# Wir verschlüsseln Dateisysteme
# Wer macht mit?

**packet@berlin.ccc.de**



`Id: cryptfs-eh02-workshop.tex,v 1.17 2002/03/28 22:40:38 packet Exp`

# Workshop-Verlauf

# Warum Dateisysteme verschlüsseln?

## Szenario 1: Laptop-Benutzer

- Laptops sind manchmal *mobiler* als man es gerne hätte – Daten darauf unsicher
- Laptops tendieren dazu – gerade bei Nerds – häufiger als andere Rechner benutzt zu werden, viele *wichtige* Daten, wie z.B. SSH- oder auch PGP-Keys lagern dort

## Szenario 2: privater Arbeitsrechner

- Auch zu Hause lagern Daten, die nicht jeder jederzeit lesen können soll
- Bei Bedarf einfach schnell die Sicherung rausziehen und wissen, dass Fremde mit den Festplatteninhalten der Rechner nichts anfangen können kann auch beruhigend sein

# Was verschlüsselte Dateisysteme bringen

- im Idealfall kann im ungemounteten Zustand niemand sagen, welche Daten sich im Dateisystem befinden
- im gemounteten Zustand verhindern nur die normalen Filepermissions den Zugriff Unbefugter
- Daten sind nicht vor Modifikation oder Zerstörung geschützt (s. a. http://www.off.net/~jme/loopdev_vul.html), auch dazu müssen die Filepermissions genutzt werden

# Verschlüsselte Dateisysteme unter Linux

**CFS:** verschlüsselnder user-space NFS-Server (http://www.ibiblio.org/pub/Linux/docs/faqs/security/Cryptographic-File-System)

**TCFS:** verschlüsselndes Netzwerk-Dateisystem (http://www.tcfs.it/)

**StegFS (2.2.x):** steganographisches Dateisystem (http://www.mcdonald.org.uk/StegFS/)

**International Kernel Patch (2.2.x/2.4.x, alt):** bei 2.4.x unbrauchbar, bietet viele versch. Cipher- und Digest-Algorithmen (http://www.kerneli.org/)

**CryptoAPI (2.4.16+):** Nachfolger des *alten* International Kernel Patches (http://cryptoapi.sourceforge.net/)

**International Kernel Patch (2.4.16+, noch testing):** Nachfolger der CryptoAPI, meine Wahl für diesen Workshop (URL s. CryptoAPI)

**LoopAES:** mit AES verschlüsselte loopback devices, *kleiner* Kernel-Patch (http://loop-aes.sourceforge.net/)

# Aufsetzen der Software

Vor dem Erstellen des verschlüsselten Dateisystems sind noch folgende Schritte notwendig:

1. Kernel patchen, konfigurieren und compilen
2. `mount` und `losetup` aus util-linux patchen, compilen, testen und installieren
3. den neuen Kernel booten

# Kernel

## Benötigte Sourcen

- Kernel-Source: http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.18.tar.bz2
- Patch für das loopback device:
  http://www.kernel.org/pub/linux/kernel/people/hvr/testing/loop-jari-2.4.16.0.patch
- International Kernel Patch:
  http://www.kernel.org/pub/linux/kernel/people/hvr/testing/patch-int-2.4.18.1.bz2

## Kernel Bauen

1. PGP Signatures der Sourcen (soweit verfügbar) verifizieren
2. im entpackten Kernel-Tree dann die Patches mit `patch` applien
3. CONFIG_BLK_DEV_LOOP, CONFIG_CRYPTO, CONFIG_CIPHERS, CONFIG_CRYPTOLOOP und mind. einen Cipher auswählen
4. Kernel bauen und booten

# `mount` und `losetup`

## Benötigte Sourcen

- util-linux Source:
  http://www.kernel.org/pub/linux/utils/util-linux/util-linux-2.11n.tar.bz2
- util-linux Patch: http://www.kernel.org/pub/linux/kernel/people/hvr/util-linux-patch-int/util-linux-2.11n.patch.bz2

## Util-Linux Bauen

1. auch hier wieder PGP Signatures verifizieren
2. im entpackten util-linux Source-Tree den Patch applien
3. `mount` mit `make -e SUBDIRS="lib mount"` compilieren
4. das neue `mount/losetup` testen
5. die alten Binaries sichern und die neuen installieren

# Erstellen des Dateisystems

1. Anlegen einer Datei für das Dateisystem mit `dd`; Alternativ: Nutzung einer freien Partition, Initialisierung mit Zufallsdaten aus `/dev/urandom`
2. loopback device auf die Datei/Partition aufsetzen mit `losetup -e <cipher> <device> <datei>`, Passphrase und Keylänge merken; Achtung: jeder, der auf das device zugreifen kann, kriegt die unverschlüsselten Daten
3. Dateisystem mit `mke2fs <device>` erstellen, danach `fsck`
4. Dateisystem mounten mit `mount <device> <mountpoint>`
5. Wenn alles geklappt hat kann man das Dateisystem jetzt unmounten, mit `losetup -d <device>` das loopback device “detachen” und mit

   ```
   mount -o encryption=<cipher>,keybits=<keylänge>,loop \
         <datei> <mountpoint>
   ```

   wieder mounten; Einträge in die `fstab` sind ebenfalls möglich

# Links

## Sourcen

**Kernel:** http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.18.tar.bz2

**loopback device Patch:**

http://www.kernel.org/pub/linux/kernel/people/hvr/testing/loop-jari-2.4.16.0.patch

**International Kernel Patch:**

http://www.kernel.org/pub/linux/kernel/people/hvr/testing/patch-int-2.4.18.1.bz2

**Util-Linux:** http://www.kernel.org/pub/linux/utils/util-linux/util-linux-2.11n.tar.bz2

**Util-Linux Patch:** http://www.kernel.org/pub/linux/kernel/people/hvr/util-linux-patch-int/util-linux-2.11n.patch.bz2

## Workshop-Folien

**PDF, PS, DVI und LaTeX 2ε:** http://berlin.ccc.de/~packet/cryptfs-eh02-workshop/

# Links

## Verschlüsselte Dateisysteme

**CFS:** http://www.ibiblio.org/pub/Linux/docs/faqs/security/Cryptographic-File-System

**TCFS:** http://www.tcfs.it/

**StegFS:** http://www.mcdonald.org.uk/StegFS/

**International Kernel Patch (alt):** http://www.kerneli.org/

**CryptoAPI / neuer International Kernel Patch:**

http://cryptoapi.sourceforge.net/

**LoopAES:** http://loop-aes.sourceforge.net/

## Sonstiges

**Schwächen der Loopback-Verschlüsselung:**

http://www.off.net/~jme/loopdev_vul.html

**Linux Encryption HOWTO:** http://encryptionhowto.sourceforge.net/