Keeping Your Data Yours: Using Encryption to Protect Entire Filesystems

Presented by
Thomas J. Munn
and
Will Stevenson

The Problem: Employers and Data

- Employers will very quickly eliminate employees, giving them little or no time to retrieve data
- Employers will not allow employees to take data that is theirs home.
- Using a USB removable hard disk, data stays with you
- Encrypting that data ensures that it won't fall into the wrong hands

Using LINUX to protect data

- Using LOOP-AES kernel module (not a patch!) allows LINUX to encrypt at the device level
- LOOP AES uses upto 256 bit symmetric encryption
- Combining LOOP-Aes with a removable USB hard disk adds even greater security
- Adding GNUPG keys on a keyring USB token enhances this further

Installing Loop-AES

- Obtain LOOP-AES sources from sourceforge
- Compile your most recent kernel (they MUST be the same for things to work without kludges)
- Get the UTIL-LINUX collection, and put in LOOP-AES directory

Installing Loop AES

- Patch UTIL-LINUX
  - patch -p0 < patchfile
  - .patch must be done one level above util-linux directory.
  - Follow readme, it tells you EXACTLY what to do!

Installing Loop AES

- Gotchyas:
  - Loop-AES requires that loopback support NOT be put in kernel
  - Make sure that you are using the correct kernel version BEFORE compiling
  - You must be root to install kernel modules
  - You must have done a “make menuconfig” BEFORE compiling loop-aes

Installing USB hard disk

- You must compile USB in the kernel as a MODULE!
- You must compile in SCSI support (as a module)
- You must compile in SCSI generic, Disk modules
- You must compile in USB-uhci (module!)
- You must compile in USB-storage
- Use a more recent kernel >=2.4.18
- For usb 2.0 you might want to try 2.5

Installing USB (continued)

- Load modules in the following order:
- scsi, scsi-generic, scsi-disk, usb, usb-uhci, and usb-storage
- See website for script to load modules
- Devices are “hot” plugabble
- USB 2.0 makes things a whole lot faster (from 1mb/s to around 10/mb/s)

Getting the filesystem made

- Follow the readme that comes with Loop-aes:
  - use losetup to setup device, commands in readme, under "setting up filesystem using GNUPG
- Format the filesystem using ext3 (so if you have to remove things, its safe!)
- manually mount the filesystem to test it

General Architecture

- Keep all scripts and modules on a separate, removable hard disk
- Keep GNUPG keys on a usb keychain
- Don’t forget to backup USB disk
- Mount keychain on /.keys
- Mount USB hard disk on /home/user
- Create a script to automatically mount disk
- Erase root’s bash_history!

Hardware Sources

- USB hard disks:
- 30 GB $150.00
- www.basoncomputers.Com
- Keychains:
  - google usb keychain
  - $50.00 for 16mb
- You might want to get a USB hub
- You might want to get a supported USB 2.0 controller (MUCH faster!)

Other possibilities

- Since filesystem is encrypted at device level, you can put stuff on NFS drives and have contents be secure from eavesdropping.
- You could GAK! Put it on a smb filesystem, and the worst thing that could happen is someone could delete it.
- E.g. put on terrapin mine.

Other possibilities

- Floppies
- ZIP disks
- Jazz disks (avoid, they die a lot!)
- Encrypted ROOT filesystems
- Encrypted SWAP

- View readme file inside of loop-aes for even more ideas!

My “Load” script

- cd /home/tjmunn/.loop
- insmod -f loop.o
- mount /dev/sda3 /.keys
- #created symlink from .gnupg in root's directory to external disk link
- cd /root/.gnupg
- gpg --decrypt < keyfile.asc | mount -p 0 -t ext3 /dev/sda2 /home -o loop=/dev/loop0,encryption=AES192

Creating the Initial Loop Device

- This is covered in the readme!!!
- head -c 45 /dev/random | uuencode -m - | head -2 | tail -1 | gpg -e -a -r ”Email/userid of key!" > keyfile.asc
- gpg --decrypt < keyfile.asc|losetup -p 0 -e AES192 /dev/loop3 /dev/sda3 (partition to destroy!)
- mkfs -t -j ext2 /dev/loop3
- losetup -d /dev/loop3