sent the request.

```
"The address of the source in an information
request message will be the destination of the
information reply message.  To form an
information reply message, the source and
destination addresses are simply reversed,..."
```

With an Information Reply (type 16) you take the source IP of the Information Request as destination IP of the reply (simply put: you send the reply to the host which requested the information). As source Ip of the reply you take the destination IP of the request...

Normally, the situation is that the sender of an Information Request sets 0 as destination address (which means "this network"). But there is also the possibility to set destination and source IP to 0 (when sending the request). In this case, the Information Reply would receive the network number of the host in the source and the destination address field, i.e., if the source address field in the request does not equal 0 the network number of the host would only be sent back in the source IP field of the reply.

```
---------------------------------------------------
|   Type   |     Code     |   Checksum             |
---------------------------------------------------
|   Pointer      |       Sequence                  |
---------------------------------------------------
```

It seems that you could only send an information request within the network (see above) but that does not have to be. Some operating systems also answer an information request where the destination IP is not in the same network. In such an information reply we would receive the IP of the host (and not the network number).

At the end, again a short tcpdump trace:

```
11:42:35.608253 Diablo > Diablo: icmp: information
request (ttl 255,
 id 13170, len 28)
    4500 001c 3372 0000 ff01 8a6c 7f00 0001
    7f00 0001 0f00 1afc d603 0000
11:42:36.608253 Diablo > Diablo: icmp: information
request (ttl 255,
 id 13170, len 28)
    4500 001c 3372 0000 ff01 8a6c 7f00 0001
    7f00 0001 0f00 19fc d603 0100
```

ICMP Address Mask Request / Reply (RFC 950): The Address Mask Request (type 17) has been described in another RFC, for more information look at RFC950 and not in RFC792. The meaning and use of an Address Mask Request is to get the subnetmask of a connected network. If a gateway receives such a request it should send back relevant information to the respective node (Address Mask Reply - type 18)

```
---------------------------------------------------
| Type  |       Code       |  Checksum             |
---------------------------------------------------
| Flag               |        Sequence             |
---------------------------------------------------
|                Address Mask                      |
---------------------------------------------------
```

With this, you can not only discover hosts in the network (which are online) but also get to know about the network configuration with further tests...

Tcpdump trace:

```
11:45:26.678253 Diablo > Diablo: icmp: address mask
request (ttl 254, id
 13170, len 32)
    4500 0020 3372 0000 fe01 8b68 7f00 0001
    7f00 0001 1100 edd7 dc03 2524 0000 0000
```

I hope, this section showed you that you have more possibilities to get information about a network and that it does not always have to be the "normal" ping... In the respective RFCs are mostly such hints which describe what you should consider if you want to "support" the different types of request... Developers of ("real") IDSs should also consider such issues. If they should be supported you should consider that it works (read RFCs). But even this is not the end as you can read in the next section...

Even more information about the target: The use of packet filters (or more common: firewalls) is surely not very special, these days. Also the use of so called firewall modules in IDSs can be found more and more often. Often, an attacker does not have the possibilities to use some of the discussed scans as they are blocked, filtered... The aim of this small section is to show possibilities with which you can work out some of the filter/firewall rules of the target.

The principle of this idea is quite simple as you try to provoke ICMP error messages, respectively send "illegal" packets from which you can draw conclusions about the set of rules.

```
"If the gateway or host processing a datagram
finds a problem with the header parameters such
that it cannot complete processing the datagram
it must discard the datagram.  One potential
source of such a problem is with incorrect
arguments in an option.  The gateway or host may
also notify the source host via the parameter
problem message.  This message is only sent if
the error caused the datagram to be discarded."
```

This section is from RFC 792 and is part of the description of the so called parameter problem (type

12). As you can see in this section a reason for the message "Parameter Problem" can be a false IP header, i.e., if we would send a packet with a false IP header to a host we should actually get back this error message. This error message has one additional advantage as support of this error message is "recommended" in RFC 1122:

```
"A host SHOULD generate Parameter Problem
messages.  An incoming Parameter Problem message
MUST be passed to the transport layer, and it
MAY be reported to the user.

DISCUSSION:
The ICMP Parameter Problem message is sent to
the source host for any problem not specifically
covered by another ICMP message.  Receipt of a
Parameter Problem message generally indicates
some local or remote implementation error."
```

On the whole, this gives a very good possibility to detect hosts in a network (for the mentioned reasons).

Additionally, I refer to RFC 1812:

```
" 4.3.3.5 Parameter Problem

A router MUST generate a Parameter Problem message
for any error not specifically covered by another
ICMP message.  The IP header field or IP option
including the byte indicated by the pointer field
MUST be included unchanged in the IP header
returned with this ICMP message.

Section [4.3.2] defines an exception to this
requirement.  "
```

Nevertheless, different routers interpret this section (and also others) differently, for which reason it is not clear that a Parameter Problem is generated.

An IDS (respectively firewalls) should check fields in the IP header anyway as it can happen at times that you receive a packet with a false header, but this should happen rarely. An attacker who scans a whole network with this method (or any specified IP range) knows that the firewall/packet filter... does not block, filter this error message. But if you do not get a Parameter Problem you know at least that this message is blocked.

This method to find out the set of rules is only the first step (the method is above all an alternative to a "normal" ping). To get an as exact as possible access control list (ACL) of possible filtering/firewall software we should get further information of the topology. To do this, you could, e.g., send the different ICMP message types to single hosts (with false IP header) and wait for a notification of a Parameter Problem. If there is a notification of a Parameter Problem on host "X" this means for us that the host does not filter this ICMP message type (and the respective host is reachable). Additionally, this means for us that the erroneous information in the IP header is not checked. If we get no Parameter Problem we can also make several conclusions. On the one hand, it would be possible that a filter filters/blocks the ICMP message type and on the other hand it is possible that the router checks the header and does not let forward this anormality, etc.. As you can see, with both results you can make conclusions on possible filter rules, last and not least you get a wide representation of what is allowed and what not. Further possibilities could be to vary used protocols (TCP, UDP, ...) so that you could find out further rules. There would be, e.g., the possibility that a certain protocol is blocked/filtered.

I think the abstract on the Parameter Problem is now clear so that we can go on looking at further error messages.

The following section is again taken from RFC 1812 and describes what Destination Unreachable error is and what can be the reason for this error:

```
"The ICMP Destination Unreachable message is sent
by a router in response to a packet which it cannot
forward because the destination (or next hop) is
unreachable or a service is unavailable.  Examples
of such cases include a message addressed to a host
which is not there and therefore does not respond
to ARP requests, and messages addressed to network
prefixes for which the router has no valid route.

A router MUST be able to generate ICMP Destination
Unreachable messages and SHOULD choose a response
code that most closely matches the reason the
message is being generated.

0 = Network Unreachable - generated by a router if
    a forwarding path (route) to the destination
    network is not available;

1 = Host Unreachable - generated by a router if a
    forwarding path (route) to the destination host
    on a directly connected network is not available
    (does not respond to ARP);

2 = Protocol Unreachable - generated if the
    transport protocol designated in a datagram is
    not supported in the transport layer of the
    final destination;

3 = Port Unreachable - generated if the designated
    transport protocol (e.g., UDP) is unable to
    demultiplex the datagram in the transport layer
    of the final destination but has no protocol
    mechanism to inform the sender;

4 = Fragmentation Needed and DF Set - generated if
    a router needs to fragment a datagram but cannot
    since the DF flag is set;

5 = Source Route Failed - generated if a router
    cannot forward a packet to the next hop in a
    source route option;

6 = Destination Network Unknown - This code SHOULD
    NOT be generated since it would imply on the
    part of the router that the destination network
    does not exist (net unreachable code 0 SHOULD be
    used in place of code 6);

7 = Destination Host Unknown - generated only when
    a router candetermine (from link layer advice)
    that the destination hostdoes not exist;

11 = Network Unreachable For Type Of Service -
    generated by a router if a forwarding path
    (route) to the destination network with the
    requested or default TOS is not available;

12 = Host Unreachable For Type Of Service -
    generated if a router cannot forward a packet
    because its route(s) to the destination do not
    match either the TOS requested in the datagram
    or the default TOS (0)."
```

If we now, e.g., try to send a packet to any port which uses a protocol that does not exist, there should actually be a notification about a Destination Unreachable with code value 2 (Protocol Unreachable). Further, with this example you should know first which protocols are "admitted". This, you can find out when you look at your /etc/protocols. After the installation on one of my hosts the /etc/protocols looked like this, e.g.:

```
--------------- /etc/protocols ----------------
# /etc/protocols:
# $Id: protocols,v 1.2 2001/01/29 17:29:30 notting Exp $
#
# Internet (IP) protocols
#
#       from: @(#)protocols     5.1 (Berkeley) 4/17/89
#
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July
1992).
#
# See also http://www.isi.edu/in-notes/iana/assignments/protocol-
numbers

ip        0       IP              # internet protocol, pseudo
protocol number
#hopopt  0       HOPOPT          # hop-by-hop options for ipv6
icmp      1       ICMP            # internet control message protocol
igmp      2       IGMP            # internet group management
protocol
ggp       3       GGP             # gateway-gateway protocol
ipencap   4       IP-ENCAP        # IP encapsulated in IP (officially
``IP'')
st        5       ST              # ST datagram mode
tcp       6       TCP             # transmission control protocol
cbt       7       CBT             # CBT, Tony Ballardie
egp       8       EGP             # exterior gateway protocol
igp       9       IGP             # any private interior gateway
                                  # (Cisco: for IGRP)
bbn-rcc  10       BBN-RCC-MON     # BBN RCC Monitoring
nvp      11       NVP-II          # Network Voice Protocol
pup      12       PUP             # PARC universal packet protocol
argus    13       ARGUS           # ARGUS
emcon    14       EMCON           # EMCON
xnet     15       XNET            # Cross Net Debugger
chaos    16       CHAOS           # Chaos
udp      17       UDP             # user datagram protocol
mux      18       MUX             # Multiplexing protocol
dcn      19       DCN-MEAS        # DCN Measurement Subsystems
hmp      20       HMP             # host monitoring protocol
prm      21       PRM             # packet radio measurement protocol
xns-idp  22       XNS-IDP         # Xerox NS IDP
trunk-1  23       TRUNK-1         # Trunk-1
trunk-2  24       TRUNK-2         # Trunk-2
leaf-1   25       LEAF-1          # Leaf-1
leaf-2   26       LEAF-2          # Leaf-2
rdp      27       RDP             # "reliable datagram" protocol
irtp     28       IRTP            # Internet Reliable Transaction
Protocol
iso-tp4  29       ISO-TP4         # ISO Transport Protocol Class 4
netblt   30       NETBLT          # Bulk Data Transfer Protocol
mfe-nsp  31       MFE-NSP         # MFE Network Services Protocol
merit-inp         32    MERIT-INP       # MERIT Internodal Protocol
sep      33       SEP             # Sequential Exchange Protocol
3pc      34       3PC             # Third Party Connect Protocol
idpr     35       IDPR            # Inter-Domain Policy Routing
Protocol
xtp      36       XTP             # Xpress Tranfer Protocol
ddp      37       DDP             # Datagram Delivery Protocol
idpr-cmtp         38    IDPR-CMTP       # IDPR Control Message
Transport Proto
tp++     39       TP++            # TP++ Transport Protocol
il       40       IL              # IL Transport Protocol
ipv6     41       IPv6            # IPv6
sdrp     42       SDRP            # Source Demand Routing Protocol
ipv6-route        43    IPv6-Route      # Routing Header for IPv6
ipv6-frag         44    IPv6-Frag       # Fragment Header for IPv6
idrp     45       IDRP            # Inter-Domain Routing Protocol
rsvp     46       RSVP            # Resource ReSerVation Protocol
gre      47       GRE             # Generic Routing Encapsulation
mhrp     48       MHRP            # Mobile Host Routing Protocol
bna      49       BNA             # BNA
ipv6-crypt        50    IPv6-Crypt      # Encryption Header for IPv6
ipv6-auth         51    IPv6-Auth       # Authentication Header for
IPv6
i-nlsp   52       I-NLSP          # Integrated Net Layer Security
TUBA
swipe    53       SWIPE           # IP with Encryption
narp     54       NARP            # NBMA Address Resolution Protocol
mobile   55       MOBILE          # IP Mobility
tlsp     56       TLSP            # Transport Layer Security Protocol
skip     57       SKIP            # SKIP
ipv6-icmp         58    IPv6-ICMP       # ICMP for IPv6
ipv6-nonxt        59    IPv6-NoNxt      # No Next Header for IPv6
ipv6-opts         60    IPv6-Opts       # Destination Options for
IPv6
#        61                       # any host internal protocol
cftp     62       CFTP            # CFTP
#        63                       # any local network
sat-expak         64      SAT-EXPAK       # SATNET and Backroom EXPAK
kryptolan         65      KRYPTOLAN       # Kryptolan
rvd      66       RVD             # MIT Remote Virtual Disk Protocol
ippc     67       IPPC            # Internet Pluribus Packet Core
#        68                       # any distributed file system
sat-mon  69       SAT-MON         # SATNET Monitoring
visa     70       VISA            # VISA Protocol
ipcv     71       IPCV            # Internet Packet Core Utility
cpnx     72       CPNX            # Computer Protocol Network
Executive
cphb     73       CPHB            # Computer Protocol Heart Beat
wsn      74       WSN             # Wang Span Network
pvp      75       PVP             # Packet Video Protocol
br-sat-mon        76    BR-SAT-MON      # Backroom SATNET
Monitoring
sun-nd   77       SUN-ND          # SUN ND PROTOCOL-Temporary
wb-mon   78       WB-MON          # WIDEBAND Monitoring
wb-expak          79    WB-EXPAK        # WIDEBAND EXPAK
iso-ip   80       ISO-IP          # ISO Internet Protocol
vmtp     81       VMTP            # Versatile Message Transport
secure-vmtp       82    SECURE-VMTP     # SECURE-VMTP
vines    83       VINES           # VINES
ttp      84       TTP             # TTP
nsfnet-igp        85    NSFNET-IGP      # NSFNET-IGP
dgp      86       DGP             # Dissimilar Gateway Protocol
tcf      87       TCF             # TCF
eigrp    88       EIGRP           # Enhanced Inter
```

What if the host does not send back a Protocol Unreachable (though you used a protocol that actually does not exist)? This can have two reasons. First, it could be that you have found an AIX, HP-UX or Digital Unix machine or the set of rules of the host does not allow access to these ports. So, at first, verify what kind of host you scan (among other possibilities with fingerprint OS detection) or else you can assume that it gets filtered/blocked.

Note: The detection of such an attack is quite simple (and belongs to the section of Protocol Anomaly Detection). Anomaly Detection will be discussed in the next chapter (Possibilities of analysis), now, it is sufficient to know that traffic is searched for "anormalities" and the use of a non-existant protocol belongs to these "anormalities".

■ Denial of Service (DoS)

DoS (Denial of Service) attacks normally aim to paralyze the target server so that it is not reachable for some time. There are so many techniques by wich you can "exhaust" the resources of the target host. In the following I present you the most common:

ICMP Flooding:

ICMP Flooding "uses" ICMP. If a host receives an ICMP echo request it will normally try to answer with an ICMP echo reply. This behaviour is used with ICMP Flooding: If you send the victim too many echo requests its resources will be working on it to reply to these requests (as echo replies). As the IP of the attacker is mostly additionally spoofed it does not get the replies but someone else.

SYN Flooding:

To understand SYN Flooding, I present again the "normal" three-way-handshake:

```
------------------        SYN        -------------
| Host     A      |  ------------->  | Host     B |
------------------                   -------------

------------------      ACK/SYN      -------------
| Host     A      |  <-------------  | Host     B |
------------------                   -------------

------------------        ACK        -------------
| Host     A      |  ------------->  | Host     B |
------------------                   -------------
```

Host A sends Host B a SYN to say that he wants a connection, B answers with ACK/SYN and waits for the final ACK with which the connection would complete. But what if the last ACK is not sent? If B sends back its SYN/ACK it waits (as said before) for the ACK of A, until then it will be queued in the connectin queue of "B". If the connection is complete (A sent B an ACK) it will be removed from the connection queue. But as mostly the IP address is spoofed B never receives an ACK (it should be so). So, you can "fill" the connection queue as the host cannot make additonal connections.

UDP Flooding:
With this flooding attack the target server is flooded with UDP packets. If you send a UDP packet to a port on the target server it will first check which service is responsible for this "request". Now, you choose random ports to which you send the packets to rise the probability that a "Port Unreachable" is sent by ICMP. As the result of such a flood the performance of a network segment suffers (often) considerably.

Land:
Later you will see a filter which detects if there is a Land attack and can initiate counteractive measures, but first the actual attack. A Land attack has similar source and destination IPs. When sending, e.g., SYN packets (with same source/destination IP) to an open port there will start a race condition on the victims host which leads to paralysis of the whole system. Here, a trace of a Land attack:

```
 23/06/02  23:12:48  194.157.1.1  80  ->
194.157.1.1
 23/06/02  23:14:57  194.157.1.1  31337  ->
194.157.1.1
```

As you can see again, source and destination IP are the same.

```
12:35:26.916369
192.168.38.110.135>192.168.38.110.135: udp
46[tos0x3,ECT,CE]
    4503 004a 96ac 0000 4011 15c7 c0a8 266e
    c0a8 266e 0087 0087 0036 8433 6920 616d
    206c 616d 6520 646f 7320 6b69 6420 6275
    7420 6920 7265
12:35:26.916566
192.168.38.110.135>192.168.38.110.135: udp
46[tos0x3,ECT,CE]
    4503 004a 2923 0000 4011 8350 c0a8 266e
    c0a8 266e 0087 0087 0036 8433 6920 616d
    206c 616d 6520 646f 7320 6b69 6420 6275
    7420 6920 7265
12:35:26.916682
192.168.38.110.135>192.168.38.110.135:udp
46[tos0x3,ECT,CE]
    4503 004a 50a0 0000 4011 5bd3 c0a8 266e
    c0a8 266e 0087 0087 0036 8433 6920 616d
    206c 616d 6520 646f 7320 6b69 6420 6275
    7420 6920 7265
```

The attack which you can see here is also called Snork.

Teardrop:
Here, the possibility of fragmentation of IP packets is used. As you see in the description of the scans there is a fragmentation if datagrams are bigger than the size limit, this size limitation is called MTU (Maximum Transmission Unit). With this attack, fragments overlap and as a result of this overlap many OSs have (had) problems and mostly crashed the system

```
10:13:32.104203 10.10.10.10.53>192.168.1.3.53: udp
28(frag 242:36@0+)(ttl64)
    4500 0038 00f2 2000 4011 8404 0a0a 0a0a
    c0a8 0103 0035 0035 0024 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000 0000
    0000 0000 0000

10:13:32.104272 10.10.10.10 >192.168.1.3: udp
28(frag 242:4@24)(ttl 64)
    4500 0018 00f2 0003 4011 a421 0a0a 0a0a
    c0a8 0103 0035 0035 0024 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000
```

Ping of Death:
Here, too, fragmentation of the IP packets again plays a role. Here, I will go into more detail (only a little ;) on fragmentation. As told before, fragmentation means that the size of datagrams is "reduced", whereas single fragments are not to be bigger than the MTU. When fragmenting you should actually consider that you cannot fragment as you want, respectively that certain fields have to exist in every fragment. So, every fragment has to contain, e.g., the IP protocol header to choose the right route. That the router can rebuild fragments to a datagram every fragment receives a 16 bit flag (of the original, "big" datagram"). With this 16 bit flag it is later possible to sort the single fragments to the right datagram. Additionally, there is a fragment offset which tells at which position the fragment was in the original datagram. But the position is in 8 octet units (as the position is measured in 8 octet units). Additionally, the "more-bit" shows if further fragments of this datagram follow or not. If it is set to 1 further will follow if set to 0 it was the last fragment of the datagram. Now, we get to the actual ping-of-death attack. Ping-of-death is given an offset of the last fragment for which is: offset + fragment size > 65535 bytes. Thus, it is possible to flood internal 16 bit variables which would result, e.g., in a system crash.

```
17:43:58.431 pinger > target: icmp echo request (frag 4321: 380@0+)
17:43:58.431 pinger > target: (frag 4321: 380@2656+)
17:43:58.431 pinger > target: (frag 4321: 380@3040+)
17:43:58.431 pinger > target: (frag 4321: 380@3416+)
```

Smurf:
A ping is sent to the broadcast address, better said, many pings are sent. Packets sent to the broadcast address are sent to all hosts of the network. If you send many pings (ICMP echo requests) to the broadcast address (e.g., 10000 per second) and you have a relatively large network with 1000 hosts, it means that there are 10000 * 1000 ICMP echo replies per second at the victim's host, that means 10000000 ICMP echo replies.

```
09:28:28.666073 179.135.168.43>256.256.30.255:
icmp: echo request (DF)
    4500 001c c014 4000 1e01 6172 b387 a82b
    c0a8 1eff 0800 f7ff 0000 0000 0000 0000
    0000 0000 0000 0000 0000 0000 0000
09:28:28.696073 68.90.226.250>256.256.30.255: icmp:
echo request (DF)
    4500 001c c015 4000 1e01 95cf 445a e2fa
    c0a8 1eff 0800 f7ff 0000 0000 3136 3803
    3133 3503 3137 3907 696e 2d61 6464
09:28:28.726073 138.98.10.247>256.256.30.255:
icmp: echo request (DF)
    4500 001c c016 4000 1e01 27ca 8a62 0af7
    c0a8 1eff 0800 f7ff 0000 0000 0332 3236
    3938 0331 3638 0769 6e2d 6164 6472
```

```
 09:28:28.756073 130.113.202.100 > 256.256.30.255:
icmp: echo request (DF)
    4500 001c c017 4000 1e01 704c 8271ca64
    c0a8 1eff 0800 f7ff 0000 0000 0231 3002
    3938 0331 3338 0769 6e2d 6164 6472
    ...
```

For some time, there also existed DDoS attacks (Distributed Denial of Service). As the name suggests it is a distributed/networked DoS attack. The attacker (client) looks for other hosts/networks... which are easy to exloit. These first infected hosts are the so called handlers. Handlers infect further hosts/networks, these infected hosts are then called agents, i.e. as a datagram:

```
                 ---------
                 | YOU   |
                 ---------
                     |
   ----------------------------------
   |                 |                 |
Handler           Handler           Handler
   |                 |                 |
--------          --------          --------
|       |         |       |         |       |
Agent  Agent      Agent  Agent      Agent  Agent
```

Later, agents execute attacks.

This is the first article out of 2. We will continue in the next issue of LinuxFocus.

This article is re-printed with permission. The originals can be found at:

http://www.linuxfocus.org/English/May2003/article292.shtml

FreeBSD 5.1 Release Process

<freebsd-qa@FreeBSD.org>
Last modified: May 31, 2003.

Introduction

This is a specific schedule for the release of FreeBSD 5.1. For more general information about the release engineering process, please see the Release Engineering section (http://www.freebsd.org/releng/index.html) of the web site.

General discussions about the release engineering process or quality assurance issues should be sent to the public FreeBSD-qa mailing list (FreeBSD-qa@FreeBSD.org). MFC requests should be sent to re@FreeBSD.org.

One of the major features of FreeBSD 5.1 will be further refinement of the re-worked SMP support introduced in FreeBSD 5.0. For specific information about the progress towards 5.1-RELEASE in this area, please see the SMP Project page (http://www.freebsd.org/smp/index.html).

FreeBSD 5.1 will continue to be released from the 5-CURRENT development stream. For more details about the milestones for reaching 5-STABLE, see the 5-STABLE Roadmap page (http://www.freebsd.org/doc/en/articles/5-roadmap).

The current release engineering TODO list is also available (http://www.freebsd.org/release/5.1R/todo.html). This list is updated periodically through the release cycle.

Schedule

Action	Expected	Actual	Description
-CURRENT code freeze	05/05/03	05/05/03	The src/ code freeze for 5.1. Commits to HEAD require re@FreeBSD.org approval.
5.1-BETA	05/05/03	15/05/03	5.1-BETA release of x86, alpha, sparc64, and ia64.
5.1-BETA2	19/05/03	22/05/03	Second 5.1-BETA release of x86, alpha, sparc64, and ia64.
RELENG_5_1 branched	30/05/03	31/05/03	Branch of src/ from HEAD for the release. Note: no branch for RELENG_5 will happen at this time.
Turn off debugging for RELENG_5_1	30/05/03	31/05/03	Turn off WITNESS, INVARIANTS, and malloc debugging options similar to what was done for 5.0.
First release candidate	30/05/03	--	x86, alpha, sparc64, and ia64 images released and uploaded to ftp-master.FreeBSD.org.
src/ unfrozen	30/05/03	31/05/03	Unfreeze HEAD src. Continue to coordinate significant check-ins with re@FreeBSD.org in order to work towards 5-STABLE.
Ports tree tagged	30/05/03	27/05/03	Tentative date of RELEASE_5_1_0 tag for ports.
Version numbers bumped	02/06/03	--	The files listed at http://www.freebsd.org/doc/en_US.ISO8859-1/articles/releng/article.html#VERSIONBUMP are updated to reflect FreeBSD 5.1.

Action	Expected	Actual	Description
src/ tree tagged	02/06/03	--	Tag the RELENG_5_1 branch with RELENG_5_1_0_RELEASE.
doc/ tree tagged	02/06/03	30/05/03	Tag the doc/ tree with RELEASE_5_1_0.
Final builds	02/06/03	--	Start x86, alpha, sparc64, ia64, and pc98 builds.
Warn hubs@FreeBSD.org	02/06/03	--	Heads up email to hubs@FreeBSD.org to give admins time to prepare for the load spike to come. The site administrators have frequently requested advance notice for new ISOs.
Upload to ftp-master	04/06/03	--	Release and packages uploaded to ftp-master.FreeBSD.org.
FreeBSD 5.1 Released	05/06/03	--	FreeBSD 5.1 is announced to the mailing lists.
FreeBSD 5.1 Press Release	05/06/03	--	A formal press release statement is in the works and should be released at this time to the www.FreeBSD.org website and various tech publications.

This article is re-printed with permission. The originals can be found at:

http://www.freebsd.org/releases/5.1R/schedule.html

SCO-vs.-IBM: the Open Source Initiative Position Paper on the Complaint

Author: Eric S Raymond <esr@thyrsus.com>
Author: Rob Landley<rob@landley.net>

Introduction

The Open Source Initiative (OSI) is a 501(c)3 nonprofit educational association with offices in Palo Alto, California. OSI is one of the principal advocacy organizations of the open-source community, which is alleged in SCO/Caldera's complaint to have been beneficiary of tortious and illegal behavior by IBM.

The principal author of this position paper (Raymond) has been a Unix developer since 1982, is a technical specialist in systems programming technologies related to those at issue, and is a historian whose writings on the open-source community and Unix ([TNHD], [CATB], [TAOUP]) are widely considered authoritative both within the community and outside it. He has been since 1997 one of the leading theorists and (both in his individual capacity and as the president of OSI) one of the principal spokespersons/ambassadors for the open-source community.

While the authors are affiliated with the Linux community, our argument is also motivated by larger concerns. Unix, Linux, and the open-source movement are vital components of the Internet and the World Wide Web. SCO/Caldera's attempt to assert proprietary control of these technologies is an indirect but potent threat against the Internet and the culture that maintains it. What is at stake here is not just the disposition of a particular volume of computer code, but what amounts to a power grab against the future.

This document, originally proposed as a draft brief of amicus curiae, has been endorsed as an OSI position paper by OSI's Board of Directors. The Board has concluded on advice of counsel that OSI cannot seek amicus status in advance of pleadings. The option to seek amicus status at a future time remains open.

This document is an evolving work in progress. SCO/Caldera's complaint against IBM disparaged the work of thousands of individual open-source contributors. These contributors feel themselves personally and professionally wronged by SCO/Caldera's unfounded allegations. In the tradition of the open-source movement, hundreds of individuals are now sending in their patches to help inform and evolve the OSI's position.

Scope of the Position Paper

This position paper is written in specific response to SCO/Caldera's complaint [1] filed on the 6th of March 2003 in the Third Judicial District of Salt Lake County, State of Utah.

It is not within OSI's competence or knowledge to address the specifics of the business relationship between SCO/Caldera and IBM, or the terms of their contract. It is, however, very much within our competence to observe that SCO/Caldera's complaint depends critically on certain historical and technical assertions which are materially false and (apparently quite intentionally) misleading.

Unlike SCO/Caldera's complaint, we have provided direct hyperlinks to browseable versions of all the sources which back our facts.

In this position paper, we focus on the following allegations, and show that they are incorrect or fundamentally misleading:

Paragraph 1.(c)

"UNIX and SCO/UNIX are widely used in the corporate, or "enterprise" computing environment"

Paragraph 23

"Except for SCO, none of the primary UNIX vendors ever developed a UNIX 'flavor' to operate on an Intel-based processor chip set."

Paragraph 57

"When SCO acquired the UNIX assets from Novell in 1995, it acquired rights in and to all (1) underlying, original UNIX software code developed by AT&T Bell Laboratories,"

Paragraph 57

"When SCO acquired the UNIX assets from Novell in 1995, it acquired rights in and to all (1) underlying, original UNIX software code developed by AT&T Bell Laboratories, "

Paragraph 75:

"The name 'Linus' [sic] was taken from the person who introduced Linux to the computing world, Linus Torvalds."

Paragraph 78:

"The primary purpose of the GNU organization is to create free software based on valuable commercial software."

Paragraph 82:

"Virtually none of these software developers and hobbyists had access to enterprise-scale equipment and testing facilities for Linux development. "

Paragraph 84:

"Prior to IBM's involvement, Linux was the software equivalent of a bicycle. UNIX was the software equivalent of a luxury car. To make Linux of necessary quality for use by enterprise customers, it must be re-designed so that Linux also becomes the software equivalent of a luxury car. This re-design is not technologically feasible or even possible at the enterprise level without (1) a high degree of design coordination, (2) access to expensive and sophisticated design and testing equipment; (3) access to UNIX code, methods and concepts; (4) UNIX architectural experience; and (5) a very significant financial investment."

Paragraph 85:

"For example, Linux is currently capable of coordinating the simultaneous performance of 4 computer processors. UNIX, on the other hand, commonly links 16 processors and can successfully link up to 32 processors for simultaneous operation."

Paragraph 90:

"To accomplish the end of transforming the enterprise software market to a services-driven market, IBM set about to deliberately and improperly destroy the economic value of UNIX and particularly the economic value of UNIX on Intel-based processors. "

Paragraph 93:

"Rather, IBM is obligated not to open source AIX because it contains SCO's confidential and proprietary UNIX operating system."

Paragraph 94:

"Over time, IBM made a very substantial financing commitment to improperly put SCO's confidential and proprietary information into Linux, the free operating system."

Paragraph 99:

"The only way that the pathway is an 'eight-lane highway' for Linux to achieve the scalability, SMP support, fail-over capabilities and reliability of UNIX is by the improper extraction, use, and dissemination of the proprietary and confidential UNIX Software Code and libraries. Indeed, UNIX was able to achieve its status as the premiere operating system only after decades of hard work, beginning with the finest computer scientists at AT&T Bell Laboratories, plaintiff's predecessor in interest. "

OSI submits that these claims are uniformly without merit, and proposes to establish that in the remainder of this position paper.

Technically-inclined readers will probably wonder why various apparently relevant topics (such as Minix, or the GNU project, or the Bell Labs research versions, or other proprietary Unixes) are not covered. Please remember that this document is not a tutorial in Unix history; history that does not bear on SCO's allegations has been omitted.

Historical and technical background. The meaning of 'Unix'

The falseness of SCO/Caldera's allegations is partly cloaked by the fact that their complaint uses the term 'Unix' in three different ways.

Among technical people and computer programmers, 'Unix' describes a family of computer operating systems with common design elements, all patterned on (but not necessarily derivative works of) the ancestral Unix invented at Bell Labs in 1969. As SCO/Caldera observes in its complaint, Unix operating systems dominate serious computing, and have for more than twenty years. There have been hundreds of different Unixes in this sense, exhibiting variations analogous to dialects within a language. Fortunately, only a handful of the principal dialects are relevant to this lawsuit.

When we wish to be clear that this is the definition we are using, we will refer to 'Unix-family' operating systems. Use of the term 'Unix' to describe any Unix-family operating system was common before SCO/Caldera's acquisition of the historical Unix codebase in 1995; AT&T's lawyers strove against it in vain as far back as the early 1980s. When we use the term 'Unix' without qualification elsewhere in this paper, this is the sense we intend.

The term 'Unix' is sometimes also used (primarily by historians of computing) more strictly, to describe only those Unix-family operating systems which are derivative works of the original Bell Labs Unix. To avoid confusion, we shall call any operating system of this kind a 'genetic Unix'.

Legally, the term 'Unix' has been since 1992 a trademark of The Open Group[2], a technical standards organization, and describes any operating system (whether genetic-Unix or not) that has been verified to conform to the published Unix standard. We will refer to an operating system of this kind as a 'trademark Unix'. The required attribution is 'UNIX is a registered trademark of The Open Group'. [3] However, The Open Group's strict construction of the term 'Unix' is more honored in the breach than the observance.

Neither SCO/Caldera nor old SCO has ever owned the UNIX trademark. IBM neither requested nor required SCO's permission to call their AIX offering a Unix. That decision lies not with the adventitious owner of the historical Bell Labs source code, but with The Open Group.

The Linux operating system is Unix-family and generally referred to as a Unix, but is neither a genetic Unix nor a trademark Unix. Linux was independently created by Linus Torvalds in 1991[4], and most versions have not been put through the rather expensive process required to verify conformance with The Open Group standards.

Linux conformance to the trademark Unix standard can be demonstrated, however. It was done once by a Linux vendor in England named Lasermoon. But the Open Group's rules require re-testing any time the operating system changes; given the pace at which Linuxes evolve, the cost to maintain certification would have been prohibitive.

The name is spelled either as 'Unix' or 'UNIX'; its inventors prefer the former.

Linux and the advent of open-source programming

SCO/Caldera's complaint cannot be understood without reference to a seismic shift now occurring in the software industry. The root of the shift lies in the approximate doubling of hardware capacity every eighteen months which has been the trend since the mid-1970s. This means that the typical complexity of software designed to fully utilize state-of-the-art hardware also doubles every eighteen months, escalating the difficulties of software engineering to previously unimagined levels.

In the mid-1990s it began to be understood that the traditional production models for software were running out of steam, increasingly unable to produce an acceptably low defect rate at these escalating complexity levels. There was much talk of a "software crisis" and attempts to resolve it through various attempts at process improvement.

These attempts at process improvement consisted largely of introducing more formality, rigor, centralization, and statistical monitoring into the software-development process. They had honorable precedents in the systematization of assembly-line manufacturing and industrial process control in the 20th century. But producing software is not like producing automobiles or soap flakes. The analogy to industrial process control turned out to be fundamentally misleading, and all these attempts failed, merely adding additional cost to the process without reliably reducing defect rates.

Relief came from an unexpected quarter -- from the loose-knit community of programmers and engineers associated with the Internet and the Unix operating system. Since the 1960s, the Internet and Unix hackers[5] had been pioneering a style of software engineering which reversed the premises of industrial software development.

Instead of centralization in large programming teams, the Internet style used small distributed programming groups. Instead of process control and hierarchy, the Internet style used peer review and open standards. Most importantly, the Internet style abolished secrecy in favor of transparency and what came to be called "open source" code.

Early examples of this mode of development included Berkeley Unix from about 1977, the GNU project from 1983, and the X Consortium from 1983. All three flourished within the Unix community. When Linus Torvalds launched Linux in 1991 he was operating within a well-established tradition.

To the surprise of all concerned, after about 1997 it became apparent that this was the answer (or, at least, an important part of the answer) that the software industry had been looking for. Defect rates and costs associated with open-source software proved dramatically lower than for closed-source software[6]. The most skilled programmers flocked to the new mode. The explosive success of Linux, and IBM's adoption of it, is a consequence of the dynamism of open-source development. Caldera Systems International itself, the company now trading as SCO, was founded to ride the Linux wave.

We did not, however, use the term "seismic shift" casually. As with previous technological revolutions, one of the prompt effects has been what the economist Joseph Schumpeter famously called "creative destruction" -- to wreck the business models of a great many companies attached to the legacy model of closed-source development.

The evolution of today's software industry is confusing to many people because it is proceeding in exactly the opposite direction from previous technological revolutions. Previously, the rationalization of production has been associated with movement away from decentralized cottage industry towards a factory system organized around concentrations of capital. This time, the move is away from the factory system, towards a new form of artisanship and individualism critically enabled by cheap PCs and the Internet. Thus, Linux.

This process panics companies like SCO/Caldera and Microsoft who stand to lose everything if they fail to adapt, but it should not be viewed with alarm by any disinterested observer. What is actually happening is that the diseconomies of corporate scale are being competed out of software production -- the market is seeking a new and more efficient equilibrium.

SCO/Caldera's complaint is only a small piece of the fallout. There will be a lot more upheaval, and wailing and gnashing of teeth and waving of legal briefs, before this process fully resolves.

The Bell Labs codebase

There is a body of code and associated intellectual property (IP), originating in Bell Labs, which old SCO purchased from Novell in 1995. This IP had previously been owned by Unix Systems Laboratories (USL), and before that by AT&T. We will refer to this IP by its location of origin, as the Bell Labs code.

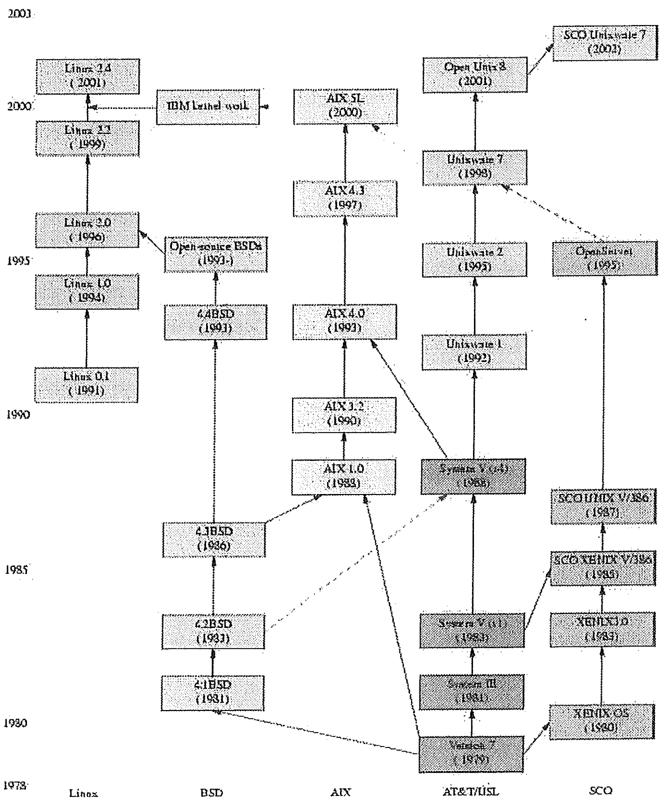
The contents of the historical Bell Labs codebase is well known; through most of its history, AT&T/USL/Novell tacitly ignored source license violations for non-commercial purposes, and many senior Unix programmers still possess bootleg copies of that source code. (The authors of this document could lay hands on several different releases without difficulty.) It is scarcely more difficult to obtain source copies of other major genetic Unixes such as AIX, HP-UX, and Solaris. The contents of these codebases, and the general pattern of copyrights and other intellectual-property claims in the source code, is therefore well known in the Unix community.

Until 19 May 2003, SCO/Caldera and old SCO before it made the Version 7 Unix source code (the root of all later versions of the Bell Labs codebase) available for free download on its website[7].

It is significant that SCO/Caldera has not asserted any direct IP claim over Linux on the basis of its ownership of the historical Bell Labs code. At best, ownership of the Bell Labs code could be construed to give SCO/Caldera certain proprietary rights with respect to genetic Unixes. Those rights are far more limited than SCO/Caldera would have one assume, a point which we will develop later in this position paper.

Relationships among the Unix variants at issue

Here is a schematic diagram of the relationships among the Unix variants at issue in this lawsuit:



Relationships among various Unixes.

This chart[8] shows the major Unix lineages at issue, indicating genetic relationships with arrows.

Vertical position on the chart indicates year of release. Horizontal position indicates which lineage the Unix belongs to. Though there are many Unix lineages not shown here, this chart covers all that are relevant to the issues raised in SCO/Caldera's complaint. We will briefly describe each lineage.

AT&T lineage

These are the Unixes directly derived from the Bell Labs codebase. The AT&T lineage through a succession of owners; first AT&T itself, then Unix

Systems Laboratories (USL), then Novell, then old SCO, now SCO/Caldera.

The early releases in this lineage (Version 7, System III, System V releases 1 through 3, and ending with Release 4 in 1988) were developed at AT&T itself[9] and are collectively known as "AT&T Unix".

The later releases are generally known as "UnixWare" after the brand name applied to them at Unix Systems Labs and Novell[10]. UnixWare is the product old SCO acquired in 1995, which Caldera acquired along with the server division of old SCO in 2001 and sold alongside of its Linux distribution.

All Unixes in the AT&T lineage are genetic Unixes, trademark Unixes, and proprietary.

AIX lineage

AIX is IBM's own Unix, originally developed 1987-1990. Genetic Unix, trademark Unix, proprietary.
SCO lineage

SCO's own versions of Unix date back to 1979, originally under the name XENIX. After old SCO acquired the Bell Labs codebase in 1995 it sold two Unixes: OpenServer based on the XENIX code, and a UnixWare descendant. All SCO Unixes are genetic Unix, trademark Unix, and proprietary.

BSD lineage

Berkeley System Distribution (BSD) is is a series of Unix releases developed primarily at the University of California at Berkeley but incorporating code from hundreds of contributors at universities and research laboratories worldwide.

BSD Unix is important for this lawsuit because its three modern variants are genetic Unix, but no proprietary rights to them can be claimed on the basis of ownership of the Bell Labs source code (for reasons we shall develop later in this position paper).

The BSDs are genetic Unixes, not trademark Unixes. They are open source.

Linux lineage

Launched in Finland in 1991, developed on the Internet, contributed to by thousands of people. Neither genetic nor trademark Unix ("Linux" is a trademark of Linus Torvalds). Open source.

Note that SCO OpenServer (at the end of the line of Unixes descended from XENIX) and SCO Unixware (at the end of the line of AT&T Unixes) are two different products of SCO; but XENIX was old SCO's original Unix, while UnixWare was what it picked up when it bought the AT&T codebase. The dashed arrow between UnixWare 2 and UnixWare 7 reflects the fact that it was a product designed to merge OpenServer 5 with UnixWare 2 (5 + 2 = 7) after old SCO bought the AT&T codebase.

The dashed red arrow from 4.2BSD to System V represents stolen property. AT&T, SCO/Caldera's predecessor in interest, took code from BSD Unix into System V, removing copyright notices and attributions in violation of the Berkeley license. We'll examine the consequences of this misappropriation later on.

The dashed blue arrow from UnixWare 7 to AIX 5L represents code incorporated into AIX from UnixWare 7 during the Monterey project. SCO/Caldera alleges that IBM misappropriated this code and merged it into Linux.

The arrow from the open-source BSDs to Linux 2.0 represents sharing of some device drivers and system utilities.

There is another major variant, Solaris, not shown on the chart. Solaris enters the discussion because it is the leading enterprise-capable proprietary Unix; its features therefore provide a standard upon which to ground assertions about which technologies fall under that rubric. It derives from System V and BSD. As of mid-2003 it is still the dominant Unix in the enterprise market (sold in conjunction with server hardware by Sun Microsystems). Genetic Unix, trademark Unix, proprietary.

Corporate history

In 1956, AT&T settled an antitrust action brought by the United States. Under the consent decree, AT&T's business was limited to "common carrier communications services." Bell Laboratories was required to license its patents on reasonable and non-discriminatory terms.[11]

The consent decree or "Final Judgement" was still in full force when work on Unix began at AT&T's Bell Laboratories in 1969.

Old SCO was founded in 1979 as a Unix porting and consulting company. The first SCO product offering, an Intel Unix port, was in 1983. [12] [13]

On January 1, 1984, the Bell System was broken up. [14] The old regime of the "Final Judgement" had been overthrown by the "Modified Final Judgement": AT&T could enter the software business. They did. That year, the corporation began to develop Unix as a commercial product.

In 1990, AT&T reorganized its business unit responsible for UNIX System V, the AT&T UNIX Software Operation, into a wholly-owned subsidiary, UNIX System Laboratories, Inc. (USL). [15] [16] The next year, AT&T sold a minority stake in USL to eleven selected companies. [17] [18] [19]

Late in 1991, the Univel joint venture was formed between Novell and USL. [20] [21]

In 1993, Novell bought USL. [22] [23] USL and Univel became the Novell UNIX Systems Group. [24]

Novell transferred the UNIX trademark to X/Open (later to become The Open Group).

In 1994, a group of Novell alumni formed Caldera Systems International with the backing of Novell's founder, Ray Noorda. Caldera was intended to be a Linux distributor, aiming at the business and enterprise market.

In 1995, Novell sold the UnixWare business to old SCO. [25] [26]

In 1998, old SCO, IBM, and Intel began cooperating on Project Monterey, a Unix port for the Intel Itanium, a 64-bit microprocessor.

Also in 1998, IBM ported its first application (DB2) to Linux.

In 2000, Caldera Systems International held an IPO as a Linux company.

Also in 2000, IBM began to support Linux kernel development.

In 2001, SCO split up. The rump of the company focused on its Tarantella product. The SCO brand, SCO OpenServer and the Bell Labs codebase were acquired by Caldera.

In 2002, Caldera began trading under the SCO name.

In the remainder of this document, we will use the following terminology:

Caldera Systems International	Caldera before the 2001 acquisition of the SCO server division.
old SCO	SCO before the 2001 acquisition.
SCO/Caldera	The merged company

SCO's history with open source

Caldera Systems International, the corporate entity now trading as SCO, was founded as an open-source centered company by a group of former Novell executives who were enthusiastic about Linux. Its only product was a Linux distribution. Caldera Linux was not a market success, however. It became an also-ran in the commercial Linux distribution market dominated by Red Hat (a North Carolina corporation) and SuSE (a German import). In mid-2002 co-founder president and CEO Ransom Love was pushed out during the shakeup that accompanied Caldera's acquisition of old SCO's server division.

Old SCO, prior to its acquisition by Caldera, did not produce its own Linux distribution, but became involved in open-source development between 1998 and 2000. It invested in Caldera and TurboLinux, bought out a popular on-line store called LinuxMall, and boasted of offering more open-source and Unix expertise than anyone else in the world.

In a press release at the time of the Project Monterey launch[27], Doug Michels (the co-founder of old SCO who remained at its head until Caldera Systems International acquired the brand in 2001) had this to say: "The whole idea of shared development has been ubiquitous in Unix for years. The Internet has magnified that and open source is bringing collaborative development to a new level." He observed, correctly, that it would be important for Unix vendors to continue their embrace of the open-source community, most notably Linux. At that time, SCO got it.

In a web page from 2000 [28], (since removed from their site) old SCO repeated this theme: "The concept of collaborative development and shared source has been ubiquitous in the UNIX system industry from the beginning. Today, the Internet has magnified that trend dramatically and led to the exciting phenomenon that is Linux."

The meaning of "enterprise scalability"

SCO/Caldera uses the term "enterprise computing" and various other derivatives in its complaint, but fails to define it. It is a marketing term suggesting very high operational reliability. The term is generally held to encompass the following technologies:

symmetric multi-processing (SMP)

The ability to allocate work in a computer among multiple processor chips in such a way that all are as efficiently utilized as possible, leading to completion of programs in the shortest possible time.

journaling file systems

Construction of an operating system's code for managing hard drives and other storage devices in such a way that data cannot be lost or garbled by power outages or most categories of hardware failure.

logical volume management (LVM)

The ability to make multiple physical disk storage devices appear to be a single large disk volume, incorporating redundant storage and error checking such that the failure of any one of the physical volumes still allows all the data to be recovered.

non-uniform memory access (NUMA)

A method of configuring a cluster of microprocessors in a multiprocessing system so that they can share memory locally, improving performance and the ability of the system to be expanded.

hot-swappable hardware

The capability to add and remove hardware while the system is running, without interrupting processing.

support for large memory address spaces via 64-bit processors

Large database

Other terms sometimes encountered include "transparent failover" and "high availability". These features are largely consequences of the application of the technologies specified above, with hardware that permits a computer system to detect and compensate for internal errors.

Solaris, the industry benchmark for a high-end enterprise-scalable Unix operating system, features all of these.

SCO/Caldera misrepresents its own history and position

SCO/Caldera's complaint is factually defective in that it implies claims about SCO/Caldera's business and technical capabilities that are untrue. It is, indeed, very cleverly crafted to deceive a reader without intimate knowledge of the technology and history of Unix; it gives false impressions by both the suppression of relevant facts, the ambiguous suggestion of falsehoods, and in a few instances by outright lying.

SCO/Caldera's claim to have been a significant enterprise player is false

SCO/Caldera's attempts to confuse the issues in this complaint begin early, in Paragraph 1.(c) where it asserts: "*UNIX and SCO/UNIX are widely used in the corporate, or 'enterprise' computing environment.*"

While this claim is literally true, it is misleading in that it fails to distinguish between the market share of old SCO's own Unixes (SCO OpenServer and Unixware) and those of competitors such as Sun, Hewlett-Packard, and IBM. By failing to so distinguish, it conveys the impression that old SCO's market share in the enterprise segment was significant, thus magnifying the putative harm done by IBM's alleged misconduct.

The truth is otherwise. Old SCO never had significant enterprise market share either before or after its purchase of the Bell Labs codebase from Novell. Their strength has been in franchise operations including McDonald's, Burger King, Pizza Hut, and Ground Round, which involve lots of parallel small deployments with no individual site requiring enterprise technology.

Examination of old SCO's 10Ks reveals that, even were we to assume that every dime of their revenue came from the enterprise market, their 2002 share could not have exceeded 3.1%[29]. This is at the level of statistical noise.

In fact, SCO/Caldera's own complaint concedes that its historical strength has been in low-end systems used by small businesses. The principal author did SCO Unix consulting in the early 1990s, configuring systems for a small-town police station and a dental practice; anyone familiar with the industry would recognize that these were entirely typical old SCO deployments. And SCO/Caldera's most recent 10K [30] states, in part "Our business is focused on serving the needs of small businesses, including replicated site franchisees of Fortune 1000 companies[31], to have reliable, cost effective Linux and Unix operating systems and software products to power computers running on Intel architecture."

Conspicuously absent in SCO/Caldera's most recent mission statement is any talk of 16-way servers or enterprise data centers. In fact, SCO's history of non-performance in the enterprise market is not only consistent from long before the beginning of IBM's involvement with Linux in 1999-2000, it predates the 1991 origin of Linux itself. SCO/Caldera's claim that IBM's behavior with regard to improving Linux's enterprise scalability did it harm should be evaluated in the light of the failure of both incarnations of SCO, over more than a decade before that, to even seriously attempt to be competitive in the enterprise market.

SCO/Caldera falsely claims to have been unique as an Intel Unix vendor

n paragraph 23 SCO/Caldera writes "Except for SCO, none of the primary UNIX vendors ever developed a UNIX "flavor" to operate on an Intel-based processor chip set."

This is false. Sun Microsystems is a primary Unix vendor by anyone's definition, and their Solaris operating system was ported to the Intel 386 and sold on that platform. IBM's AIX was also ported to the 386 in 1987 and sold until 1995[32].

The complaint misleadingly implies that Unix was not generally available on PCs other than from old SCO. But, in fact, AT&T Unix was ported to Intel chips by no fewer than six different software houses " and that's not counting "own brand" ports maintained by PC hardware vendors such as Dell.

Up to 1994, when Linux made them irrelevant, the principal author maintained an on-line product comparison listing of all Intel Unixes known to him. The list of vendors from the final archival version [33] reads, in part:

```
Univel UnixWare Release 4.2
Consensys System V Release 4.2
UHC UnixWare Release 4.2
ESIX System V Release 4.0.4.1
Micro Station Technology SVr4 UNIX
Microport System V Release 4.0 version 4
UHC Version 4.0.3.6
SCO Open Desktop 3.0
BSD/386 1.0
NEXTSTEP 3.1
Yggdrasil Linux/GNU/X
Soft Landing Software
```

or personally ran two of these " Microport and Yggdrasil " and a third not listed, which was the Dell own-brand port.

As far back as 1983, old SCO had already had serious competition in the 386 Unix market from Interactive Systems Corporation (later bought by Sun Microsystems).

Not only was old SCO far from unique as an Intel Unix vendor, but SMP Unix implementations date as far back as 1985. The Sequent Corporation produced machines[34] featuring 2 to 30 80386 processors at that date. These machines ran DYNIX, a variant of Berkeley Unix.

Better yet, consider the following quote from a 1991 old SCO press release[35] (emphasis added):

> or the benefit of the entire user base, as well as the industry as a whole, SCO encourages all *UNIX System vendors for Intel processors* to join SCO, USL, Intel, ISC and OSF in supporting the iBCS-2 standard for x86 applications.

SCO/Caldera misrepresents the scope of its rights over Unix

SCO/Caldera alleges (Paragraph 57): *"When SCO acquired the UNIX assets from Novell in 1995, it acquired rights in and to all (1) underlying, original UNIX software code developed by AT&T Bell Laboratories."*

SCO/Caldera neglects to mention that those rights had been substantially impaired before its acquisition of the ancestral Bell Labs source code. There was a legal action in 1992-1993, in which Unix Systems Laboratories and Novell (SCO/Caldera's predecessors in interest) sued various parties including the University of California at Berkeley and Berkeley Systems Design, Inc. for alleged copyright infringement, trade secret disclosures, and trademark violations with regard to the release of substantial portions of the 4.4BSD operating system[36].

The suit was settled after AT&T's request for an injunction blocking distribution of BSD was denied in terms that made it clear the judge thought BSD likely to win its defense. The University of California then threatened to countersue over license violations by AT&T and USL. It seems that from as far back as before System V Release 4 in 1985, the historical Bell Labs codebase had been incorporating large amounts of software from the BSD sources. The University's cause of action lay in the fact that AT&T, USL and Novell had routinely violated the terms of the BSD license by removing license attributions and copyrights.

The exact terms of final settlement, and much of the judicial record, were sealed at Novell's insistence. The key provisions are, however, described in *Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributable*, , [McKusick99]. Only three files out of eighteen thousand in the distribution were found to be the licit property of Novell (and removed). The rest were ruled to be freely redistributable, and continue to form the basis of the open-source BSD distributions today.

Ten years ago " at a time when Linux was in its infancy " the courts already found the contributions of other parties to what is now UnixWare to be so great, and Novell's proprietary entitlement in the code so small, that Novell's lawyers had to settle for a minor, face-saving gesture from the University of California or walk away with nothing at all.

If the current lawsuit proceeds, justice requires that the court and settlement records in the AT&T-vs.-Berkeley lawsuit be unsealed, with a view to determining the degree to which SCO/Caldera's IP claims are nullified by the results.

This history is well-known in the open-source community, and helps explain why SCO/Caldera's claim that ownership of the historical Bell Labs code gives it substantial rights over other Unixes such as AIX is regarded among old Unix hands with near-universal disdain. Some of the court documents, including the 1993 ruling, are now available on the web [37].

If, as SCO/Caldera says, it inherited AT&T/USL/Novell's rights to Unix, it also inherited the *res judicata* hat there are many sources of code and engineering experience in the Unix design tradition entirely independent of AT&T/USL/Novell's intellectual property. And that, seven years before IBM's behavior with respect to AIX and Linux became an issue, AT&T/USL/Novell's proprietary stake in at least one leading-edge Unix was already so diluted in comparison with the contributions it had received from elsewhere that said stake could scarcely be said to exist at all.

SCO/Caldera would, for understandable reasons, prefer that the courts remain ignorant of the history, outcome, and implications of the BSD lawsuit. But, of course. it bears directly on SCO/Caldera's claim in Paragraph 93: *"Rather, IBM is obligated not to open source AIX because it contains SCO's confidential and proprietary UNIX operating system."*

The implied theory here is that "SCO's confidential and proprietary UNIX operating system" encompasses the entirety, or at least a preponderance, of the AIX code, including those portions related to enterprise scalability issues; and that SCO/Caldera therefore may exercise ex post facto control, even for anti-competitive purposes, over IBM's use of Unix in its normal course of business.

In fact, SCO/Caldera's complaint relies on confusing three separate scopes of control. One: those rights that pertain to the SCO shared libraries mentioned early in the complaint (paragraph 36 and following) only to disappear from the exposition shortly afterwards. Two: those rights entailed in SCO/Caldera's ownership of the SCO OpenServer codebase. Three: putative rights entailed in SCO's ownership of the historical Bell Labs source code (Unixware).

Since IBM's AIX is well known to contain large portions of Berkeley code (towards which IBM has by all accounts met its license obligations) SCO/Caldera's theory is at best extremely dubious. In other words, to prove its right to relief SCO/Caldera will need to show that whatever code IBM gave to the open-source community was neither legally obtained by both IBM and old SCO from a common source nor independently developed.

SCO/Caldera's claim to own Unix scalability techniques is weak

We observed previously that substantial contributions from outside sources to the historical Bell Labs codebase actually date back to before System V Release 4 in 1985. But even if we were to stipulate SCO/Caldera's undiluted ownership of the entire Bell Labs code base, its claim to own the class of enterprise scalability techniques at issue in its complaint would be very weak.

A major reason that the historical Bell Labs code base became nomadic among USL, Novell, and old SCO after 1990 is that it was already at that time senescent relative to newer Unixes like Solaris, Irix, HP-UX, Ultrix, and others. The Vax and 3B series minicomputers for which the late versions of the Bell Labs codebase were designed were years obsolete by 1995; the internal architecture of those variants of Unix is now primarily of historical interest. We noted previously that SCO/Caldera and old SCO made the "ancient Unix" Version 7 source code available for free, which rather disposes of the theory that the original Unix code had any residual IP value in the marketplace of today. (SCO belatedly terminated this offering on May 19th 2003, apparently realizing how badly it damaged their trade-secret claims.)

Furthermore, as previously noted, many Unix developers possess copies of SVr1 through SVr4 versions of the historical Bell Labs source code. We can therefore state that of the component technologies for enterprise scaling, the Bell Labs codebase includes a journaling file system (in the form of the VxFS Veritas journaling file system) and LVM (in the form of VxVM). But SMP for Intel processors only entered the line in 1995 with UnixWare 2. PCI hot-swapping came in only in 1998 with Unixware 7.

Ironically, UnixWare did not get usable SMP on Intel until *after Linux*.The UnixWare implememtation was unstable [38] until mid-1997; Linux got working SMP in 1996 with the release of 2.0[39].

As for 64-bit support, Linux had this in 1994, five years before IBM became involved in Linux development. Neither of SCO's products has this capability yet in 2003.

In fact, *not one* of the key enterprise-scalability technologies was present in the ancestral Unix code before UnixWare got a JFS in 1992. SCO's complaint alludes to this: in paragraphs 46 to 48 they observe that it required three years of development (1995-1998) to "harden" UnixWare for enterprise use. On SCO's own representation, the Bell Labs minicomputer-centered codebase of 1989-1995 is hardly more relevant to today's enterprise scalability challenges on today's PCs than the inner workings of a WWII-era jeep would be to the design of this year's Formula One racing cars.

SCO/Caldera's claim to own the scalability techniques certainly cannot be supported from the feature list of its own SCO OpenServer, a genetic Unix. The latest version[40] advertises SMP up to only 4 processors (a level which SCO's complaint dismisses as inadequate), no LVM, no NUMA, and no hot-swapping. That is, SCO/Caldera is alleging that IBM misappropriated from SCO technologies which do not appear in SCO's own product.

SCO/Caldera ignores its own role in motivating the development it now decries

SCO/Caldera charges (paragraph 82): *"Virtually none of these software developers and hobbyists had access to enterprise-scale equipment and testing facilities for Linux development."*

In making this claim, SCO/Caldera blithely ignores the existence of facilities such as the Open Source Development Lab[41], an organization funded by twenty-one companies including technology giants such as Intel, Hewlett-Packard, Cisco Systems, NEC, Dell, and Hitachi - *and* IBM. OSDL has lab facilities in Beaverton, Oregon, and Yokohama, Japan. OSDL opened its first lab in January 2001, four months before IBM's withdrawal from Project Monterey. From October 2000 to October 2002, one of its sponsors was Caldera Systems International!

OSDL is explicitly dedicated to assisting projects aiming towards carrier-grade and data-center Linux. It has supported over a hundred projects, most directly concerned with Linux scalability, performance improvements, fail-over and precisely those technical areas in which SCO/Caldera alleges that Linux could have made no progress without the intervention of IBM.

The existence of OSDL demonstrates industry-wide interest from large hardware vendors in scalable Linux, sufficient to sustain development with or without IBM's participation. But there may be a more direct linkage.

When OSDL spun up, IBM gained a choice: work with one small partner that lacks demonstrated expertise or focus on the enterprise market, or join a large consortium of industry heavyweights with man-centuries of relevant experience.

That seems just about enough time for an astute IBM strategist to conclude that old SCO was the less likely alternative to sustain a serious Linux development and support effort over time. To any technical person, SCO's own failure to develop expertise beyond its small-business roots seems a more plausible explanation for the switch to OSDL than some nefarious anti-SCO conspiracy by top IBM executives. To establish its right to relief, SCO/Caldera would have to show that switching horses to OSDL was not defensible as a normal business decision.

But the earlier role of Caldera in hoisting itself on its own petard was far more direct and less conjectural than this.

Symmetric multiprocessing (SMP) takes an operating system from being able to manage a single processor to utilizing two. The steps from two to four, four to eight, and eight to sixteen are not trivial but are far less challenging by comparison. Thus, SMP was perhaps the single most significant barrier between the Linux of the early 1990s and what SCO/Caldera itself characterizes as enterprise scaling.

Alan Cox (a key Linux developer generally considered Linus Torvalds's chief lieutenant) led the early work on symmetric multiprocessing in 1995. A web page[42], confirmed by a public newsgroup posting from an employee of Caldera[43] establishes that the dual-processor motherboard on which he performed the development was provided by none other than Caldera itself!

The timing is notable. Caldera began contributing directly to development of an enterprise-scalable Linux at around the same time old SCO acquired the historical Bell Labs codebase in 1995, *five years* before IBM became seriously involved in Linux development in 1999-2000.

aldera acquired the SCO brand in 2001. During the more than eighteen months between the cancellation of Project Monterey and the filing of the complaint, Caldera continued to garner revenues from the SMP-enabled kernel it distributed in SCO Linux.

If Darl McBride and complainants did not know at the time of the complaint that Caldera itself had played a lead role in the very development they accuse IBM of having unfairly and unlawfully pursued, they are incompetent. If they did know, their complaint appears to verge closely upon perjury.

SCO/Caldera misrepresents the capability and efforts of the open-source community

Most of the allegations that we have so far discussed in the complaint have been greeted among Linux and Unix developers merely with derision. Now, we are beginning to get to the claims that have stimulated a tidal wave of anger at SCO/Caldera among the open-source community in the weeks since their complaint was published.

SCO/Caldera misrepresents the efforts of the open-source community

When SCO/Caldera asserts (Paragraph 75): *"The name "Linus"(sic) was taken from the person who introduced Linux to the computing world, Linus Torvalds."* its use of the verb "introduced" appears to be an attempt to insinuate that Linux was in some way copied or pre-existent rather than an invention that Linus Torvalds originated.

Similarly, when SCO/Caldera asserts (Paragraph 78): *"The primary purpose of the GNU organization is to create free software based on valuable commercial software."*it portrays the GNU organization's original works as being mere derivatives or clones. In doing so, it flatly contradicts the evidence of major GNU projects such as the Emacs editor that is shipped by SCO/Caldera itself not merely on its Linux but on its Unix product as well. The Emacs editor predated every commercial product with even roughly comparable features.

Both implied claims cannot but be characterized as false, self-serving attempts to denigrate the work of others in order to magnify SCO/Caldera's imputed importance as the present owner of the historical Bell Labs code. Furthermore, they are offensive to the tens (perhaps hundreds) of thousands of skilled programmers who have collaborated in the invention of modern open-source Unixes.

SCO/Caldera misrepresents the state of Linux now

In paragraph 85, SCO/Caldera claims: "*For example, Linux is currently capable of coordinating the simultaneous performance of 4 computer processors. UNIX, on the other hand, commonly links 16 processors and can successfully link up to 32 processors for simultaneous operation..*"

32-processor SMP was already implemented under

Linux in 2000.[44] 24-processor operation, three times the 8-processor limit of UnixWare, was demonstrated in 1998 on a Sun E10000[45].

Today, SGI is shipping Altix 3000 cluster computers that run Linux over 64 processors[46].

SCO/CALDERA GROSSLY MISREPRESENTS THE STATE OF LINUX BEFORE IBM

A major part of SCO/Caldera's complaint turns on (a) representing pre-IBM Linux as a primitive makeshift being slapped together by garage-band amateurs. Their implied narrative is that (b) only the corporate intervention of IBM made Linux a competitive product, and that (c) IBM's intervention was in turn only efficacious due to the ineffable superiority of the primal Bell Labs code base.

All three of these assertions are not merely false, they are profoundly disrespectful to the many, many developers worldwide who labored with sweat and brilliance to craft Linux into a world-class operating system for eight years before IBM came on the scene.

The fun begins in paragraph 84, where SCO/Caldera alleges: "*prior to IBM's involvement, Linux was the software equivalent of a bicycle.*"

This was a 'bicycle' that, by actual measurement, could pedal data over the Internet faster than SCO OpenServer. And which by 1996, three years before IBM involvement, already featured SMP capabilities absent in SCO OpenServer and a more stable SMP implementation than UnixWare.

SCO/Caldera continues: "*UNIX was the software equivalent of a luxury car. To make Linux of necessary quality for use by enterprise customers, it must be re-designed so that Linux also becomes the software equivalent of a luxury car. This re-design is not technologically feasible or even possible at the enterprise level without (1) a high degree of design coordination, (2) access to expensive and sophisticated design and testing equipment; (3) access to UNIX code, methods and concepts; (4) UNIX architectural experience; and (5) a very significant financial investment.*"

This paragraph depends on the presumption that the open-source community consists of amateurs and incompetents, incapable of coordinating to produce high-quality work. In fact, the Linux developers have consistently out-thought, out-imagined, and out-coded old SCO's and SCO/Caldera's - which is precisely why old SCO went into the Linux professional services business, why it added Linux application compatibility to OpenServer, and why SCO/Caldera now finds itself without a business model and reduced to suing the handiest pair of deep pockets.

Let us take SCO/Caldera's's pre-requisites in order:

(1) a high degree of design coordination
Earlier, we noted that the success of Linux has motivated a sweeping reappraisal of some of the central premises of software engineering. It is now generally accepted that a high degree of design coordination can be achieved within the decentralized structure of open-source and Linux development; this has been demonstrated repeatedly by open-source projects such as the Apache webserver (with around 66% market share), the Perl and Python scripting languages, and Linux itself.

SCO/Caldera's implication that this is impossible is false and insulting. It is also dishonest. SCO/Caldera, and Caldera before it, participated in Linux development for eight years before this complaint and demonstrated understanding of the process through their actions (such as supplying Alan Cox with SMP hardware). They know better.

(2) access to expensive and sophisticated design and testing equipment
Caldera itself had a significant role in providing that access to Linux developers.

(3) access to UNIX code, methods and concepts
We noted earlier that access to UNIX code, methods and concepts is general in the community from which Linux development springs, and that said access is through codebases not subject to SCO's proprietary control or IP rights. For more than thirty years there has been a flourishing technical literature describing Unix operating system architecture and concepts. UNIX system internals and architecture are routinely taught in university computer science courses. Indeed, old SCO's and SCO/Caldera's own free publication of the Version 7 source code gave many programmers licit access to the ancestral Unix code, methods, and concepts.

(4) UNIX architectural experience
The open-source community encompasses more man-centuries of Unix architectural experience than SCO/Caldera (or even the vastly larger IBM) could ever dream of hiring. A vast literature on Unix design and internals has been available for over a quarter century.

A significant number of Linux developers (including the principal author) are old Unix hands whose experience stretches back to Unix's formative years in the 1970s and early 1980s. We read SCO/Caldera's animadversions not merely as an insult to us and our peers, but as a tendentious distortion of history.

(5) a very significant financial investment
While this might appear plausible to a layperson, and was true until relatively recently, it is no longer so. One of the characteristics of open-source development is that it is an inexpensive process. Not without costs; people still need to be

fed and salaried. But PCs are cheap, and the personnel load of open-source development is spread across a wider range of organizations and user consortia than was the case when process secrecy required a high degree of centralization and formal management structure.

Part of the reason SCO/Caldera is in distress is that the functional role it filled as a nexus of capital and management skills is near to being obsolete. Software development is simply outgrowing the need for such organizations. SCO/Caldera is on the wrong side of history.

In paragraph 99, SCO/Caldera continues: "*The only way that the pathway is an 'eight-lane highway' for Linux to achieve the scalability, SMP support, fail-over capabilities and reliability of UNIX is by the improper extraction, use, and dissemination of the proprietary and confidential UNIX Software Code and libraries. Indeed, UNIX was able to achieve its status as the premiere operating system only after decades of hard work, beginning with the finest computer scientists at AT&T Bell Laboratories, plaintiff's predecessor in interest.*"

We hope it is clear at this point just how meretricious this claim is. By wrapping itself in the mantle of Bell Labs, SCO/Caldera hopes to obscure the fact that its own non-Linux products do not in fact exhibit the enterprise-ready quality it accuses IBM of stealing, and never have. SCO/Caldera further attempts to confuse present-day claims arising from the Monterey project with claims putatively arising from the obsolete Bell Labs source code.

OTHER DEFECTS IN THE FACTS AND LOGIC OF THE COMPLAINT

THE PUBLIC EVIDENCE DOES NOT SUPPORT A CLAIM OF MISAPPROPRIATION

SCO/Caldera alleges (paragraph 90): "*To accomplish the end of transforming the enterprise software market to a services-driven market, IBM set about to deliberately and improperly destroy the economic value of UNIX and particularly the economic value of UNIX on Intel-based processors*" In paragraph 94 they continue: "*Over time, IBM made a very substantial financing commitment to improperly put SCO's confidential and proprietary information into Linux, the free operating system.*"

e do not claim to be able to read the minds of IBM executives. We do, however, know what IBM's Linux kernel hackers thought their marching orders were. In a June 2002 Slashdot interview [47], Dave Hansen and other members of IBM's multiprocessor-Linux team addressed the very point at issue, nine months before SCO/Caldera's complaint:

Q: As Linux developers inside IBM, do you get to see the AIX source code? If you do, are you allowed to "steal" some ideas from AIX and implement them in Linux? If not, why not, and what's the IBM official line?

A: First of all, before any of us were allowed to contribute to Linux, we were required to take an "Open Source Developers" class. This class gives us the guidelines we need to participate effectively in the open source community - both IBM guidelines and lessons learned about open source from others in IBM.

We are definitely not allowed to cut and paste proprietary code into any open source projects (or vice versa!). There is an IBM committee who can and do approve the release of IBM proprietary or patented technology, like RCU.[48]

On public evidence, not one of the five key technologies for enterprise scalability in Linux can plausibly be traced to historical Bell Labs code through IBM. These key technologies are:

symmetric multi-processing (SMP)

As we've seen, Linux SMP was worked up by Alan Cox and others using equipment provided by Caldera itself.

journaling file systems

IBM's Journaling File System (JFS) was contributed by IBM deriving from IBM's OS/2 and AIX operating systems[49], *not* from earlier Unix efforts.

Furthermore, Linux features three *other* journaling filesystems, contributed by Red Hat, Namesys, and SGI. Any of these three would be sufficient for enterprise scalability, and in fact the Red Hat EXT3 journaling system (and not IBM JFS) is the one in most common use.

logical volume management (LVM)

It is a matter of record [50] that IBM's approach to LVM was rejected by Linus Torvalds in favor of a different approach. Accordingly, even if we were to stipulate that IBM had access to old SCO's LVM technology, any attempted misappropriation came to naught.

non-uniform memory access (NUMA)

IBM's NUMA work derives from the NUMA-Q technology of the former Sequent corporation and others such as SGI, NEC and Fujitsu[51], not the historical Bell Labs codebase or any SCO development effort (neither of which included NUMA). The extent to which this page credits non-IBM organizations should be noted.

hot-swapping

Hot-swapping capability on PCs is dependent on special hardware capabilities. When the hardware supports it, hot-swapping tends to be natively and independently enabled by all modern Unixes (though not by the ancestral Bell Labs code).

SCO/Caldera's claim that IBM misappropriated SCO technologies for enterprise scalability should be evaluated in light of the following two facts: (a) The Bell Labs codebase contains neither LVM, nor hot-swapping; and (b) the SCO OpenServer codebase contains neither JFS, LVM, nor NUMA.

SCO/Caldera implies claims it is barred from explicitly making

Through phrases like "misusing and misappropriating SCO's proprietary software", and through the enumeration of five categories of rights in paragraph 68, SCO/Caldera's complaint implies the existence of relevant intellectual-property rights based on patent, copyright, trade-secret, and trademark law as a background to the explicit matter of its licensing dispute with IBM over Linux.

It is notable that the complaint does so without ever actually stating what those claims are. We have previously observed that the outcome of the USL/Novell-vs.-BSD lawsuit places the very existence of such rights in serious doubt. But there are other reasons for SCO/Caldera's coyness which should not escape notice.

One is that, despite misleading claims implied on SCO/Caldera's web pages by phrases like "exclusive licensing", SCO/Caldera does not own or control the Unix trademark. As we have previously observed, that trademark - and the privilege of suing IBM for relief on a trademark-violation theory - belongs to The Open Group.

Furthermore, SCO/Caldera is barred by the terms of the GNU General Public License from making copyright or patent-infringement claims on any technology shipped in conjunction with the Linux kernel that SCO/Caldera itself has been selling for the last eight years. Therefore, SCO/Caldera may accuse IBM of misappropriating SCO-owned software to improve the Linux kernel only if that software does not actually ship with the Linux kernel it is alleged to be improving!

Finally, SCO/Caldera is barred from making trade-secret claims on the contents of the Linux kernel, not merely by the fact that the kernel source is generally available, but by the fact that SCO/Caldera has made the sources of its Linux kernel available for download from SCO's own website! [52].

SCO/Caldera, as a matter of fact and law, clearly does retain proprietary rights with respect to the SCO OpenServer binary distribution (which has never been published in source-code form and is not under the GPL). It is not the purpose of this position paper to dispute those rights. But to the extent that SCO/Caldera uses those proprietary rights to attempt to cast a shadow over Linux, it maintains a position which is factually untenable.

Indeed, the effect of SCO/Caldera's complaint is to systematically mislead and obfuscate on the issue of what background rights SCO/Caldera actually has at issue in its claim of tort and license violations. The clear intent is to deceive observers into believing that SCO/Caldera has a licit claim.

But that emperor has no clothes. Ultimately, SCO/Caldera's argument would appear to boil down to asserting that "IBM had no right to give the community technologies that SCO/Caldera had made freely available on its download site."

Who owns Unix, anyway?

The issue of who owns Unix has always aroused passions of an intensity and nature difficult for strangers to the issue to understand. The drama here is not merely about money or the competing agendas of corporations, but about the Unix community's sense of ownership of its own work. That community, and that sense of ownership, is exceptionally powerful for reasons which bear materially on the matter of SCO/Caldera's complaint.

Unix was born in 1969 at Bell Laboratories among computer science researchers. After 1975, much of its development was actually done by contributors outside Bell Labs, especially at UC Berkeley and elsewhere in academia. However, AT&T continued to formally own the results of that collective work. Indeed, they spent five years after 1983 in a largely fruitless quest to commercialize it as a product - a role which was played much more effectively by Sun Microsystems and other licensees, including old SCO and IBM.

Even during the early days of Unix commercialization, the Unix code base was widely regarded as a commons worked by many hands. As time went on and Unix evolved, possession of an AT&T source license came to be seen as more a pro-forma gesture in the direction of history than a concession that AT&T's intellectual property still contributed a dominating part of the value. This was especially so after the Berkeley hackers added Internet capability to Unix around 1980.

Thus, the community of Unix hackers that had grown up around the pre-commercial releases never lost the conviction that, ethically, the Unix code belonged to them - the people who had the ideas and wrote the code - regardless of what the legal paperwork said. The outcome of the USL-vs.-Berkeley lawsuit in 1993, which severed the claims of AT&T and its successor Novell to the BSD source code, was universally regarded in the community as no more than simple and overdue justice.

From 1975 to about 1995, therefore, the Unix vendors and the Unix hackers existed in a kind of half-symbiotic, half-antagonistic embrace. The Unix hackers (who needed the jobs vendors were providing to practice their craft) expressed their conviction of ownership by freely passing around bootlegged Unix sources among themselves for study and problem-solving. Vendors (who needed the hackers to fill job slots) looked the other way, routinely winking at behavior that was technically a massive theft of critical intellectual property as long as it stayed in the family and nobody's bottom line got hurt. But given this history, any attempt to make a trade-secret claim based on the historical Bell Labs source code would be at best highly disingenuous.

This tacit truce began to disintegrate after 1990. The rise of the PC meant that the hackers had less need of massive corporate infrastructure and capital concentrations to support their art. The USL-vs.-Berkeley lawsuit was the first major confrontation that the hackers won. Under the settlement terms, the Berkeley source code - and the Unix tradition with it - achieved the autonomy in law that it had always deserved in the minds of Unix programmers.

Two related developments were going on at the same time between 1990 and 1995. One was the rise of open-source development, and the other was the senescence of the historical Bell Labs codebase. While corporate Unix vendors continued to pay formal obeisance to the Bell Labs codebase by buying Unix source code licenses from AT&T's successors in interest, the rise of Linux and the open-source BSDs made the disposition of the ancestral Unix sources increasingly seem a meaningless game played among lawyers, of little remaining interest to Unix hackers. The technical leading edge of the Unix tradition had moved elsewhere, notably to Linux. Nobody, neither the vendors nor the hackers, really needed the Bell Labs source code any more.

Indeed, when the old-school Unix hackers who at that time ran old SCO purchased the ancestral Bell Labs code in 1995, it was widely viewed as little more than a bit of clever marketing, or even as a pure nostalgia trip by techies who had bought the ancestral source code just because they could. (It appears that this is not quite correct; old SCO employees who have spoken off the record with us say that Caldera was also buying access to old SCO's channel partners and distributor network.) At that time (as we have noted in the section called "SCO's history with open source") old SCO gave every evidence of understanding and endorsing Linux.

But SCO/Caldera is no longer run by Unix old hands. Their complaint, once again, has thrust the question of "who owns Unix" into the foreground of debate. This time around, the hacker community has corporate allies (IBM among them) who understand the new world of open source - and that it is to their own business advantage to respect the Unix hackers as the owners of their art.

SCO/Caldera's complaint, in all its brazen mendacity, is the last gasp of proprietary Unix. We in the open-source community (and our allies) are more than competent to carry forward the Unix tradition we founded so many years ago. We pray that all assertions of exclusive corporate ownership over this tradition be given a swift and definitive end.

Policy implications and recommendations

A judgment in favor of SCO/Caldera could do serious damage to the open-source community. SCO/Caldera's implication of wider claims could turn Linux into an intellectual-property minefield, with potential users and allies perpetually wary of being mugged by previously unasserted IP claims, and ever-more-outlandish theories of entitlement being propounded by parties with only the most tenuous relationship to anyone who ever wrote actual program code.

On behalf of the community that wrote most of today's Unix code, and whose claims to have done so were tacitly recognized by the impairment of AT&T's rights under the 1993 settlement, we protest that to allow this outcome would be a very grave injustice. We wrote our Unix and Linux code as a gift and an expression of art, to be enjoyed by our peers and used by others for all licit purposes both non-profit and for-profit. We did not write it to have it appropriated by men so dishonorable that after making profit from our gift for eight years they could turn around and insult our competence.

Damage to the open-source community would matter, because we are both today's principal source of innovation in software and the guardians and maintainers of the open Internet. Our autonomy is everyone's bulwark against government and corporate control of the digital media that are increasingly central in political, commercial, and personal communications. Our creative energy is what perpetually renews and finds ever more exciting uses for computers and networks. The vigor of our culture today will translate into more possibilities for everyone tomorrow.

On behalf of Unix developers over the last thirty-five years, of today's Linux and open-source developers, and of all Internet users everywhere, we therefore express these hopes with respect to court findings:

- To find against SCO/Caldera in its complaint against IBM, or for IBM on any motion for dismissal or summary judgment.
- To ground the finding in terms which will foreclose any future claims by SCO/Caldera of proprietary control over technologies contributed to Linux.
- To confirm that SCO/Caldera cannot re-litigate the USL/Novell-vs.-BSD action by stealth, and thus

that SCO/Caldera's ownership of the ancestral Bell Labs source code gives it no authority or proprietary entitlement over the works of the open-source community and Unix developers at large.

We further suggest that SCO/Caldera's complaint is knowingly deceptive to a degree that recommends sanctions under the Utah and Federal Rule 11 of Civil Procedure.

OSI is *not* requesting any more general finding on broad issues of intellectual property in software, even supposing that were within the purview of the court. We feel the facts of Unix history are sufficiently compelling and particular that the court would be justified in ruling as we recommend without attempting to challenge and re-construct the entire legal theory of software intellectual property.

Bibliography

[TNHD] The New Hacker's Dictionary. Third Edition. Eric S. Raymond. Copyright © 1996. ISBN 0-262-68092-0. MIT Press. 547pp.. HTML at the Jargon File Resource Page. .

[CATB] The Cathedral and the Bazaar. Second Edition. Eric S. Raymond. Copyright © 1999. ISBN 0-596-00131-2. O'Reilly & Associates. 240pp.. How and why the Linux development model works. HTML here .

[TAOUP] The Art of Unix Programming. Second Edition. Eric S. Raymond. Copyright © 2003. ISBN 0-13-142901-9. Addison-Wesley. 240pp. Work in progress, to be published August 2003. HTML of the draft is here .

[McKusick99] Open Sources: Voices from the Open Source Revolution. Sam Ockman and Chris DiBona. Copyright © 1999. ISBN 1-56592-582-3. 280pp. O'Reilly & Associates. "Twenty Years of Berkeley Unix". From AT&T-Owned to Freely Redistributable. McKusick Kirk Marshall. Available on the Web .

A. Work in Progress

This position paper is maintained in DocBook XML format. Please obtain the latest copy of the source before diffing any patches for submission.

[1] http://www.sco.com/scosource/complaint3.06.03.html

[2] There is a list of the trademarks owned by The Open Group at http://www.opengroup.org/legal.htm#The Open Group's Trademarks

Old SCO could call OpenServer and UnixWare Unix because the source passes the tests necessary for Unix branding by The Open Group. See http://www.unix.org/what_is_unix/the_brand.html.

The Open Group maintains a list of vendors of registered Unix products at http://www.unix-systems.org/vendors/

[3] The UNIX System -- Trademark Usage -- Unix trademark.

[4] Strictly speaking, what Linus wrote was the Linux kernel. He and others combined the kernel with pre-existing software, much of it written or sponsored by the GNU project of the Free Software Foundation, to produce a full operating system.

[5] We use the term "hacker" in its correct and original sense here, as an enthusiast or artist of computer programming.

[6] One of the best-known papers on this topic is Fuzz Revisited.

[7] See http://shop.caldera.com/caldera/ancient.html and http://minnie.tuhs.org/pipermail/pups/2000-April/000181.html

[8] The material in the Unix relationships chart is from the Unix History Timeline.

[9] See The Creation of the UNIX* Operating System: Early versions of the UNIX* System for a more detailed history.

[10] UnixWare Frequently Asked Questions (General).

[11] The Decision to Divest: Incredible or Inevitable? By Trudy E. Bell. (Reprinted from IEEE Spectrum Online June 2000. Volume 37. Number 6.)

[12] History of SCO

[13] History of Tarantella, Inc.

[14] AT&T History - The Bell System

[15] USO renamed UNIX System Laboratories, Inc.News release 25 Jun 1990.

[16] The Creation of the UNIX* Operating System: Business gets the word Lucent Technologies.

[17] 11 computer companies purchase equity in UNIX System Labs News release 3 Apr 1991.

[18] Annual meeting in Chicago; AT&T income up 6.6 percent News release 17 Apr 1991.

The gain on the sale of ownership interests in UNIX System Laboratories added $43 million to other income.

[19] The Creation of the UNIX* Operating System: UNIX moves on Lucent Technologies.

[20] Novell and USL to form joint venture for UNIX and Netware News release 15 Oct 1991.

[21] Novell and Unix System Labs create Univel, a networking company News release 12 Dec 1991.

Novell and USL are contributing cash and technology rights to Univel. Novell holds a 55 percent share in the new company. Although specific financial terms of the joint venture were not disclosed, the companies acknowledged that Univel has been underwritten with $30 million cash and other assets.

In addition, the joint venture will have access to technical resources and the education, training, sales, marketing and distribution capabilities of the parent companies.

[22] Novell signs letter of intent to purchase UNIX System Labs News release 21 Dec 1992.

[23] Novell signs definitive agreement to buy AT&T's UNIX System Labs News release 16 Feb 1993.

[24] Novell completes acquisition of UNIX System Laboratories News release 14 Jun 1993.

[25] HP, Novell and SCO To Deliver High-Volume UNIX OS With Advanced Network And Enterprise Services Novell press release 20 Sep 1995.

SCO has purchased the UnixWare business from Novell and will consolidate its SCO OpenServer system and Novell's UnixWare into a merged high-volume Intel-based UNIX operating system hat provides interfaces in common with HP-UX.

[26] Novell Completes Sale of UnixWare Business to The Santa Cruz Operation Novell press release 6 Dec 1995.

Novell, Inc. today completed the sale of its UnixWare business to The Santa Cruz Operation, Inc. (SCO), finalizing an agreement first announced in September, 1995. Under the agreement, Novell receives approximately 6.1 million shares of SCO common stock, resulting in an ownership position of approximately 17 percent of the outstanding SCO capital stock. The agreement also calls for Novell to receive a revenue stream from SCO based on revenue performance of the purchased UnixWare business. This revenue stream is not to exceed $84 million net present value, and will end by the year 2002.

[27] http://www.cnn.com/TECH/computing/9908/18/scoforum.idg/

[28] http://web.archive.org/web/20000816145931/www.sco.com/linux/

[29] SCO/Caldera filed an annual report on Jan 27, 2003, (available online here). Revenue is broken down on page 19, with a total of $64.2 million reported for fiscal year 2002, of which $53.0 million comes from "products" and $11.3 million from "services".

The size of the Unix market in dollars is given in the February 10, 2003 CNET article "Sales Increase for U.S. Linux Servers" (available online at http://news.com.com/2100-1001-984010.html). The figure for 2003 is $1.69 billion.

Product revenue of $53 million divided by $1,690 million is 3.14 percent, the figure used. Assuming every dime of old SCO's revenue came from "the Unix server market", their market share could not exceed 3.80 percent (I.E. $64.2/$1690).

[30] http://www.sec.gov/Archives/edgar/data/1102542/000104746903003091/a2101798z10-k.htm

[31] E.g., McDonald's restaurants. The McDonald's chain was, and apparently remains, SCO's biggest customer.

[32] http://os2ports.com/docs/aix/withdraw.html

[33] http://www.catb.org/esr/faqs/clone-unix-guide.txt

[34] http://www.cs.berkeley.edu/~culler/machines/sequent2.ps http://www.cs.berkeley.edu/~culler/machines/sequent2.ps

[35] http://groups.google.com/groups?q=iBCS+www.intel.com&hl=en&lr=&ie=UTF-8&oe=UTF-8&selm=20016%40scorn.sco.COM&rnum=1

[36] Unix System Laboratories v. Berkeley Software Design, Inc., 1993 U.S. Dist. LEXIS 19503; 27 U.S.P.Q.2D (BNA) 1721, issued 30 Mar 1993. There was a subsequent case, 832 F. Supp. 790, issued 7 September 1993, but it was largely a technical ruling on sovereign immunity and does not bear on the issues in this brief.

[37] http://cm.bell-labs.com/cm/cs/who/dmr/bsdi/bsdisuit.htm

[38] http://groups.google.com/groups?hl=en&lr=&ie=UTF-8&oe=utf-8&selm=5jlkiv%24hm5%241%40news3.texas.net

[39] http://ouray.cudenver.edu/~etumenba/smp-howto/SMP-HOWTO-2.html

[40] http://www.caldera.com/products/openserver/enterprise.html; note the sentence "Support for systems with up to 4 CPUs"

[41] http://www.osdl.org/

[42] http://www.linux.org.uk/SMP/title.html

[43] http://groups.google.com/groups?selm=44qe0m%24gno%40caldera.com,

[44] A dated boot log of a 32-processor SMP Linux box (albeit one running with only 31 processors active) is at http://lists.insecure.org/linux-kernel/2000/Sep/5014.html.

[45] http://samba.org/~anton/e10000/dmesg_24

[46] http://www.sgi.com/servers/altix/

[47] http://interviews.slashdot.org/article.pl?sid=02/06/18/1339201&mode=nested

[48] RCU is Read-Copy-Update, a multiprocessor technology developed by the Sequent Corporation before its acquisition by IBM.

[49] http://oss.software.ibm.com/developer/opensource/jfs/project/pub/faq.txt

[50] http://lwn.net/Articles/14215/; search down for 'EVMS'

[51] The sources of the NUMA work are described at http://news.com.com/2100-1001-982651.html?tag=fd_top.

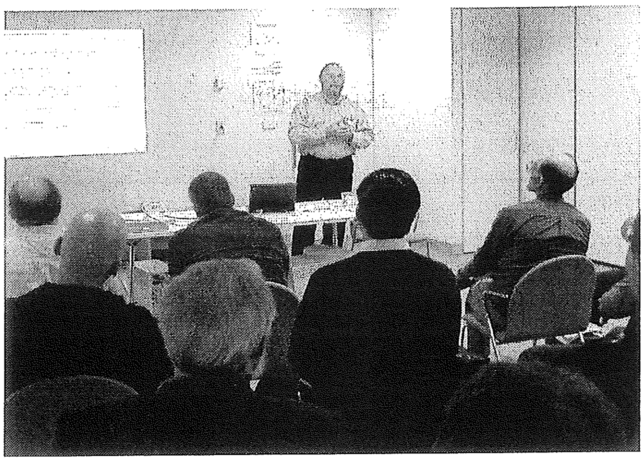
[52] ftp://ftp.sco.com/pub/scolinux/server/updates/4.0/SRPMS/kernel-source-2.4.19.SuSE-82.nosrc.rpm

This article is re-printed with permission. The originals can be found at:

URL://http://www.opensource.org/sco-vs-ibm.html

AUUG 2003 Systems Administration Symposium Photo Gallery

Courtesy of Enno Davids <enno@doc.metva.com.au>



More images available here:
http://www.vic.auug.org.au/sysadm03/

This quarter's CD-R: NetBSD 1.6.1

Greg Lehey <Greg.Lehey@auug.org.au>

This quarter your AUUGN comes with NetBSD (http://www.NetBSD.org/) version 1.6.1. which was released on 21 April. NetBSD is the oldest free UNIX-like operating system still in circulation: the NetBSD project was founded on 21 March 1993. Like FreeBSD, it was based on Bill Jolitz's now-defunct 386BSD.

In previous editions of AUUGN we have distributed OpenBSD (which was derived from NetBSD in 1995) and FreeBSD, which like NetBSD was derived from 386BSD and is currently also celebrating its tenth anniversary (19 June 1993). You'll find NetBSD very similar in feel to FreeBSD and OpenBSD, more so even than one Linux distribution resembles another.

NetBSD prides itself on being portable: ``Of course it runs NetBSD". The full distribution doesn't fit on one CD-ROM, but this CD contains installable versions for i386, Apple Mac and 32 and 64 bit SPARC implementations. Based on our surveys, that should satisfy almost everybody. If the CD-R doesn't support your platform, please let us know: we'll get a free CD-R to you.

Installing NetBSD

You'll find detailed instructions on the CD-R in a number of formats. The installation instructions for i386, for example, run to over 3,000 lines, enough to fill this issue by themselves. In this article, I'll just go over the basics.

The root directory of the CD-R includes some documentation and boot aids, but the most important thing are the directories i386, macppc (for Mac), sparc (for 32 bit SPARC) and sparc64 (for 64 bit SPARC). Each of these directories contains platform-specific installation instructions in a number of formats: INSTALL.txt, INSTALL.html and INSTALL.ps. There's also an INSTALL.more for
use during the install process.

As with other operating systems, installation involves partitioning the disk, deciding what to install, creating file systems, installing the software and configuring the system. Most machines can boot from the CD-R, where you get a menu-based installation tool. See the installation instructions on the CD-R for further details. It's probably worth printing them out before you start.

MARK YOUR DIARY!!
AUUG 2003

AUUG Inc

Open Standards, Open Source, Open Computing

When and Where?

The Duxton Hotel, Milsons Point, Sydney.

- The Conference will be held from 3 to 5 September.
- Tutorials will be conducted prior to the conference from 31 August to 2 September.

Come along and hear presentations relating to:

- Open Systems or other operating systems
- Open Source projects
- Business cases for Open Source
- Technical aspects of Unix, Linux and BSD variants
- Managing Distributed Networks
- Performance Management and Measurement
- System and Application Monitoring
- Security in the Enterprise
- Technical aspects of Computing
- Networking in the Enterprise
- Business Experience and Case Studies
- Cluster Computing
- Computer Security
- Networking, Internet (including the World Wide Web).

Sponsors

AUUG wishes to gratefully acknowledge the generous assistance given by the following organisations.

Diamond Sponsors

Sun microsystems

Platinum Sponsor

ca Computer Associates™

Gold Sponsor

IBM

Pearl Sponsor

EverythingLinux.com.au

Participating Sponsor

silicon breeze www.linuxjewellery.com

Why not take this opportunity to highlight your company's commitment to Unix and Open Systems? Sponsorship of this premiere event is a great way to showcase your business and products. Full details can be found at:

http://www.auug.org.au/events/2003/auug2003/sponsors/

Further Information

Regular updates, including Registration Forms, which will be available shortly, can be found at:

http://www.auug.org.au/events/2003/auug2003

For all other enquiries relating to AUUG 2003 please contact Liz Carroll, AUUG Business Manager on:

Phone: 1-800-625 655 or (02) 8824 9511
Email: busmgr@auug.org.au

Message from the Programme Chair

The Call for Papers for AUUG 2003 has recently closed. We received a great many good quality submissions and with room for only a limited number of presentations the choice was not easy. I would therefore like to thank all those who submitted proposals.

Our speakers are drawn from a wide spectrum, from local technology enthusiasts through to internationally recognised specialists. I am confident that, between the conference and tutorial programmes, you will find many items of interest, not just on the hard-core technical side, but for Unix and open systems in general. Full programme details can be found on the AUUG website at:

http://www.auug.org.au/events/2003/auug2003/

One of the great aspects of the conference is the opportunity to meet and network with people with a serious interest in Unix and open source solutions. In addition to the Networking Reception on Wednesday and the Conference Dinner on Thursday, there will be plenty of time during breaks to meet with your fellow delegates and discuss the latest technologies with our sponsors.

I look forward to greeting you personally at the conference in September.

Adrian Close
Programme Chair
AUUG 2003

AMERICAN
BOOK STORE

10% DISCOUNT
TO AUUG MEMBERS
ON OUR COMPLETE RANGE
OF COMPUTER, BUSINESS
AND GENERAL BOOKS

173 Elizabeth St, Brisbane Queensland 4000
Ph: (07) 3229 4677 Fax: (07) 3221 2171 Qld Country Freecall: 1800 177 395
american_bookstore@compuserve.com

Name: ______________________________ Date: ____________

Address: ____

______________________________ Post Code: ____________

Phone Number: ______________________________

Payment Method: ☐ Cheque ☐ Money Order ☐ Amex ☐ Bankcard
☐ Diners ☐ Mastercard ☐ Visa

Card Number: ____

Expiry Date: ______________ Signature: ______________________

This is a: ☐ Special Order ☐ Mail Order ☐ Book on Hold

QUANTITY	TITLE	PRICE

SUBTOTAL	$ ....
LESS 10% DISCOUNT	$ ....
POST & PACK	$ ....
TOTAL	$ ....

POSTAGE AND HANDLING FEES: 1 BOOK $6.00 2-4 BOOKS $7.00
BOOKS OVER $70.00 WE WILL SEND CERTIFIED - PLEASE ADD ANOTHER $1.50 OR WAIVE CERTIFIED DELIVERY.

FOR SPECIAL ORDERS, PLEASE ENCLOSE $10.00 PER BOOK AS A DEPOSIT.

AUUG MEMBERSHIP RENEWAL

MEMBERSHIP RENEWAL INVOICES WERE MAILED IN MAY, FOR THOSE OF YOU WHOSE MEMBERSHIP EXPIRES ON 30 JUNE 2003.

IF YOU HAVE NOT YET RECEIVED YOURS, PLEASE CONTACT:

LIZ CARROLL, AUUG BUSINESS MANAGER

EMAIL: **busmgr@auug.org.au**

PHONE: 1-800 625 655

FAX: (02) 8824 9522

IF...

YOU DO NOT SEND IN YOUR MEMBERSHIP RENEWAL

<u>THIS WILL BE YOUR LAST COPY OF AUUGN!</u>

AUUG Chapter Meetings and Contact Details

CITY	LOCATION	OTHER
ADELAIDE	We meet at Internode, Level 3/132 Grenfell St aka 'the old AAMI building', at 7 pm on the second Wednesday of each month.	Contact sa-exec@auug.org.au for further details.
BRISBANE	Inn on the Park 507 Coronation Drive Toowong	For further information, contact the QAUUG Executive Committee via email (qauug-exec@auug.org.au). The technologically deprived can contact Rick Stevenson on (07) 5578-8933. To subscribe to the QAUUG announcements mailing list, please send an e-mail message to: <majordomo@auug.org.au> containing the message "subscribe qauug <e-mail address>" in the e-mail body.
CANBERRA	Australian National University	
HOBART	University of Tasmania	
MELBOURNE	Various. For updated information See: http://www.vic.auug.org.au/	The meetings alternate between Technical presentations in the even numbered months and purely social occasions in the odd numbered months. Some attempt is made to fit other AUUG activities into the schedule with minimum disruption.
PERTH	The Victoria League 276 Onslow Road Shenton Park	
SYDNEY	Meetings start at 6:15 pm Sun Microsystems Ground Floor 33 Berry Street (cnr Pacific Hwy) North Sydney	The NSW Chapter of AUUG is now holding meetings once a quarter in North Sydney in rooms generously provided by Sun Microsystems. More information here: http://www.auug.org.au/nswauug/

For up-to-date details on chapters and meetings, including those in all other Australian cities, please check the AUUG website at http://www.auug.org.au or call the AUUG office on **1-800-625655.**

Application / Renewal Individual or Student Membership of AUUG Inc.

Use this tax invoice to apply for, or renew, Individual or Student Membership of AUUG Inc. To apply online or for Institutional Membership please use **http://www.auug.org.au/info/**

This form serves as Tax Invoice.

Please complete and return to:

AUUG Inc, PO Box 7071, BAULKHAM HILLS BC NSW 2153, AUSTRALIA

If paying for your membership with a credit card, this form may be faxed to AUUG Inc. on +61 2 8824 9522.

Please do not send purchase orders.
Payment must accompany this form.

Overseas Applicants:

- Please note that all amounts quoted are in Australian Dollars.
- Please send a bank draft drawn on an Australian bank, or credit card authorisation.
- There is a $60.00 surcharge for International Air Mail
- If you have any queries, please call AUUG Inc on +61 2 8824 9511 or freephone 1800 625 655.

Section A:

Personal Details

Surname: ....

First Name: ....

Title: .... Position: ....

Organisation: ....

Address: ....

....

Suburb: ....

State: .... Postcode: ....

Country: .... Phone Work: ....

Phone Private: .... Facsimile: ....

E-mail: ....

Membership Number (if renewing): ....

Student Member Certification

For those applying for Student Membership, this section is required to be completed by a member of the academic staff.

I hereby certify that the applicant on this form is a full time student and that the following details are correct:

Name of Student: ....

Institution: ....

Student Number: ....

Signed: ....

Name: ....

Title ....

Date Signed: ....

Section B: Prices

Please tick the box to apply for Membership. Please indicate if International Air Mail is required.

Renew/New* Individual Membership	$110.00 (including $10 GST)	☐
Renew/New* Student Membership	$27.50 (including $2.50 GST)	☐
Surcharge for International Air Mail	$60.00	☐

* Delete as appropriate.

GST only applies to payments made from within Australia. Rates valid from 1st October 2002.

Section C: Mailing Lists

AUUG mailing lists are sometimes made available to vendors. Please indicate whether you wish your name to be included on these lists:

Yes ☐ No ☐

Section D: Payment

Pay by cheque

Cheques to be made payable to **AUUG Inc.** Payment in Australian Dollars only.

OR Pay by credit card

Please debit my credit card for A$ ....

Bankcard ☐ Mastercard ☐ Visa ☐

Card Number: .... Expires: ....

Name on card: .... Signature: ....

Date Signed: ....

Section E: Agreement

I agree that this membership will be subject to rules and bylaws of AUUG Inc as in force from time to time, and this membership will run from the time of joining/renewal until the end of the calendar or financial year as appropriate.

Signed: ....

Date Signed: ....

This form serves as Tax Invoice. AUUG ABN 15 645 981 718