

Advanced search

*Linux Journal Issue #112/August 2003*



*Features*

Implementing Encrypted Home Directories *by Mike Petullo*

Keep your files safely encrypted when you're logged out, and automatically get access when you log in.

Take Control of TCPA *by David Safford, Jeff Kravitz and Leendert van Doorn*

The free code behind IBM's new security chip. Menace or protector?

The Power of the Incredible Hulk—The ILM Linux Death Star *by Robin Rowe*

This fully operational battle station is a 750-node Linux cluster running a custom batch scheduling program.

Root for All on the SE Linux Play Machine *by Russell Coker*

Set visitors loose as root and see what they break—can SE Linux alone keep the system safe?

*Indepth*

Eleven SSH Tricks *by Daniel R. Allen*

You know it's the secure way to connect to your server. But OpenSSH is fast and convenient too.

VTun *by Ryan Breen*

Need to make a secure connection from home? Set up a simple virtual private network?

2003 Editors' Choice Awards

With all the great Linux stuff introduced in the past year, these are some of the hardest decisions we've ever made.

*Embedded*

Driving Me Nuts Device Classes by Greg Kroah-Hartman

*Toolbox*

**Kernel Korner** NSA Security Enhanced Linux by Faye Coker

**At the Forge** CMF Types by Reuven M. Lerner

**Cooking with Linux** Illuminating Your Network's Darkest Corners  
by Marcel Gagné

**Paranoid Penguin** Authenticate with LDAP by Mick Bauer

*Columns*

**Linux for Suits** Practical Penguin Progress by Doc Searls

**EOF** Consider Accessibility by Janina Sajka

*Reviews*

Red Hat 9 by Marco Fioretti

*Departments*

Letters

upFRONT

**From the Editor** : Security: Yes, It's Part of Your Job

On the Web

Best of Technical Support

New Products

Archive Index

Advanced search

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## Implementing Encrypted Home Directories

**Mike Petullo**

Issue #112, August 2003

Take advantage of loopback encrypted filesystems and `pam_mount` to secure your data.

When used correctly, encrypted filesystems can be an effective way to protect sensitive data stored on a computer. Standard encryption packages, such as the GNU Privacy Guard (GPG), are good for encrypting e-mail. They are not as convenient, however, for encrypting files one wishes to read or modify many times throughout the files' lifetimes.

Unlike GPG, an encrypted filesystem is transparent to users. There is no hassle of manually decrypting a file before reading it or encrypting it again after modifying it. Potential user forgetfulness also is mitigated. After introducing some encrypted filesystems available for Linux, this article explains how to create an encrypted home directory that is automatically mounted at login time and unmounted at logout. Finally, this article introduces some concepts that demonstrate the dangers of improperly implemented encryption techniques.

Why would one want to encrypt the data stored on a computer? Isn't protecting sensitive data what filesystem permissions are for? Although useful, filesystem permissions quickly lose their effectiveness when an attacker has complete control of the storage medium the permissions are used to protect. For example, if someone steals my Linux laptop, an Apple iBook, its filesystems permissions are of little use against the thief who can simply boot from a diabolical CD-ROM. The same is true if I send my laptop to Apple for repairs. A dishonest employee conceivably could read my files. Correctly encrypting the files on a computer is a safe form of protection, because the process does not depend on the integrity of the operating system after the encryption takes place.

I have chosen to encrypt only the home directories on my iBook. Encrypting the entire filesystem, starting with root, was not acceptable in my case due to

performance penalties and other factors. Information on implementing this technique can be found on the Internet—it requires using Linux's initial ramdisk capability. In my experiences with an x86-based system, the encryption technique I chose is around 50% slower than a non-encrypted XFS filesystem when writing to disk using buffered I/O. Encrypting only home directories obviously leaves many files, such as system logs, as plain text, but these were not significantly sensitive in my case. Encrypting only home directories was a sensible compromise for me.

Linux supports a few options for encrypting filesystems. Systems such as the Transparent Cryptographic File System provide an encrypted extension to NFS-served ext2 volumes. Other filesystems contain integrated cryptography in their design for local use. For my application, the best solution seemed to be the concept of loopback encrypted filesystems. As you will see, loopback encryption can be used to encrypt any filesystem type supported by Linux, including the proven ext2, XFS and ReiserFS.

Linux loopback filesystem support simply allows a user to mount an ordinary file as if it were a device, such as /dev/hda1. This traditionally is useful for doing things like mounting a CD-ROM filesystem image to populate and test before burning it to CD-R media or distributing bootable floppy disk images. Herbert Valerio Riedel's GNU/Linux Cryptographic API (CryptoAPI) and util-linux patches add support for mounting encrypted filesystem images to the loopback driver.

Before delving into the details of loopback encrypted filesystems, let's take a look at how to create and mount a vanilla loopback filesystem. First, create a file to contain the filesystem. This example creates a file large enough to host a 20MB filesystem:

```
dd if=/dev/zero of=plaintext.img bs=1M count=20
```

Second, associate the newly created file with a loopback device:

```
losetup /dev/loop0 plaintext.img
```

Next, create a filesystem within the file, using the newly associated loop device:

```
mkfs -t ext2 /dev/loop0
```

Finally, mount the filesystem and use it as if it were any other mounted volume:

```
mount /dev/loop0 mount point
```

Now, let us move on to the encrypted case. In order to use loopback encrypted filesystems, the kernel must support them. Most distributions do not include

this functionality, so a custom-built kernel probably is necessary. A cryptographic API package similar to the one I use is being merged into the mainstream 2.5 kernel. However, for the stable 2.4 tree, the GNU/Linux CryptoAPI patches are necessary and available at [www.kernel.org](http://www.kernel.org). Once you apply the patch-int and loop-hvr patches, Cryptographic options should be available when you configure your kernel. The following options must be enabled:

- cryptographic API support (CONFIG\_CRYPTO)
- generic loop cryptographic filter (CONFIG\_CRYPTOLOOP)
- cryptographic ciphers (CONFIG\_CIPHERS)

You have to enable at least one cipher as well. Remember which cipher you choose. I chose AES, originally called Rijndael, and use AES in my examples.

Build and install the newly configured kernel. All of the kernel's encryption code may be compiled as modules. If you choose to build kernel modules, don't forget to insert them before you try to use their functionality. It also is necessary to patch util-linux, compile the tools and install them. The relevant util-linux patch is available at [www.kernel.org/pub/linux/kernel/people/hvr/util-linux-patch-int](http://www.kernel.org/pub/linux/kernel/people/hvr/util-linux-patch-int). You should find that you end up with modified mount and losetup commands.

Now we are ready to create a loopback encrypted filesystem using a process similar to that which we used to create a vanilla loopback filesystem. First, in order to make it difficult to differentiate between encrypted blocks and unused disk space, the file that will hold the loopback filesystem is created using /dev/urandom instead of /dev/zero:

```
dd if=/dev/urandom of=ciphertext.img bs=1M count=20
```

After creating the host file, it must be temporarily associated with a loopback device, as before. This time, however, we must tell losetup that the loopback device is to be encrypted, in this case with the AES cipher:

```
losetup -e aes /dev/loop0 ciphertext.img
```

Enter the password and possibly—depending on the cipher you decided to use—the key size you wish to use for the volume when prompted by losetup.

Creating the filesystem is done in a manner identical to that for creating a vanilla loopback device. The encryption was set up in the previous step and is now handled by the loopback driver:

```
mkfs -t ext2 /dev/loop0
```

In addition to modifying `losetup`, the `util-linux` patch also makes the `mount` command crypto-aware. Mounting an encrypted loopback volume is a simple process, given the correct command parameters:

```
mount -o loop,encryption=aes ciphertext.img \  
mount point
```

**mount** prompts you for a password and possibly for a key size.

Now that you understand how to mount and unmount encrypted loopback filesystems manually, an introduction to `pam_mount` is appropriate. `pam_mount` is a PAM module that simplifies the management of volumes and should be mounted when a user logs in to a system. It can handle mounting things like Samba-hosted volumes, Novell-hosted volumes and encrypted filesystems. The original author of `pam_mount` is Elvis Pftzenreuter. Mukesh Agrawal wrote the patch that first added support for loopback encrypted volumes. The author of this article now maintains `pam_mount`, which is available at [www.flyn.org](http://www.flyn.org).

Instead of having to mount encrypted volumes manually, a system administrator can configure `pam_mount` to mount and unmount the volumes automatically when users log on and off. This can be configured so the system password also unlocks the encrypted volume, essentially creating a completely transparent encrypted volume.

`pam_mount` can employ three different techniques to unlock an encrypted volume. The first technique is rather boring. When the encrypted volume's key is unrelated to the system's login password, `pam_mount` simply prompts users for the key to their encrypted volume. In order to use this technique on a system, `pam_mount.so` and `pmhelper` must be installed and configured. The standard `./configure`, `make` and `make install` commands build and install `pam_mount`'s binaries and configuration file.

You should find the stock `pam_mount.conf` in `/etc/security`. Inspect and tailor it to your own system. The stock `pam_mount.conf` file is well documented. The most important change necessary is to add definitions for the volumes that should be mounted to the end of the file. The following is the definition format for encrypted loopback filesystems, as documented in the stock file:

```
volume user local ignored  
loopback file  
mount point mount options  
fs  
key cipher  
fs key path
```

Here is an example that mounts an AES-encrypted loopback filesystem hosted by /home/mike.img at /home/mike when Mike logs on:

```
volume mike local - /home/mike.img /home/mike
loop,user,exec,encryption=aes,keybits=256 - -
```

Next, add the lines `auth required pam_mount.so try_first_pass` and `session required pam_mount.so try_first_pass` to the configuration files of the PAM-supporting services you want to support loopback encrypted filesystems. As an example, here is the `/etc/pam.d/login` file from my laptop:

```
auth      requisite pam_securetty.so
auth      requisite pam_nologin.so
auth      required pam_env.so
auth      required pam_unix.so nullok
account   required pam_access.so
account   required pam_unix.so
session   required pam_unix.so
session   optional pam_lastlog.so
session   optional pam_motd.so
session   optional pam_mail.so standard noenv
password  required pam_unix.so nullok obscure \
          min=4 max=8 md5
auth      required pam_mount.so try_first_pass
session   required pam_mount.so try_first_pass
```

Finally, create the user's loopback encrypted filesystem using the steps listed in the introduction to encrypted loopback filesystems.

The second technique for `pam_mount` to unlock a volume is more convenient for users. If, when creating the encrypted volume using the same method as the first technique, a user specifies his or her login password as the volume key, then `pam_mount` unlocks the volume using the same password the user enters to login.

The third technique is the most flexible and requires a more sophisticated description. Here are a few terms to help you understand how this technique works:

- `sk`: system key, the key or password used to log in to the system.
- `fsk`: filesystem key, the key that allows you to use the filesystem you want `pam_mount` to mount for you.
- `E` and `D`: an OpenSSL-supported synchronous encryption/decryption algorithm, `bf-ecb`, for example.

- efsk: encrypted filesystem key,  $\text{efsk} = \text{E\_sk}(\text{fsk})$ , stored somewhere on the local filesystem (that is, `/home/user.key`).

`pam_mount` reads `efsk` from the local filesystem, performs  $\text{fsk} = \text{D\_sk}(\text{efsk})$  and uses `fsk` to mount the filesystem. This technique has the advantage of allowing users to change their login passwords without having to re-encrypt their home directories using this new key. If the login password is changed, simply regenerate `efsk` (that is, `/home/user.key`) using  $\text{efsk} = \text{E\_newsk}(\text{D\_oldsk}(\text{efsk}))$ . A script named `passwdhd` is included in `pam_mount` to do this for you.

To implement this third technique, begin by creating the file to host the encrypted filesystem (as before):

```
dd if=/dev/urandom of=/home/user.img \
bs=1M count=image size in MB
```

Then, create a file (`efsk`) containing the volume's password (`fsk`) using `/dev/urandom`, encrypted using the user's login password as the key:

```
dd if=/dev/urandom bs=1c count=keysize / 8 | \
openssl enc -bf-ecb > /home/user.key
```

Next, create an encrypted loopback filesystem. The filesystem's key should be `fsk` (generated using `/dev/urandom`, encrypted and stored as `/home/user.key` in step 2).

```
openssl enc -d -bf-ecb -in /home/user.key | \
losetup -e aes -k keysize -p0 /dev/loop0 \
/home/user.img
mkfs -t ext2 /dev/loop0
umount /dev/loop0
losetup -d /dev/loop0
```

Finally, in `pam_mount.conf`, set the `fs key cipher` variable to the cipher used to encrypt `fsk`, in this case `bf-ecb`, and set the `fs key path` variable to `efsk`'s path, in this case, `/home/user.key`.

In his definitive text, *Applied Cryptography*, Bruce Schneier states, "Software encryption is scary." What he means is, it is difficult to design truly secure encryption software for computers running general-purpose operating systems such as Linux. For example, modern operating systems can swap memory to disk at any time, and this memory could contain plain text or encryption keys. An encrypted volume is useless if its key has been written to disk by the operating system. One way to reduce the effects of this is to encrypt one's swap



volume. CryptoAPI still cannot do this safely, but it is in development. A similar project, LoopAES, already can encrypt a system's swap space.

Consider again the example where I sent my iBook to Apple for repairs. Though my home directory is encrypted, my data still may not be completely safe. A dishonest employee could boot his or her diabolical CD-ROM and replace, for example, the login binary on my system with the employee's own design. When my computer is returned and I log in, my encryption key could be shipped off to a remote computer by the newly installed login program. An intrusion detection system would make this scenario much less likely.

Another possible weak point in a system employing encrypted home directories using pam\_mount is the system's login password. Because the login password is used, directly or indirectly, to unlock an encrypted filesystem, it must be strong. Countless studies have shown that most passwords chosen by users are quite weak. Rather than blindly increasing the required length of passwords, spend some time reading Bruce Schneier's *Secrets and Lies*. A strong passphrase, written down and stored in your wallet may be more secure than a memorized password. The addition of a physical authentication token might be even better. Remember, if your system login password is not secure, your encrypted filesystem is as good as read.

Finally, encrypted filesystems can be a double-edged sword. What if you forget your encryption key? What if you use the third technique described above and accidentally delete all records of your encrypted filesystem key? What if my or someone else's encryption-related software is buggy? All of these problems can result in you having to try  $2^{128}$  or so different encryption keys to get your filesystem back. Your data may be as good as gone. As with any system administration endeavor, make filesystem backups. Ideally, these backups are *not* encrypted and are physically locked up somewhere secure.

The bottom line is many subtle considerations and procedures are required to administer a reasonably secure system beyond the use of a modern encryption algorithm like AES. To quote Matt Blaze's contribution to *Applied Cryptography*:

High-quality ciphers and protocols are important tools, but by themselves poor substitutes for realistic, critical thinking about what is being protected and how various defenses might fail (attackers, after all, rarely restrict themselves to the clean, well-defined threat models of the academic world).

After reading this article, you should be familiar with the concept of an encrypted loopback filesystem. In addition, you should be able to deploy encrypted filesystems on the systems you administer and manage them with

the pam\_mount PAM module. In the future, I would like to see the CryptoAPI patches and pam\_mount supported by the major Linux distributors. I also would like to see the CryptoAPI patch rolled into the mainstream util-linux package. Properly administered encrypted home directories are a powerful security tool.

Mike Petullo is a platoon leader in the US Army, stationed in Germany. He jumps out of airplanes by day, fights C code bugs by night and has been tinkering with Linux since early 1997. He welcomes your comments sent to [lj@flyn.org](mailto:lj@flyn.org).

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## Take Control of TCPA

**David Safford**

**Jeff Kravitz**

**Leendert van Doorn**

Issue #112, August 2003

Can you trust "Trusted Computing"? Learn how it works with free software that lets you store your own keys securely.

The Trusted Computing Platform Alliance (TCPA, [www.trustedcomputing.org](http://www.trustedcomputing.org)) has published open specifications for a security chip and related software interfaces. The TCPA chip is designed to provide client machines with a minimal but essential hardware base for client-side security. It provides two important security functions: secure storage of signature and encryption keys and system software integrity measurement. TCPA's secure storage can be used to protect an individual's RSA authentication private key or a loopback filesystem's master key from theft or disclosure. TCPA's integrity measurement can be used to detect software compromise, such as a rooted kernel, and to lock down use of protected keys and data if a compromise is found.

IBM is now shipping models of ThinkPad and desktop machines with TCPA chips. IBM has published a set of open-source tutorial code for TCPA, available at [www.research.ibm.com/gsal/tcpa](http://www.research.ibm.com/gsal/tcpa). This tutorial package is intended to help teach people about TCPA and to introduce programming of the TCPA chip under Linux.

In this article, we try to help you better understand the TCPA specification and tutorial package by introducing the fundamentals of TCPA, describing the IBM open-source TCPA tutorial package for Linux and explaining how you can use TCPA to sign documents and store the key for an encrypted loopback filesystem.

## The TPCA Chip

A TPCA security subsystem includes both hardware and software components. Functions provided by hardware are called TPM (trusted platform module) functions; those provided by software are TSS (trusted support services). From a programmer's perspective, the IBM version of the TPM (or TPCA chip) looks like Figure 1.

Functional Units	Non-volatile memory	Volatile memory
RNG	Endorsement Key (2048b)	RSA Key Slot-0 ...
Hash	Storage Root Key (2048b)	RSA Key Slot-9
HMAC	Owner Auth Secret (160b)	PCR-0 ...
RSA Key Generation		PCR-15
RSA Encrypt/Decrypt		Key Handles Auth Session Handles

Figure 1. Programming View of the TPCA Chip

The TPM includes five cryptographic functional units. It has a hardware random number generator (RNG), which provides a source of high-quality random numbers for on-chip key generation, as well as for application use. It has a hash unit (SHA-1) and an associated hashing for message authentication calculator (HMAC). It also has the ability to generate RSA keys of up to 2,048 bits on the chip, based on random numbers supplied by the RNG. Finally, it has an RSA engine that can perform signatures, encryption and decryption. The TPM does not do signature verification, as this is not a sensitive operation and can be done more easily and quickly with software.

The TPM stores three important keys in non-volatile memory. The endorsement key is a 2,048-bit RSA public and private key pair, which is created randomly on the chip at manufacture time and cannot be changed. The private key never leaves the chip, while the public key is used for attestation and for encryption of sensitive data sent to the chip, as occurs during the TPM\_TakeOwnership command. Because this key is sensitive from a privacy perspective, its use can be disabled completely by the TPM owner.

The storage root key (SRK) is a 2,048-bit RSA key pair. It is initially empty and is created as part of the TPM\_TakeOwnership command. This key never leaves the chip. It is used to encrypt (wrap) private keys for storage outside the TPM

and to decrypt them when they are loaded back into the TPM. The SRK can be cleared by the system owner.

The Owner Authorization secret key is a 160-bit secret shared with the owner of the TPM. The owner loads it into the TPM as part of the TPM\_TakeOwnership command. This secret key is used to authorize sensitive owner command requests.

The volatile memory section contains ten slots for temporary storage of RSA key pairs. Any number of wrapped keys can be stored externally and loaded as needed into these slots for use. Although loaded keys are considered volatile and are not guaranteed to persist across power down, in the case of the IBM chip version, keys normally do persist and may need to be evicted to make room for the loading of new ones.

The volatile memory section also contains 15 platform configuration registers (PCRs). These registers contain 160-bit measurements (hashes) of software integrity. During system boot, measurements are taken of the BIOS, extension BIOSes, MBR, GRUB bootstrap stages and any designated files, such as the kernel. These measurements are added to various PCRs. The BIOS actively resets all PCR values at boot time power-on. When a system resumes from a suspended state, the BIOS tries to start the TPM in a mode that restores saved PCR values. For this to work, the TPM device driver must have issued a TPM\_SaveState command right before the chip was suspended.

Volatile memory also is used to store two types of handles. Key handles are used to give temporary names to loaded keys, so subsequent commands can indicate which key should be used, if multiple keys are loaded. Key handles are cleared when the respective key is evicted. Authorization session handles are used to identify authorization state data across multiple commands. Authorization handles are created by TPM\_OSAP and TPM\_OIAP authorization commands, and they must be cleared with a TPM\_Terminate\_Handle command when no longer needed.

### **Enabling and Clearing the TPM**

Before looking at these specific TPM commands, we should cover one of the more mystifying aspects of the TPM—how to get it started. Fortunately, the BIOS is responsible for starting up and clearing the TPM, so this really is not as complex as it looks to be in the TPM specification. At power-on, the TPM is activated but not started. The BIOS then must issue a TPM\_Startup command. This command can do one of three things: deactivate the TPM, start up the TPM with a reset of the PCR registers or start up the TPM with a restore of PCR values from their saved states (as with a resume). If the BIOS deactivates the TPM, it remains deactivated until the next power cycle; no software command

can reactivate it. A startup with clearing of the PCRs is done at boot time, so all PCR values are calculated correctly during boot. The TPM device driver is responsible for making a TPM\_SaveState request at suspend time to ensure that valid PCR values are available at resume time.

The BIOS also is responsible for performing a TPM\_ForceClear if desired. The clear command is a complete reset of the TPM, and it unloads all keys and handles and clears the SRK and owner authorization secret. TPM\_ForceClear requires proof of physical presence, which normally is given by holding down the Fn key (blue key at the bottom left) when powering on the system.

The control of TPM deactivation and clearing by the BIOS is set in the BIOS setup mode. To get started with the TPM, then, hold down the Fn key and press the Power-On button. When the BIOS screen appears, release Fn, and press F1 to enter BIOS setup mode. Next, select Config→Security System, then select Enable and Clear entries. These steps enable operation of the TPM and clear the chip, so it is ready for us to take ownership.

### Talking to the TPM

The TPM device driver, tpm.o, is a loadable kernel module that provides a character device interface to the TPM chip. It is registered officially as Linux major number 10, minor number 224. Applications normally access it through the special file /dev/tpm.

To send a command to the TPM, /dev/tpm is opened for read/write, a command packet is written and the response packet is read. The TPM can process only one command at a time, so the entire request must be sent and the entire response must be read before another request can be made.

All command packets have a common structure:

16-bit unsigned TAG	type of packet
32-bit unsigned Length	length of total packet
32-bit unsigned Ordinal	TPM command number
variable	command data

All response packets have a similar structure:

16-bit unsigned TAG	type of packet
32-bit unsigned Length	length of total packet
32-bit unsigned Return	return code
variable	returned data

All 16- and 32-bit values are in network byte order (big endian) and must be converted to and from host byte order. On writes to the TPM, write exactly the number of bytes in the packet, as indicated in the packet's total length field. When reading the response, you should attempt to read 4,096 bytes (the defined maximum TPM packet size), and the return value of the read indicates how many bytes are in the returned packet. This should match the returned packet's length field exactly. The return code is zero for a successful command, and a positive value is a specific error code.

A function for sending/receiving TPM packets can look something like the following (error handling omitted for clarity):

```
uint32_t TPM_Transmit(unsigned char *blob)
{
    int tpmfp, len;
    uint32_t size;

    tpmfp = open("/dev/tpm", O_RDWR);
    size = ntohl(*(uint32_t *)&blob[2]);
    len = write(tpmfp, blob, size);
    len = read(tpmfp, blob, 4096);
    return(ntohl(*(uint32_t *)&blob[6]));
}
```

### Some Simple TPM Commands

Once the TPM is enabled and cleared through the BIOS setup and the TPM device driver is loaded, we can try some simple TPM commands. The TCGA main specification details some 73 TPM commands. Fortunately, we can demonstrate the desired signing and sealing functionality in this tutorial with only 14 of these commands.

The simplest command is TPM\_Reset, a request to flush any existing authorization handles. TPM\_Reset is a nice command to test a driver and library, as it is short, fixed and should always succeed, returning a result code of zero. Here is the example code for TPM\_Reset:

```
uint32_t TPM_Reset()
{
    unsigned char blob[4096] = {
```

```

    0, 193,      /*TPM_TAG_RQU_COMMAND*/
    0, 0, 0, 10, /* blob length, bytes */
    0, 0, 0, 90}; /*TPM_ORD_Reset */
return(TPM_Transmit(blob));
}

```

It is important to size blob[] to allow the returned TPM data to be up to the maximum allowed packet size of 4,096 bytes.

The TPM\_GetCapability command is another simple function that can return several items of information about a given TPM. It can return the version of the current TPM, the total number of key slots in the TPM (typically ten), the number of loaded keys and their handles and the number of PCR registers (typically 16). Here is the example code for using TPM\_GetCapability to read the TPM version:

```

uint32_t TPM_GetCapability_Version()
{
    unsigned char blob[4096] = {
        0, 193,      /* TPM_TAG_RQU_COMMAND */
        0, 0, 0, 18, /* blob length, bytes */
        0, 0, 0, 101, /* TPM_ORD_GetCapability */
        0, 0, 0, 6, /* TPCA_CAP_VERSION */
        0, 0, 0, 0}; /* no sub capability */
    return(TPM_Transmit(blob));
}

```

TPM\_PcrRead returns the 20 bytes (160 bits) of a specified PCR register. It is useful to check that any desired TPM measurements are being made by the modified GRUB loader.

TPM\_ReadPubek is used to read the TPM's fixed public endorsement key (Pubek). Pubek initially must be read so it can be used by the owner to encrypt sensitive data in the TPM\_TakeOwnership command. Once ownership is established, the owner typically disables reading of the Pubek for privacy reasons; after that, then this command fails.

### TPM Authorization Protocols

Some TPM commands require authorization. Owner-related commands normally require authorization based on knowledge of the owner authorization 160-bit secret. Similarly, the use of keys may require authorization based on the key's authorization secret. Normally, this is done in the form of a hash of password, or PIN, applied to the key when it is created.

The TPM supports two protocols for this authorization: Object Independent Authorization Protocol (OIAP) and Object Specific Authorization Protocol (OSAP). Both protocols are similar in that they create an authorization context



with a handle returned to the user, and they both use rolling nonces. The main difference is OIAP creates a long-term session with a new session secret key, and it can be used across multiple objects within a session. OSAP relates to a single object, such as a given key. In the case of TPM\_TakeOwnership, OIAP must be used because the objects and secrets have not yet been established. In most other cases, either authorization protocol may be used.

TPM\_OIAP and TPM\_OSAP both create authorization handles that should be terminated (freed) when finished. This is done with the TPM\_Terminate\_Handle command.

### **TPM\_TakeOwnership**

We are ready to perform the essential TPM\_TakeOwnership. This command executes four critical functions: it installs the owner-supplied owner authorization secret, creates the SRK, applies the owner-supplied SRK authorization secret and, optionally, returns the Public SRK portion to the owner. With the SRK available, we now have a functional TPM and are able to create and use signature and encryption keys.

### **Creating and Using Keys**

TPM\_CreateWrapKey generates a new RSA key on the chip, using the hardware RNG. A key must be typed as being either for signing or for encryption/decryption. The TPM does not allow a signature key to encrypt or an encryption key to sign, as this can lead to attacks. A key optionally may be given a secret that it is required to produce to use the key in the future. In addition, keys can be wrapped to specified PCR values. If this is done, both the authorization data and specified PCR data must match to use the key. All keys must have a parent key—it may be the SRK—that is used to encrypt the private part of the key, before the key structure is returned to the user. The returned key data must be stored by the user for future loading.

TPM\_LoadKey is used to load a key into one of the volatile key storage slots in the TPM. This command requires the authorization password for the parent key; once loaded, the TPM uses the parent key to decrypt the loaded key's private data for use. If the key has an authorization secret, it is not needed to load the key, but it is required for any subsequent command that tries to use the key for encryption or signing.

Because a limited number of key slots are available in the TPM, when a key is no longer needed, it must be evicted to make the slot available for other keys.

The TPM\_Sign command uses a loaded key to sign presented data, normally the hash of the actual data. TPM\_Seal is used to perform RSA encryption of data; it

requires a loaded encryption key and any authorization secret for that key. TPM\_Seal also may specify PCR values to be used in the seal. If a future unseal is attempted without matching PCR values, the unseal fails. TPM\_Seal also applies a used supplied data authorization value (password) to the sealed data. Thus, to unseal the data, the user may require the password for the sealing key and for the data, and the PCR values may have to match. TPM\_Unseal performs the corresponding unseal operation.

### **The T CPA Linux Tutorial Package**

The IBM T CPA tutorial package provides source code for five major components: device driver, libtcpa, examples, GRUB patch and loopback patch.

The device driver code allows you to compile a tpm.o loadable module for your kernel. The libtcpa code provides easy-to-use C interfaces for the application level TPM commands discussed in this article. The example programs demonstrate how to use libtcpa to do common actions, including taking ownership, creating keys, loading keys, signing, sealing and unsealing. The GRUB patch is a source code patch to the GRUB bootloader. It adds support for PCR measurement of grub itself and of any designated files, such as the kernel. The loopback patch is a source code patch to the loopback driver and associated utilities. This patch allows the loopback encryption key to be stored in TPM sealed form and releases it only if presented with the corresponding password and only if the PCR values match. With this patch installed, loopback mounting appears normal; it asks for a password, but this password is used to authorize only the unsealing of the actual loopback key data.

So, what does the use of the T CPA chip for signing and sealing/unsealing do for us? Our private keys are created on the chip, and they never leave the chip unless encrypted under a protected public key. The use of the PCRs also can protect our keys by refusing to authorize their use if the system has not been booted in the proper way, or if the integrity of measured files has been compromised. Sealing a loopback key similarly can protect against alternative booting and compromised software.

### **Next Steps**

The IBM T CPA tutorial package is not a complete TSS implementation, as it was mainly intended to make T CPA easier to understand. It does not do TPM resource management for handles and loaded keys nor does it give access to the TPM's key backup and migration facilities. These are topics for future development and articles.

David Safford is a researcher at IBM's T. J. Watson Research Center where he leads a security analysis group and gets to play with fun things like TCPA, Linux and wireless security. He can be contacted at [safford@watson.ibm.com](mailto:safford@watson.ibm.com).

Jeff Kravitz works in IBM's T. J. Watson Research Center where he has worked on various projects, including communications gateways, multitasking operating systems for embedded systems and software for controllers of gigabit optical networks. Jeff currently works on the uses of public key cryptography.

Leendert van Doorn is a researcher at IBM's T. J. Watson Research Center where he runs the secure embedded systems group. He has actively hacked on many versions of UNIX (starting with V6), Amoeba, Paramecium and Linux. He even has been known, but strongly denies, to have written Windows drivers. His current interests include operating systems, security, secure coprocessors, simulators and hypervisors. He can be contacted at [leendert@watson.ibm.com](mailto:leendert@watson.ibm.com).

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## The Power of the Incredible Hulk—the ILM Linux Death Star

**Robin Rowe**

Issue #112, August 2003

The story in the July 2002 issue of *Linux Journal* about the Linux desktop and renderfarm conversion at Industrial Light & Magic (ILM) drew record interest from readers. Linux has advanced at ILM in the past year, meeting the special challenges ILM faced in creating the movie *The Hulk* (release June 2003).

The *Star Wars* Death Star exists! But, not as a menacing planet-killing weapon. ILM's Death Star renderfarm is the computing power behind the many motion pictures produced at Industrial Light & Magic. "Linux will just continue to grow", says ILM CTO Cliff Plumer. "Our renderfarm has over 1,500 processors currently, and almost 1,000 more are added every evening with desktops." The renderfarm utilizes both dedicated CPUs and the computing power of idle desktops.

### The Death Star Renderfarm

"Our core renderfarm is comprised almost entirely of Linux boxes since we switched over from SGI machines", says ILM Systems Developer Mike Thompson. "We have about 750 nodes—1,500 CPUs." The renderfarm exists as a row of computing towers made up of 1U rackmount dual-processor PCs. However, this isn't a supercomputer in the classic sense. Each machine operates semi-independently in a grid configuration, not bound together as a supercomputer running a single job. At ILM, a proprietary batch-scheduling program called ObaQ manages the workload across machines.



Figure 2. ILM RackSaver Linux renderfarm has 1,500 processors now, which will double for *Star Wars Episode III*.

As the raw horsepower of systems increased, so did the demand for electricity and cooling in ILM's machine room. "It's important to reduce power and heat", says Thompson. "We went with AMD Athlon 1600 CPUs, a low-power variant that can be a little difficult to find." Each node has 2GB of memory, expandable to 4GB, and each node usually runs two jobs at once to utilize the dual processors, but sometimes this is reduced to a single job for full use of memory.

The AMD-powered nodes are RS-1100 units produced by RackSaver. RackSaver caught ILM's attention at a National Association of Broadcasters (NAB) convention and was given the opportunity to bid on building the renderfarm. RackSaver competes mostly with heavyweights IBM and Dell. RackSaver CEO David Diggers says RackSaver's advantage is servers with double the density of competitors. "We're very strong in this vertical market with sales to ILM, Pixar and Warner Brothers", he said.

The RackSaver renderfarm servers are connected via 100BASE-TX into a Foundry 8000 switch that aggregates network traffic into a gigabit link into the network core. "Just recently we added a 10-gigabit link into our core which helped a lot", says Thompson. "We have a file server core and 2,500 rendering cores with total aggregate traffic of about 70TB a day."

### **Why Green Is Scary**

"*The Hulk* is not a typical comic book movie", says Technical Director Doug Sutton. "It is some of the most challenging work we've done in years. Making a

15-foot-tall green guy look real is an incredible challenge!" The Hulk is a computer-generated (CG) digital actor with emotions and complex green skin.

"Film is designed to make people not look green, to push green away", says Principal Software Engineer Rod Bogart. "We're using Kodak Premiere print stock that has deeper color than Kodak Vision. If the character is green, as with Hulk, it's hard to make it look like green skin. The green dinosaurs of *Jurassic Park* are not as hard. Dinosaurs don't have the same sort of skin highlights as people." The audience is less forgiving about human faces—even green ones. "Green passes through yellow as it goes to white", adds Sutton. "You need to see that on a monitor in order to counteract or accept it."

Live action was shot in the streets of San Francisco, California. Jennifer Connelly would be in the middle of a street acting to a big green guy who isn't there. A grip holding a pole with green head on top would be her only cue. "They did a few really cool practical effects, explosions, but mainly it is CG", says Sutton, "a combination of Maya and SOFTIMAGE".

### Cineon and OpenEXR

Capturing film images demands higher dynamic range than the typical JPEG or PNG supports. Kodak Cineon has long been the standard for digitized film. Cineon is a 10-bit logarithmic format. Compared to JPEG, which is 8-bit linear (that is, 24-bit RGB), a Cineon image has more dynamic range and is especially rich in colors near black. As computing power has increased, most computation has switched to floating point, and working in 10-bit log has become a limitation. OpenEXR is a new floating-point image format.

"The OpenEXR file format is a better digital representation of film because it has a dynamic range of over 30 f-stops without loss of precision", says Plumer. "Previous 8-bit file formats have the dynamic range of only around seven to ten f-stops and cannot accurately reproduce images with extreme contrast." ILM created the EXR format in the summer of 2000 and has used it on *Harry Potter and the Sorcerer's Stone*, *Men in Black II*, *Gangs of New York*, *Signs*, *Dreamcatcher*, *The Hulk*, *Van Helsing*, *Peter Pan*, *Timeline* and *Pirates of the Caribbean*.

OpenEXR uses lossless compression like PNG, not lossy compression like JPEG. Actually, there is an unused Piz12 lossy compression option. Unlike PNG or JPEG, OpenEXR uses wavelet encoding—basically a tree structure containing the signed differences between pixels. Because the magnitude of the numbers is smaller and there are fewer unique values, Huffman encoding can compress that more efficiently. EXR supports 32-bit float, 32-bit integer and 16-bit float to any number of channels. Channels can be different depths; for instance, RGBAZ

images need more precision in Z depth—with 16-16-16-16-32 typical. The Z channel is physical depth, not a color or alpha mask—think of Z sort of like sonar. In January 2003, ILM released open-source OpenEXR. The first open-source application to support it was CinePaint.

### CinePaint

CinePaint, which until recently was called Film Gimp, is a frame-by-frame motion picture retouching system that branched from GIMP in 1998. I became the CinePaint project leader serendipitously after I wrote some articles about Film Gimp for *Linux Journal*. In addition to the usual still-image formats, CinePaint supports file formats popular in the motion picture industry. Those formats include Cineon, RnH 16-bit float (a format created by studio Rhythm & Hues that chops off half of a 32-bit float), Radiance HDR, LogLuv TIFF and now OpenEXR. ILM wrote the OpenEXR plugin for CinePaint.

### SDevs, LUTs and Lattices

“We never want to look at an image without an appropriate SDev”, says Sutton. “It’s important that what you see is what you get. An SDev—a simulation device—is how we make monitors look like film. Since switching to OpenEXR (more on that in a moment) we don’t use LUTs much anymore, but we use SDevs all the time.” A LUT is a LookUp Table used to adjust an image to correct gamma. Monitor brightness is not proportional to the input voltage, but rather to the input voltage raised to a power. This exponent is called gamma and varies depending on the display. Macs are usually about 1.8 and PCs about 2.2.

“The way LUTs work is mainly to change contrast”, says Bogart. “What a LUT can’t do is increase or decrease saturation.” Instead of LUTs, a more complex lattice computation is used at ILM. Think of a lattice as a  $13 \times 13 \times 13$  cube in space—a 3-D indexed array. An odd number is used so the center is gray. In lattice, each RGB value is mapped—not like a LUT that maps per channel. Using three independent lookup tables is not sufficient for getting the look of film—especially with saturated green. The lattice adjusts to a 12-point film curve using a calculated table with 64k entries. An index into the lattice between 0 and 1 returns three values using trilinear or tetrahedral interpolation that are then gamma-corrected. The process is slower, because each pixel must be handled together, but more accurate than the typical RGB channels-based lookup. “Lattices are not just a *Hulk* thing”, points out Bogart. “For *Minority Report* that was a bleach process print—very desaturated. We simulated that look with lattices. You can’t do desaturation with LUT either.”



## GPU Programming with NVIDIA Cg

The raw 16-bit OpenEXR data format is called Half, as in half of a 32-bit floating-point number. The Half data format is an internal format of NVIDIA graphics cards. It would be nice if the lattice calculation, which consumes CPU cycles, could instead be run directly on the graphics processing unit (GPU) on the graphics card. In fact, that's becoming possible due to advances in graphics cards. "We're looking forward to that", says Bogart. "We intend to offload image calculations to the GPU running a pixel shader." NVIDIA offers a new C-like compiler/library called Cg to run bits of pixel code, commonly called shaders, on the GPU. ATI offers a similar technology called High Level Shading Language, and 3Dlabs has OpenGL Shading Language.

GPU programming is something like embedded systems programming, where code is compiled on a host platform then downloaded to the embedded system. GPU programs can be compiled and downloaded to the graphics card at runtime. The compiler is part of the runtime library.

Some 3-D packages, such as SOFTIMAGE and Maya, already are beginning to use Cg to improve rendering performance.

### Comparison of Floating-Point Number Formats

Format	Total Bits	Sign	Exponent	Mantissa
IEEE 754 Double	64	1	11	52
IEEE 754 Float	32	1	8	23
NVIDIA/ILM Half	16	1	5	10
RnH Float16	16	0	8	8

## Modeling and Rendering

Alias|Wavefront Maya was used for particles and some of the character animation models. Pixar RenderMan and Mental Images Mental Ray software were used for rendering. Raytracing renders reflective surfaces better but takes longer. Raytracing is becoming more practical, thanks to the faster, cheaper Linux systems. Both RenderMan and Mental Ray support shader programming to give images a custom look. RenderMan provides its own shader language, which is considered easy to learn. Mental Ray uses C, which is considered more challenging but more powerful. Which software to use is decided on a scene-by-scene basis. Each scene is rendered under the control of a batch scheduler.



## The ObaQ Batch Scheduler

“Florian Kines, who was also behind OpenEXR, wrote our batch scheduler along with a couple others a long time ago for SGI Irix”, says Hess. “That made use of big iron and desktops. When we started our move to Linux we wanted better resource management.” The first version of ObaQ divided machines by show—not a very efficient utilization of resources.



Figure 3. ILM proprietary ObaQ scheduler running a Linux batch render for the movie *The Hulk*.

“Our attempt to replace ObaQ with a centralized resource management system called the IMP Project didn't work out”, says Hess. “We went back to ObaQ, and the Linux port of that took about two weeks. Three or four months ago, Florian decided he was going to fix that so any show could use any machine.” ObaQ is a peer-to-peer (P2P) scheduler system. The advantage of a P2P scheduler is that a scheduler server failure won't knock the entire system off-line. ObaQ2 uses a single machine for global scheduling, but it advises only independent machines running ObaQ. Losing the ObaQ2 server won't bring the entire facility down. There are scheduler system alternatives, such as the popular proprietary product Platform LSF or the open-source Condor and OpenPBS schedulers, but ILM plans to continue to use ObaQ.

SGI had added functionality in the IRIX kernel for process monitoring, such as CPU time and temp space. Those values determine how ILM machine time gets charged back by central accounting to projects. ILM discovered the Linux /proc filesystem didn't provide all those statistics or created excessive overhead, and that it couldn't support ObaQ without changes.

## Death Star Kernel Hacking

“Florian asked me to address some of the Linux kernel issues”, says Hess. “For one thing, Linux provides no way to tell if something is a thread or a process. In **ps**, every thread shows as a separate process.” Some jobs, such as Mental Ray, can run multiple threads per frame in parallel. Linux **top** or **ps** shows each thread using 1GB RAM, but that's shared memory being counted twice. Linux also couldn't tell which job is opening temporary files. ObaQ needs to know that in order to clean up temporary files if it kills a job.

Hess created a Linux kernel module to trap opens, forks, clones, vforks, exits and renames, to make accurate statistics possible. The kernel module does most of work, but the hooked calls should ignore any job not being run by ObaQ. To do that required hacking the kernel. “I used one of the unused bits in the ptrace flag”, says Hess. “Every x86 job has a 32-bit ptrace vector. As of 2.4.20, 10 bits are used to indicate ptrace modes, such as single step. Sometime last year Linux or glibc changed how the ptrace flag works so it clears on fork. I found all places the kernel clears those bits and keep bit 32.” Hess says the OPROFILE feature in the 2.5 kernel has enhanced accounting facilities, so his hack might not be needed in 2.5. Commandeering an unused bit in the ptrace flag was a quick hack to mark jobs as being ObaQ tasks. “This is one of the great things about Linux”, says Hess. “Because we had the source, we could make this change ourselves, and very quickly. No third-party vendor had to be involved to do custom engineering, as in the IRIX case.”

## NFS Troubles

“Now that we have all this firepower in the renderfarm it can overwhelm any file server”, says Thompson. “In *The Hulk* we have these nuclear explosion renders that are really crunchy—causing major grief for us lately. It is easy for an artist to proc-up a render [add more processors to a task] to the point that it brings a file server to its knees. We're doling out 700 times the data we used to!”

ILM uses a Sun T3 disk array to serve NFS. Adopting Linux as an NFS client presented a number of problems when brought on-line a year and a half ago. Due to a Linux NFS UDP-packets-out-of-order bug (fixed in 2.4.18), after a couple hours the Sun Solaris server would spin up to 100% and be dragged down. Sun came to the rescue with a proprietary Solaris kernel module and IP stack patch to work around the Linux bug.

A nagging issue from choosing Linux NFS UDP is no flow control. “When we get into hot spot problems on file servers, the renderfarm makes a denial-of-service attack on our file servers”, says Thompson. “We're going to try TCP NFS

on a Linux client again, now that it's a year and a half later. We'll start testing that next week." TCP adds about 5% overhead.



Figure 4. ILM's Michael Thompson and the 20TB EMC File Server

ILM is scaling up, from about 20TB of file server storage now to double that next year. "For *Star Wars Episode III* we're going to double the size of our renderfarm", says Thompson. "We can do that by ordering another 3,000 nodes from RackSaver—but that could destroy our file servers." Thompson plans to head off that NFS server meltdown by going to a clustered file server—Sistina GFS or something like that. File serving isn't limited to only within the ILM facility.

### Digital Dailies

ILM ships dailies worldwide over the ILM Conduit, a proprietary file-transfer system that uses an encrypted-SSL transport. Everything is doubly encrypted with Blowfish. ILM has playback software for Windows, Macintosh and a Web-based Java applet version that works everywhere. "MJPEG-A QuickTime is our core movie container format", says Thompson, "but Conduit can carry anything—match-move data, digital pictures, dailies. People can play back dailies on Linux desktops across regular network connections. That's pretty impressive. It used to be you could do that only on SGI equipment". For dailies, ILM has 20TB in EMC Clarrion FC4700 arrays, fronted by 4-proc Sun E420R servers with 4GB memory and gigabit Ethernet. "Shot disks" are arranged in quarter-terabyte chunks of storage.

## **The Linux Greenlight**

“Linux means having incredible amounts of processing power to solve any problem”, says Sutton. “In *The Hulk*, we had I don't know how many layers of textures for skin and hair. We can do incredibly complex scenes using Linux.” What movies a studio makes is influenced by cost and schedule. Faster, cheaper Linux means more movies.

## **Acknowledgements**

Jimmy Perry ([jimmy@racksaver.com](mailto:jimmy@racksaver.com)), marketing coordinator of RackSaver, Inc., for his conception and aid in the development of this feature.

Robin Rowe ([Robin.Rowe@MovieEditor.com](mailto:Robin.Rowe@MovieEditor.com)) is a partner in the motion picture technology company MovieEditor.com, the release manager of Film Gimp and the leader of LinuxMovies.org and OpenSourceProgrammers.org.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

## Root for All on the SE Linux Play Machine

**Russell Coker**

Issue #112, August 2003

SE Linux gives you an extra layer of security that protects the system even from root. Russell decided to show how it works by giving everyone root access.

Since the middle of 2001, I have been working on NSA Security Enhanced Linux [see page 20 in this issue] both on packaging it for Debian and in general development. When describing the project to Linux users, I find much confusion exists about what SE Linux does; it is difficult to gain a full understanding of SE Linux from reading the documentation or attending a presentation. Also, many people who have prior experience in security want to gain some practical experience but don't have the time to install SE Linux to experiment. I decided that a good way to teach people about SE Linux would be to set up a machine with public access for anyone to use.

Demonstrating SE Linux in a regular configuration is not particularly exciting, as the only noticeable operations it restricts for non-root users are **ps ax** and **dmesg**. In a default configuration, **ps ax** shows an unprivileged user only the other processes in the same user domain, and **dmesg** is blocked. This is similar to the restrictions imposed by OpenWall and is nothing new in itself. I decided to grant root access to the world using only SE Linux for security, so users can see exactly what it is capable of doing.

On June 6–9, 2002, at LinuxTag in Karlsruhe, Germany, I ran an SE Linux demo machine at the Debian stand. This was the first SE Linux play machine. At the time, the default policy was less restrictive than it currently is. It allowed **setuid** and **DAC\_OVERRIDE** capabilities for regular users (**user\_t** domain). For a regular SE Linux configuration, this is fine. SE Linux does not use **uids** when deciding whether to grant access, and **DAC\_OVERRIDE** allows overriding the UNIX access controls, but not any SE controls. The reason these capabilities were granted was to allow running **setuid** programs from the **user\_t** domain without needing SE Linux domains for such programs. So although those capabilities were satisfactory for the typical user, they were not suitable for the unusual situation

of having a root user who should be banned from accessing other uids in the same domain. I removed these capabilities from user\_t, restricted the root account to the user\_r role and it was ready to go.

In recent releases, the default policy has changed to not grant setuid or DAC\_OVERRIDE capabilities to user\_t. So, the most significant security policy difference between my play machine and a real server is that on the play machine unprivileged users are permitted to read the kernel message log (dmesg) and the security policy source as an aid to understanding SE Linux.

My SE Linux challenge is based on a machine deliberately configured to be less secure than real servers, by granting log file access, granting read access to the security policy and allowing unprivileged users root access. In spite of these factors, little success was had in breaking the security.

On the first day of LinuxTag, a potential issue with /boot files was reported. A user believed he could determine the LILO password from the LILO map file. I immediately changed the policy to restrict the access to /boot to prevent such problems. Of course, if you have physical access to a machine you usually can break the security somehow, but we want to make it as difficult as possible.

During the event, I started work on support for multiple user roles. The initial reason for this was one of my colleagues used the play machine for more serious purposes. He lost all his files, because they were created by the root:user\_r:user\_t security context as uid root, the same as users who were testing the security. The standard test that *everyone* ran as root was `rm -rf /`, which deleted all his files. The system itself was unharmed, as files in /bin, /etc and other system directories cannot be unlinked or written to by user\_t. After I gave my friend an account with the domain user1\_t, his files could not be accessed by a root user in domain user\_t.

On June 17, 2002, I set up an SE play machine on a Cobalt Qube that is available on the Internet for everyone to use. The first machine was on-line intermittently until July 11. The uptime for the play machines has not been great, because they need to be monitored continually. Such a machine would have the potential to become a risk to everyone, including me, my ISP and people who use it, if it was cracked and I didn't respond fast enough. So whenever I go on holidays or am busy with work, I have to take it off-line.

### **How It Was Set Up**

The machine had its own iptables setup to prevent undesired network access from leaving the local machine. It also was placed behind a firewall, which applied similar restrictions on data transfers. This setup prevented any user from even probing my firewall from the inside unless they first cracked the



security of the play machine. I initially allowed most outbound network connections other than SMTP, but I soon changed this to allow only outbound connections to a Web proxy. SSH tunnels could be used for other Net access. Also, I denied X forwarding so that if a user mistakenly enables it on his client, his machine can't be attacked by other users on the play machine.

### How Secure Was It?

When the play machine had been on-line for less than a day, a user reported that `/etc/shadow` could be read. This directory was declared to be outside the scope of the LinuxTag demonstration, but I should have fixed it before putting the machine on-line. I changed the shadow file to have the type `shadow_t`, which required changes to the `spasswd` wrapper program and the SE policy for it.

Adding full support for `shadow_t` was difficult because, in many instances, the same program changes `/etc/passwd` and `/etc/shadow` by re-creating them, thus giving them the default context of `etc_t`. I could have modified those programs to use the `open_secure(2)` system call to specify the security context at file creation time. I decided not to, however, because it would involve a lot of work on security critical applications, creating the risk that an error might weaken security. Instead, I wrote wrapper code to run those programs and set `/etc/passwd` back to `etc_t` after the program exits. I also made `shadow_t` the default type for those programs when creating files in `/etc`. Still, even when `/etc/shadow` had the type `etc_t`, it prevented unauthorized root users from writing to it. It was read-only to root users in the `user_t` domain.

The next day, someone discovered that `/dev/nvram` was not adequately protected. It was writable by everyone, therefore any user could make the machine unbootable by scrambling the BIOS setting. Potentially, they could have made the Qube BIOS pass different parameters to the kernel to weaken security on the next boot. The Cobalt BIOS performs the functions that a bootloader such as LILO would perform on other machines. I changed the policy to fix that glitch immediately. It is important to note that different platforms, either different CPU architecture or a different hardware, may require similar minor changes to the security policy to match different device nodes in `/dev`. With the current policy there is little risk of this causing insecurity, as the default is to deny most operations related to device nodes.

Some people were concerned that I had not appropriately granted authorization and wanted reassurance that they were not doing anything illegal, so I changed the `/etc/motd` to state that the machine was put on-line for the purpose of security testing. I explained that it would be acceptable to break the security in any way, including methods that may render the machine unusable, as long as I was informed of the method used. I also stated that it

was not to be used for launching attacks on other machines, although I tried to make that impossible with firewall rules. Finally, I requested that no one try denial-of-service (DoS) attacks, as they are boring and that is not the aim of the exercise.

From June 20 on, the operation of the play machine was fairly uneventful. In February 2003, I put a play machine on-line at the Debian stand at OSDDEM and announced it as a capture the flag contest. This received a surprising amount of interest; at times there were up to 30 people watching one person trying to crack the security. A user managed to get the easy flag, accessing a file in a specified non-root account after logging in as root. He did this by setting the EDITOR environment variable and running `crontab -e`. The crontab program ran the editor with more SE privileges than a regular program and allowed greater access. Even though the exploit would not work in a typical server configuration, because you don't give untrusted users root access even if you are running SE Linux, I changed the policy for the crontab program to prevent doing so. Even with this, the crontab attack still was confined to a single user role. Therefore, any accounts that were in different domains, such as the one I used for running the play machine, could not be touched.

One ongoing problem I experienced was that of resource usage. Many users thought they had achieved something by filling the filesystem or running the machine out of other resources. The message about DoS attacks didn't seem to receive much notice.

Another interesting problem I had was trying to convince users they actually were root. I had GCC installed, and many users compiled their own versions of **ps** and other utilities in the belief that they weren't really root and that it was all a trick with modified utilities. One user even had assembly code to call the `getuid()` system call to determine whether I had modified `libc6`. Although that user really was root, it would be a fun exercise to modify `libc6` to pretend that someone had logged in as root when they really had not. I encourage readers to try this out for themselves.

Of course, not all users were so difficult to convince. I gave the password to a "black hat" person who was seeking machines for the installation of a rootkit. He tried installing his rootkit but found that all the relevant directories (`/bin`, `/sbin` and `/etc`) and the files they contained were not writable. He asked for assistance in installing, but I was unable to help him.

### **How to Run Your Own Security Test/Challenge Machine**

If you want to run your own security test machine, the first thing to do is find a suitable place to house the server. This is easier said than done, as such a machine attracts a lot of network scans and penetration attempts on the



network. The terms of service of most ISPs prohibit such things, and you risk being disconnected.

Once you have arranged the hosting, you have to devise a good method for taking the machine off-line in a hurry in case something goes wrong. Direct physical access to the power switch is convenient for this purpose. A method of controlling the power or hardware reset over the Internet also is a good option. Failing that, you should install the test machine on its own switch port, for a managed switch, or, as a cheaper option, on a crossover cable to a Linux machine running Netfilter. This way, you can disable network access to the entire machine quickly.

The next thing to do is choose suitable hardware. For example, an iPAQ is not ideal for this type of machine, as it is possible to render it unusable through software. Commodity desktop PC hardware is a good option, though. The worst-case scenario would be replacing the motherboard, which is cheap and easy. Another good option is to obtain free hardware, so you won't have lost any money if the system dies. Some nice hardware seems to end up in the rubbish nowadays.

Once you have the machine basically configured, you have to set up suitable packet filters to prevent it from being used for attacks on other machines. How strict these filters are depends on the agreement you have with your ISP. If you have no specific agreement regarding such access—if you are using a home broadband connection—then the filters should be very strict. If your contract specifically permits running servers, you can allow greater access, even the ability to host Web pages. Granting more network access allows more interesting tests to be performed. A frequent complaint from users was the test machine didn't have enough access granted to allow a wide enough range of testing. For the next play machine, I plan to provide full network access, so users can receive mail on the machine, host Web pages and do most other things that they request.

The firewall should be set up both on the test machine and on any other machines on the same physical network. The test machine can be configured reasonably with Netfilter to discard or reject the packets silently, without logging them, although you may want to log them for interest. The router should be configured to log all such packets when it drops them, so you know if someone gets past the filter on the test machine or cracks its security in any other way. If your ISP knows of your plans for a security test server, then a minimal firewall should work. This will prevent SMTP connections, spoofed source IP addresses on packets being sent and connections to Web-mail services such as Hotmail, which includes blocking access to Web proxies and

configuring any local Web proxies to not allow the test machine access to Web mail.

Having any machines other than the test machine and the router on the same LAN may be a bad idea, as it may allow the test machines to be used to attack the other machines. Having several security test machines on the same physical network may be fun, though, as it would allow them to be used to attack each other. If you have only one test machine, connecting it to the router by a crossover Ethernet cable or a null-modem cable running PPP probably is a good option.

Once the machine is connected and all firewalls are arranged, the difficult work begins. You have to determine how to limit the access that is granted and audit it as appropriate. For SE Linux, all that needs to be done is to change the root entry in the users' file to `user root roles { user_r };`. Another option is to remove the root entry from the database entirely, as the default identity of user\_u is permitted only the role user\_r and gives the extra restriction of preventing password changes. To change the password of a nonprivileged account, the identity must match the user name.

The policy database then has to be recompiled and loaded into the kernel to apply the change. After that, the root user has no significant access to the system, so make sure you grant some other account administrative privileges first.

The next time I set up a test machine, I plan to get someone with legal experience to review the usage conditions to make sure they state what is permitted in a clear and legally binding language. I will place the password on a Web page that has the usage conditions and change it regularly, so people can't get in without reading the conditions. Too many people were obviously not reading the conditions, particularly regarding local DoS attacks through fork bombs and using all available disk space.

If you run an SE Linux play machine, please let me know so I can publicize it on my Web page.

I have been using the IRC channel #selinux on [irc.debian.org](http://irc.debian.org) for supporting the play machine and for answering general SE Linux questions. I encourage anyone else who is running such a security test machine, whether SE Linux or some other system, to join that channel to discuss it.

## **Acknowledgements**

The Cobalt division of Sun generously supported my work through the gift of a RaQ server. All SE Linux play machines after LinuxTag were run on Cobalt hardware.

Russell Coker has been using Linux for ten years. Through his work in UNIX administration for ISPs, he has become convinced that security is the area of UNIX that needs the most improvement.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## Eleven SSH Tricks

**Daniel R. Allen**

Issue #112, August 2003

Run remote GUI applications and tunnel any Net connection securely using a free utility that's probably already installed on your system.

SSH is the descendant of rsh and rlogin, which are non-encrypted programs for remote shell logins. Rsh and rlogin, like telnet, have a long lineage but now are outdated and insecure. However, these programs evolved a surprising number of nifty features over two decades of UNIX development, and the best of them made their way into SSH. Following are the 11 tricks I have found useful for squeezing the most power out of SSH.

### Installation and Versions

OpenSSH is the most common free version of SSH and is available for virtually all UNIX-like operating systems. It is included by default with Debian, SuSE, Red Hat, Mandrake, Slackware, Caldera and Gentoo Linux, as well as OpenBSD, Cygwin for Windows and Mac OS X. This article is based on OpenSSH, so if you are using some other version, check your documentation before trying these tricks.

### X11 Forwarding

You can encrypt X sessions over SSH. Not only is the traffic encrypted, but the DISPLAY environment variable on the remote system is set properly. So, if you are running X on your local computer, your remote X applications magically appear on your local screen.

Turn on X11 forwarding with `ssh -X host`. You should use X11 forwarding only for remote computers where you trust the administrators. Otherwise, you open yourself up to X11-based attacks.

A nifty trick using X11 forwarding displays images within an xterm window. Run the web browser w3m with the in-line image extension on the remote machine; see the Debian package w3m-img or the RPM w3m-imgdisplay. It uses X11 forwarding to open a borderless window on top of your xterm. If you read your e-mail remotely using SSH and a text-based client, it then is possible to bring up in-line images over the same xterm window.

### Config File

SSH looks for the user config file in `~/.ssh/config`. A sample might look like:

```
ForwardX11 yes
Protocol 2,1
```

Using `ForwardX11 yes` is the same as specifying `-X` on the command line. The `Protocol` line tells SSH to try SSH2 first and then fall back to SSH1. If you want to use only SSH2, delete the `, 1`.

The config file can include sections that take effect only for certain remote hosts by using the `Host` option. Another useful config file option is `User`, which specifies the remote user name. If you often log in to a machine with `ssh -l remoteuser remotehost` or `ssh remoteuser@remotehost`, you can shorten this by placing the following lines in your config file:

```
Host remotehost
ForwardX11 yes
User remoteuser

Host *
ForwardX11 no
```

Now, you can type `ssh remotehost` to log on as user `remoteuser` with the `ForwardX11` option turned on. Otherwise, `ForwardX11` is turned off, as recommended above. The asterisk matches all hosts, including hosts already matched in a `Host` section, but only the first matching option is used. Put specific `Host` sections before generic sections in your config file.

A system-wide SSH config file, `/etc/ssh/ssh_config`, also is available. SSH obtains configuration data in the following order: command-line options, user's configuration file and system-wide configuration file. All of the options can be explored by browsing `man ssh_config`.

## Speeding Things Up: Compression and Ciphers

SSH can use gzip compression on any connection. The default compression level is equivalent to approximately 4× compression for text. Compression is a great idea if you are forwarding X sessions on a dial-up or slow network. Turn on compression with `ssh -C` or put `Compression yes` in your config file.

Another speed tweak involves changing your encryption cipher. The default cipher on many older systems is triple DES (3DES), which is slower than Blowfish and AES. New versions of OpenSSH default to Blowfish. You can change the cipher to blowfish with `ssh -c blowfish`.

Cipher changes to your config file depend on whether you are connecting with SSH1 or SSH2. For SSH1, use `Cipher blowfish`; for SSH2, use:

```
Ciphers blowfish-cbc,aes128-cbc,3des-cbc,cast128-cbc,arcfour,aes192-cb
```

## Port Forwarding

Ports are the numbers representing different services on a server; such as port 80 for HTTP and port 110 for POP3. You can find the list of standard port numbers and their services in `/etc/services`. SSH can translate transparently all traffic from an arbitrary port on your computer to a remote server running SSH. The traffic then can be forwarded by SSH to an arbitrary port on another server. Why would you want to do this? Two reasons: encryption and tunneled connections.

## Encryption

Many applications use protocols where passwords and data are sent as clear text. These protocols include POP3, IMAP, SMTP and NNTP. SSH can encrypt these connections transparently. Say your e-mail program normally connects to the POP3 port (110) on `mail.example.net`. Also, say you can't SSH directly to `mail.example.net`, but you have a shell login at `shell.example.net`. You can instruct SSH to encrypt traffic from port 9110 (chosen arbitrarily) on your local computer and send it to port 110 on `mail.example.net`, using the SSH server at `shell.example.net`:

```
ssh -L 9110:mail.example.net:110 shell.example.net
```

That is, send local port 9110 to `mail.example.net` port 110, over an SSH connection to `shell.example.net`.

Then, simply tell your e-mail program to connect to port 9110 on localhost. From there, data is encrypted, transmitted to `shell.example.net` over the SSH

port, then decrypted and passed to mail.example.net over port 110. As a neat side effect, as far as the POP3 daemon on mail.example.net knows, it is accepting traffic from shell.example.net.

### **Tunneled Connections**

SSH can act as a bridge through a firewall whether the firewall is protecting your computer, a remote server or both. All you need is an SSH server exposed to the other side of the firewall. For example, many DSL and cable-modem companies forbid sending e-mail from your own machine over port 25 (SMTP).

Our next example is sending mail to your company's SMTP server through your cable-modem connection. In this example, we use a shell account on the SMTP server, which is named mail.example.net. The SSH command is:

```
ssh -L 9025:mail.example.net:25 mail.example.net
```

Then, tell your mail transport agent to connect to port 9025 on localhost to send mail. This exercise should look quite similar to the last example; we are tunneling from local port 9025 to mail.example.net port 25 over mail.example.net. As far as the firewall sees, it is passing normal SSH data on the normal SSH port, 22, between you and mail.example.net.

A final example is connecting through an ISP firewall to a mail or news server inside a restricted network. What would this look like? In fact, it would be the same as the first example; mail.example.net can be walled away inside the network, inaccessible to the outside world. All you need is an SSH connection to a server that can see it, such as shell.example.net. Is that neat or what?

### **Limitations/Refinements to Port Forwarding**

If a port is reassigned on a computer (the local port in the examples above), every user of that computer sees the reassigned port. If the local system has multiple users, tunnel only from unused, high-numbered ports to avoid confusion. If you want to forward a privileged local port (lower than 1024), you need to do so as root. Forwarding a lower-numbered port might be useful if a program won't let you change its port, such as standard BSD FTP.

By default, a tunneled local port is accessible only to local users and not by remote connection. However, any user can make the tunneled port available remotely by using the -g option. Again, you can do this to privileged ports only if you are root.

Any user who can log in with SSH can expose any port inside a private network to the outside world using port forwarding. As an administrator, if you allow

incoming SSH connections, you're really allowing incoming connections of any kind. You can configure the OpenSSH daemon to refuse port forwarding with `AllowTcpForwarding no`, but a determined user can forward anyway.

A config file option is available to forward ports; it is called `LocalForward`. The first port-forwarding example given above could be written as:

```
Host forwardpop
  Hostname shell.example.com
  LocalForward 9110 mail.example.com:110
```

This way, if you type `ssh forwardpop` you receive the same result as in the first example. This example uses the `Host` command described above and the `HostName` command, which specifies a real hostname with which to connect.

Finally, a command similar to `LocalForward`, called `RemoteForward`, forwards a port from the computer to which you are connected, to your computer. Please read the `ssh_config` man pages to find out how.

### Piping Binary Data to a Remote Shell

Piping works transparently through SSH to remote shells. Consider:

```
cat myfile | ssh user@desktop lpr

tar -cf - source_dir | \
ssh user@desktop 'cat > dest.tar'
```

The first example pipes `myfile` to `lpr` running on the machine named `desktop`. The second example creates a tar file and writes it to the terminal (because the tar file name is specified as dash), which is then piped to the machine named `desktop` and redirected to a file.

### Running Remote Shell Commands

With SSH, you don't need to open an interactive shell if you simply want some output from a remote command, such as:

```
ssh user@host w
```

This command runs the command `w` on `host` as `user` and displays the result. It can be used to automate commands, such as:



```
perl -e 'foreach $i (1 .. 12) \
{print `ssh server$i "w"}`}'
```

Notice the back-ticks around the SSH command. This uses Perl to call SSH 12 times, each time running the command **w** on a different remote host, server1 through server12. In addition, you need to enter your password each time SSH makes a connection. However, read on for a way to eliminate the password requirement without sacrificing security.

### Authentication

How does SSH authenticate that you should be allowed to connect? Here are some options:

- By hostnames only: uses .rhosts file; insecure; disabled by default.
- By hostnames and host-key checking.
- The S/Key one-time password system.
- Kerberos: private-key encryption with time-expired “tickets”.
- Smart card.
- Password prompt.
- Public key.

The most common authentication method is by password prompt, which is how most SSH installations are run out of the box.

However, public key encryption is worth investigating; it is considerably more secure than passwords, and by using it you can do away with all or most of your password typing.

Briefly, public key encryption relies on two keys: a public key to encrypt, which you don't keep secret, and a private key to decrypt, which is kept private on your local computer. The general idea is to run `ssh-keygen` to generate your keys. Press Return when it asks you for a passphrase. Then copy your public key to the remote computer's `authorized_keys` file.

The details depend on whether the computer to which you are connecting uses SSH1 or SSH2. For SSH1 type `ssh-keygen -t rsa1`, and copy `~/.ssh/identity.pub` to the end of the file `~/.ssh/authorized_keys` on the remote computer. For SSH2, type `ssh-keygen -t rsa`, and copy `~/.ssh/id_rsa.pub` to the end of the file `~/.ssh/authorized_keys` on the remote computer. This file might be called `~/.ssh/authorized_keys2`, depending on your OpenSSH version.

If the first one doesn't work, try the second. The payoff is you can log in without typing a password.

You can use a passphrase that keeps the private key secret on your local computer. The passphrase encrypts the private key using 3DES. At no time is your passphrase or any secret information sent over the network. You still have to enter the passphrase when connecting to a remote computer.

### **Authentication Agent**

You might wonder: if we want to use a passphrase, are we stuck back where we started, typing in a passphrase every time we log in? No. Instead, you can use a passphrase, but type it only once instead of every time you use the private key. To set up this passphrase, execute `ssh-agent` when you first start your session. Then execute `ssh-add`, which prompts for your passphrase and stores it in memory, not on disk. From then on, all connections authenticating with your private key use the version in memory, and you won't be asked for a password.

Your distribution may be set up to start `ssh-agent` when you start X. To see if it's already running, enter `ssh-add -L`. If the agent is not running already, you need to start it, which you can do by adding it to your `.bash_login`, logging out and logging back in again.

### **Authentication Agent Forwarding**

If you connect from one server to another using public key authentication, you don't need to run an authentication agent on both. SSH automatically can pass any authentication requests coming from other servers, back to the agent running on your own computer. This way, it never passes your secret key to the remote computer; rather, it performs authentication on your computer and sends the results back to the remote computer.

To set up authentication agent forwarding, simply run `ssh -A` or add the following line to your config file:

```
ForwardAgent yes
```

You should use authentication agent forwarding only if you trust the administrators of the remote computer; you risk them using your keys as if they were you. Otherwise, it is quite secure.

### Traveling with SSH Java Applet

Many people carry a floppy with PuTTY or another Windows SSH program, in case they need to use an unsecured computer while traveling. This method works if you have the ability to run programs from the floppy drive. You also can download PuTTY from the web site and run it.

Another alternative is putting an SSH Java applet on a web page that you can use from a browser. An excellent Java SSH client is Mindterm, which is free for noncommercial use. You can find it at [www.appgate.com/mindterm](http://www.appgate.com/mindterm).

### Conclusion

An SSH configuration can go wrong in a few places if you are using these various tricks. You can catch many problems by using `ssh -v` and watching the output. Of course, none of these tricks is essential to using SSH. Eventually, though, you may encounter situations where you're glad you know them. So give a few of them a try.

Daniel R. Allen ([da@coder.com](mailto:da@coder.com)) discovered UNIX courtesy of a 1,200-baud modem, a free local dial-up and a guest account at MIT, back when those things existed. He has been an enthusiastic Linux user since 1995. He is president of Prescient Code Solutions, a software consulting company in Kitchener, Ontario and Ithaca, New York.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## VTun

**Ryan Breen**

Issue #112, August 2003

Link your home and office securely with a virtual private network.

Back in the halcyon days of the dot-com era, I was the first employee at a P2P software startup. Because we were building an intranet and development environment from the ground up, we were free to use Linux everywhere. As everyone now knows, the world changed, and the dot-coms went the way of the dodo. So too did the independence of my little startup, which was acquired by a larger company with an established Windows developer base. Although the new firm was liberal enough to allow me to continue developing on and for Linux, I largely was left to fend for myself on system administration tasks.

The only area where I encountered significant difficulty was with VPN setup. At my old job, every developer had an inbound SSH port mapped to her development workstation. Not only did the new office lock down external access to SSH ports, the sanctioned VPN solution was not Linux-friendly. Technical inertia guaranteed that a cross-platform solution such as FreeS/WAN would not be available in the foreseeable future. Fortunately, VTun, the VPN solution I used at my old job, is flexible enough to handle even this inhospitable environment.

### **How Does It Work?**

VTun works by seamlessly integrating IP-tunneling technology with existing packet routing programs. True to the UNIX spirit of modularity, VTun is directly responsible only for tunneling packets between two systems, leveraging established network management tools to provide a cohesive VPN solution.

For the sake of analogy, imagine your home and office networks are a set of discrete and isolated railroad networks. Each machine represents a station. The Linux kernel controls the track switches, determining how trains from one

station reach the next. These facilities can be manipulated through the **route** program, allowing the end user to add or remove destinations.

The Linux kernel also provides facilities for rerouting trains. For example, let's add the Internet, a vast railroad system, to our train station analogy. The home and office networks are merely tiny spurs in this system. Typically, only one station, a firewall or gateway router, has direct access to the larger Internet railroad. If another station on the home tracks wants to dispatch trains to the Internet, these trains first are rerouted through the gateway station. This rerouting process, technically known as IP masquerading or network address translation (NAT), is managed through the **iptables** program. **iptables** is the user-space half of the Netfilter firewall code in the 2.4 Linux kernel.

So where does VTun fit into this analogy? Recall that the home and office networks are isolated train systems. A train from home generally is not allowed to cross over to the work tracks due to restrictions at the office firewall station. VTun gives us a facility to lay a virtual track between two stations—for example, your home and office desktops—on the separate networks. Once this track has been laid, the stations are configured using iptables and routed such that trains originating from home can access the work system as freely as if they originated from the office desktop.

### Conventions and Caveats

Now that we've looked at the components of a VTun VPN, we are ready to examine a complete implementation. The most obvious scenario connects a single remote workstation (your home desktop) to the office LAN by way of your work desktop. To keep this example simple, assume you can establish an SSH connection to your office desktop from home but that the machine is otherwise inaccessible from the Internet. Assume the home network is configured on the 192.168.1.0/24 subnet, and the office network has subnets 192.168.5.0/24 and 192.168.100.0/24.

VTun is a client/server system. The server machine listens for connections from VTun clients on a specified port. The client initiates the creation of the tunnel by connecting to the server port. For this example, the home desktop is the client, and the office desktop is the server.

Before we begin installation, we should take a moment to discuss security. Creation of a VPN can mean the office network now is only as secure as the home network. As such, it is imperative that your home machines are protected by a firewall that is up to date on all security patches and routinely audited for intrusion. Most importantly, never create a VPN without the consent of your office system administrators.

## Installation

With the caveats and disclaimers out of the way, we proceed to the fun stuff. VTun needs to be installed on both the client and server, so the procedure outlined below should be completed on each system. This procedure has been tested on recent versions of Red Hat Linux. If you discover this installation path fails for your distribution, please send me an e-mail at [ryan@ryanbreen.com](mailto:ryan@ryanbreen.com). I will use these responses to track a distribution-specific errata file at [www.ryanbreen.com/vtun](http://www.ryanbreen.com/vtun).

Some distributions already have packages for VTun, so you might be able to save a step by using your package manager to install VTun from your distribution's update site.

As with most VPN solutions, VTun requires the support of kernel-level facilities, in this case provided by the TUN point-to-point network driver. The TUN module is included in the stock kernel distribution, so you most likely do not need to recompile your kernel. As a test, attempt to load the driver by running `insmod tun` as root. If the module is not found, download the latest version (currently tun-1.1) from [vtun.sourceforge.net/tun/index.html](http://vtun.sourceforge.net/tun/index.html). Install it with:

```
tar xzf tun-1.1.tar.gz
cd tun-1.1
su -c 'make install'
```

If you would like the TUN module to be loaded automatically whenever a process attempts to access the virtual tunnel device, add the following line to `/etc/modules.conf`:

```
alias char-major-10-200 tun
```

Next, configure and install the user-space vtund program. You can find the latest VTun package at [vtun.sourceforge.net/download.html](http://vtun.sourceforge.net/download.html). For the sake of generality, here we install from source, but if your distribution supports RPMs or debs, feel free to grab one of the precompiled packages. The latest source tarball at press time is vtun-2.5.tar.gz. Compilation follows the standard procedure:

```
tar xzf vtun-2.5.tar.gz
cd vtun-2.5
./configure
make
su -c 'make install'
```

Depending on your distribution, configuration might fail with an error that LZO is not installed. LZO is a compression library used by VTun. It can be

downloaded from [www.oberhumer.com/opensource/lzo/download](http://www.oberhumer.com/opensource/lzo/download). Build and install LZO, then retry VTun installation.

Upon installation, VTun places its configuration file at `/usr/local/etc/vtund.conf`. This can be extremely confusing as the client and server need separate entries in the tunnel specification section. To avoid confusion, I suggest moving `vtund.conf` to `vtund-client.conf` and `vtund-server.conf` as appropriate. Then, manually specify a path to the relevant configuration file on startup. This recommendation is used throughout the following configuration discussion.

### VTun Configuration Files

The VTun configuration file format is relatively straightforward (see Listings 1 and 2). The file is organized into three discrete units. First is a set of global options defining basic parameters, such as server port number and paths to helper programs. Second is a set of default session options that define the networking properties of the tunnel. These properties can be overridden as needed in the configuration of a specific tunnel.

#### Listing 1. Simple `vtund-client.conf`

```
options {
    port 5000;

    # Path to various programs
    ifconfig /sbin/ifconfig;
}

# Default session options
default {
    compress no;      # Compression is off
    encrypt no;      # ssh does the encryption
    speed 0;         # By default maximum speed
    keepalive yes;
    stat yes;
}

my_tunnel {
    pass XXXXXXXX;   # Password
    type tun;        # IP tunnel
    proto tcp;       # TCP protocol

    up {
        # 10.3.0.1 = fake tunnel interface (home-end)
        # 10.3.0.2 = fake tunnel interface (work-end)
        # 192.168.5.0/24 = actual work network 1
        # 192.168.100.0/24 = actual work network 2
        ifconfig
```

```

        "%% 10.3.0.1 pointopoint 10.3.0.2 mtu 1450";
    };
    down{
        ifconfig "%% down";
    };
}

```

## Listing 2. Simple vtund-server.conf

```

options {
    port 5000;

    # Path to various programs
    ifconfig /sbin/ifconfig;
}

# Default session options
default {
    compress no;    # Compression is off
    encrypt no;    # ssh does the encryption
    speed 0;      # By default maximum speed
    keepalive yes;
    stat yes;
}

my_tunnel {
    pass XXXXXXXX;    # Password
    type tun;        # IP tunnel
    proto tcp;       # TCP protocol

    up {
        # 10.3.0.1 = fake tunnel interface (home-end)
        # 10.3.0.2 = fake tunnel interface (work-end)
        # 192.168.1.0/24 = actual home network
        ifconfig
            "%% 10.3.0.2 pointopoint 10.3.0.1 mtu 1450";
    };
    down{
        ifconfig "%% down";
    };
}

```

One tunnel configuration parameter that deserves special attention is `keepalive`. Office system administrators often set a low idle time on active connections through their firewalls. If your tunnel is inactive for longer than this deadline, even a few minutes, your connection times out. Enabling `keepalive` instructs VTun to circumvent this behavior by periodically sending packets from client to server, convincing the firewall the connection is in active use.



The final unit of options defines the configuration for a specific tunnel. The configuration file can contain any number of settings of this type, allowing clients and servers to be involved in multiple VPNs. Each tunnel configuration group begins with a name. I have chosen the name `my_tunnel`, but the name is an arbitrary designation. Each tunnel can configure a password, though this option generally is ignored when the tunnel is created over SSH. The `up` and `down` blocks describe a set of commands run when the tunnel is created and destroyed, respectively.

The simple configuration files in Listings 1 and 2 instruct VTun to create the tunnel interface on each system once the connection is established. The configuration files use the pattern `%%` to represent the tunnel interface, so multiple tunnels can be created in any order. The actual name of the tunnel interface begins with the prefix `tun` followed by a digit. The first tunnel created is `tun0`.

### Creating a VTun VPN

Let's put this basic understanding of VTun configuration into practice, using Listings 1 and 2 to create a simple tunnel. You can find the Listings at [ftp.linuxjournal.com/pub/lj/listings/issue112/6675.tgz](http://ftp.linuxjournal.com/pub/lj/listings/issue112/6675.tgz) if you would prefer not to enter them by hand. Save `vtund-server.conf` to `/usr/local/etc/` on the office machine, and save `vtund-client.conf` to `/usr/local/etc/` on the home machine. With the config files in place, initiate the VTun processes on each machine. As root, start the server on the office desktop:

```
vtund -f /usr/local/etc/vtund-server.conf -s
```

The `-s` option tells `vtund` to run as the server, listening for connections on port 5000.

To access the server, you must be able to reach port 5000 on the office machine. Recall that, for the sake of this example, the office is accessible only by SSH, so you must use OpenSSH's port-forwarding mechanism to tunnel port 5000 from the office machine. From home, run:

```
ssh mydesktop.work.com -L 5000:localhost:5000
```

The `-L` option tells OpenSSH to forward port 5000 on the home machine to port 5000 on the office desktop. Connections to port 5000 on the home machine then are tunneled transparently through SSH to port 5000 on the office machine. This configuration has the additional benefit of encrypting all VPN traffic.

With the running server on the office machine now accessible from the home desktop, all that remains is to start the client. As root on the home desktop, run:

```
vtund -f /usr/local/etc/vtund-client.conf
↳my_tunnel localhost
```

The `my_tunnel` parameter tells the client and server what tunnel is being created. Both systems query their respective configuration files and run the commands within the `up` block of the `my_tunnel` stanza. The final parameter, `localhost`, specifies the hostname of the VTun server. In this case, the VTun server is `localhost` because you forwarded port 5000 from the home machine to the office desktop.

If the tunnel was created successfully, running `ifconfig` on each machine should list a `tun0` interface. The home machine then has an IP address of 10.3.0.1 on `tun0`, and the office machine has IP 10.3.0.2. Drawing on the train station analogy, the track between the office desktop and home desktop has been laid, and you can now route trains between the machines over this track. To demonstrate this, create an SSH connection from your home desktop to 10.3.0.2.

### **Making It Real**

You now have a working tunnel from home to the office. Next, you need to configure route and iptables so packets from home are masqueraded through the work desktop to the rest of the office LAN. Fortunately, this is as simple as adding a few lines to the configuration files on the client and server and restarting the vtund processes. VTun executes the appropriate route and iptables commands when the connection is established.

Returning to the train station analogy, you need to instruct the home desktop station that any trains destined for the office network should be routed through the newly created VTun track. You can accomplish this manually by running:

```
route add -net 192.168.5.0 netmask
↳255.255.255.0 gw 10.3.0.2
route add -net 192.168.100.0 netmask
↳255.255.255.0 gw 10.3.0.
```

Alternatively, you can add the commands as shown in Listing 3 to `vtund-client.conf`. These commands instruct iptables to forward all packets from the `tun` interface and to masquerade these packets as coming from the office

desktop. Alternatively, we can add the commands shown in Listing 4 to vtund-server.conf and restart the server.

### Listing 3. Complete vtund-client.conf

```
options {
    port 5000;

    # Path to various programs
    ifconfig    /sbin/ifconfig;
    firewall    /sbin/iptables;
    route       /sbin/route;
}

# Default session options
default {
    compress no;    # Compression is off
    encrypt no;    # ssh does the encryption
    speed 0;       # By default maximum speed
    keepalive yes;
    stat yes;
}

my_tunnel {
    pass XXXXXXXX;    # Password
    type tun;         # IP tunnel
    proto tcp;        # TCP protocol

    up {
        # 10.3.0.1 = fake tunnel interface (home-end)
        # 10.3.0.2 = fake tunnel interface (work-end)
        # 192.168.5.0/24 = actual work network 1
        # 192.168.100.0/24 = actual work network 2
        ifconfig
            "%% 10.3.0.1 pointopoint 10.3.0.2 mtu 1450";
        route "add -net 192.168.5.0 netmask
            ↪255.255.255.0 gw 10.3.0.2";
        route "add -net 192.168.100.0 netmask
            ↪255.255.255.0 gw 10.3.0.2";
    };
    down{
        ifconfig "%% down";
        route "del -net 192.168.5.0 netmask
            ↪255.255.255.0 gw 10.3.0.2";
        route "del -net 192.168.100.0 netmask
            ↪255.255.255.0 gw 10.3.0.2";
    };
}
}
```

### Listing 4. Complete vtund-server.conf

```

options {
    port 5000;

    # Path to various programs
    ifconfig    /sbin/ifconfig;
    firewall    /sbin/iptables;
    route       /sbin/route;
}

# Default session options
default {
    compress no;    # Compression is off
    encrypt no;    # ssh does the encryption
    speed 0;       # By default maximum speed
    keepalive yes;
    stat yes;
}

my_tunnel {
    pass XXXXXXXX;    # Password
    type tun;         # IP tunnel
    proto tcp;        # TCP protocol

    up {
        # 10.3.0.1 = fake tunnel interface (home-end)
        # 10.3.0.2 = fake tunnel interface (work-end)
        # 192.168.1.0/24 = actual home network
        ifconfig
            "%% 10.3.0.2 pointopoint 10.3.0.1 mtu 1450";
        route "add -net 192.168.1.0 netmask
            ↵255.255.255.0 gw 10.3.0.1";
        firewall "-t nat-A POSTROUTING -o %%
            ↵-j MASQUERADE";
        firewall "-AFORWARD -i %% -j ACCEPT";
    };
    down{
        ifconfig "%% down";
        route "del -net 192.168.1.0 netmask
            ↵255.255.255.0 gw 10.3.0.1";
    };
}

```

Once route and iptables are configured, you should have access to your entire corporate intranet from your home desktop. Browse around your internal Web servers, connect to the source code server and try exporting a graphical widget such as an xterm. Performance should be more than adequate for all these tasks, and the SSH tunnel ensures that all traffic is encrypted from prying eyes.

Now that you have a working tunnel, you may want to configure the server to start automatically. This process is distribution-specific. The VTun tarball

includes a set of init scripts for different distributions, so you should consult the Readme to determine which will work best for you.

### **Advanced Configuration**

Astute readers may have noticed that only the home desktop has access to the office intranet. Trains originating from other stations within the home network currently are not rerouted through the home desktop station. I feel that this configuration is at least marginally more secure, as it reduces the exposure of the office network to compromises at home. If you desire connectivity from other machines on the home network, simply add the appropriate iptables rules to the up directive in vtund-client.conf. I leave that as an exercise for the interested reader.

The above configuration works perfectly if you can connect by SSH to any machine on your office network. Unfortunately, many offices do not provide any open incoming ports. This was precisely the situation I found upon arrival at my new job, but the flexibility of VTun allowed me to overcome even this obstacle. The solution is to reverse the configuration, using the office desktop as the VTun client and originating the SSH tunnel from within the office.

To make this solution work, we must be able to access our home machine from within the office. However, most broadband connections have dynamic IP addresses. We can sidestep this issue by using a DNS service tailored for dynamic IPs, such as that provided by DynDNS.org.

The greatest downside to this approach is its relative fragility. In a secure setup, the client does not start automatically because the SSH connection requires authentication, leaving you out in the cold if the office machine goes down due to a power outage. If you are less worried about security, you can automate login using SSH public key authentication without a passphrase or expect scripting. I do not encourage either method.

If your office machine is on a UPS, you rarely should encounter this problem. In the six months that I have used this setup, only one power outage lasted long enough to kill the client side of my VPN. This setup also is robust on the home network side. You can take your machine off-line for days, and the VPN re-initializes as soon as you start the vtun server, thanks to the intelligent keepalive and retry facilities in the client.

### **Conclusion**

Hopefully, you now have an appreciation for the versatility and power of a VTun VPN and possess the technical know-how to set one up for yourself. Unfortunately, a comprehensive discussion of VTun's feature set is well beyond

the scope of this article. Beyond the basic setups described above, VTun allows Ethernet, PPP or SLIP tunneling of protocols other than IP. VTun also provides native support for encryption, compression and bandwidth shaping, so it is adaptable to every imaginable connection scenario. VTun belongs in the toolkit of every network user and deserves mention alongside breakthrough applications such as OpenSSH, rsync and screen.

### **Acknowledgements**

Many thanks to Jennifer Edwards and James Manning for reviewing this article.

### **Resources**

Ryan's VTun Info Page: [www.ryanbreen.com/vtun](http://www.ryanbreen.com/vtun)

Universal TUN/TAP Driver Home: [vtun.sourceforge.net/tun](http://vtun.sourceforge.net/tun)

VTun Home: [vtun.sourceforge.net](http://vtun.sourceforge.net)

Ryan Breen ([ryan@ryanbreen.com](mailto:ryan@ryanbreen.com)) is a 2000 graduate of Duke University with degrees in Computer Science and Economics. He is currently living in Boston with his girlfriend of three years and dog of two and a half years. At work, he builds high-throughput browser simulations, is a devoted KDE user and occasional KDE developer.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## 2003 Editors' Choice Awards

**LJ Staff**

Issue #112, August 2003

When time for the Editors' Choice awards comes around, we ask for suggestions from all our columnists and contributing editors. This year, we have some familiar names among the winners and a few newcomers to Linux.

### **Server Appliance (hardware)**

**Sputnik AP 120** [www.sputnik.com/products/ap120.html](http://www.sputnik.com/products/ap120.html)

Want to offer wireless Net access to your customers or neighbors? You can build a custom box with NoCatAuth, sign with one of the expensive startups or take your chances and go wide open—until now. Sputnik performed a marvel of Linux miniaturization to get a usable portal onto a relatively inexpensive access point.



Sputnik's access point includes centrally managed authentication.

### **Security Tool (hardware or software)**

#### **Netfilter/iptables [kernel.org](http://kernel.org)**

Security Editor Mick Bauer writes, "The packet-filtering code in the Linux 2.4 kernel, although not new to 2003, really came into its own, bringing Linux firewalling up to the level of many commercial products. It's flexible and intelligent, with impressive connection-state-tracking capabilities." Mick also points out that you can use the ubiquitous Netfilter right on the bastion host to add an extra layer of firewall protection, even if you use another firewall at the network edge.

### **Server**

#### **Newisys 2100 [www.newisys.com/products/2100.html](http://www.newisys.com/products/2100.html)**

Michael Baxter called this dual Opteron, 1U server "superbly engineered", as the 64-bit Opteron breaks through the memory limitations of x86 while keeping backward compatibility. Newisys-based servers are a hot item in today's competitive Linux server market, with many Linux server vendors whose integration and service we like offering them. And, they start GNU Emacs almost as quickly as most people's computers start Vim.

### **Workstation**

#### **Dell Precision 650n [www.dell.com/precision](http://www.dell.com/precision)**

Our reviewer Glenn Stone calls this dual-Xeon desktop system "serious hardware for serious work", and admires the performance of its 320MB/s SCSI RAID subsystem and Dell's on-site service plan.

### **Web Browser or Client**

#### **Mozilla 1.4 [www.mozilla.org](http://www.mozilla.org)**

Tabbed browsing, pop-up blocking, bookmark keywords—when we're stuck with other browsers they simply seem archaic, restrictive and awful. Konqueror is good too, but this time Mozilla barely beat it out as the browser for people who want to make the Web work their way.

### **Graphics Software**

#### **Jahshaka [www.jahshaka.com](http://www.jahshaka.com)**

No, this isn't the special prize for "not being The GIMP because they always win". Greg Kroah-Hartman brings this bleeding-edge, alpha-stage video editing



application to our attention, and we can't wait to do a full tutorial. More than only video editing, Jahshaka offers animation, effects, a character generator and file-sharing capabilities.

### **Communication Tool**

**Gaim** [gaim.sourceforge.net](http://gaim.sourceforge.net)

Marcel Gagné writes, "I used to scoff at instant messaging, but in the last few months, I have discovered it to be an amazingly useful communications tool. Sometimes, nothing beats a real-time, ongoing conversation when trying to resolve technical issues." Gaim is, well, instant messaging for people whose friends don't agree on instant-messaging systems. As we go to press, Gaim supports AOL Instant Messenger, ICQ, MSN Messenger, Yahoo, IRC, Jabber, Gadu-Gadu and Zephyr.

### **Desktop Software**

**OpenOffice.org** [www.openoffice.org](http://www.openoffice.org)

Marcel also recommends OpenOffice.org, citing "nearly perfect support of Microsoft Office documents". Everyone seems to like the word processor, but other useful parts of the suite include a drawing program and a presentation package.

### **Development Tool**

**Perl 5.8.0** [www.perl.org](http://www.perl.org)

Reuven Lerner covers a different web development tool every month but keeps coming back to good old Perl. In the new version, he writes, "Most important is its support for threading and Unicode, both of which will help to propel Perl forward for years to come." We like browsing the Comprehensive Perl Archive Network (CPAN) for modules that empower a short script to do exactly what you want.

### **Database**

**PostgreSQL** [www.postgresql.org](http://www.postgresql.org)

Reuven also is a fan of PostgreSQL, a database that makes its fair share of appearances in our pages. "The PostgreSQL team has demonstrated that it is possible to produce a database with the price and ease of administration of MySQL, but with the feature set of Oracle. The only real competition is Firebird and SAP DB, both of which might make serious inroads in 2003–2004", he writes.

The new 7.3 version offers table functions (functions that return multiple rows), schemas and prepared queries, as well as Unicode by default, improved logging and nonlocking vacuuming of tables.

Marcel points out that “What we need, however, is a dead-simple database application for new users who might simply want to create a Christmas card list or whatever. Just because we can create a full-blown relational SQL database and have it for free, doesn't mean it is always what the user needs.”

### **Management or Administration Software**

**Webmin** [www.webmin.com](http://www.webmin.com)

Marcel calls Webmin “a wonderful, low-resource tool that crosses distros and operating systems” and praises the integrated SSH application. Webmin standard modules can be used to administer any server software you can imagine, from Apache to voice mail to WU-FTP. Third-party modules make it easy for ISPs to delegate mail and virtual web host administration to customers.

### **Mobile Device**

**Lindows Mobile PC** [www.lindows.com/lindows\\_feature\\_preinstall.php](http://www.lindows.com/lindows_feature_preinstall.php)

Finally, a notebook computer with Linux pre-installed. Doc Searls would be happy with this box, based on a VIA processor and equipped with 256MB of RAM and all the expected extras, if he could get it away from his six-year-old kid.

### **Game**

**Frozen Bubble** [www.frozen-bubble.org](http://www.frozen-bubble.org)

Move along, people, nothing to see here, back to work. This could be the next *Tetris*. You won't often hear this from us, but whoever ported this thing to Microsoft Windows and Mac OS, thank you, because now it won't be only Linux users' productivity down /dev/crapper. New in version 1.0: 100—yes, 100—levels and a level editor.



Just this one screenshot, then *Frozen Bubble* is off our computers for good.

### Book

*Understanding the Linux Kernel, 2nd Edition* by Daniel P. Bovet and Marco Cesati [www.oreilly.com/catalog/linuxkernel2](http://www.oreilly.com/catalog/linuxkernel2)

This is a good one to keep handy if you get stuck on something in Kernel Korner, or if you dig building custom kernels and want to know how things work. We won't say which of the contributing editors voted for their own books.

### Web Site

**Linux Weekly News** [lwn.net](http://lwn.net)

Everybody's doing metanews—selecting the best articles from every news site—but Linux Weekly News does a good job of filtering the important Linux news from the drivel. And, they offer original content on diverse topics such as security alerts and kernel hacking—something to keep you happy between issues of *Linux Journal*.

### Product of the Year

**SGI Altix 3000** [www.sgi.com/servers/altix](http://www.sgi.com/servers/altix)

Remember the world's biggest single-system-image Linux box from the February 2003 cover? We recently heard from SGI that Professor Stephen Hawking's research group at Cambridge University just bought one.

Steve Neuner of SGI wrote in February 2003, "When you look at the list of kernel additions included...the list is actually surprisingly small, which speaks highly of Linux's robust original design. What is even more impressive is that many of these and other changes are already in the 2.5 development kernel." Congratulations to SGI for taking Linux up several steps on the food chain, and congratulations to all our winners.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## Driving Me Nuts

*Device Classes*

**Greg Kroah-Hartman**

Issue #112, August 2003

More necessary instructions for making your new device driver play nice in the 2.6 kernel.

In the last Driving Me Nuts column [see *LJ*, June 2003], we introduced the kernel driver model framework with an explanation of how the generic bus and driver and device code works. The i2c core was used as an example to show how these different subsystems work. This month, we cover how the driver class code works, again using the i2c code to provide some working examples.

As discussed in the last column, device classes do not meet the general object-oriented definition of a class; rather they are something that provides a single type of function to the user. For example, kernel classes are used for tty devices, block devices, network devices, SCSI hosts and, in the near future, filesystems.

In the 2.5.69 kernel, the driver class support was rewritten radically. In previous kernel versions, class support was tied tightly to the driver and device support. A class would be bound to the device at the same time it was registered to a driver. This did work for a number of devices and classes, but some real-world devices did not fit very well into this model. Now, class support is tied only loosely to devices and drivers; in fact, a device or driver is not even needed to use the class code now, as the tty class code shows. The class code is now split into three different types of structures: classes, class devices and class interfaces.

### Classes

Classes in the kernel are defined with a simple struct class structure. Yes, class is not a reserved word in C. (Everyone who wants to build a kernel with a C++

compiler, go flame the author of the new class code.) To create a class structure, only the name variable in the struct class structure needs to be defined for it to be a valid class. This can be done with the following code:

```
static struct class i2c_adapter_class = {  
    .name = "i2c_adapter"  
};
```

After the class structure is defined, it can be registered with the driver core by calling the `class_register` function:

```
if (class_register(&i2c_adapter_class) != 0)  
    printk(KERN_ERR "i2c adapter class failed "  
           "to register properly\n");
```

After the `class_register` function returns without reporting an error, the `/sys/class/i2c_adapter` directory has been created successfully. Later, when the class needs to be unloaded, the `class_unregister` function should be called:

```
class_unregister(&i2c_adapter_class);
```

### Class Devices

Classes are used to manage a set of different class devices. A class device is defined in the kernel with the `struct class_device` structure. This structure contains a lot of variables the driver core uses, and it can be ignored by the driver writer. Only the following variables should be set:

- `class`: should point to the struct class that is going to manage the class device.
- `dev`: should be set to the address of the struct device associated with the class device, if any. A single struct device can be pointed to by multiple class device structures. This is the main difference between the previous kernel class support and the current implementation. This variable does not have to be set for the kernel to work properly. If it is set, a device symbolic link is created in the `sysfs` entry for the class device that points to the struct device. See below for an example.
- `class_id`: an array of characters used to describe the class device. It must be unique among all class device structures assigned to a single class structure.
- `class_data`: used to store a pointer to any private data the class driver wants to associate with the class device. This variable should not be

accessed directly, but the `class_set_devdata` and `class_get_devdata` functions should be used to set and retrieve the value of this variable.

To register a properly set up `struct class_device` structure, the `class_device_register` function should be called. An example of how to initialize a `struct class_device` and register it with the driver core can be seen in the following code from the `drivers/i2c/i2c-core.c` file:

```
/* Add this adapter to the i2c_adapter class */
memset(&adap->class_dev, 0x00,
       sizeof(struct class_device));
adap->class_dev.dev = &adap->dev;
adap->class_dev.class = &i2c_adapter_class;
strncpy(adap->class_dev.class_id,
        adap->dev.bus_id, BUS_ID_SIZE);
class_device_register(&adap->class_dev);
```

First, the `struct class_device` variable (embedded in the `struct i2c_adapter` variable) is initialized to zero. All driver model structures need to have all variables set to zero before they are registered, in order for the driver core to use them properly.

Then the `dev` variable is set to point to the `i2c_adapter`'s `struct device` variable; in this case, the same structure, `struct i2c_adapter`, contains both a `struct device` and a `struct class_device`. The `class` variable is set to the address of the `i2c_adapter_class` variable, and then the `class_id` variable is set to the same value as the device's `bus_id`. Because the `i2c_adapter` device's `bus_id` is unique, it also ensures that the `i2c_adapter class_device`'s `class_id` is unique. Finally, the class device structure is registered with the kernel driver core by a call to the `class_device_register` function.

With the above code and two `i2c` adapters loaded on a test machine, the `/sys/class/i2c_adapter` tree might look like the following:

```
$ tree /sys/class/i2c_adapter/
/sys/class/i2c_adapter/
|-- i2c-0
|   |-- device -> ../../../../devices/pci0/00:07.3/i2c-0
|   `-- driver -> ../../../../bus/i2c/drivers/i2c_adapter
`-- i2c-2
    |-- device -> ../../../../devices/legacy/i2c-2
    `-- driver -> ../../../../bus/i2c/drivers/i2c_adapter
```

As you can see by the above tree output, a device and driver symbolic link are created automatically by the driver core to point to the proper place within the

sysfs tree that represents those values. If the dev pointer was not set to point to a struct device, those symbolic links would not have been created. If you look in the /sys/class/tty directory, the majority of those class device entries do not have a corresponding struct device, so those symbolic links are not present.

## Class Interfaces

Class interfaces simply are a way for your code to be notified whenever a struct class\_device is registered or unregistered from a specific class. A class interface is defined with the struct class\_interface structure. This structure is simple and looks like:

```
struct class_interface {
    struct list_head node;
    struct class *class;
    int (*add) (struct class_device *);
    void (*remove) (struct class_device *);
};
```

The class variable needs to be set to the class about which we want to be notified. The add and remove variables should be set to a function that is called when any devices are added or removed, respectively, from that class. It is not necessary to set both the add and remove variables if you do not want to be notified about one of those events.

To register a class interface with the kernel, the class\_interface\_register function is called. Likewise, to unregister a class interface, the class\_interface\_unregister function is called. An example of code that uses class interfaces is the CPU frequency core; this code can be found at kernel/cpufreq.c in the kernel source tree.

## Creating Files

As described above, the i2c-adapter class is useful for easily determining all of the different i2c adapters present in the system and their specific location in the driver tree. But i2c adapters are not directly addressable by a user. To talk to an i2c adapter, an i2c chip driver needs to be loaded, or the i2c-dev driver can be used. The i2c-dev driver provides a character driver interface to all i2c adapters present in the system. Because it is useful to determine exactly which i2c-dev devices are attached to which i2c adapters, a i2c-dev class was created:

```
static struct class i2c_dev_class = {
    .name = "i2c-dev"
};
```



Then, when every i2c adapter is found by the i2c-dev driver, a new i2c class device is added to the driver core. This addition is done in the `i2c_add_class_device` function:

```
static void
i2c_add_class_device(char *name, int minor,
                    struct i2c_adapter *adap)
{
    struct i2c_dev *i2c_dev;
    int retval;

    i2c_dev = kmalloc(sizeof(*i2c_dev), GFP_KERNEL);
    if (!i2c_dev)
        return;
    memset(i2c_dev, 0x00, sizeof(*i2c_dev));

    if (adap->dev.parent == &legacy_bus)
        i2c_dev->class_dev.dev = &adap->dev;
    else
        i2c_dev->class_dev.dev = adap->dev.parent;
    i2c_dev->class_dev.class = &i2c_dev_class;
    snprintf(i2c_dev->class_dev.class_id,
             BUS_ID_SIZE, "%s", name);
    retval =
        class_device_register(&i2c_dev->class_dev);
    if (retval)
        goto error;
    class_device_create_file (&i2c_dev->class_dev,
                             &class_device_attr_dev);

    i2c_dev->minor = minor;
    spin_lock(&i2c_dev_list_lock);
    list_add(&i2c_dev->node, &i2c_dev_list);
    spin_unlock(&i2c_dev_list_lock);
    return;
error:
    kfree(i2c_dev);
}
```

This function looks almost like the `i2c_adapter` class registration code, with two exceptions. First, the `class_dev.dev` field is set to be either the adapter's parent device or the adapter's device. This is done because some i2c adapters do not have a real parent in the global kernel device tree, as they live on a bus that has not been converted to the kernel driver model (like ISA) or they do not really live on a bus at all (like some i2c embedded controllers). When an i2c adapter does not have a place in the kernel device tree, it is assigned to the legacy bus. The legacy bus, located at `/sys/devices/legacy`, is used for these kinds of devices.

The second thing that is different with this class device is the line:

```
class_device_create_file (&i2c_dev->class_dev, &class_device_attr_dev)
```

The `class_device_create_file` function is used to create a file in the class device's directory. The filename and attributes are defined with the `CLASS_DEVICE_ATTR` macro as:

```
static ssize_t
show_dev(struct class_device *class_dev, char *buf)
{
    struct i2c_dev *i2c_dev = to_i2c_dev(class_dev);
    return sprintf(buf, "%04x\n",
                  MKDEV(I2C_MAJOR, i2c_dev->minor));
}
static
CLASS_DEVICE_ATTR(dev, S_IRUGO, show_dev, NULL);
```

The `CLASS_DEVICE_ATTR` macro is itself defined as:

```
#define CLASS_DEVICE_ATTR(_name, _mode, _show, _store) \
struct class_device_attribute \
class_device_attr_##_name = { \
    .attr = { .name = __stringify(_name), \
              .mode = _mode }, \
    .show = _show, \
    .store = _store, \
};
```

The arguments within the `CLASS_DEVICE_ATTR` macro are:

- `_name`: both the name of the file to be created in sysfs and part of the variable name that describes this whole attribute.
- `_mode`: the file access mode with which the file is created. Use the standard access macros to specify the proper value.
- `_show`: points to a function that is called when the file is read from. This function must have the following return value and parameters. This variable does not have to be set if the file is not to be read from.

```
ssize_t
show (struct class_device *class_dev, char *buf);
```

- `_store`: points to a function that is called when the file is written to. This function must have the following return value and parameters. This variable does not have to be set if the file is not to be written to.

```
ssize_t
store (struct device *dev, const char *buf,
       size_t count);
```

Almost all driver model structures have an `ATTR()` macro that declares a file within the sysfs tree.

In this example, a file named `dev` is created when the `class_device_create_file` function is called. This file is created to be read-only by any user. If the file is

read from, the `show_dev` function is called by the driver core. The `show_dev` function fills in the buffer passed to it with the information it wants to give the user. In this case, the major and minor number for this specific device are passed to the user. All class devices using a major and minor number should have a dev file within their sysfs class device directory.

The `class_device_remove_file` function can be used to remove any files created by the `class_device_create_file` function. But it is not necessary to remove manually any file created if the device is about to be removed. When devices are removed from sysfs, all files created in their directories are removed automatically by the sysfs core. So, when the `i2c-dev` class device is removed from the system, all that is needed is the following:

```
static void
i2c_remove_class_device(int minor)
{
    struct i2c_dev *i2c_dev = NULL;
    struct list_head *tmp;
    int found = 0;

    spin_lock(&i2c_dev_list_lock);
    list_for_each (tmp, &i2c_dev_list) {
        i2c_dev = list_entry(tmp, struct i2c_dev,
                             node);
        if (i2c_dev->minor == minor) {
            found = 1;
            break;
        }
    }
    if (found) {
        list_del(&i2c_dev->node);
        spin_unlock(&i2c_dev_list_lock);
        class_device_unregister(&i2c_dev->class_dev);
        kfree(i2c_dev);
    } else {
        spin_unlock(&i2c_dev_list_lock);
    }
}
```

### What It All Looks Like

With the `i2c-dev` driver and two `i2c` adapter drivers (the `i2c-piix4` and `i2c-isa` drivers) loaded, the `/sys/class/i2c-dev` directory might look like the following:

```
$ tree /sys/class/i2c-dev/
/sys/class/i2c-dev/
|-- i2c-0
|   |-- dev
|   |-- device -> ../../../../devices/pci0/00:07.3
|   `-- driver -> ../../../../bus/pci/drivers/piix4-smbus
`-- i2c-2
    |-- dev
    |-- device -> ../../../../devices/legacy/i2c-2
    `-- driver -> ../../../../bus/i2c/drivers/i2c_adapter
```

The dev file in the `/sys/class/i2c-dev/i2c-2/` directory would contain the following string:

```
$ cat /sys/class/i2c-dev/i2c-2/dev
5902
```

which corresponds to major number 86 and minor number 2, the character major and minor numbers for this specific device.

Also, the `/sys/bus/i2c/` directory with a few i2c client drivers loaded looks like:

```
$ tree /sys/bus/i2c/
/sys/bus/i2c/
|-- devices
|   |-- 0-0050 -> ../../../../devices/pci0/00:07.3/i2c-0/0-0050
|   |-- 0-0051 -> ../../../../devices/pci0/00:07.3/i2c-0/0-0051
|   |-- 0-0052 -> ../../../../devices/pci0/00:07.3/i2c-0/0-0052
|   |-- 0-0053 -> ../../../../devices/pci0/00:07.3/i2c-0/0-0053
|   `-- 2-0290 -> ../../../../devices/legacy/i2c-2/2-0290
`-- drivers
    |-- dev driver
    |-- eeprom
    |   |-- 0-0050 -> ../../../../devices/pci0/00:07.3/i2c-0/0-0050
    |   |-- 0-0051 -> ../../../../devices/pci0/00:07.3/i2c-0/0-0051
    |   |-- 0-0052 -> ../../../../devices/pci0/00:07.3/i2c-0/0-0052
    |   `-- 0-0053 -> ../../../../devices/pci0/00:07.3/i2c-0/0-0053
    |-- i2c_adapter
    `-- w83781d
        `-- 2-0290 -> ../../../../devices/legacy/i2c-2/2-0290
```

And, the actual `/sys/devices/` directories for the i2c adapters look like:

```
$ tree /sys/devices/pci0/00:07.3
/sys/devices/pci0/00:07.3
|-- class
|-- device
|-- i2c-0
|   |-- 0-0050
|   |   |-- eeprom_00
|   |   |-- name
|   |   `-- power
|   |-- 0-0051
|   |   |-- eeprom_00
|   |   |-- name
```

```
| | | `-- power
| | | |-- 0-0052
| | | | |-- eeprom_00
| | | | |-- name
| | | | `-- power
| | | |-- 0-0053
| | | | |-- eeprom_00
| | | | |-- name
| | | | `-- power
| | | |-- name
| | | `-- power
|-- irq
|-- name
|-- power
|-- resource
|-- subsystem_device
|-- subsystem_vendor
`-- vendor
```

and:

```
$ tree /sys/devices/legacy/i2c-2/
/sys/devices/legacy/i2c-2/
|-- 2-0290
|   |-- alarms
|   |-- beep_enable
|   |-- beep_mask
|   |-- fan_div1
|   |-- fan_div2
|   |-- fan_div3
|   |-- fan_input1
|   |-- fan_input2
|   |-- fan_input3
|   |-- fan_min1
|   |-- fan_min2
|   |-- fan_min3
|   |-- in_input0
|   |-- in_input1
|   |-- in_input2
|   |-- in_input3
|   |-- in_input4
|   |-- in_input5
|   |-- in_input6
|   |-- in_input7
|   |-- in_input8
|   |-- in_max0
|   |-- in_max1
|   |-- in_max2
|   |-- in_max3
|   |-- in_max4
```

```

|-- in_max5
|-- in_max6
|-- in_max7
|-- in_max8
|-- in_min0
|-- in_min1
|-- in_min2
|-- in_min3
|-- in_min4
|-- in_min5
|-- in_min6
|-- in_min7
|-- in_min8
|-- name
|-- power
|-- pwm1
|-- pwm2
|-- pwm_enable2
|-- sensor1
|-- sensor2
|-- sensor3
|-- temp_input1
|-- temp_input2
|-- temp_input3
|-- temp_max1
|-- temp_max2
|-- temp_max3
|-- temp_min1
|-- temp_min2
|-- temp_min3
|-- vid
`-- vrm
|-- name
`-- power

```

I think the best description of the kernel driver model's use of interconnected structure pointers and representation to the user was issued by Jonathan Corbet: “web woven by a spider on drugs” ([lwn.net/Articles/31185/](http://lwn.net/Articles/31185/)). Hopefully, these two articles have helped you unravel the loony web, showing the true interconnectedness of all devices within the kernel.

### **Acknowledgements**

I would like to thank Pat Mochel for creating such a powerful and complete framework in which all kernel drivers and devices easily can be shown to the user. Also, a big thanks to all of the kernel driver subsystem maintainers who have gladly converted their subsystems over to this model; without their help, the driver core code would have been little more than a nice academic exercise.

Greg Kroah-Hartman is currently the Linux USB and PCI Hot Plug kernel maintainer. He works for IBM, doing various Linux kernel-related things and can be reached at [greg@kroah.com](mailto:greg@kroah.com).

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Advanced search

## Kernel Korner

*NSA Security Enhanced Linux*

**Faye Coker**

Issue #112, August 2003

With fine-grained mandatory access controls in your system, you can even put limits on what root can do.

NSA Security Enhanced Linux has its roots in the distributed trusted operating system (DTOS) and Flask (Flux advanced security kernel) architecture. The DTOS Project was a collaborative effort between the US National Security Agency (NSA) and Secure Computing Corporation (SCC) in the early and mid-1990s. The goal was to provide stronger operating system security mechanisms than those provided by standard security methods. The Flask architecture was the result of a joint effort between the NSA, SCC and the University of Utah's Flux Project, which was "enhanced to provide better support for dynamic security policies" ([www.cs.utah.edu/flux/flask](http://www.cs.utah.edu/flux/flask), "Flask: Flux Advanced Security Kernel" by Stephen Smalley, NAI Labs, December 26, 2000).

SE Linux implements mandatory access control, or MAC, while regular UNIX systems employ discretionary access control, or DAC. With DAC, users can control what access is applied to objects they own at their discretion. On a UNIX system, for example, they can use the `chmod` command to change permissions on directories they own. With MAC, access control is decided by a more authoritative user who configures security policies that determine what access rights an object possesses. If a policy is preventing Bob from accessing Alice's home directory, and Alice runs `chmod 777` on her home directory, Bob still would not be able to access it.

When utilizing MAC, processes are run with a minimum amount of privilege, and a compromised process cannot grant other processes inappropriate access to its own resources. This reduces the amount of damage that might occur if a `dæmon` was compromised. Security decisions are based on a number of factors, such as the role of the user, what type of program is being run, how



trusted that program is and the secrecy level or integrity of the data being accessed.

### **What Is SE Linux?**

SE Linux is an implementation of flexible, fine-grained mandatory access controls in the Linux kernel that is now implemented using the LSM framework (see "Using the Kernel Security Module Interface" by Greg Kroah-Hartman, *LJ*, November 2002). In its current implementation, the LSM interface supports only restrictive access controls. Therefore, if the standard UNIX permissions deny an operation, SE Linux cannot permit it. SE Linux generally is used to apply additional restrictions to a system that employs UNIX permissions, and it is quite capable of enforcing all necessary access controls on its own. However, it is strongly recommended that a combination of UNIX permissions and SE Linux be used for "defense in depth" on production servers. SE Linux is comprised of a kernel patch and patches to utility programs such as login and cron.

The NSA is responsible for official releases. A number of other people outside the NSA also contribute code to the project. Packages are maintained constantly for the Debian stable and unstable releases. As SE Linux is licensed under the GPL, anyone can contribute and make her own modifications. SE Linux can be used on 2.4.19 kernels and above, and at the time of this writing, May 2003, it is being redeveloped for the 2.5 kernel.

### **Why Are Modified Utilities Required?**

As previously mentioned, SE Linux is comprised of a kernel patch and modified utility programs. The modified utilities ensure that all files on the system possess the correct security context. Modified versions of utilities, such as login, cron and logrotate, and programs, such as ps and ls, are available. With login, for example, it is crucial to have the correct security contexts when a user logs in to the system. If not, he might not be able to log in at all.

Installing the login package is covered in the Getting Started with SE Linux HOWTO (see Resources) and is beyond the scope of this article. Forgetting to install the login package during the SE Linux installation, however, results in not having the right type assigned to the terminal device from which you are logging in after a reboot, which renders you unable to log in. An unmodified login program also runs a shell in a security context that is denied access to files in the user's home directory. The patches for login and cron, for example, tell the kernel which security context to use. The actual enforcement of these measures is done by the kernel. Labeling is imperative, hence the need for some modified programs. It is possible to create your own security policies that

provide basic levels of protection without having to install modified packages, but the default configuration provides finer-grained security.

### Frequently Used Terms

When reading documents on SE Linux or mailing list posts, the following terms always are used. It is important to familiarize yourself with them as much as possible before you attempt to install SE Linux. Doing so makes things much easier later on.

*Domain:* a domain details what processes can and cannot do or, rather, what actions a process can perform on various types. If you are in the `user_t` domain (the standard unprivileged user domain) and you run the command `ps aux`, you see only the processes running in the `user_t` domain. Some examples of domains are `sysadm_t`, the system administration domain, and `init_t`, the domain in which `init` runs. The domain in which the `passwd` program is run by an unprivileged user is `passwd_t`.

*Role:* a role determines what domains can be used. The domains that a user role can access are predefined in the policy database. If a role is not authorized to enter a domain (in the config files), it is denied. Some examples of roles are the general unprivileged user role (`user_r`) and the system administrator role (`sysadm_r`).

Consider the following example: in order for a user from the `user_t` domain to execute the `passwd` command, `role user_r types passwd_t ;` is specified in the relevant config file. In addition, other domain transitioning rules must be set that are not covered here. This added code states that a user in the user role (`user_r`) is allowed to enter the `passwd_t` domain, so he can run the `passwd` command. Another consideration is whether the old domain is allowed to transition to the new domain.

Now that we have defined domains and roles, we can look at comparisons between SE Linux and the standard UNIX uid (user ID). If root owns a program with UNIX permissions 4777 (making the program setuid root), any user on the system can execute that program, resulting in a security issue. With SE Linux, however, if a process triggers a domain transition to a privileged domain and the role of the process is not authorized to enter a particular domain, the program cannot be run. Every process on an SE Linux system runs in a domain that determines what access rights a process possesses.

*Identity:* an identity under SE Linux is not the same as the traditional UNIX uid with which most readers might be familiar. Identities under SE Linux form part of a security context that controls what you can and cannot do. An SE Linux

identity and a regular UNIX login name may have the same textual representation (and in most cases they do for ease of use), but it is important to understand that they are two different beings. The default is to have them be the same, if the SE Linux identity in question exists. Therefore, if I log in as user faye on an SE Linux system, and if the policy database has the identity faye compiled into it, then my processes would be assigned to the faye identity.

To illustrate that standard UNIX user IDs are different from SE Linux identities, consider the `su` command. Running `su` does not change the user identity under SE Linux, but it does change the `uid` in the same way it would on a non-SE Linux system. If user faye, on the SE Linux system, types `su -` to switch to root and then runs the `id` command, which returns her security context and other information, she would see that her identity still is faye and not root, but her `uid` has changed. To illustrate this further, if an unprivileged user with the login name faye runs the `id` command, she would see the security context of:

```
uid=1000(faye) gid=1000(faye)
groups=1000(faye) context=faye:user_r:user_t
sid=45
```

The identity portion of the security context in this case is faye. Notice the `uid` of 1000. Next, say faye does an `SU` to root and runs the `id` command again; she now would see:

```
uid=0(root) gid=0(root) groups=0(root)
context=faye:user_r:user_t sid=453
```

The identity has not changed to root as might be expected, but the `uid` has changed to 0. However, if user faye has been granted access to enter the superuser role, or `sysadm_r`, she can do so by either logging in at the console and specifying she wants the superuser role or entering the `newrole -r` command covered later. If she then runs the `id` command again, she now sees `context=3Dfaye:sysadm_r:sysadm_t`.

So again, the identity remains the same, but the role and domain (second and third fields, respectively) have changed. Maintaining the identity in this manner is useful where user accountability is required. It also is crucial to system security because the user identity determines what roles and domains can be used. With regular UNIX, if you have a `setuid` or `setgid` program that is not world-executable, whether it is executed is determined not by the permissions of the user you are logged in as, but by the user you last did an `su` to reach. This restriction does not exist under SE Linux, as your identity is tracked throughout all operations. If your domain has not been granted the access to execute that `setuid/setgid` program, you cannot run it even if you did an `su` to root. The domains you are permitted to enter are determined by your role, and

the roles you are permitted to enter are determined by your identity. Thus, identity indirectly controls the list of domains you may enter.

*Type:* each object has a type assigned to it, and that type determines what can access the object. Objects here are files, directories, sockets and other processes. A type is similar in concept to a domain, the difference being that a domain applies only to processes. More specifically, a domain is a type that can be applied to a process.

*Transition:* a transition refers to the change in security context for a requested operation. Transitions fall into one of two categories. The first category is a transition of process domains. When you execute a program of a given type, a transition may be made from the current domain of the process to a new domain. To illustrate this, I'll use the `newrole` command. The `newrole` command is used to change your role, say, from `user_r` to `sysadm_r`, assuming you have been granted access to `sysadm_r`. If you start off as `user_r`, the general unprivileged user role, and run `newrole -r sysadm_r` to change to `sysadm_r`, the system administrator role, a transition is made from your `user_t` domain to `newrole_t` (the domain in which the `newrole` process runs) and from there on to the `sysadm_t` domain.

The second transition category is the transition of file type when you create a file under a particular directory. If a user creates a file in his own home directory, that file is labeled as `user_home_t`. But if that same user creates a file in `/tmp`, that file is labeled `user_tmp_t`. `user_tmp_t` is derived from the type of `/tmp`, which is `tmp_t`, and the domain of the creating process, which is `user_t`. When the user creates a file under `/tmp`, a transition to the `user_tmp_t` type is made.

*Policies:* a policy determines what actions can be taken on various types by various domains. All daemons have their own policies, and the naming convention is of the form `daemon-name.te`—`postfix.te`, `apache.te` and so forth. As the system administrator of an SE Linux machine, you can edit your policy files to suit your requirements. The policy database is a compiled form of the policy source files and is loaded by the kernel at boot time.

The `spasswd` program on an SE Linux system is used to change your password. `spasswd` actually is a wrapper for the standard `passwd` command used on Linux systems; it ensures that the `passwd` program is run in the correct domain. It also ensures that your SE Linux identity matches your regular UNIX account name. Earlier I mentioned that regular UNIX user IDs are quite different from SE Linux identities, so why do they have to match when running `spasswd`? `spasswd` requires you to have the same SE Linux identity name as your UNIX account name. Recall that on an SE Linux system, your identity is the

only unique method of determining who you are. So if you're not currently the corresponding UNIX user, you cannot change the password.

If you are the system administrator user (`sysadm_r`), the `sadminpasswd` program is used to change the password of another user. `sadminpasswd` does not have the same matching user name/identity restriction that `spasswd` has, but `sadminpasswd` can be run only by `sysadm_t`.

### **Permissive and Enforcing Modes**

SE Linux can be run in one of two modes, permissive or enforcing. Permissive mode is used for debugging purposes as everything gets logged, but SE Linux is not actually enforcing your policies. You still can do things as root that you could do on a regular Linux system. It is best to run your machine in permissive mode until you are satisfied that all your policies are correct. Labels are assigned to objects on the system, but nothing is enforced.

Enforcing mode applies the policies you have configured, such as access restrictions. You should boot in to enforcing mode only when you are convinced that everything is working properly, after running in permissive mode for a while. Remember, if your kernel is compiled with no development support, you cannot specify permissive mode. If your kernel is compiled with development mode support turned on, it means that your machine boots into permissive mode, but you have to switch it to enforcing mode manually. This can be done easily by creating a startup script.

Alternatively, you can make a link between `/etc/rc.boot/avc` and `/sbin/avc_toggle`. Another option is to specify `enforcing=1` on the kernel command line. The `avc_toggle` command can be used to switch between permissive and enforcing mode, and the `avc_enforcing` command can be used to determine whether you are in enforcing mode.

### **Where to Go from Here**

Hopefully this article has you interested in trying out SE Linux. I have omitted installation instructions deliberately, as you can install with RPMs, source tarballs or Debian packages. Including even the basics of each here would fill an entire article. There's quite a lot to learn before, during and after installation, and new users often find themselves rather confused. If you read the documents referred to in the Resources section before you do anything else and become familiar with frequently used terms, you should find it a little easier. If you get stuck, fire up your favorite IRC client and go over to channel `#selinux` on [irc.debian.org](http://irc.debian.org), or subscribe to the SE Linux mailing list.

## Resources

Flask (Flux Advanced Security Kernel): [www.cs.utah.edu/flux/flask](http://www.cs.utah.edu/flux/flask)

Getting Started with SE Linux HOWTO: [sourceforge.net/docman/display\\_doc.php?docid=15285&group\\_id=21266](http://sourceforge.net/docman/display_doc.php?docid=15285&group_id=21266)

NSA Official SE Linux Site: [www.nsa.gov/selinux](http://www.nsa.gov/selinux)

NSA SE Linux FAQ: [www.nsa.gov/selinux/faq.html](http://www.nsa.gov/selinux/faq.html)

NSA SE Linux White Papers: [www.nsa.gov/selinux/docs.html](http://www.nsa.gov/selinux/docs.html)

SE Linux Mailing List: [www.nsa.gov/selinux/list.html](http://www.nsa.gov/selinux/list.html)

SE Linux Mailing List Archives: [marc.theaimsgroup.com/?l=selinux](http://marc.theaimsgroup.com/?l=selinux)

SourceForge SE Linux Project Page: [sourceforge.net/projects/selinux](http://sourceforge.net/projects/selinux)

Faye Coker currently works as a freelance systems administrator and often finds herself running the systems at ISPs and converting servers to Linux. She has worked in Europe and Australia. She also has been asked “are you lost?” far too many times at Linux conferences.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## At the Forge

### *CMF Types*

**Reuven M. Lerner**

Issue #112, August 2003

Every content management system requires extensive customization. Start with one that has the power to make your web site work the way your organization does.

Over the last few months, we have discussed content management systems (CMS) in general and Zope's content management framework (CMF) in particular. Zope's CMF is designed to give developers the tools they need to create their own content management systems. Of course, anyone who has worked with a CMS knows that even the most proprietary of the bunch requires extensive modification, reworking and customization before it can be used. Zope thus not only reduces the price of the base software, but provides a rich environment that makes it relatively easy to develop and customize the CMS.

When you create a CMF site, you (as the site manager) can add, modify and delete documents. Click on the folder contents link, click on the New... button, indicate which type of document you want to add and what ID it should have and then click on the add button. Enter the metadata (that is, title, description, subject and content type), click on the change & edit button, add some content and you're off and running.

However, although the existing content types are sufficient for simple sites, more sophisticated sites will want to create their own, custom types. CMF provides several ways to do exactly this. This month, we look at types in CMF—how we can work with them, customize their behavior, install new ones and even create new types to handle custom content.

## CMF Types

The simplest way to create a new type is to use CMF's built-in, Web-based type extension system. It allows you to create a new type that shares its methods, properties, actions, presentation templates and icons with another type by default. When you create a new type using the Web-based extension system, you can modify any of these items, except for the methods and properties. In other words, the new type you create can have a different look and feel from its parent type, but it continues to behave much as the parent did.

For example, let's go to what is known as the types tool, available by clicking on `portal_types` within the management interface for a CMF site. If you don't have a CMF site already defined in Zope, you can create one by choosing CMF Site from the Add... menu in the upper-right corner of the Web-based Zope management interface. Once you have created the site, clicking on its icon from within the management interface displays a number of different customization tools, each with an icon that looks like a wrench.

When you first enter the types tool, you see a list of the currently defined CMF types, including folders, documents, news items, links and topics. You can examine and modify the properties and actions associated with these types by clicking on the name of the type you want to change. For example, if you want to examine or change the way the File content type does things, click on File. This brings up a new set of management tabs at the top of the page, with properties (the default) and actions being the only ones not standard to other parts of Zope. Actually, properties is a standard Zope tab, but CMF types have a number of unusual property names.

In addition to the standard properties you expect to see, each type has the following properties that affect what it does:

- Icon: a string that describes which icon should be displayed for items of this type.
- Product metatype: describes the Zope product meta-name. Meta-names are used in the Add... menu in the Zope Management Interface. This also is the name used in the similar Add... menu in the CMF.
- Product name: indicates the Zope product in which the CMF type was defined. Because both the File and News item types were defined in the default CMF installation, they are listed as being in the CMFDefault product. And indeed, if you look in `/lib/python/products/CMFDefault`, which is a symbolic link to `CMF-1.3/CMFDefault` in CMF 1.3, you should see both `File.py` and `NewsItem.py`, Python modules that define the content types. To see how the initial values for properties are set, look at the



factory\_type\_information variable in any module for any defined CMF type.

- Product factory method: describes the method CMF should invoke to create a new instance of the type.
- Filter content types and Allowed content types: these work together, even though they are separate properties. Although both of these properties exist for all CMF types, they are relevant only for folder-like objects, such as Folders and Topics. The first, Filter content types, is a boolean value that indicates whether Allowed content types is active. The second, Allow content types, lets you specify which types may be contained within the current type. So if you were interested in creating a folder that would contain only News items, you could do so by clicking yes and then indicating which types may be included.

### Creating a New Type

The easiest way to create a new CMF type is to base it on an existing type with the Web-based CMF type creation tool. This method does not allow you to modify the fields or methods associated with a type, but it does let you change the permissions associated with the type's actions, whether the type can be discussed and even the way in which this data type is displayed.

For example, go to the portal\_types tool and choose Factory-based type information from the Select type to add... menu in the top-right corner. You are prompted for two pieces of information, the ID or name of the new type and the existing type on which it should be based. We are creating the ATFDocument, which means we are basing ourselves on CMF Default: Document.

Once you create the new type, it is available and visible from all of the type listings, including the types tool and the contents view in which you create a new instance of a type. Indeed, anyone with administrative privileges on the portal can now see your new ATFDocument type in the menu of options from which they can choose a new type to create.

What's the point of doing this, if ATFDocument and Document are the same? Well, they're not exactly the same; rather, they share methods and an overall class definition. Other information about this type, such as properties, permissions and skins, default to be the same as Document, but they can be made to look quite different. This means that if you want instances of Document to be displayed in black-on-white text without discussions and ATFDocument to be displayed in yellow-on-maroon text with discussions, you can do that quickly and easily with this method. And, if and when you upgrade

your copy of CMF, ATFDocument will be updated automatically, along with Document.

### Under the Hood

Of course, there will be times when you want to create a type that has fields or behavior significantly different from an existing type. Several options exist for doing this, but the most flexible (and challenging and poorly documented) method is to create a new Zope product that adheres to CMF rules. For example, all Python packages must contain an `__init__.py` file in the package's root directory. This file may be empty, or it may contain statements that are evaluated when the package is first loaded into memory. In the case of a product, `__init__.py` is where the class is first registered into Zope by use of the `initialize()` method, which takes a single argument commonly called `context`. A bare-bones Zope product thus has an `__init__.py` that looks something like the following mythical `MyProduct`:

```
import MyProduct

def initialize(context):
    context.registerClass(
        MyProduct.MyProduct,
        constructors=(MyProduct.manage_addMyProductForm,
                      MyProduct.manage_addMyProduct)
    )
```

When Zope starts up, it looks through the products and invokes the `initialize()` method with an appropriate context. Context is part of Zope's system of acquisition, in which an object's attributes are defined by its location in the hierarchy as well as by its class definition. In the above example, `MyProduct` registers itself with two constructors, the methods `manage_addMyProductForm` and `manage_addMyProduct`.

A CMF type must register itself not only with Zope but also with CMF, so it can appear in the various CMF tools. Our product's `initialize()` method thus needs to include CMF-specific registration, which means that `__init__.py` needs to import modules from CMF. Moreover, every type in CMF must register itself with one of the CMF-specific initialization routines in `Products.CMFCore.utils`. For example, `__init__.py` from `CMFDefault`, which comes with CMF, first defines the different classes it will register:

```
contentClasses = ( Document.Document
                  , File.File
                  , Image.Image
```

```

        , Link.Link
        , Favorite.Favorite
        , NewsItem.NewsItem
        , SkinnedFolder.SkinnedFolder
    )

```

It then defines the constructor for each of the classes:

```

contentConstructors = \
    ( Document.addDocument
    , File.addFile
    , Image.addImage
    , Link.addLink
    , Favorite.addFavorite
    , NewsItem.addNewsItem
    , SkinnedFolder.addSkinnedFolder
    )

```

And, of course, every type can have its own specific tool:

```

tools = ( DiscussionTool.DiscussionTool
    , MembershipTool.MembershipTool
    , RegistrationTool.RegistrationTool
    , PropertiesTool.PropertiesTool
    , URLTool.URLTool
    , MetadataTool.MetadataTool
    , SyndicationTool.SyndicationTool
    )

```

Finally, the initialize() method, abbreviated slightly here, within the package registers these classes using CMF with `utils.ToolInit()`, for tools, or `ContentInit`, for content. It then invokes `initialize(context)` on what it receives back, thus registering the new object with Zope:

```

def initialize( context ):

    utils.ToolInit('CMFDefault Tool', tools=tools,
                  product_name='CMFDefault',
                  icon='tool.gif',
                  ).initialize( context )

    utils.ContentInit( 'CMFDefault Content'
                      , content_types=contentClasses
                      , permission=AddPortalContent
                      , extra_constructors=contentConstructors
                      , fti=Portal.factory_type_information
                      ).initialize( context )

```

```
context.registerClass(Portal.CMFSite,  
                      constructors=(Portal.manage_addCMFSiteForm,  
                                   Portal.manage_addCMFSite,  
                                   ))
```

The final statement in the above version of `initialize()`, as you can see, is similar to the final statement in the version of `initialize()` from the sample `MyProduct()`, demonstrating that CMF types are Zope products, only with some extra hooks included.

### Should You Use CMF?

This article concludes our look at Zope as a platform for content management, which began with Plone and concluded with CMF and CMF types. Now that we've looked at CMF in a bit more detail, let's consider whether it is worth using for projects that require a CMS.

The good news is that CMF is a powerful and flexible system. In the hands of a skilled and knowledgeable developer, CMF makes it possible to produce a custom CMS with lower cost and greater flexibility than the proprietary systems now on the market. The fact that everything is built on top of Zope, which is designed for rapid development, makes it quick and easy to create new types, modify templates and develop functionality.

But CMF, like much open-source software, suffers from a terrible lack of up-to-date and useful documentation. I'm sure that one of the reasons for the success of the Plone CMS is the excellent documentation that comes with Plone.

So if you're going to use CMF, be ready and willing to read through a great deal of Python code, to experiment quite a bit and to ask other CMF developers for help. Given the central role that CMF already is playing in the Zope world already, I expect that the amount and quality of CMF documentation will continue to increase. But until it does, working with CMF will require patience, reading the source code and a lot of trial and error.

The current state of the CMF is such that I would be somewhat hesitant to use it for anything but the largest and most complex content management systems. That said, the flexibility and power of the CMF is designed to solve problems of precisely this magnitude. In short, as inappropriate as CMF might be for small jobs, it probably is quite appropriate for large ones. And as time goes on, I expect CMF to play an increasingly prominent role in the world of open-source content management, providing a framework for the rapid development of custom CMS software.

## Conclusion

Zope's CMF is an impressive framework for building a custom CMS. I have no doubt that CMF makes it easy to create a CMS, at a significantly lower cost and with far less effort than would be the case with a full-fledged proprietary solution. That said, CMF still is not quite ready for prime time for anyone who is not intimately familiar with it or willing to spend a great deal of time learning it. I would argue that Plone has pushed CMF into the spotlight, and the fact that Zope 3 will be largely or completely merged with CMF means there is now greater incentive at Zope Corporation to make CMF more impressive and better-documented than was previously the case.

If you have a fair amount of programming experience with Python and Zope, you almost certainly can use CMF to create your own custom types as Zope products—and with those, create impressive, interesting sites for yourself and your clients. However, until the type-creation system becomes easier to understand, CMF will not get the attention it deserves from outside the Zope community. Creating Zope products is no longer the black art that it used to be, and I expect that creating CMF types will be treated similarly in the near future.

Next month, we will shift gears dramatically, looking at another open-source CMS known as Bricolage. Bricolage, which uses Mason, mod\_perl and PostgreSQL, has gained a great deal of ground in the past year, and it is an increasingly prominent player in the open-source CMS community.

## Resources

The home page for Zope CMF is [cmf.zope.org](http://cmf.zope.org). I not only found the site difficult to navigate, but I could not easily find good, useful information on CMF.

The best introduction to CMF types I found was not actually on the CMF site but on the Plone site, at [www.plone.org](http://www.plone.org). For example, the document at [plone.org/documentation/CMFTypesBook/backtalk](http://plone.org/documentation/CMFTypesBook/backtalk) book view is the *CMF Types Book*, which is both readable and contains examples. Chapter 8 of *The Plone Book* also contains some good information about CMF types, at [plone.org/documentation/book/8](http://plone.org/documentation/book/8).

As always, ZopeLabs, at [www.zopelabs.com](http://www.zopelabs.com), has a good amount of sample code and minitutorials describing how to accomplish certain tasks in CMF.

Finally, if you are interested in creating new types for CMF, consider Archetypes, a SourceForge project designed to make it easier for people to create new CMF types. And indeed, the CMF Collective is a SourceForge project containing a number of CMF types that might interest you. Be sure to look through the CVS

repository, rather than depending on the files and types that have been made available on SourceForge.

Reuven M. Lerner ([reuven@lerner.co.il](mailto:reuven@lerner.co.il)) is a consultant specializing in open-source Web/database technologies. He and his wife, Shira, recently celebrated the birth of their second daughter, Shikma Bruria. Reuven's book *Core Perl* was published by Prentice Hall in early 2002, and a second book about open-source Web technologies will be published by Apress in 2003.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## Cooking with Linux

*Illuminating Your Network's Darkest Corners*

**Marcel Gagné**

Issue #112, August 2003

Marcel has cooked up a menu of network monitoring tools.

No, François. This kind of sniffer has nothing to do with wine. Wine is one area when the human nose performs far better than any software, no matter how clever the programmer. Honestly, *mon ami*, sampling the wine, for quality control reasons of course, is not a task I wish to automate. Other uses exist for the kinds of sniffers we are likely to encounter when cooking with Linux.

Look here, *mon ami*. Notice how much of our bandwidth is being used here and here. Are you curious as to what those connections really represent? François, why are you looking away? Ah, our guests are here. Why did you not say something?

*Bonsoir, mes amis!* Welcome once again to *Chez Marcel*, home of tantalizing Linux fare, great wines from the world over and a general *penchant* for all things open source. Please sit and make yourselves comfortable. Before you walked in, I was telling François about the many hidden bits of information flying across the average network. Speaking of hidden delights, François, please hurry to the wine cellar. Head to the west wing and bring back the 1995 Rioja Imperial Gran Reserva. This Spanish red is the perfect networking wine, *non?*

As I was telling my faithful waiter, a great deal is happening on the average network, and many people are completely oblivious to all but those connections they themselves have initiated. The simplest tool for checking out active network connections is included in every Linux distribution, **netstat**. By using the `-a` and `-p` options, you can find out about almost every open connection (or port) on your system and what programs are using them.

Notice what happens when I run the program. I'm going to use the `-n` option as well, which tells `netstat` not to worry about resolving IP addresses into symbolic addresses. This makes the program run a bit faster because no name resolution is performed. The result can be quite a long listing, so I pipe the output to **more**:

Ah François, you are back with the wine. Excellent. Please pour for our guests.

The listing I've shown you is only a partial listing, but the entire listing is incomplete itself. The reason for this is iptables masqueraded connections are not visible to `netstat`; that information is in another location, specifically `/proc/net/ip_contrack`. The PID is the process ID of the running program using the connection. Now, we could do a `cat` on `/proc/net/ip_contrack`, but the output doesn't make for eye-friendly reading. Look at the following sample (the output is a single, wrapped line):

```
tcp      6 431253 ESTABLISHED src=192.168.22.5
↳dst=192.168.22.10 sport=34212 dport=22
↳src=192.168.22.10 dst=192.168.22.5 sport=22
↳dport=34212 [ASSURED] use=1
```

Patrick Lagacé obviously found this unpleasant to read as well. His `contrack` viewer script is available at [cv.intellos.net](http://cv.intellos.net). Because it is a Perl script, simply change the permissions to make the script executable after you have downloaded it, then run the command:

```
chmod +x contrack-viewer.pl
./contrack-viewer.pl
```

By default, the output shows all connections, including the masqueraded ones. To limit the output to masqueraded connections only, use the `-m` option. The reverse effect (no masqueraded connections) can be achieved with the `-d` option. Have a look at Figure 1 for a sample of the output.



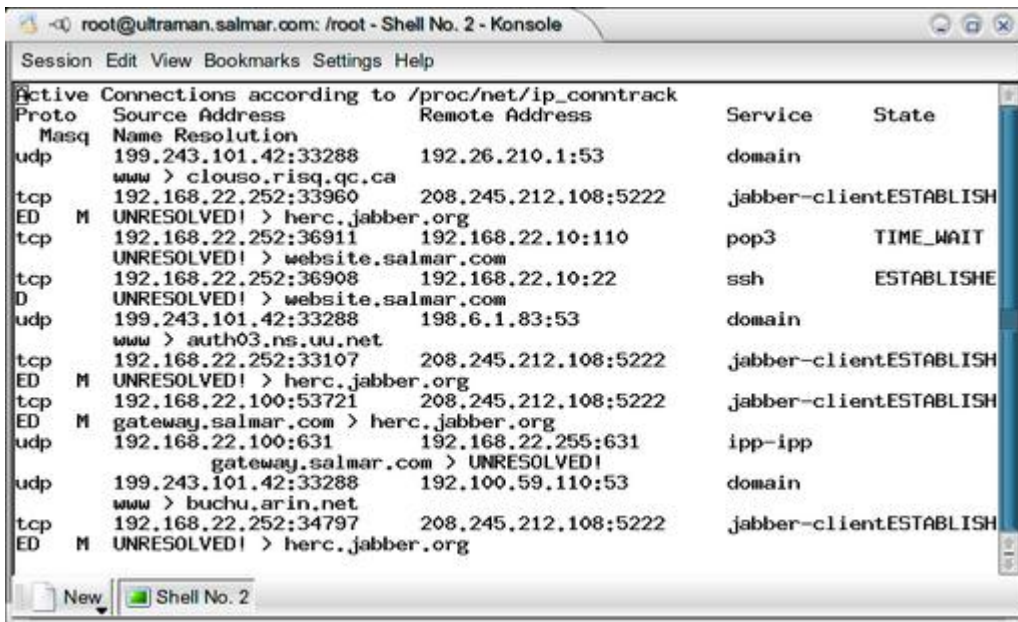


Figure 1. contrack Viewer Displays Masqueraded Connections

Alexander Neptun's **Nnetstat** is a nice graphical tool for displaying active connections, routing tables and so on. To get your copy, visit [www.aneptun.de/linux/Nnetstat](http://www.aneptun.de/linux/Nnetstat) and download the latest version. This is basically a Perl script, so no real installation has to be done other than making sure Nnetstat.pl is executable. As it turns out, Nnetstat also requires the Gtk.pm modules libraries, and while Perl should be on your system, this module likely is not. The easiest way to get it is from the Perl CPAN repository, and the command line still is your friend here:

```
perl -MCPAN -e "install Gtk"
```

If this is the first time you install Perl modules in this manner, you'll go through a little question-and-answer session. Follow through, accept the defaults and trust the system. What you need to decide is the location of the closest CPAN mirrors. Select your continent and country when asked, then select the available local mirrors. Once this is done, the Gtk installation continues on its own.

Installing the Gtk Perl modules does take some time. I probably should warn you that at some point near the end of the installation, a set of tests is performed. Don't be surprised when a graphical box pops up asking you to click Run to test all sorts of graphical magic associated with the package. When you are happy with the result, click Close to terminate the tests and complete the installation.

The screenshot shows a window titled "Nnetstat v0.1Alpha" with a menu bar (File, Options, Help) and a tabbed interface (tcp, udp, raw, unix, route). The "tcp" tab is active, displaying a table of network connections. The table has columns: Nr, Local, Port, Remote, Port, rx\_q, tx\_q, Status, User, Inode, PID, and Program. The data is as follows:

Nr	Local	Port	Remote	Port	rx_q	tx_q	Status	User	Inode	PID	Program
11	199.243.101.42	53	*	0	0	0	LISTEN	named	2468805	13815	named
12	192.168.22.10	53	*	0	0	0	LISTEN	named	2468805	13815	named
13	localhost	53	*	0	0	0	LISTEN	named	2468805	13815	named
14	*	22	*	0	0	0	LISTEN	root	1202	703	sshd
15	*	5432	*	0	0	0	LISTEN	postgres	2005641	18963	postmaster
16	*	25	*	0	0	0	LISTEN	root	2473210	19891	smtpd
17	localhost	953	*	0	0	0	LISTEN	named	2468805	13815	named
18	localhost	6010	*	0	0	0	LISTEN	root	2491670	15424	sshd
19	199.243.101.42	47451	194.201.147.71	25	0	36	ESTABLISHED	postfix	2492072	19845	smtp
20	199.243.101.42	47500	213.228.0.166	25	0	0	ESTABLISHED	postfix	2492076	19828	smtp
21	localhost	47356	localhost	6010	416	2048	ESTABLISHED	root	2491730	19729	Nnetstat.pl
22	199.243.101.42	47465	24.93.67.222	25	0	0	ESTABLISHED	postfix	2492073	19855	smtp
23	199.243.101.42	47526	130.227.187.7	25	0	0	ESTABLISHED	postfix	2492678	19774	smtp
24	199.243.101.42	47454	192.108.102.204	25	0	0	TIME_WAIT	root			
25	localhost	6010	localhost	47356	0	0	ESTABLISHED	root	2491730	15424	sshd
26	192.168.22.10	22	192.168.22.4	34402	0	2096	ESTABLISHED	root	2491670	15424	sshd
27	192.168.22.10	25	192.168.22.4	34563	0	0	TIME_WAIT	root			
28	localhost	25	localhost	47374	0	0	TIME_WAIT	root			
29	199.243.101.42	47529	212.205.126.30	25	0	0	ESTABLISHED	postfix	2492760	19790	smtp
30	199.243.101.42	34218	66.187.232.100	443	0	0	ESTABLISHED	root	3202338	3597	rhn_check
31	localhost	47493	localhost	783	0	0	TIME_WAIT	root			
32	localhost	47499	localhost	783	0	0	TIME_WAIT	root			

Figure 2. Nnetstat is a nice, graphical netstat.

For a truly terrifying (or amusing, depending on your perspective) view of exactly what is flying across your system, run **Driftnet**. The name itself should be enough to send shivers up your spine. Simply put, Driftnet listens on a selected interface for image or video traffic (MPEG only), then displays the images it finds. Whether this display is more frightening to the system administrator who finds out what users are watching or to the users themselves, depends on more factors than we adequately can cover here. That said, this collection of images is completely indiscriminate and doesn't in any way point to a specific user.

To get your copy of Driftnet, head on over to Chris Lightfoot's web site at [www.ex-parrot.com/~chris/driftnet](http://www.ex-parrot.com/~chris/driftnet) and pick up the source. Before the Pythonists among you ask, last time I checked, his web site had not yet ceased to be nor was it pining for the Fjords.

Some prerequisite libraries are required to build Driftnet, most notably libungif, libjpeg and libpcap. If you don't have them installed already, the links are in the Resources section of this article, but check your distribution CDs first. Building the package is then a simple matter of extracting the tarball and running a make in the source directory. You then can run the resulting program from the directory itself or copy it to a more useful location:

```
./driftnet -i eth0
```

Because Driftnet needs to set the interface to promiscuous mode, you need to run it as root. Look at Figure 3 for a sample of Driftnet in action.



Figure 3. Driftnet in action: all your images belong to us.

Sure, looking at pretty pictures flying across your network is fun if you don't consider the bandwidth costs, but what other interesting things are moving across those wires? There are Web requests, file downloads, e-mail messages, instant messaging sessions and more. Most network monitors, netstat included, show you active connections, but what precisely do those connections represent?

David Leonard has created an ncurses-based program called **pktstat** ([www.itee.uq.edu.au/~leonard/personal/software/#pktstat](http://www.itee.uq.edu.au/~leonard/personal/software/#pktstat)) that does a nice job of showing you what percentage of bandwidth each connection uses. It also keeps a running load average in the style of uptime but tracks network transfer rates rather than processes in a run queue. What sets the program apart is its ability to display filenames associated with the packets sailing across your Web server or files being downloaded from client PCs on your network. Building pktstat is a matter of extracting the source, switching to the directory and typing make:

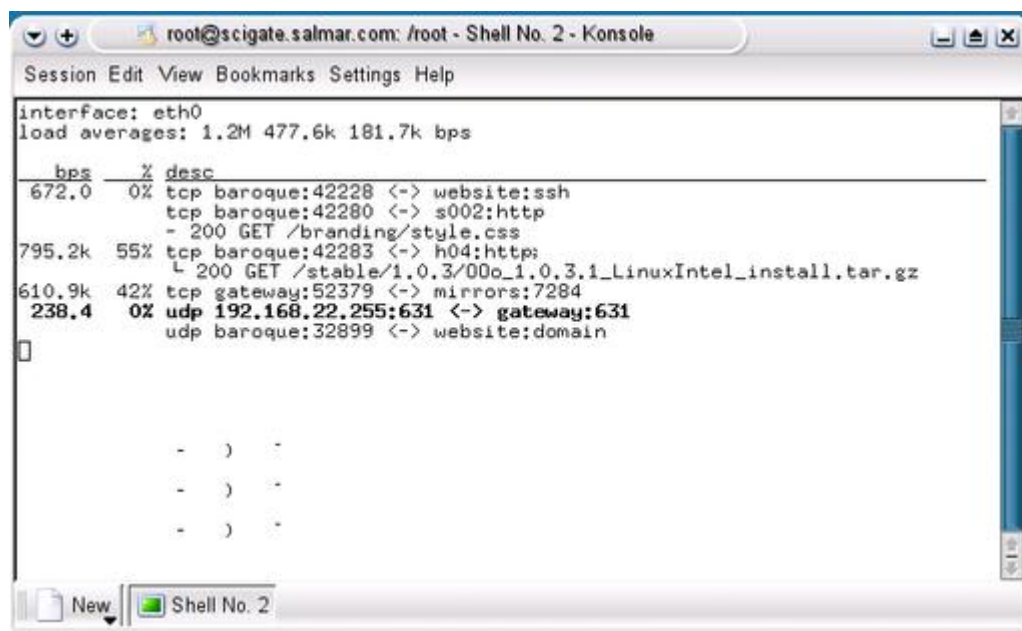
```
tar -xzvf pktstat-1.7.2q.tar.gz
cd pktstat-1.7.2q
```

```
make
su -c "make install"
```

To run the program, use the `-i` parameter to specify the interface on which you wish to listen:

```
pktstat -i eth1
```

A window appears, similar to the one in Figure 4. As you can see, I've started a download of the latest OpenOffice.org software. The actual filename is displayed below the connection information; the same is true with HTTP Web requests. You can see not only the address of the file being downloaded but the filename too, whether it be an HTML page or an image.



```
interface: eth0
load averages: 1.2M 477.6k 181.7k bps

  bps   % desc
-----
672.0   0% tcp baroque:42228 <-> website:ssh
        tcp baroque:42280 <-> s002:http
        - 200 GET /branding/style.css
795.2k  55% tcp baroque:42283 <-> h04:http
        L 200 GET /stable/1.0.3/00o_1.0.3.i_LinuxIntel_install.tar.gz
610.9k  42% tcp gateway:52379 <-> mirrors:7284
238.4   0% udp 192.168.22.255:631 <-> gateway:631
        udp baroque:32899 <-> website:domain

- ) -
- ) -
- ) -
```

Figure 4. Filenames are resolved in `pktstat`'s display.

Speaking of traffic, if you are looking to concentrate your efforts simply on what and where your network is being used, the final item on tonight's menu may be more appropriate. **IPTraf** is one of your humble chef's favorite IP-traffic monitoring tools, one that I go back to time and again. This is a ncurses-based application that displays IP traffic, byte and packet counts (including non-IP packets), UDP traffic, incoming vs. outgoing traffic and more. IPTraf is a package every person in charge of a network should have handy.

Visit Gerard Paul Java's web site at [iptraf.seul.org](http://iptraf.seul.org) to pick up your copy of IPTraf. Extract the tarred and gzipped source, then `cd` to that directory and run the **Setup** to build the package. The installation process finishes by copying the binary to `/usr/local/bin`. To run IPTraf, type `ipt raf`, press Enter and you are on your way (Figure 5 shows an active IPTraf session).



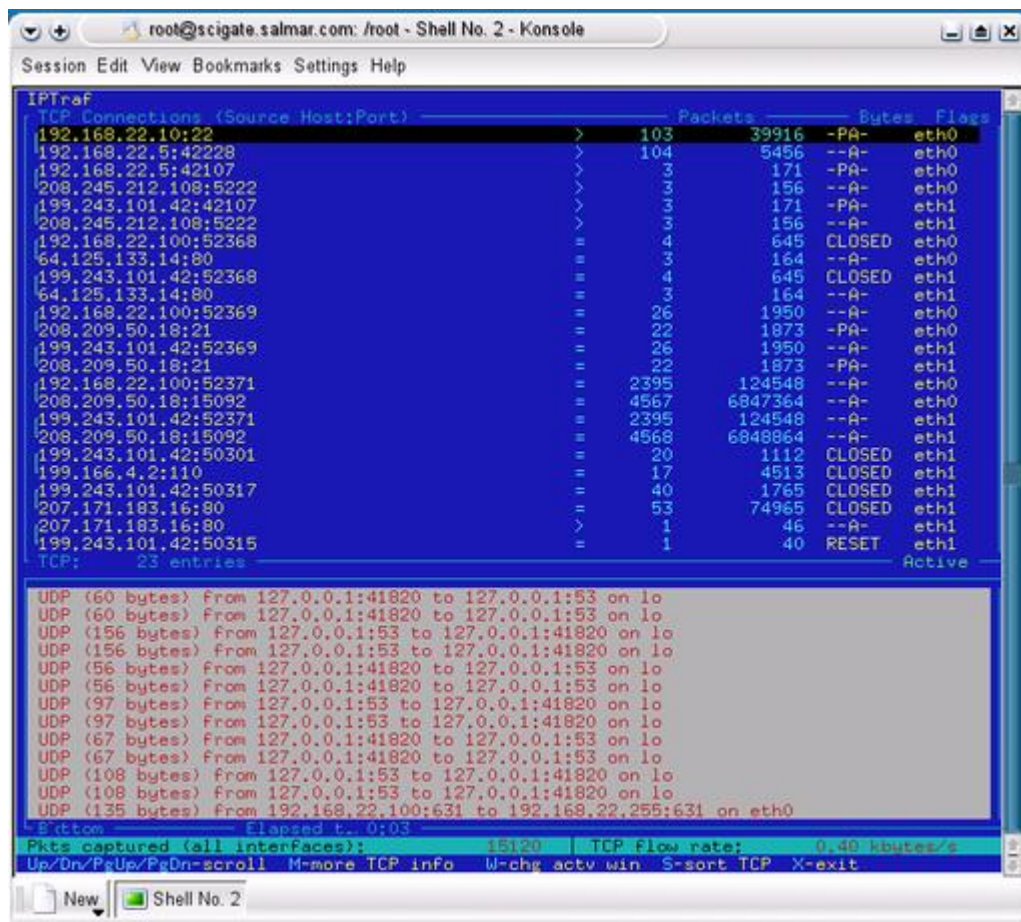


Figure 5. IPTráf's Default Monitoring Window

As IPTráf gathers and displays information, the screen may get very busy, very quickly. What I like to do is run the program in a larger X terminal, say 80 × 40. Pressing the Esc key lets you back out of the current function or view. From there, you can change settings, add or remove filters and continue with your data gathering. IPTráf also provides different views, from the default station-to-station traffic, basic and detailed interface traffic stats and physical stats to packet size breakdowns. Don't be fooled by the apparent simplicity of this package. IPTráf is flexible enough to satisfy a great many IP monitoring needs.

Well, *mes amis*, closing time is rapidly approaching. As François refills your glasses, I hope you will walk away with an appreciation of exactly how much is happening on your network. Keep in mind, however, that along with the richness of information these tools can deliver, moral and social implications are attached. Good system administrators know what is happening on their networks. They also know when to look the other way. On that note, I raise my glass to you, *mes amis*. *A vôtre santé! Bon appétit!*

## Resources

Contrack Viewer: [cv.intellos.net](http://cv.intellos.net)

Driftnet: [www.ex-parrot.com/~chris/driftnet](http://www.ex-parrot.com/~chris/driftnet)

IPTraf: [iptraf.seul.org](http://iptraf.seul.org)

Libjpeg (Independent JPEG Group): [www.ijg.org](http://www.ijg.org)

Libpcap (packet capture library): [www.tcpdump.org](http://www.tcpdump.org)

Libungif: [prtr-13.ucsc.edu/~badger/software/libungif](http://prtr-13.ucsc.edu/~badger/software/libungif)

Nnetstat: [www.aneptun.de/linux/Nnetstat](http://www.aneptun.de/linux/Nnetstat)

Pktstat: [www.itee.uq.edu.au/~leonard/personal/software/#pktstat](http://www.itee.uq.edu.au/~leonard/personal/software/#pktstat)

Marcel's Wine Page: [www.marcelgagne.com/wine.html](http://www.marcelgagne.com/wine.html)

Marcel Gagné lives in Mississauga, Ontario. He is the author of *Linux System Administration: A User's Guide* (ISBN 0-201-71934-7), published by Addison-Wesley, and is currently at work on his next book. He can be reached via e-mail at [mggagne@salmar.com](mailto:mggagne@salmar.com).

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## Paranoid Penguin

*Authenticate with LDAP*

**Mick Bauer**

Issue #112, August 2003

The directory server is running, so now it's time to configure crypto and add some users.

Last month, we did some of the preliminary work in setting up an OpenLDAP server. We installed the base, server and, where applicable, client packages for OpenLDAP, and we entered some basic configuration information into the file `/etc/openldap/slapd.conf` (slapd is OpenLDAP's server dæmon).

This month, we configure TLS encryption, start the dæmon and begin building an LDAP database. We won't have a finished authentication server yet, but we'll be pretty close. Next month, in this series' third and final installment, we'll get there.

### **TLS for Secure LDAP Transactions**

By default, OpenLDAP transactions over networks are conducted in clear text. If you're using OpenLDAP, for example, as a centralized address book server on a trusted network, that's probably fine. But if you're using it to authenticate users, regardless of whether the networks involved are trusted, you really ought to encrypt your LDAP communications to protect your users' passwords from eavesdroppers.

The LDAP v.3 protocol, support for which was introduced in OpenLDAP 2.0, provides encryption in the form of Transport Layer Security (TLS), the same mechanism used by web browsers and mail transport agents. TLS is the successor to SSL, the Secure Sockets Layer. All you need to do to take advantage of this is create a server certificate on your LDAP server; add a couple more lines to `/etc/openldap/slapd.conf`, and optionally, tweak slapd's startup options.

To generate a server certificate, you need OpenSSL. This already should be present on your system, because binary OpenLDAP packages depend on OpenSSL.

What sort of certificate you should use as your LDAP certificate actually is a fairly subtle question. Will the server need a certificate that has been signed by some other certificate authority (CA), such as Thawte or VeriSign? That is, will your LDAP clients need to see an externally verifiable certificate when connecting to your server? Or, will your organization be its own CA? If so, will the LDAP server also act as your local CA, issuing and signing both its own and other hosts' and users' certificates?

If your needs match any of those scenarios, you'll need to do a bit more work than I'm able to describe here. Suffice it to say that the certificate slapd uses can't have a password associated with it—its key can't be DES-encrypted—so a self-signed certificate, though technically a CA certificate, shouldn't be used as an actual CA certificate for signing other certificates. If you want to use your LDAP server as a real CA, you'll need to create two keys, a password-protected CA key and a password-free slapd key. Vincent Danen's article "Using OpenLDAP for Authentication" ([www.mandrakesecure.net/en/docs/ldap-auth.php](http://www.mandrakesecure.net/en/docs/ldap-auth.php)) discusses this.

For many if not most readers, it will be enough to create a self-generated TLS-only certificate to be used by slapd and slapd alone. If you don't care about being a CA and you don't need your LDAP clients to be able to verify the server certificate's authenticity via some third party, you can create your certificate like this:

```
bash-$> openssl req -new -x509 -nodes \  
-out slapdcert.pem -keyout slapdkey.pem \  
-days 365  
Using configuration from /usr/share/ssl/openssl.cnf  
Generating a 1024 bit RSA private key  
.....+++++  
.....+++++  
writing new private key to 'slapdkey.pem'
```

On this command line I told OpenSSL to generate a new X.509 certificate, without password protection, with the certificate (public key) stored in the current working directory in the file slapdcert.pem, and the private key stored in the file slapdkey.pem, with a lifetime of 365 days,

After issuing this command, you will be prompted for distinguished name information for the new certificate and key. For OpenLDAP's purposes, the most important field here is the common name. This must be set to your LDAP server's DNS name, which is the name your LDAP clients will see associated



with this certificate. If your LDAP server's IP address, for example, reverse resolves to bonzo.lamemoviesfromthepast.com but its server certificate shows a CN of bonzo.lm.com, LDAP clients will reject the certificate and therefore will be unable to negotiate TLS connections (with unpredictable results, depending on your client software).

Once you've got certificate and key files, copy them into /etc/openldap if you weren't already in that directory when you created them. Make sure that both of these are owned by ldap or whatever user your Linux distribution runs slapd as, Red Hat and SuSE use ldap, and that your key file has very strict permissions, such as -r - - - - - . Your certificate file may, however, be world-readable, because this contains a public key.

It's possible to specify the same filename after both the -out and -keyout options, resulting in both the certificate and private key being stored in a single file. This is fine if you don't intend to share the certificate. Keeping the two separate, however, allows you to distribute the server certificate while still keeping the server (private) key secret.

Naturally, it isn't enough to have certificate/key files in place; you need to tell slapd to use them. As with most slapd configuration, this happens in /etc/openldap/slapd.conf. Listing 1 shows the sample slapd.conf entries from last month's column (in case you've forgotten what we covered), plus three additional ones: TLSCipherSuite, TLSCertificateFile and TLSCertificateKeyFile.

### Listing 1. Customized Part of /etc/openldap/slapd.conf

```
database          ldbm
suffix            "dc=wiremonkeys,dc=org"
rootdn            "cn=ldapguy,dc=wiremonkeys,dc=org"
rootpw            {SSHA}zRsCkoVVVDXObE3ewn19/Imf3yDoH9
directory         /var/lib/ldap
TLSCipherSuite    HIGH:MEDIUM:+SSLv2
TLSCertificateFile /etc/openldap/slapdcert.pem
TLSCertificateKeyFile /etc/openldap/slapdkey.pem
```

TLSCipherSuite specifies a list of OpenSSL ciphers from which slapd will choose when negotiating TLS connections, in decreasing order of preference. To see which ciphers are supported by your local OpenSSL installation, issue this command:

```
openssl ciphers -v ALL
```

In addition to those specific ciphers, you can use any of the wild cards supported by OpenSSL, which allow you to specify multiple ciphers with a single word. For example, in Listing 1 TLSCipherSuite is set to HIGH:MEDIUM:+SSLv2; as it happens, HIGH, MEDIUM and +SSLv2 all are wild cards.

HIGH means “all ciphers using key lengths greater than 128 bits”; MEDIUM is short for “all ciphers using key lengths equal to 128 bits”, and +SSLv2 means “all ciphers specified in the SSL protocol, version 2, regardless of key strength”. For a complete explanation of OpenSSL ciphers, including all supported wild cards, see the `ciphers(1)` man page.

`TLSCertificateFile` and `TLSCertificateKeyFile` are more obvious. They specify the paths to your certificate file and private key file, respectively. If both certificate and key are combined in a single file, you can specify the same path for both parameters.

### slapd Startup Options

We've done everything we need (on the server end) for TLS encryption to work. Only one detail to consider remains. Should we force the use of TLS for all LDAP requests from the network or keep it optional?

By default, `slapd` listens for LDAP connections on TCP port 389 and accepts either clear text or TLS-encrypted connections on that port. However, if you're using LDAP for authentication, you probably don't want to make TLS optional. A better approach in that case is to have `slapd` listen for clear text-only LDAP connections on TCP 389 on the loopback interface only and have `slapd` listen for TLS-enabled (ldaps) connections on TCP 636 (the standard port for ldaps) for all other local addresses.

This behavior is controlled by `slapd`'s startup option `-h`, used to specify the URLs to which `slapd` will respond. For example, `slapd -h ldap://127.0.0.1/ldaps:///` tells `slapd` to listen on the loopback address (127.0.0.1) for ldap connections to the default ldap port (TCP 389) and to listen on all local addresses for ldaps connections to the default ldaps port (TCP 636).

If you run Red Hat 7.3 or later, this actually is the default behavior: `/etc/init.d/ldap` checks `/etc/openldap/slapd.conf` for TLS configuration information and if it finds it, sets the `-h` option exactly like the one in the previous paragraph's example. If you run SuSE 8.1 or later, you can achieve the same thing by editing `/etc/sysconfig/openldap` such that the value for `OPENLDAP_START_LDAPS` is `yes`, and then editing `/etc/init.d/openldap` to set the value for `SLAPD_URLS` to `ldap://127.0.0.1`. This variable is defined early in the script, with a default value of `ldap:///`.

Other Linux distributions may have different ways of passing startup options like `-h` to `slapd`, but hopefully by now you get the idea and can figure out how to make `slapd`'s listening ports work the way you want.

## Testing

So, does our TLS-enabled LDAP server actually work? A quick local test will tell us. First, start LDAP:

```
bash-$ /etc/init.d/ldap start
```

Next, use the `ldapsearch` command to do a simple query via loopback:

```
bash-$ ldapsearch -x -H ldaps://localhost/ \
-b 'dc=wiremonkeys,dc=org' '(objectclass=*)'
```

Naturally, your own LDAP server will have a different base DN than `dc=wiremonkeys,dc=org`. If you prefer, you can run this last command from a remote host, specifying the LDAP server's name or IP address in place of `localhost` in the `-h` option. If the LDAP server returns a dump of the LDAP database, which actually is empty at this point, followed by the string `result: 0 Success`, your test has succeeded.

If you get an error about an invalid certificate, try adding this line to your client system's `/etc/openldap/ldap.conf` file:

```
TLS_REQCERT    allow
```

This allows your OpenLDAP or OpenLDAP-based client software (for example, `gq`) to accept self-signed server certificates.

## LDAP Schema

You're almost ready to start populating the LDAP database. On the one hand, tools like `gq` and `ldapbrowser` can reduce the ugliness and toil of LDAP data entry and administration greatly. But to get to the point where these tools can be used, you first have to settle on a combination of LDAP schema, and this is where things can get unpleasant.

For purposes of this discussion, two types of LDAP data matter, attributes and object classes. Attributes are the things that make up a record. A user's phone number, e-mail address, nicknames and so on are all attributes. You can use as many or as few attributes in your LDAP database as you like. You even can invent your own. But for a record to contain a given attribute, that record must be associated with the proper object class.

An object class describes the type of record you're trying to build. It defines which attributes are mandatory for each record and which attributes are

optional. You might think, “that’s easy, then I simply need to choose an object type that provides the group of attributes I want to store for my users and associate each user record with that object class.” If you thought that, you’d be only partly right.

In practice, you’ll probably want to use attributes from a variety of object classes. “Well, fine”, you think, “I’ll just specify multiple object classes in each user record and get my full complement of attributes *à la carte*. Whatever.” Right again, but there’s more to it than that. Chances are the object classes that provide the attributes you need are spread across a number of schema files (these are text files, each of which contains a list of attributes and the object classes that reference them). So, even before you can begin composing your user records, each containing a stack of object class statements and a bigger stack of attribute settings, first you need to make sure `/etc/openldap/slapd.conf` contains include statements for all the schema files you need, usually present in `/etc/openldap/schema`.

For example, suppose that because we’re going to use our sample LDAP server for authentication, we want to make sure that no matter what, we’re able to specify the attributes `userid` and `userPassword`. Doing a quick `grep` of the files in `/etc/openldap/schema` shows that `uid` appears in the file `inetorgperson.schema` in the MAY list (of allowed attributes) for the object class `inetOrgPerson`. This has two ramifications. First, `/etc/openldap/slapd.conf` needs to contain this line:

```
include /etc/openldap/schema/inetorgperson.schema
```

Second, whenever I create a user record, I need to make sure that an `objectclass: inetOrgPerson` statement is present.

### Creating and Adding User Records

So, how do you create user records? Ideally, with the GUI of your choice. Last month I mentioned `gq`, which is a standard package in many distros; another excellent tool is `ldapbrowser`, available at [www.iit.edu/~gawojar/ldap](http://www.iit.edu/~gawojar/ldap). Initially, however, you’ll probably want to add at least your organizational entry manually, by creating an `ldif` file and writing it to the database via the `ldapadd` command. An `ldif` file is a text file containing a list of attribute/object class declarations, one per line; a simple one follows:

```
dn: dc=wiremonkeys,dc=org
objectclass: top
objectclass: organization
o: Wiremonkeys of St. Paul
```

Here, we're defining the organization wiremonkeys.org. We specify its distinguished name, associate it with the object classes' top (mandatory for all records) and organization and specify the organization's name (Wiremonkeys of St. Paul), which is the only mandatory attribute for these two object classes.

To write this record to the database, issue this command:

```
bash-$ ldapadd -x -H ldaps://localhost/ \  
-D "cn=ldapguy,dc=wiremonkeys,dc=org" \  
-W -f wiremonkeys_init.ldif
```

As with most OpenLDAP commands, `-x` specifies simple password authentication, `-H` specifies the LDAP server's URL, `-D` specifies the DN of the administrator account and `-W` causes a prompt for the administrator's password. The `-f` option specifies the path to our ldif file.

Confused yet? I've packed a lot of information into this month's column, but our LDAP server is very nearly done. To finish yours without waiting for next month, see the OpenLDAP Administrator's Guide at [www.openldap.org/doc](http://www.openldap.org/doc) for more information about TLS, startup flags, schema and ldif files.

Mick Bauer, CISSP, is *Linux Journal's* security editor and an IS security consultant for Upstream Solutions LLC in Minneapolis, Minnesota. Mick spends his copious free time chasing little kids (strictly his own) and playing music, sometimes simultaneously. Mick is author of *Building Secure Servers With Linux* (O'Reilly & Associates, 2002).

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## Linux for Suits

*Practical Penguin Progress*

**Doc Searls**

Issue #112, August 2003

What we're learning from Linux's growing silent majority.

Here at *Linux Journal* we've been deeply involved in the Linux community ever since Phil Hughes started the magazine back in 1994 when Linux was turning 1.0. Through that whole time, we've served two different overlapping communities of interest—one political, the other practical. The political community cares about issues and the characters who drive them. The practical community cares about the nuts-and-bolts challenges of putting Linux to work.

During the past year, we've seen a growing split between the two, because the practical penguin population has been snowballing. And, on the whole, newborn practical penguins are joining the species for practical purposes. They don't care much about licensing, software patents or threatening legislation. To them, free software means free-as-in-beer. The free-as-in-speech stuff is fine, but it's not their issue.

What practical penguins care about is getting stuff done. If they're shucking off Microsoft, it's mostly because other stuff is cheaper, does a better job or is a smaller pain in the butt. The fact that it's free, and seems to grow wild in nature, makes Linux, along with the free stuff that runs on it, highly appealing. The fact that programmers get to keep and share their improvements to code is another bonus.

Of course, many practical penguins are political ones, too. But the political penguin percentage is getting smaller, even though their absolute numbers are getting larger.

Sound familiar? This is very much like what happened to the Internet in the mid-90s. Prior to that time, the Net was the province of academics and hackers working for organizations privileged to have Net connections. Politics and utility were closely connected. What made the Net important was also what made it handy. At a certain point, however, the appeal got turned around. What made it handy made it important. And as it became handier for more and more people, the percentage of people who continued to care about its founding virtues grew smaller. Today a noisy minority still exists that loves the Net's founding virtues and inveighs tirelessly on its behalf. That's why I wrote "World of Ends" ([worldofends.com](http://worldofends.com)) with David Weinberger, for example. But most people who use the Net really don't care. They only like what the Net does.

That's where we're going with Linux. It's the somewhat unexpected yet entirely predictable consequence of World Domination. When Linux becomes the default operating system—ubiquitous, or close enough—it still will be important for some of us to care about why it got that way and why its founding virtues remain as operative as ever. But as a percentage of the whole community, that "some" will be a progressively smaller group, even if they grow in absolute numbers. The only problem is the practical majority still is mostly a silent one, while the political minority still is widely heard. This creates distortions in the press, which often treats Linux as a political cause rather than a market effect. It also denies the political voices much of the validation they deserve. Unfortunately, the silence is likely to persist for a while. Linux is long past the Not Going Away stage, but still a long way from World Domination.

So, I've been making it my job to watch what the practical penguins have been up to and to talk to as many as I can. I outlined my first findings last month in "How Linux Makes Smart Companies Smarter". Since then I've condensed my conclusions to a few key points:

1. Linux is a project, not a product. So are the other members of the LAMP suite: Apache, MySQL, Perl, Python, PHP and PostgreSQL.
2. Free software and open source are ways that the demand side supplies itself. Call this DIY-IT, or Do It Yourself Information Technology. In some cases, DIY-IT is so well developed that customers hardly need vendors at all.
3. DIY-IT is causing a shift in market power from supply to demand.
4. Clueless vendors fight this shift. Smart vendors learn from smart customers, often by working on the same development projects.
5. IBM and Oracle are two standout examples of suppliers who are hip to the smart stuff happening on the demand sides of their markets—largely because it's happening inside their companies as well.

6. This shift in power, and in the way clues are passed among practitioners, is turning the software industry into something very much like the construction industry, which also has architects, designers, builders, tools, platforms, frameworks, a hearty DIY tradition, project-based work and open-source know-how.

Figure 1 presents the old view of the software marketplace. Vendors had a controlling position, because they supplied the goods—and the expertise to go with them. This is what made 90% gross profit margins not only possible, but standard.

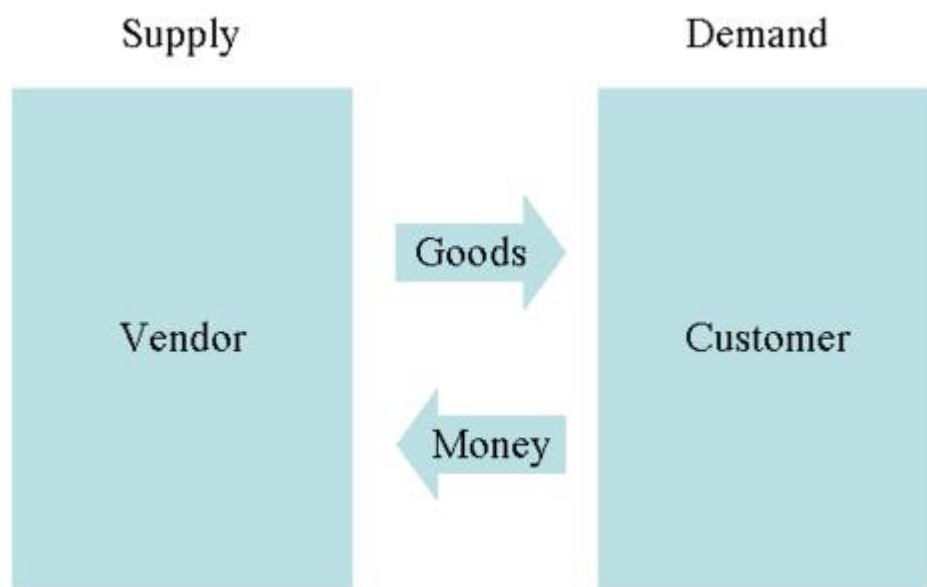


Figure 1. The Old View of the Software Marketplace

Figure 2 shows how the new marketplace looks. Here the demand side—the customer—is in a position to supply itself. It still buys goods from vendors. In fact, smart vendors' goods are improved by participation in the same development community as their customers.



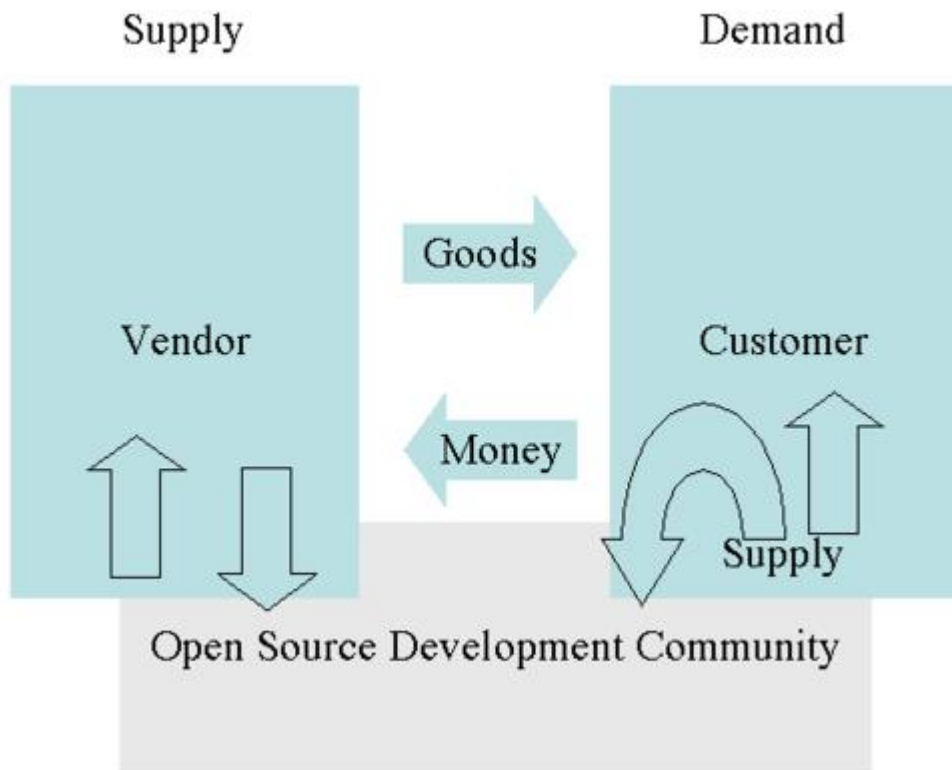


Figure 2. The New Marketplace

As the market continues to mature, the open-source development community will come to be understood as a collection of know-how. In a mature industry like construction, this community includes everyone who participates expertly in the industry. When that happens in the software industry, topics like patents and licensing will acquire more confined meanings than they have today. We won't need patents on "business methods", and most of us won't need licenses to tell us it's good to share what we know. We'll do what comes naturally, which is what free software and open-source licenses predicted in the first place.

Meanwhile, the political remains as important as it ever was. Let's remember, if it weren't for the political penguins who got the snowball rolling, the practical penguin movement never would have picked up its mass and momentum.

Doc Searls is senior editor of *Linux Journal*.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## EOF

### *Consider Accessibility*

**Janina Sajka**

Issue #112, August 2003

How does Linux measure up for users with disabilities?

Is Linux accessible? In other words, how well does Linux work for persons with disabilities? The answer today depends, in great degree, on which disability we're talking about. However, several factors have begun driving Linux to become a friendlier environment for users who are persons with disabilities.

Perhaps the most fundamental factor spurring Linux toward a more cohesive response to the accessibility challenge is the Linux ethos itself. The communitarian values that undergird free and open-source software simply cannot tolerate the notion that Linux should exclude anyone. Doesn't everyone have the right to run the program? Isn't everyone free to study and improve the source? It goes against the grain to leave anyone out simply because they can't read text on a video display or press both the meta key and F1 simultaneously. Linux always has valued and encouraged active participation from its community of users. We pride ourselves that quality contributions, whether code or documentation, float to the top and become incorporated in the next release. And, indeed, much in Linux exemplifies accessibility. Unfortunately, it is generally incidental accessibility—a by-product of the fact that Linux is deeply rooted in ASCII on the one hand, and on Linux's stellar ability to accept input from most any kind of device on the other. Not that developers intentionally would exclude support for accessibility—it just hasn't been a design criterion in the code review process.

Another potent contributory factor is the emergence of laws and policies giving preferences to software and systems that are accessible. Chief among the examples of such factors is the legal mandate in the US government procurement known as Section 508. This recently strengthened US law requires the US government to procure accessible “electronic and information

technology” for use in the federal workplace and in systems that provide information and services electronically to the public if accessible technology is commercially available. Section 508 has expanded interest in accessibility greatly simply because the US government is a big IT customer, currently spending about \$40 billion annually on technology—a number expected to grow by 50% in merely five years. Clearly, most vendors that bundle Linux products and services might find Uncle Sam a customer to court. Coupled with the high esteem Linux people have for Linux communitarian values, it rankles one to think that Linux itself might, just might, not measure up against this social performance yardstick.

Over the past few years several developers, funded principally by Sun Microsystems, but also by IBM, Red Hat and Ximian, among others, systematically have been putting the new GNOME 2.0 desktop accessibility framework together. Frankly, it is hard to believe this would have happened had it not been for Section 508, but the benefits of these efforts will extend well beyond the US. The GNOME Accessibility web page ([developer.gnome.org/projects/gap](http://developer.gnome.org/projects/gap)) is possibly the best current source of information, in a Linux context, of what constitutes accessibility and how it can be achieved.

However, we might summarize accessibility as an I/O issue. The fundamental problem, though varied and complex in its myriad permutations, is simple. For every way by which a computer can accept input from a human user, there is someone who can't do things that way. Likewise, for every output modality intended for human consumption, there are users who cannot accept that particular medium as input. So, the challenge, the subject of articles to come, is simply how we get this intrinsically binary instrument to accept input from whatever device might be needed. And, how do we reformat and re-purpose output to encompass the rest of the user base?

The ethical imperative is clear. The software commons must be for everyone if it is to be a true commons. Even those whose software isn't directly related to providing interface support to users with disabilities must help, at some point and in some manner, to make their applications work for the widest possible user base. It is indeed possible this demand will prove disruptive in some areas. For the most part, however, it should prove minimally invasive, simply because of the socketed nature of Linux. Clearly, much of the work will be performed by specialists, because there is a good deal of specialized knowledge involved in getting solutions that work well. Ultimately, of course, the various communities of users with disabilities will determine for themselves how to do things. But, what they will want to do are the same things the rest of us want to do—which takes us back full circle to the communitarian ethos.

Janina Sajka is the director of Technology Research and Development, at the American Foundation for the Blind (AFB).

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

Advanced search

## Red Hat 9

**Marco Fioretti**

Issue #112, August 2003



### Product Information.

- Manufacturer: Red Hat
- URL: [www.redhat.com](http://www.redhat.com)
- Price: \$39.95 US

### The Good.

- Nice fonts and overall look and feel.
- Best-of-breed applications chosen for consistency.
- UTF-8 support.

- Educational software included.
- Stable and good hardware recognition.

### **The Bad.**

- CD burning may be problematic in certain conditions.
- Some package dependencies are incorrect.
- Excessive startup time for desktop and OpenOffice.org.

Red Hat 9, code-named Shrike, was released on April 7, 2003. It took everyone by surprise, including this reviewer, who was expecting an 8.1 release. As a matter of fact, starting with this release, the offering from Red Hat has changed, clearly splitting into two lines with different purposes and targets.

### **Product Lines**

The Enterprise line is designed for professional use on mission-critical servers or corporate workstations with powerful hardware and the need for the highest possible stability. This product line will have release cycles of 12–18 months and five years of support for every version, which is great for all those companies that have to develop and maintain products for several years. The top product is the Advanced Server for up to eight CPUs and 16GB of main memory. The ES server is basically the same thing but optimized and scaled down for systems with no more than two CPUs and 4GB of RAM. The Enterprise Workstation is the version for the corporate desktop. The first and last products also are available for the Itanium 2 processor.

Standard Red Hat Linux is defined by Red Hat itself as a “community product for SOHO users, independent professionals, students and hobbyists with minimal support needs”. It also has a professional edition (\$149.95 US), with an extra multimedia CD, manuals and phone support for 60 days. This line marks the beginning of a different numbering scheme. Patches and updates will be released, of course, but for periods shorter than the Enterprise Line (Red Hat says “at least until April 30, 2004”), and the next version will be 10, not 9.1. This review covers Red Hat Linux 9.

### **Base System**

The kernel shipped with the CDs is 2.4.20. Everything has been compiled with GCC 3.2.1 and with GNU libc 2.3.2. Both the kernel and the libraries already have been updated on Red Hat's web site, so be sure to grab the latest versions from <https://rhn.redhat.com/errata/rh9-errata.html> after you install. The Web server installed is Apache httpd 2.0.

Installation didn't really differ from previous releases, except for more refined Red Hat commercials, but produced better end results. This is the first Red Hat on my computer that allowed me to use my UMAX Astra 610S scanner without manual tweaking.

In user space, all the most popular applications are provided. The shipped versions often are a bit behind those provided, for example, by Mandrake 9.1, but unless one really wants the bleeding edge, this is not a big deal. The versions of some of the most popular programs are reported in Table 1.

**Table 1. Popular Programs Included in Red Hat 9**

Emacs	21.2
Evolution	1.2.2
Gaim	0.59.8
Galeon	1.2.7
GIMP	1.2.3
GnuPG	1.2.1
Kdebase	3.1
KOffice	1.2.1
Mozilla	1.2.1
Mutt	1.4
OpenOffice.org	1.0.2
OpenSSH	3.5p1
Perl	5.8.0
Quanta	3.1

Some applications may have benefited from a more modular packaging. OpenOffice.org, for example, requires two extra RPMs, `openoffice-libs` and `openoffice-i18n`. The files to manage all conceivable languages are placed on disk, no matter what you choose at install time. The end result is that on Shrike, OpenOffice takes almost 200MB of space.

System-wide support for UTF-8 is great, in spite of one issue, which doesn't depend on Red Hat. There is no clean, unique solution to guarantee that all, possibly old, Perl scripts will continue to process all, possibly old, text files as expected. As serious as it is, this problem comes from the simple fact that text files are plain and cannot specify how their content is encoded, unlike e-mail messages and XML documents. The script must then be helped from the outside by setting environment variables to work properly.

This release is the first Red Hat to support the Native POSIX Thread Library (NPTL). This should increase performance if an application has been coded or modified to use it. On the other hand, it may interfere with some old applications or with ones operating at a very low level, such as WINE. If this is the case on your system, NPTL can be turned off at the user level by adding the following to the `cshrc` or `bashrc` files, where `kernel-version` is 2.4.1 or 2.2.5:

```
export LD_ASSUME_KERNEL=kernel-version
```

or system-wide by adding `nosysinfo` at the end of the kernel load line in your bootloader configuration. The release notes also warn that “kernel support for the new NPTL feature changes several internal kernel programming interfaces significantly. As a result, several external kernel modules may not compile without modifications.. Examples currently include the NVIDIA and ATI 3-D modules.”

### User Interface

Fonts are anti-aliased and are beautiful. The integration with the `xft2/fontconfig` system has matured from Red Hat 8. The most common problem annoying early users, the fact that the dash and other characters in man pages were not displayed properly under an UTF-8 locale, now is gone. Some applications still work outside the system, however. OpenOffice.org is the main case, but being a cross-platform application it will move to `fontconfig` later, and Red Hat configured OpenOffice.org fonts properly anyway.

The process started with Red Hat 8—customizing GNOME and KDE to offer a consistent look called Bluecurve and the same default choice for the most common tasks such as Web browsing and e-mail—continues. I deliberately chose to review the KDE desktop partly because it is not Red Hat's first choice, and partly because the difference is smaller in Red Hat. Figure 1 shows an almost vanilla Bluecurve/KDE screenshot. My only changes were placing the panel vertically, not installing Evolution and choosing different colors for the main panel icons. The icons for the text files include the beginning of the text of the file.



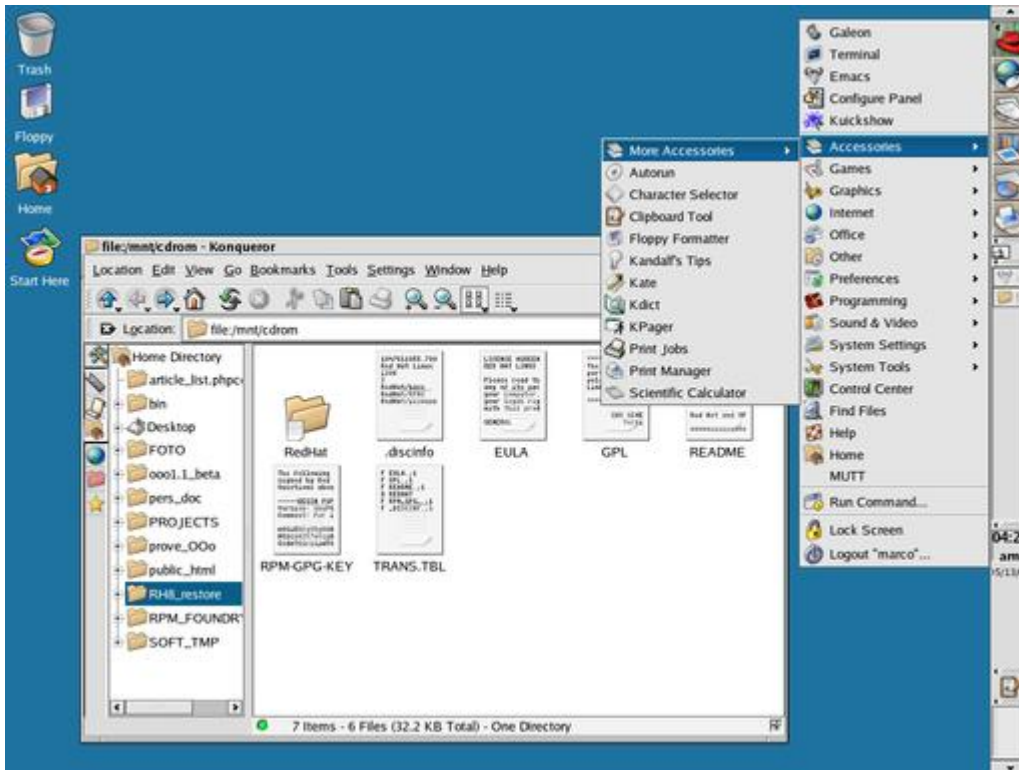


Figure 1. KDE in Red Hat 9

Figure 1 also shows a change from Red Hat 8 that came from popular demand—a different menu organization. Each submenu lists only five to ten applications and has a “More programs of this type” submenu. This definitely makes searching for programs easier. The first five entries of the main menu are filled dynamically with the five most used or most recently used programs. For some reason, not all the menu entries are considered when doing this. I added Mutt, opened it continuously, and it never showed up at the top.

Another minor annoyance with the desktop is the fact that, although automount works nicely and opens a file manager window as soon as you insert a CD-ROM, it works too much in at least one case. When I inserted the first Red Hat 9 CD, simply to read the release notes, the system said that to run the rh-install-helper, I had (rightly) to type the root password. When I clicked cancel, it exited with “unknown exit code”.

### Multimedia

The short story is that Red Hat 9 can play music and movies fine, it simply doesn't want to by default. The distribution does not include MP3 players, deCSS or anything else, including the fortune program, that cannot be certified as freely redistributable with respect to current law. Please don't whine about this, as it does the right thing, which is to force the end user to choose between asking her government representative if certain laws, such as the controversial Digital Millennium Copyright Act (DMCA) can be reformed, or deliberately installing the missing packages herself, which is really easy anyway.

## System Management

Under both GNOME and KDE, normal users can do everything they typically are allowed to do without problems. The only misconfiguration I found is shown in Figure 2. LPD is declared as the currently used printing system, although CUPS had been chosen. Everything printed fine, but the text is misleading.

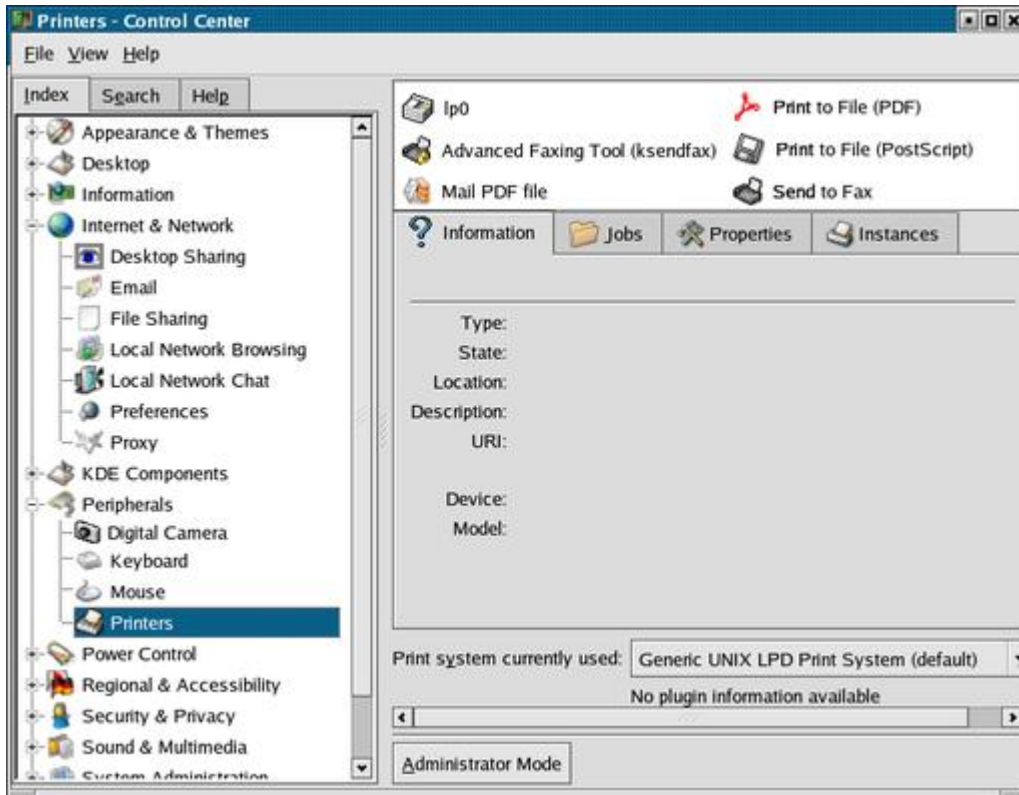


Figure 2. The Shrike/KDE Control Center

Red Hat provides its own set of system administration tools, most of them named `redhat-config-*` (simply type `redhat -`, then press Tab while logged in as root to see them all). All are documented in the downloadable or printed Red Hat manuals and are adequate for beginning and intermediate system administrators. Figure 3 shows the security/firewall tool, which is limited but sufficient for home users. The only problem found with other services is that the `redhat-switch-mail` tool would not work in the text version.



Figure 3. The Firewall Manager

### Miscellanea

The CD-ROM includes the KDE-EDU package, a nice collection of educational and recreational programs. This household's favorite is KStars, shown in Figure 4.

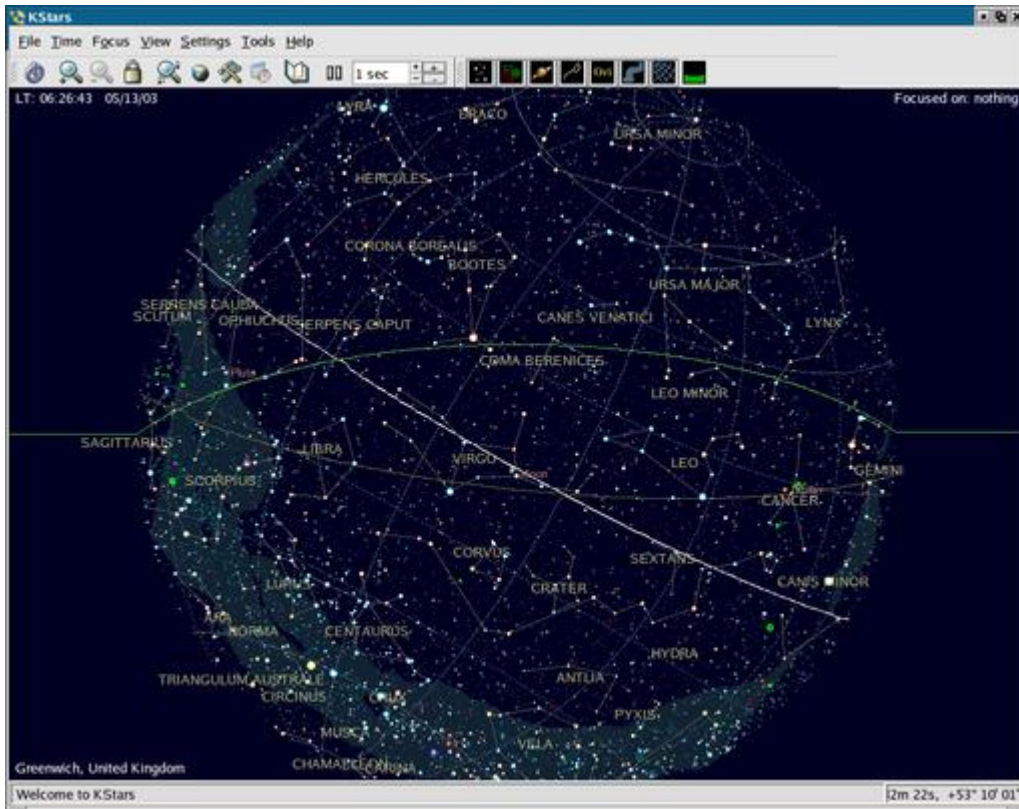


Figure 4. To Infinity and Beyond with KStars

### Scanning and OCR

As already mentioned, Shrike was the first Red Hat version to let me use my scanner without changing any settings. I was able to scan without a glitch, so when I saw the OCR button I immediately pressed it (I consider this the number one application area still sorely missing good software for Linux, be it free or proprietary). The system answered with "gocr: command not found". I didn't find this program on the CD-ROMs, so it does seem to have escaped the dependency checks. I found the gocr RPM on-line and am still testing it.

### CD Burning

During the first weeks of life of the Shrike users list, a noticeable amount of traffic was devoted to CD burning problems. On the test system used for this review, using Xcdroast on a Philips CDRW1600 device, no problems were observed. Everything was recognized without manual intervention, and no disks were wasted. Several users reported that problems disappeared by removing the magicdev package. This tool is supposed to perform several user-friendly actions when removable media are inserted—playing audio CDs, opening a burn window in Nautilus and so on. The fact that Nautilus (and its dependencies, like magicdev) were not installed on the test system seems to confirm the hypothesis that magicdev, at least as packaged in Red Hat 9 version 1.1.4, is not ready for prime time, at least not for all systems.

## Conclusion

Red Hat 9 is indeed a nice desktop. Overall performance, even on a relatively limited system, is not slower than with the previous release. The convergence imposed on KDE and GNOME is much less dramatic than it may seem and hopefully will lead to less work to maintain future versions and fix the quirks reported here.

Marco Fioretti is a hardware systems engineer interested in free software both as an EDA platform and (as the current leader of the RULE Project) as an efficient desktop. Marco lives with his family in Rome, Italy.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## Letters to the Editor

### **rsync Solves Networking Problems**

Thank you and thanks to Mick Bauer for the articles on rsync [*LJ*, March and April 2003]. rsync's ability to "pull" as well as "push" solved a long-standing problem of how to synchronize through a firewall without too much hassle. rsync is also more intuitive than rdist.

—

Tom Kuiper

### **Pascal vs. C**

Pascal and C arrived on the scene about the same time. As far as it went, Pascal was a better idea than C, but Pascal did not go far enough, and C won. Pascal inspired Ada, which was awful. C grew into C++, even more half-baked than C. It took Niklaus Wirth two decades to get Pascal fully grown into a finished product of the quality of the original idea. The result is Oberon-2. Unfortunately, it arrived after the train had left the station. For more see [www.waltzballs.org/other/prog.html](http://www.waltzballs.org/other/prog.html).

—

Donald Daniel

### **How to Authenticate Wireless Connections?**

I want to set up something to regulate my wireless network so the person has to log on to the domain.

—

Aaron Egbert

NoCatAuth, [nocat.net](http://nocat.net), does what you need. We plan to cover it next issue. —Ed.

#### **Latin for Accountants**

In response to a letter in the June 2003 issue, I was motivated to clear up a mistake. Kathy claims that "Debit on the left, Credit on the right" in double-entry accounting can be explained by the Latin for left and right. The Latin words for left and right are *sinister* and *dexter*, respectively. Credit actually comes from *credo*, to trust, and debit comes from *debeo* meaning to owe.

—

Phil

#### **How to Pronounce Linux?**

I have an ongoing debate with my son on the correct pronunciation of Linux. I say that it is pronounced either as in finish or as in helix, and my son says it is pronounced as in Lennox or tennis. Who is correct?

—

Ira Friedman

Listen to Linus Torvalds say it at [www.kernel.org/pub/linux/kernel/SillySounds](http://www.kernel.org/pub/linux/kernel/SillySounds). —Ed.

#### **Why Review SCO?**

I've been a fan and a subscriber of your magazine for some time now. However, I was recently disappointed to see you running a review of SCO's Linux product, in light of their lawsuit against IBM [*LJ*, June 2003]. To get directly to the point, given SCO's opinion, and now their opening terrorism tactics to Linux customers, publishing a SCO product review is in bad taste and insulting to the entire Open Source Development community.

—

Dave Crown

SCO pulled their Linux distribution from the market, and made sweeping accusations that Linux contains code copied from SCO UnixWare, after we already had gone to press. But our timing wasn't that bad after all. Because we still have access to the SCO update service, we can substantiate the fact that SCO continues to offer Linux source code under the GPL: [linuxjournal.com/article/6899](http://linuxjournal.com/article/6899). SCO's position on Linux, and the responses from developers and companies, are changing too rapidly to cover in a monthly publication. Check our web site for the latest. —Ed.

## GNOME 2 Tips

I generally agree with the opinion from Dr Mark Alford on GNOME 2 [Letters, *LJ*, June 2003], but I believe some comments will help him and other readers.

You can still use sawfish on Red Hat 8 and configure it from the Extras→Preferences menu, including the keyboard shortcuts and window layout. You need to add the following lines in your `.bash_profile` file:

```
WINDOW_MANAGER=/usr/bin/sawfish
export WINDOW_MANAGER
```

See the `gnome-wm` script in `/usr/bin`.

You can recover some of the chopped-off functionality of GNOME 2 by writing a Nautilus script. For example, if you want to see the list of files contained in an RPM file, write an `rpm` command in a script:

```
rpm -qlp $@ >/tmp/rpmlist.txt
gview /tmp/rpmlist.txt
```

Many script examples are at [g-scripts.sourceforge.net](http://g-scripts.sourceforge.net).

—

Hiroshi Iwatani

## Mind Your License Ps and Qs

Although my employer is very supportive of Linux (it's our primary operating system) and open-source development in general, our legal department recently has become quite sensitive to the contents of open-source licenses. Specifically, they are concerned about some of the licenses that require any modifications to code to be distributed back to the original authors—even if it is not otherwise publicly redistributed. They also have discovered some odd parts to licenses, such as “you can use this software, but you have to buy me a beer if you're ever in Boston.” The net result is that we now have to obtain legal approval before downloading, installing, using and especially modifying any open-source software. (They're equally restrictive about proprietary licenses, but that seems justified.) Although I appreciate the levity some open-source “licenses” have, it seems that we may need better standards in licensing to encourage its adoption in litigation-heavy corporate environments.

—

Doug Cooper



Once you add terms like the ones you mention, the license is no longer a Free Software license or an Open Source one. These issues do not affect the standard free software licenses such as the GNU GPL and the Apache license. Can you get your legal department to approve the standard licenses, so you don't need a program-by-program review? —Ed.

#### **Freedom Powers Software Market**

In the June 2003 issue of *Linux Journal*, a letter from John was run under the title "Freedom Threatens Some Companies". In this letter, John wrote that he felt the integrated library system Koha, [www.koha.org](http://www.koha.org) and other free software projects represented a threat to small- to medium-sized commercial software companies. He argued that it seemed likely that a single free software project eventually could dominate a particular market niche and thus drive out the commercial competition. I think he's wrong, both in the case of Koha and in the larger case of free software in general. I can't give specific information about other projects, but I can see how Koha can help, not hurt, the library automation marketplace. Koha is a thriving little free software project. Several mailing lists are devoted to it, nearly 40 people have CVS access, and a growing number of sites use it. Currently, at least three commercial ventures work on Koha actively, and several others offer commercial support for the platform. Any library automation vendor is perfectly able to pick up Koha and create an offering built around it—in fact, I'd encourage them to do so. If Koha doesn't meet their needs, perhaps one of the other free library systems will. Libraries who adopt Koha take on a level of commitment resulting in their giving back to the larger Koha community. In some cases this means developing new features or fixing bugs in Koha themselves, in other cases they might hire someone to do so. Some libraries invest in Koha by reporting bugs, writing documentation, answering questions on the mailing list or explaining library-specific knowledge to developers without library backgrounds. This same kind of commitment is seen from many other users of free software. Not everyone gives back, but enough do to keep the community viable.

—

Pat Eyler  
kaitiaki/manager, the Koha Project

#### **Learning to Document Software**

For every line of code we write, every class or function that we make, there must be documentation. My professors all have commented on my documentation or the lack of it. As a self-taught programmer, I see coding as more of an art form than a how-to guide. I had written hundreds of lines of code and projects that do amazing things, but no one could follow my source code. Now that I am helping to develop open-source projects, using documentation is imperative. The Open Source community has brought me

into the light. With documentation we can all make beautiful software together. Thanks to *Linux Journal* for making me a comprehensive programmer.

—

Carl Jones  
Student at Southeastern Louisiana University

*C'est magnifique!*

Being an apprentice DIY chef and a passionate Linux worshiper, I always enjoy reading *Cooking with Linux*. Some days ago I was preparing some cookies that reminded me of you wearing a chef hat. I couldn't resist taking some photos and sending them to you! Hope you enjoy them.

—

Gianluca Insolubile



**`/var/spool/fanmail`**

I've got to comment on the article "Introducing the 2.6 Kernel" by Robert Love [LJ, May 2003]. Training strictly as a network engineer for 14 years, programming jargon was always over my head. Mr Love kept the developer's jargon in sync with the rest of us out here in the network field, which made it the most pleasurable article I've read to date. I actually understood what the kernel was doing and as such, now have a greater appreciation for what's happening on the inside. I hope you can convince Mr Love to write ALL future kernel release articles for *Linux Journal*.

—

Lyndon Tynes  
Intel Solutions Center Network Engineer

Keep watching Kernel Korner for more from Robert Love. —Ed.

#### **Fighting C++ Rumors**

Bravo on the C++ editorial. The clock is ticking, and die-hard C/procedural programmers are running out of excuses to eschew the notion of C++ as a suitable language for small- to mid-sized projects. The year 2003 is upon us, hence the time to dispel some old rumors (or, in some cases, review some old facts):

- “C++ is slow”: modern compilers are closing the C/C++ speed gap. Some of C++'s fabled performance lags may be due to proper (and automatic) calls of constructors and destructors, the sort of initialization and cleanup that you already should have in your C code anyway. To close the remaining gap, we can replace certain object hierarchies with template programming, in which we perform decision making at compile time rather than runtime.
- “Compiled C++ libraries are incompatible with one another”: compilers are stepping up to the line drawn by The Standard, which means this soon will be a thing of the past.
- “C++ is the worst of both worlds”: this is purely an issue of perspective: imagine the cleanliness of encapsulation, constructors/destructors, exceptions and inheritance when needed, plus the speed of pointers when wanted.
- “C++ is fat and complex”: is this really C++ or just the OO paradigm? It takes some exercise to get one's mind around it and to write true object-oriented software, but once you understand it you'll never go back. For those thorough procedural programmers, imagine being able to wrap those cleanup-style calls into a function that is automatically (or automagically) called at scope exit.
- “Not everyone knows C++, and a common language is key for community-style projects”: although is this true in a certain sense, the only way it will change is if people buckle down and experiment with C++. At one point in its lifetime, every computer language was new and therefore not as well known as some others.

With those complaints out of the way, developers can now make a more educated decision as to what language should be used for a given project.

—

Q

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## UpFront

- [diff -u: What's New in Kernel Development](#)
- [Bartleby:](#)
- [cdwrite:](#)
- [Helpdesk:](#)
- [iBackup:](#)
- [LJ Index—August 2003](#)
- [Inventory:](#)
- [jigl:](#)
- [New Sputnik Access Point Achieves Orbit](#)
- [They Said It](#)
- [tvlisting:](#)

### **diff -u: What's New in Kernel Development**

**Zack Brown**

Issue #112, August 2003

**Interrupt request handlers** have changed their return values in the 2.5 tree to handle certain error conditions involving peripheral devices better. One result of this is that many, many drivers were broken at the source level and had to be fixed. For drivers included in the official kernel source, this was a repetitive, undesirable chore, but still feasible. Countless external drivers, however, can be fixed only when someone notices they no longer work; although proprietary drivers may take much longer than that. This is par for the course of the kernel development cycle, though such a large change is never undertaken lightly.

**Bartłomiej Zolnierkiewicz** is officially at the top of **IDE** maintainership. **Andre Hedrick** has (for the moment at least) stepped aside to focus on **Serial ATA (SATA)** and vendor chipset issues. **Alan Cox** remains the official liaison to **Linus** for all **IDE** patches. Andre also has decided to release all his kernel contributions retroactively under a dual license, instead of under only the **GNU General Public License**. The second license is now **Lawrence E. Rosen's Open Software License**.

**Benjamin Herrenschmidt** forked the **RADEON** Framebuffer code, when **Ani Joshi**, the official maintainer, failed to apply patches or respond to e-mails. With Ani's disappearance, Benjamin collected a variety of patches that had been floating around and released a new RadeonFB update. As of early May 2003, this driver still needs a lot of work; Benjamin's plan is to continue working with the current code base for 2.4 and try to get a complete rewrite into the 2.5 tree in spite of the feature-freeze. Some developers, like Alan Cox, have said the feature-freeze is not really in effect anymore. If so, the first attempt to stabilize 2.5 in preparation for 2.6 (or 3.0) has been a failure.

**I/O Controls** (ioctl's) have been on the way out ever since **SysFS** emerged out of the 2.5 planning discussions. In the old days, folks used to rail against ioctl functions in kernel drivers. Not only were there hordes and hordes of them, but it was impossible to know exactly how many there were, and it was therefore impossible to document all of them properly. Although there are still plenty to get rid of, more and more are migrating to the saner SysFS interface. And, some that have never been used at all, like `SCSI_IOCTL_BENCHMARK_COMMAND` and `SCSI_IOCTL_SYNC`, are being removed as part of that general cleanup. It looks as though SysFS soon will become the primary interface between userland and the kernel, replacing ioctl's and the unruly **ProcFS**.

**Greg Kroah-Hartman** has been working on a replacement for **devfs** (and `/dev`) since early 2003 and finally released an initial version of **udev** in mid-April, based on designs and ideas by **Dan Stekloff**. The `/dev` directory always has been a mess, typically including hundreds and hundreds of unnecessary files. Although **udev** is only one among a crowd of possible replacements, including **devfs** itself, it seems clear that this area of the kernel will be undergoing extensive modifications in the 2.5 time frame and the next unstable series.

**Bartleby:** [www.dahak.com/bartleby/bartleby-current.tar.gz](http://www.dahak.com/bartleby/bartleby-current.tar.gz)

**David A. Bandel**

Issue #112, August 2003

Three years ago I reviewed several good programs I still use today, such as `arping`, `SendEmail`, `SICKnotes` and `CIDR`. But the most uniquely useful one is `Bartleby`. What I like best about `Bartleby` is its flexibility and usefulness for all kinds of things not foreseen by the author. As I get older, my mind doesn't remember like it used to. But I can remember to send off a quick note, check it later and act on it. It's as simple as `bartleby "start writing linux web book"`. Yes, the script `bartleby` is mine and saves me writing `echo "message" | syslog@localhost`, but that's small peanuts. If you try this and get in the habit of using it, you'll see how valuable it is. Requires: Perl, Perl

modules DBI, CGI, DBD::Mysql or DBD::Pg, MySQL or Postgres database, MTA, Web server, Web browser.

**cdwrite:** [strony.wp.pl/wp/c\\_kruk](http://strony.wp.pl/wp/c_kruk)

**David A. Bandel**

Issue #112, August 2003

Got X? No? But there's too much to remember to burn a CD on that VT-only server with the brand new high-performance CD burner? Miss your xcdroast, gcombust, k3b, whatever? Try cdwrite. It takes the pain out of creating a data image and writing it or reading and writing audio tracks. If you have the tools installed, you don't have to keep re-reading the man pages simply to burn an image. Requires: bash, cdparanoia, cdrdao, mkisofs, cdrecord.

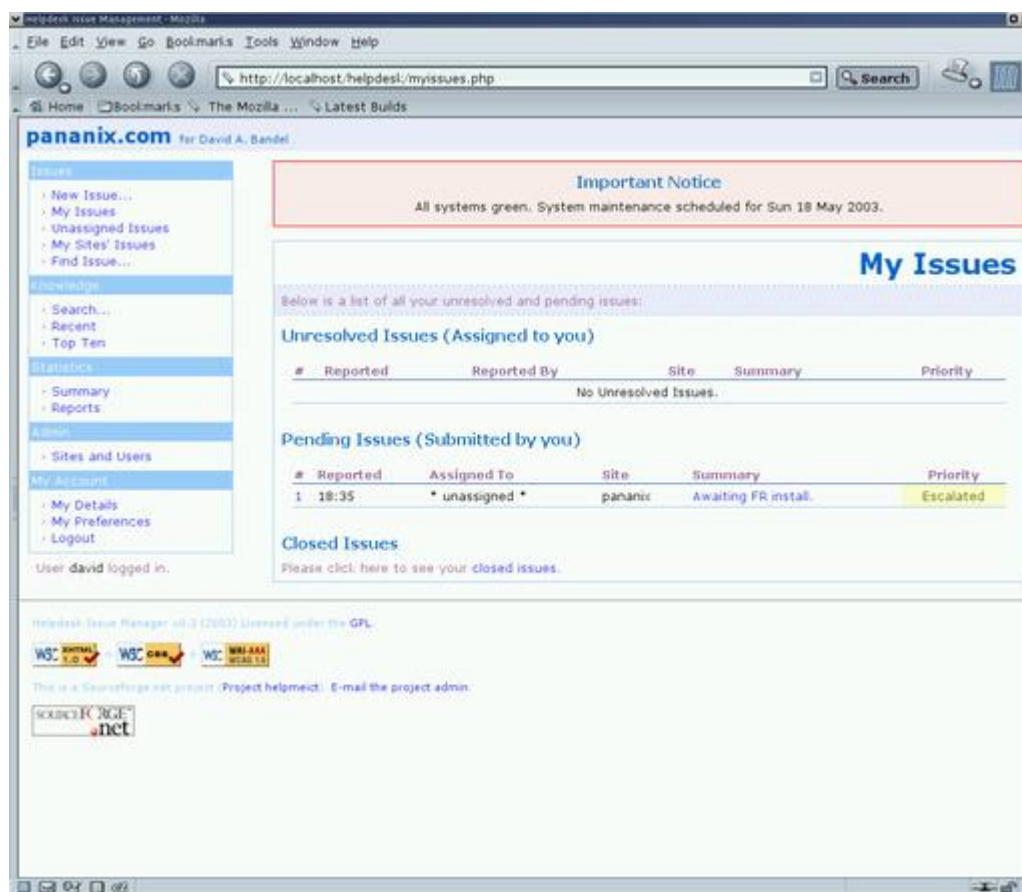


**Helpdesk:** [www.jrobst.freemove.co.uk/helpdesk.html](http://www.jrobst.freemove.co.uk/helpdesk.html)

**David A. Bandel**

Issue #112, August 2003

This Web help desk allows various levels of users, from clients to support personnel to site contacts to administrators, to track trouble tickets. The system is easy to set up and use. Each user can configure their preferences as needed. If e-mail support is desired, you'll need an MTA, such as sendmail, running on the system. Requires: Web server with PHP and PostgreSQL support, PostgreSQL server, Web browser.



**iBackup:** [www.linuks.mine.nu/ibackup](http://www.linuks.mine.nu/ibackup)

**David A. Bandel**

Issue #112, August 2003

This particular backup utility isn't meant to back up your system, rather it backs up configuration files. It creates an HTML file with all the data in it, tars it up and compresses it if requested. iBackup also can upload this file to another system for safekeeping. This makes saving and restoring critical data (like your password, group and shadow files, BIND files, etc.) quick and easy. Requires: BASH, tar, gzip, ifconfig, netstat, standard UNIX tools.

**LJ Index—August 2003**

- 1. Linux percentage range of new server operating system shipments in late 2002: 15–20
- 2. Percentage share of new OS shipments that will be Linux on Intel, or Lintel, by 2006 or 2007: 45
- 3. Current percentage rate of increase in base IT salaries: 5
- 4. Number of Linux servers that will replace SCO UNIX at branches of the Farmlands rural retail cooperative in New Zealand: 28



- 5. Number of farmers served by Farmlands: 15,000
- 6. Number of Linux server shipments to Asia-Pacific in 2002: 18,000
- 7. Sales in millions of US dollars for those servers: 58
- 8. Projected number of Linux server shipments to Asia-Pacific in 2007: 47,000
- 9. Sales in millions for those servers: 146
- 10. Minimum top number of computing nodes that will run Linux on IBM's new Blue Gene system: 65,000
- 11. Number of CPUs in a single Blue Gene chip: 32
- 12. Number of 32-CPU chips in a single Blue Gene computing node: 64
- 13. Number of nodes per Blue Gene rack: 8
- 14. Number of racks required to obtain a petaflop (quadrillion floating-point math operations per second): 64
- 15. Size in millions of dollars earmarked for the Blue Gene research initiative, announced in 2000: 100
  
- 1-3: Meta Group, Inc.
- 4, 5: *New Zealand Herald*
- 6-9: Gartner Group
- 10-15: CNet

**Inventory:** [qballsinventory.sourceforge.net](http://qballsinventory.sourceforge.net)

**David A. Bandel**

Issue #112, August 2003

It slices. It dices. You can sort it, search it, export it or import it. You create the categories and subcategories, then create the columns as any of Boolean, Integer or String. In fact, this looks like a nice utility to use for any number of applications, not only for inventory, but how you abuse it is strictly up to you. I doubt anyone will say anything if you used it as a to-do list or address book. Few things beyond pencil and paper are this flexible, and almost none are as powerful. Requires: libgtk-x11, libgdk-x11, libatk, libgdk\_pixbuf, libm, libpangoxft, libpangox, libpango, libgobject, libgmodule, libdl, libglib, libmysqlclient, glibc, libXi, libXext, libXft, libX11, libz, libXrender, libfontconfig, libfreetype, libcrypt, libnsl, libexpat.

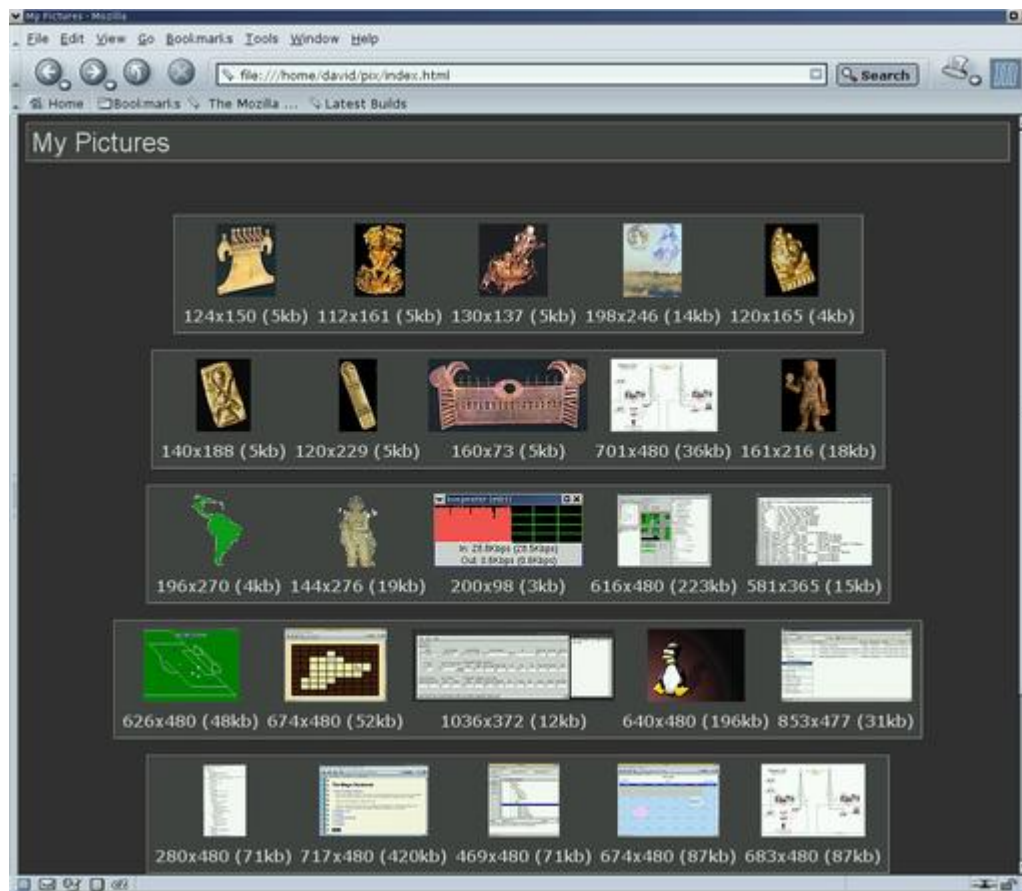
Index	Item	Title	Author	Publisher	ISBN	Cost
▷ misc	1	Forensic Pathology	DiMaio	Elsevier	0-444-01506-X	\$150.00
▼ books	2	Principles of Internal Medicine, Volume I	Harrison	McGraw-Hill	968-435-019-8	\$75.00
SCI-FI	3	Principles of Internal Medicine, Volume II	Harrison	McGraw-Hill	968-435-019-8	\$75.00
Mystery	4	Pathologic Basis of Disease	Robbins	Saunders	0-7216-7597-2	\$125.00
Computer Books						
Medical Books						
<add sub-category>						
▷ living room						
▷ dining room						
▷ master bedroom						
▷ children's rooms						
▷ computer office						
▷ medical office						
<add category>						

**jigl:** [xome.net/projects/jigl/](http://xome.net/projects/jigl/)

**David A. Bandel**

Issue #112, August 2003

This Perl script is simply too easy. You create a directory, copy all the files you want to show on a web page into it, then run `jigl.pl` from inside that directory. This gives you instant thumbnails, web pages for each thumbnail, and you can look through the slides one by one. You can customize using various templates that aren't difficult to manage. Requires: Perl, ImageMagick, jhead.



## **New Sputnik Access Point Achieves Orbit**

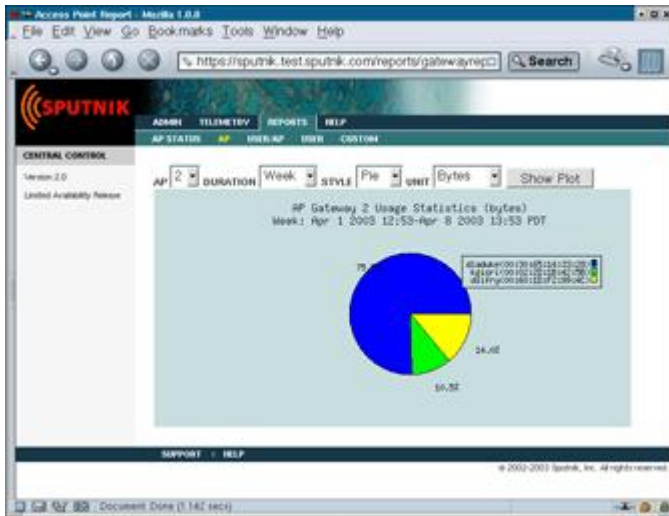
### **Doc Searls**

Issue #112, August 2003

You've got an enterprise. Your enterprise has a network. You want to go wireless, but you want to manage the unwired parts of your network as well as you manage the wired parts—maybe even better. An inexpensive answer comes in the form of Sputnik's new AP 120, an intelligent wireless access point (WAP) that lets you know who is on your wireless network and how much they're using it, through Sputnik Central Control software.



Sputnik's New AP 120



Central Control Screenshot

Sputnik, by the way, is the company started last year by David LaDuke, David Sifry and Arthur Tyde—the same guys who founded Linuxcare. Both the AP 120 and Sputnik Central Control software run on Linux and are available through the Sputnik site and OEM channels.

### They Said It

Obviously Linux owes its heritage to UNIX, but not its code. We would not, nor will not, make such a claim.

—Darl McBride, CEO, The SCO Group, August 2002 [www.linuxjournal.com/article/6293](http://www.linuxjournal.com/article/6293)

The only question I asked was “My business uses Linux exclusively. Can you guarantee that Centrino will work with the computers we use?”—and got an affirmative, in front of around ten UK tech industry journalists. Just thought you would like to know.

—Peter Spark, CEO and Founder, Ecsponet, reporting from an Intel briefing in the UK

Again, I don't understand why you all are so threatened by this, but from a careful look at the lobbyists in this room that are representing Microsoft, and all of you here representing proprietary software companies, which—let's face it, that's where the big money is, it's not in Open Source, it's in proprietary—it's rather transparent as to why you all feel so threatened by this language. And I'll tell you, this [bill] is innocuous, but next session I'll be on a crusade.

—Texas State Senator John Corona (D-Dallas) to Mario Correa, who represented the Business Software Alliance in opposition to a bill introduced by Corona that

would allow Texas to consider acquisition of open source as well as proprietary software

We have all seen many movies like *Hackers* that pass off ridiculous 3-D animated eye-candy scenes as hacking. So I was shocked to find that Trinity does it properly in *The Matrix Reloaded*. She whips out Nmap version 2.54BETA25, uses it to find a vulnerable SSH server, and then proceeds to exploit it using the SSH1 CRC32 exploit from 2001.

—Fyodor, author of Nmap, on [www.insecure.org](http://www.insecure.org)

**tvlisting:** [www.cherrynebula.net/projects/tvlisting/tvlisting.html](http://www.cherrynebula.net/projects/tvlisting/tvlisting.html)

**David A. Bandel**

Issue #112, August 2003

Want to know what's on the tube tonight? This simple Perl program shows you in any of a number of formats, although HTML is probably the easiest to read (and is the default). So, if you have free time coming up soon, you can check out the movie listings ahead of time. I think I'll check for the next episode of *FarScape*. Requires: Perl, Perl Modules HTML::TreeBuilder, HTML::Tagset, HTML::Parser, Web browser.

Indianapolis - Bright House  
Monday May 12, 2003

	30	35	40	45	50	55	00	05	10	15	20	25	30	35	40	45	50	55	00	05	10	15	20	25
	2:30 PM						3:00 PM						3:30 PM						4:00 PM					
3 DCTVI	Digital Cable TV Information																							
3 WISH	As the World Turns						Guiding Light						Access Hollywood											
4 RTTV	Paid Programming						Scooby Doo, Where Are You!						Jackie Chan Adventures						Pokemon					
6 WFPX	Paid Programming																							
7 WRVY	General Hospital						Judge Joe Brown						Judge Joe Brown						Judge Judy					
8 TheC	The Weather Channel																							
9 HSN	USA Gold						USA Gold						Focus on Beauty											
10 WNDV	700 Club						Buzz Lightyear of Star Command						Recess						Digimon: Digital Monsters					
11 WWSN	Jenny Jones																							
12 WTHR	Dr. Phil						Montel Williams						Oprah Winfrey											
	2:30 PM						3:00 PM						3:30 PM						4:00 PM					
23 WTBU	Seeking with Nancy						Charlie Rose						Lidia's Italian-American Kitchen											
14 WGN	Street Smarts						Family Matters						Fresh Prince of Bel-Air						Cosby Show					
16 WCTY	Local Access Programming																							
17 GOVACC	Government Access Programming																							
18 CSPAN2	Public Affairs																							
19 EQUACC	Educational Access Programming																							
20 WDNH	Music Channel						Music Channel						Hip-Hop Rhythm and Vibes											
21 WFTS	Reading Rainbow						Dragon Tales						Between the Lions						Clifford the Big Red Dog					
22 WHMB	Martha Stewart Living						Recipe TV						Green Acres						Family Ties					
23 TBS	Roseanne						Roseanne						Drew Carey						Drew Carey					
	2:30 PM						3:00 PM						3:30 PM						4:00 PM					
24 TNB	Star Trek: The Next Generation						Seven Days						Highlander											
25 COMEDY	Teyz																							

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## From the Editor

***Security: Yes, It's Part of Your Job***

**Don Marti**

Issue #112, August 2003

If evil doesn't get you, ignorance will. Learn what everyone needs to know about SE Linux and TCPA.

Welcome to our annual issue about necessary information technology security tools for the enterprise, I mean sinister tools of massive repression.

What's the difference? In most cases, only the use to which you put the tool. Security is a fascinating subject because it exercises both your logical, problem-solving side—what would an attacker have to compromise to get from point A to point B—and your conscience.

You've often heard that security has to be designed in, not bolted on. That makes everyone in information technology a security professional, whether it says "security" on your business card or not. And as a security professional, you have to consider security threats at two levels: the many small attacks from people who want to copy credit-card numbers, send spam and deface web sites, and the larger, slower attack from those who want to destroy our civilized way of life on the Net, with all its messy free speech, and institute a tidy regime of surveillance and "digital rights management".

Professor Lawrence Lessig, in *Code and Other Laws of Cyberspace*, makes the most powerful case for considering your beliefs and your politics when you go to work on technology. Code is law. How you build a system affects how some users of the system can regulate others. So the security you put into place to protect you from small attacks should not facilitate the one large attack on freedom itself.

It's important to let your conscience guide your technical decisions, but it's just as important to back up your political positions with the facts about the

technologies to which they apply. Proposals for “trusted computing” are the subject of justifiable concern among freedom lovers. Nobody wants to give up the PC for a sealed box with a so-called Fritz chip, named after authoritarian US Senator Ernest “Fritz” Hollings, that would prevent you from running a free operating system or recording your own music.

But Fritz chip hysteria is sometimes misdirected at new technologies or proposed specifications that wouldn't take away your freedom to run the software of your choice and might even have some beneficial applications. Is the Trusted Computing Platform Alliance unfairly maligned? Read the article on TCPA by David Safford, Jeff Kravitz and Leendert van Doorn on page 50, then get their free TCPA code and decide for yourself.

You can give a big boost to your personal information security by encrypting your home directory. Making it work seamlessly is tricky, though, and Mike Petullo addresses the hard parts head-on on page 62.

The US National Security Agency's SE Linux is one of the hottest topics in security today, and Faye Coker gives us an introduction in Kernel Korner on page 20. Russell Coker follows up on page 56 with a report on what happens if you give out the root password—can the SE Linux rules alone protect the system?

Daniel R. Allen has written a helpful article on one of the most common Linux security tools, OpenSSH, and Mick Bauer continues his series on OpenLDAP, a multifunctional directory service. There's plenty of thought-provoking information this issue, so stay informed and, in the immortal words of the Google employee handbook, “Don't be evil.”

Don Marti is editor in chief of *Linux Journal*.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.



[Advanced search](#)

## On the Web

### *Make the Most of Your Summer*

**Heather Mead**

Issue #112, August 2003

There won't be a decent thing on TV until the fall season starts, and probably not even then. So take on a project.

The sun is out, the temperature is steadily rising, pale legs find themselves thrust into the glare of day and multiplexes are showing 12 screens' worth of sequels—must be summer. Remember being a kid, when summer meant doing things you never had the time to do when school was in session? These days, few of us have the luxury of summers off, but that doesn't mean we can't do something adventurous this August. So stop mourning Buffy, quit complaining about *Reloaded* and do something you've wanted to do but didn't have the time for: build your own workstation, put a Linux installation on your laptop that actually works, transform a spare computer into a server. And to help you with these projects, the *Linux Journal* web site offers the following articles and tutorials.

First off, if you've been following Jay Docherty's Linux on the laptop series, he wrapped it up with "Polishing Your Linux Laptop Setup" ([www.linuxjournal.com/article/6891](http://www.linuxjournal.com/article/6891)). This article discusses how to go wireless, how to install ALSA sound support and how to set up the ACPI power management component. Jay admits "ACPI can be a beast to set up", but it can reduce the clock speed when the laptop is idle to increase battery life and control the system's fans for thermal protection. All in all, if you're wanting to put Linux on your laptop, Jay's complete series is worth a read.

In "High Availability Linux with Software RAID" ([www.linuxjournal.com/article/6412](http://www.linuxjournal.com/article/6412)), Micah Silverman describes how he recycled a system to create an HA server. He explains how to use "software RAID Level 5 under a fresh installation of Red Hat 8.0" and how to test the fault tolerance of the RAID. Before going live with this setup, Micah built a testbed by using VMware "to set up a Linux virtual

machine with six 9GB SCSI drives...on a machine with only one real physical IDE drive”.

If you've been wanting to build your dream Linux machine, you might want to check out Glenn Stone's weekly articles about the components he's thinking of including in this year's Ultimate Linux Box. “Getting Serial: the Ultimate Linux Box S-ATA Disk Subsystem” ([www.linuxjournal.com/article/6902](http://www.linuxjournal.com/article/6902)) weighs the benefits of using serial ATA drivers against their higher cost compared to traditional ATA controllers. Specifically, Glenn discusses the performance of 3ware's 8500-4 serial ATA card. When used in our testbed machine, Glenn found that serial ATA offers “some fairly serious bang” for not too many more bucks. Glenn has also been thinking about what sort of video card the “Ultimate” machine should have (“Some Graphic Remarks: VGA for the Ultimate Linux Box”, [www.linuxjournal.com/article/6922](http://www.linuxjournal.com/article/6922)). To that end, ATI's new Fire GL X1 workstation card may be just the card to get us closer to the dream. This card has dual DVI-I digital/analog outputs that each have a DVI-to-VGA adapter, so you can use a standard monitor instead of a digital one.

These are only a few of the projects you can take on with the help of the *Linux Journal* web site. Search the site for other ideas; new projects are posted every week. If you'd like to share your own project, send the proposal to [info@linuxjournal.com](mailto:info@linuxjournal.com).

Heather Mead is senior editor of *Linux Journal*.

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.

[Advanced search](#)

## Best of Technical Support

### How to FTP through an iptables Firewall

I've been trying to configure iptables to work properly with incoming SSH and FTP. For some reason, every time I want to FTP from a remote site, I have to disable the POLICY for the INPUT chain. Can you explain how to deal with this issue—configuring FTP and iptables together without having to disable the policy? I'm running Red Hat 8.0.

—

Ramiro Albarracin

[ramiro@sergiolub.com](mailto:ramiro@sergiolub.com)

Without having your list of rules it is difficult to find the problem, but clearly some of the rules (in the INPUT chain) are preventing the traffic. Try adding LOG rules before each actual rule (in /var/log/messages) to see which one is causing the packets to stop. For example:

```
iptables -A INPUT -p TCP -s 0/0 -d 0/0 \
--dport ftp -j LOG --log-prefix "FTP :"  
iptables -A INPUT -p TCP -s 0/0 -d 0/0 \
--dport ftp -j ACCEPT
```

—

Mario Bittencourt

[mneto@argo.com.br](mailto:mneto@argo.com.br)

You should read up on firewalling and FTP. Basically, FTP is a hard protocol to filter, and actually it's two protocols in one, depending on the client. Active FTP is not too hard to filter on the server side; you simply need to allow incoming connections on port 21 (the control connection). For passive FTP, however, the server doesn't open the data connection to the client; the client opens the data connection to you on some high TCP port (>1024). With iptables, you can make

use of connection tracking, which opens only the one port used for that FTP connection:

```
iptables -A $IF -p tcp --dport ftp -j ACCEPT
iptables -A $IF -p tcp --dport 1024:65535 \
-m state --state RELATED -j ACCEPT
```

You also have to load the `ip_conntrack_ftp` module for the above to work (`modprobe ip_conntrack_ftp`).

—

Marc Merlin

[marc\\_bts@google.com](mailto:marc_bts@google.com)

#### **Synchronize Your Watches**

How can I manually time synchronize my computer? When I install my distribution, Mandrake 9.0, it lets me choose an NTP source, but I don't leave my machine powered on all the time. How can I manually sync to be sure its happening?

—

Rick Shores

[rshores@ispwest.com](mailto:rshores@ispwest.com)

Simply run `ntpdate timeserver`. This command synchronizes your time to the time server and also reports how far off your clock was. You probably should follow this by saving the time to your hardware clock to preserve it if you reboot: `hwclock --systohc`.

—

Chad Robinson

[crobinson@rfgonline.com](mailto:crobinson@rfgonline.com)

#### **Dual-Boot System Skips LILO Menu**

I had Red Hat 7.1 installed on my PC, with another partition used for Microsoft Windows. I recently re-installed Windows using `mssetup`. When the system reboots I am not being asked whether to switch to Windows or Linux. Now the system starts up directly in Windows. Is there some way to restore Linux?

—

Kunal S Doddanavar

[kunal\\_s\\_d@indiatimes.com](mailto:kunal_s_d@indiatimes.com)

Windows removed or disabled the Linux bootloader, which is LILO on Red Hat 7.1. Boot with your rescue floppy, mount your Linux root partition with, for example, `mount /dev/hda1 /mnt` and run `lilo -R /mnt` before rebooting. If you were running GRUB, `grub-install` should do the trick.

—

Marc Merlin

[marc\\_bts@google.com](mailto:marc_bts@google.com)

On newer Red Hat distributions that use the GRUB bootloader, boot from the rescue floppy and re-install GRUB with `grub-install`. If you didn't make a boot disk, boot with the first install CD in rescue mode.

—

Christopher Wingert

[cwingert@qualcomm.com](mailto:cwingert@qualcomm.com)

#### **Cleaning Up Old Kernels**

I am using Red Hat Network to upgrade my software and keep it current. I have allowed the up2date program to include my kernel. Now my /boot partition is getting too full. How do I remove some of the old kernels? I really don't think I need five different kernels in /boot.

—

Bob Wooden

[bobwooden@netwalk.com](mailto:bobwooden@netwalk.com)

Simply remove the undesired boot images. You could run `rpm -qa | grep kernel` to find which kernel packages you have installed, and use `rpm -e` to remove the older ones. As a suggestion, keep at least two options, so that if something goes wrong with the current one you have a backup.

—

Mario Bittencourt

[mneto@argo.com.br](mailto:mneto@argo.com.br)

This is not only okay, it is a good administration habit. You should keep only useful kernels around, and generally only two are required: the primary kernel file and a backup in case something happens to the primary. Saving as many versions as you have is rarely necessary unless you have special requirements, such as if you are developing and testing kernel drivers.

—

Chad Robinson

[crobinson@rfgonline.com](mailto:crobinson@rfgonline.com)

**USB Flash Drive?**

How do I mount a USB flash drive? I can see my flash drive when I check `/proc/bus/usb/devices/`. When I run the hardware browser, it shows up as `hda4` (fat32), but I can't mount it or access the files.

—

Callum Benepe

[callumb@yahoo.com](mailto:callumb@yahoo.com)

It looks like you do not have the `usb-storage` driver loaded, which is needed for this device. Take a look at the Linux USB Guide at [www.linux-usb.org](http://www.linux-usb.org) for more information on how to load the proper drivers and mount the device.

—

Greg Kroah-Hartman

[greg@kroah.com](mailto:greg@kroah.com)

**Support for Intel Video?**

My video card is a built-in Intel 82845G/GL that fails with Linux (Red Hat 8.0). Linux probes it during installation but fails to start up in graphic mode; `startx` shows a fatal error.

—

Jafar Borhan

[jafar\\_borhan@yahoo.com](mailto:jafar_borhan@yahoo.com)

Searching on Google, I found a page on how to configure a system with this video card, [www.linuxcare.com/labs/certs/ibm/netvista-m42/rh80-config.epl](http://www.linuxcare.com/labs/certs/ibm/netvista-m42/rh80-config.epl). Upgrade the listed packages, then run Xconfigurator.

—

Marc Merlin

[marc\\_bts@google.com](mailto:marc_bts@google.com)

**Connections Time Out**

Telnet and SSH connections seem to time out and I get disconnected. I use tcsh for my shell, and the pty device I am logged in on is listed in /etc/securetty. This is not an issue with autologout. Even if I disable autologout, the connection still is dropped after about an hour. When this happens, the user still is listed as being logged in and the shell still is active. It has to be terminated by killing its process ID.

—

Floyd Miller

[floyd@studiodust.org](mailto:floyd@studiodust.org)

This smells of a firewall-level issue. In common NAT and masquerading setups, if there is no traffic on a link for some time the router will forget about the connection, assuming it was closed improperly. This is because some clients do not issue closure requests correctly, and it would be unwise to allow these stale connections to continue to tie up kernel resources.

—

Chad Robinson

[crobinson@rfgonline.com](mailto:crobinson@rfgonline.com)

You may be going through a NAT gateway that expires idle TCP connections after one hour of inactivity. Try (as root):

```
echo 600 > /proc/sys/net/ipv4/tcp_keepalive_time
```

Then, when you use SSH, you should ask for keepalive TCP packets to keep the connection up:

```
ssh -o 'KeepAlive=yes' targethost
```

—

Marc Merlin

[marc\\_bts@google.com](mailto:marc_bts@google.com)

You can save typing and put:

```
ProtocolKeepAlives 300
```

in `~/.ssh/config` to make SSH send keepalive packets for all connections every five minutes.

—

Don Marti

[info@linuxjournal.com](mailto:info@linuxjournal.com)

[Archive Index](#) [Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.



[Advanced search](#)

## New Products

### **SnapGear Embedded Linux**

A new free embedded Linux distribution is available from SnapGear, Inc. SnapGear Embedded offers support both for microprocessors lacking MMUs, such as ColdFire, ARM and SPARC, and those with MMUs, including SuperH, XScale and x86. Based on SnapGear's work maintaining  $\mu$ Clinux patches, this distribution includes toolchains, API standardizations and library support for a single executable and source collection. Available as a free download on the SnapGear Embedded web site, it also is available for a fee in CD-ROM form.

SnapGear, Inc., 7984 South Welby Park Drive #101, West Jordan, Utah 84088, 801-282-8492, [www.snapgear.com](http://www.snapgear.com) (company site), [www.snapgear.org](http://www.snapgear.org) (downloads).



### **SmartFLeX SFT-CXC Network Terminal**

Based on SmartFLeX Technology's embedded Flash Linux system, the SFT-CXC is a dual-mode network terminal that supports operations as both a character and X terminal. In character terminal mode, the SFT-CXC can have up to five different simultaneous sessions in full-screen mode over Ethernet or a serial connection. In X mode, the client provides one XDMCP session to a network host system. Shape extensions are included to enable compatibility with window managers. Remote management of the SFT-CXC system settings is available through a browser.

SmartFLeX Technology, Inc., 623 Selvaggio Drive, Suite 220, Nazareth, Pennsylvania 18064, 610-746-2390, [www.smartflextech.com](http://www.smartflextech.com).



### **Address Object for Linux**

Address Object for Linux is software that allows programmers to add address verification and routines to custom PC or web applications. Addresses are verified in batch applications or in real time by comparing the submitted address to a time zone, congressional district or county. Latitude and longitude coordinates are returned as well. Address Object uses shared object technology to provide easy installation on existing hardware components. It can operate in

any environment running on an x86 platform and is CASS-certified by the US Postal Service.

Melissa Data Corporation, 22382 Avenida Empresa, Rancho Santa Margarita, California 92688, 949-589-5200, [www.melissadata.com](http://www.melissadata.com).

### **ATG 6**

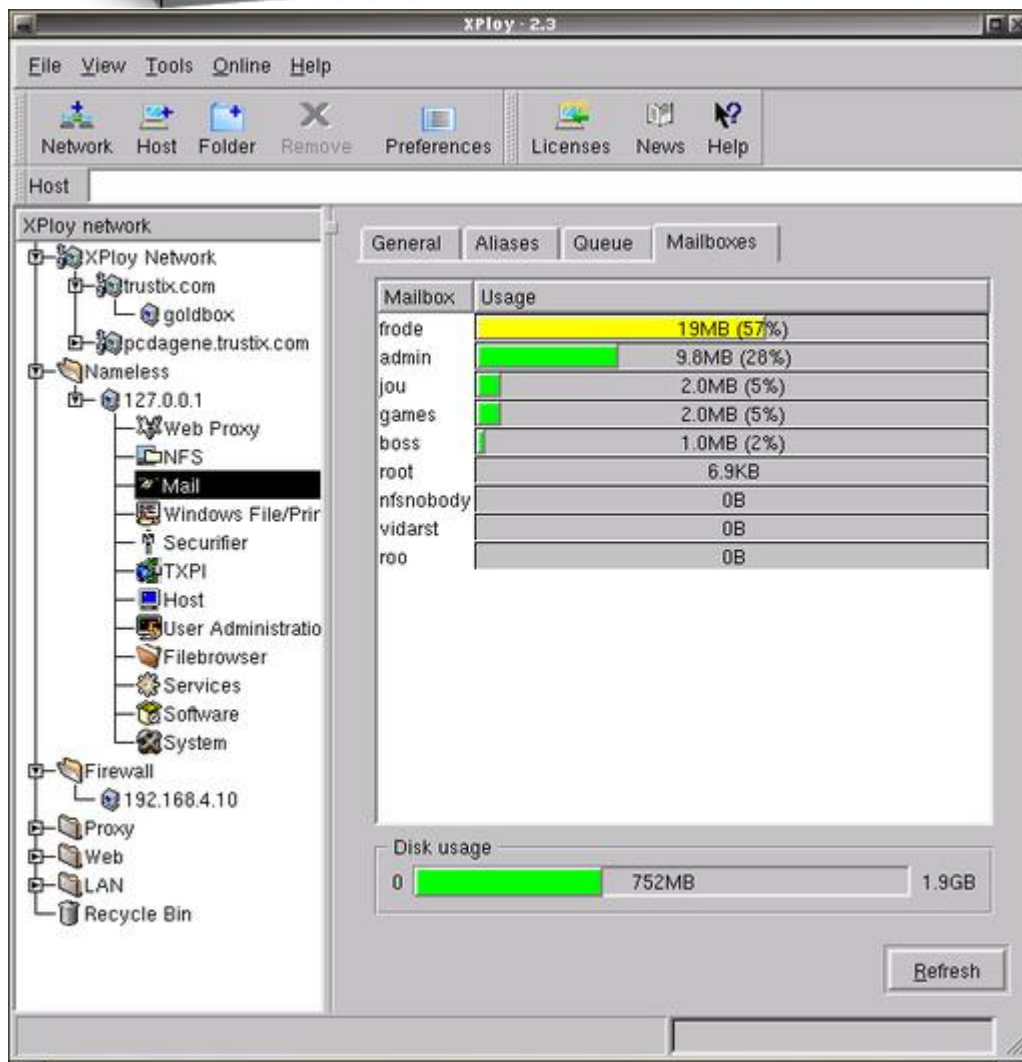
Offering support for Red Hat Advanced Server 2.1, ATG 6 builds on-line commerce and self-service applications, in addition to managing business transactions and relationships. Functions include self-service order entry, account administration and customer-end tasks such as product comparisons, gift registration and express checkouts. Automation tools direct the logical workflow of projects and automate entire sequences of interactions. Interconnected modules handle publishing, search, analytics, payments and fraud protection duties. ATG integrators are provided to connect ATG 6 with various existing ERP and CRM systems.

ATG, 25 First Street, Second Floor, Cambridge, Massachusetts 02141, 617-386-1000, [www.atg.com](http://www.atg.com).

### **Trustix Small Office Server**

Trustix has released the Trustix Small Office Server, designed for environments of up to 25 networked users and upgradable to 50 users. Small Office Server includes the Trustix distribution and provides Web, mail, proxy and LAN server capabilities. It can be installed on existing hardware or pre-installed on IBM xSeries hardware. RAV antivirus and antispam software is included, as is the NetVault backup and restore application. Small Office Server supports centralized storage for user files, network caching and a centralized logon.

Trustix, 4819 Emperor Boulevard, 4th Floor, Durham, North Carolina 27703, 919-313-4599, [www.trustix.com](http://www.trustix.com).



### 3DBOXX M4 Opteron Workstations

BOXX Technologies announced a new line of 3-D rendering workstations based on dual Opteron processors; machines built on the 240, 242 and 244 processors are now available. M4 workstations use NVIDIA Quadro architecture for modeling and rendering 3-D content and animation with Maya, 3ds max,

SOFTIMAGE XSI, LightWave 3D and Houdini. The standard workstation includes the AMD-8111 HyperTransport PCI tunnel, the AMD-8151 HyperTransport AGP tunnel, 128-bit dual-channel memory bus, up to 8GB ECC registered 333MHz DDR, four DIMM slots, dual channel UltraDMA 133 IDE controller and six channel audio. Custom-configured workstations also are available. The workstations have lightweight aluminum chassis for heat dissipation, and two 92mm fans provide airflow.

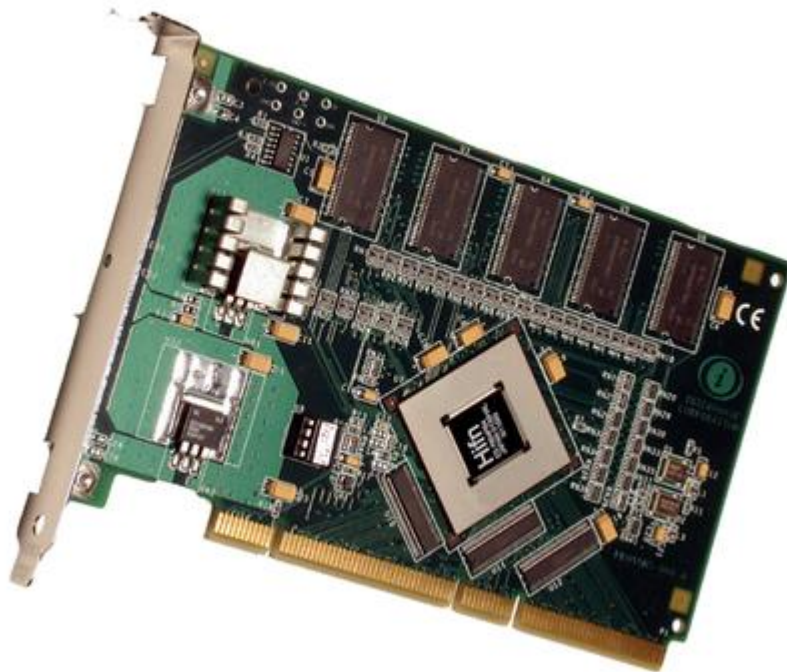
BOXX Technologies, Inc., 10435 South Burnet Road, Suite 120, Austin, Texas 78758, 877-877-2699, [www.boxxtech.com](http://www.boxxtech.com).



#### **Interphase IPsec Accelerator Cards**

The first products in Interphase's new network security product line are the 45NS (PMC) and 55NS (PCI) network security acceleration adapters. Designed to eliminate traffic bottlenecks caused by VPNs, gateways, routers and firewalls, the accelerators off-load bandwidth-intensive IPsec processing from the host CPU. The accelerators handle header analysis, payload extraction, compression, encryption, authentication and packet assembly. Both adapters offer 500Mbps 3DES throughput and accelerate DES, MD5, SHA-1, RC4 and AES security algorithms. They also offer a 64-bit 66MHz PCI bus, 64MB of private memory and support for full duplex OC-3 rates and 512K simultaneous sessions.

Interphase, Parkway Centre, Phase 1, 2901 North Dallas Parkway, Suite 200, Plano, Texas 75093, 800-327-8638, [www.interphase.com](http://www.interphase.com).



[Archive Index Issue Table of Contents](#)

[Advanced search](#)

Copyright © 1994 - 2019 *Linux Journal*. All rights reserved.