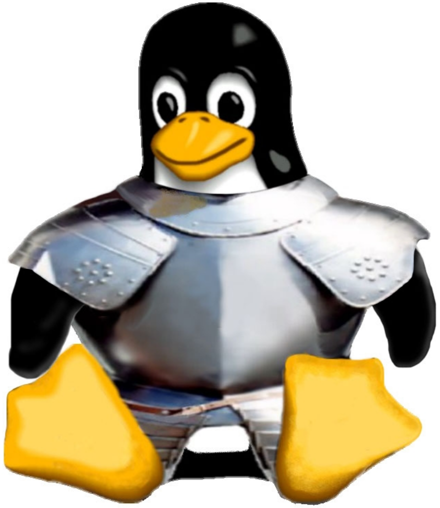
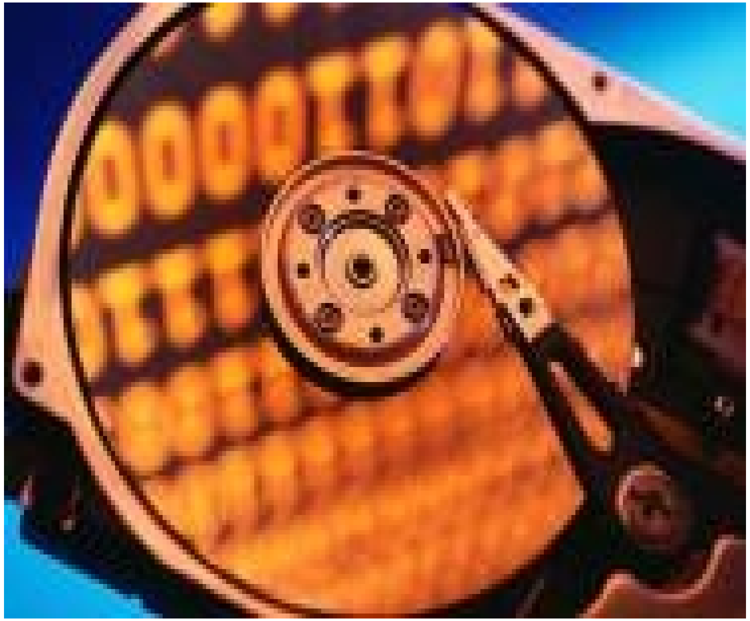
Open Source Encrypted Filesystems for Free Unix Systems





George Kargiotakis

kargig@noc.uoi.gr

Introduction

- Users seek ways to secure their data with maximum comfort and minimum requirements.
- No application specific encryption is wanted.
- Performance considerations.
- Encryption can be more secure than physical security.
- Protection of stolen equipment.
- Lifetime of protected data could be years (backups).

Presentation Topics

- Threat Models
- Linux Solutions: *CryptoAPI, StegFS, CFS, PPDD, CryptFS, TCFS*
- BSD Solutions: *TCFS, OpenBSD Encrypted Virtual Filesystem*

Threat Models

- The theft of the computer while it is powered off or if the thief has to power it off to remove it.
- The theft or copying of the discs from the computer.
- The theft or copying of backups.
- Copying of discs after booting the computer from a boot floppy.

Not all presented solutions cover ALL these threat models.

Linux Solutions

- CryptoAPI – Loop-AES
- StegFS
- PPDD (Practical Privacy Disk Driver)
- CFS
- CryptFS
- TCFS

CryptoAPI -- Loop-AES

(Loopback Encrypted Filesystems)

- **Usable Ciphers**:
  - **loop-AES**: AES
  - **CryptoAPI**: XOR, DES, twofish, blowfish, cast128, serpent, MARS, RC6, DFC, and IDEA.
- **Procedure of Installation:**
  - **Kernel Patching**

    make patchkernel KDIR=<kernel source dir> LOOP=iv (or LOOP=jari provided with loop-AES)

    make modules; make modules_install
  - **Loading of cryptoloop device:** *modprobe cryptoloop*
  - **Creation of 100Mb file:** *dd if=/dev/urandom of=/home/kargig/cryptfile bs=1M count=100*

- **Loading of desired cipher:** *modprobe twofish*
- **Loading and encrypting the file:** *losetup -e twofish /dev/loop0 /home/kargig/cryptfile*
- **Formating and mounting the new device:** *mke2fs -j /dev/loop0 ; mount -t ext3 /dev/loop0 /mnt*
- **Unmounting and securing the device:** *umount /dev/loop0; losetup -d /dev/loop0*
- **Positive**:
  - Very easy to install and use.
  - Relatively fast based on the selected algorithm.
  - Can encrypt whole filesystems like /home (but not the booting device!!!
- **Negative**:
  - Once mounted anyone with access on the dir can read the files.
  - Encryption on whole devices is trivial.

StegFS

- **Usable Ciphers:** AES/Rijndael (default), Serpent, Twofish and MARS
- **Procedure of Installation and Usage:**
  - **Patching the kernel creating new modules**

```
make patch ; make patch LINUX=/path/to/kernel-source ; patch -p1 <
   /path/to/patch ; make modules; make modules_install
```

  - **Create a filesystem and turn it to a StegFS partition.**

```
mke2fs /dev/device ; mkstegfs /dev/device /path/to/btab
```

  - **Mount the partition:**

```
mount /dev/device /mnt/mntpoint -t stegfs -o btab=/path/to/btab
```

  - **Open N security levels:** *stegfsopen /mnt/mntpoint N*
  - **Close N security levels:** *stegfsclose /mnt/mntpoint N*

StegFS

- **Positive:**
  - Various levels of security.
  - An attacker cannot even see the existence of more levels than he has already acquired.
- **Negative:**
  - Speed.
  - Waste of Space.

CFS

- **Usable Ciphers**: Older versions DES running in CBC mode. Newer versions use Blowfish.
- **Procedure of Installation:**
  - **Compiling sources and copying files to** */usr/local/sbin* **with ownership** *root:wheel* **and accessmode** *551*
  - **Creation of** */.cfsfs* **dir with ownership** *root:root* **and accessmode** *000*
  - **Creation of** */securefs* **dir.**
  - **Starting the daemon and mounting the filesystem:**

```
/usr/local/sbin/cfsd > /dev/nulll
/bin/mount -o port=3049,intr localhost:/.cfsfs /securefs
```

CFS

- **Creation of CFS protected dir:**

  cmkdir secret

- **To make it readable we have to attach it:**

  cattach secret MYSecret

  /securefs/MYSecret **Will appear.**

- **To secure the dir:**

  cdetach MYSecret

- **Positive:**
  - No need for system modifications.
- **Negative:**
  - Lack of speed

PPDD

- **Usable Ciphers:** Blowfish
- **Procedure of Installation and Usage:**
  - **Patching the kernel and rebooting from the new one**
  - **Compiling the sources and making the necessary devices.**

    Make; make devices; make install
  - **Create a filesystem.**

    ppddinit /dev/ppdd0 /dev/XXXX (where XXXX is a partition eg. hdc1)
  - **Setup the device:**

    ppddsetup -s /dev/ppdd0
  - **Create a new filesystem:** *mke2fs /dev/ppdd0*
  - **Mount it where we want:** *mount /dev/ppdd0 /home/kargig/crypto*

PPDD

- **To unmount and secure the filesystem:**

```
mount /dev/ppdd0 ; ppddsetup -d /dev/ppdd0
```

- **Positive:**
  - Ease of use.
  - Possibility to use without kernel modifications.
  - Secure backups
  - Support for read-only media
  - PGP support
  - Support for data integrity using MD5 hashes
  - Possibility for encryption of the root partition
- **Negative:**
  - Not so strong algorithm
  - Block size of the filesystem is locked to 1024

CryptFS

- **CryptFS operates by "encapsulating" a client file system with a layer of encryption transparent to the user.**
- **Cipher: Blowfish**
- **2 working modes (UID – UID+PID checking)**
- **Performance**
- **Longer Passphrases**
- **Encrypted filenames**
- **Secured even from root user.**

TCFS

- **Usable Ciphers**: 3DES,RC5, Blowfish.
- **Procedure of Installation:**
  - **Kernel and sources recompilation**

    option TCFS
  - **Directory Creation**

    ```
    mkdir /crypto; mkdir /mnt/tcfs ; mkdir /crypto/kargig
    chown kargig:wheel /crypto/kargig ; chmod 700 /crypto/kargig
    ```
  - **/etc/fstab modification**

    ```
    /crypto /mnt/tcfs tcfs rw,label=crypto, cipher=2
    0=3DES 1=RC5 2=Blowfish
    ```
  - **Mount the device**

    ```
    mount /crypto
    ```

TCFS

- **Creation of user and keys:**

```
tcfsmgr adduser
tcfsuse genkey
```

- **Using the Filesystem:**

```
tcfsuse putkey -f crypto
```

- **Setting the X flag to a dir and testing the filesystem:**

```
tcfsuse flags +x /mnt/tcfs/kargig
cp ./foo.txt /mnt/tcfs/kargig
cat /mnt/tcfs/kargig/foo.txt (we see clear output)
umount /crypto
cat /crypto/kargig/foo.txt  (we se garbage)
```

OpenBSD Encrypted Virtual Filesystem

- **Usable Ciphers**: Blowfish.
- **Procedure of Installation:**
  - **Creation of a file**

    ```
    dd if=/dev/urandom of=/home/kargig/cryptfile bs=1024 count=100000
    ```

  - **Association of cryptfile with a svnd device**

    ```
    vnconfig -ck -v /dev/svnd0c /home/kargig/cryptfile
    ```

  - **Creation of new filesystem**

    ```
    newfs /dev/svnd0c
    ```

  - **Mount the new filesystem**

    ```
    mount /dev/svnd0c /home/kargig/secrets
    ```

OpenBSD Encrypted Virtual Filesystem

- **Unmounting and securing the filesystem:**

  umount /dev/svnd0c

  /usr/sbin/vnconfig -u -v /dev/svnd0c

- **Positive:**
  - ➢Ease of use.
  - ➢Performance.
- **Negative:**
  - ➢Size Limit.

Conclusion

- In most encrypted filesystems a major problem appears with multi-user environments.
- Security of a system is as strong as it's weekest link.
- Choose an encryption scheme according to the current needs.
- Other Problems incude:
  - Filesystem damage
  - Data integrity checking
  - Low Performance