Proseminar
„Konzepte von Betriebssystem-Komponenten Schwerpunkt Sicherheit“

CFS und TCFS

Franz Hirschbeck
franz@hirschbeck.net

1. Ziele eines verschlüsselten Dateisystems

Seine Dateien vor ungewollten Einblicken zu schützen ist wohl den meisten Benutzern ein mehr oder weniger wichtiges Anliegen bei der Auswahl ihres EDV Systems. Die Kommunikation ins Internet lässt sich mit firewalls und der Verwendung von Verschlüsselungssoftware recht gut absichern, doch bei entwendeten Hardware oder einem Einbruch in das interne Netz wird die Sache schwieriger. Da sehr viele Lösungsansätze vorstellbar sind, aber nur wenige wirklich effektiv und passend, ist es wichtig zunächst einmal fest zu legen, was mit dem System einmal erreicht werden soll.

Wie wollen uns folgende Ziele definieren:

- Es werden viele unterschiedliche Programme zum bearbeiten von sensiblen Daten eingesetzt, diese müssen natürlich auch weiterhin alle funktionieren. Die Standard Unix Dateisystemschnittstelle, also die Systemcalls zum öffnen/schließen/bearbeiten von Dateien müssen unabhängig von der Verschlüsselung arbeiten.
- Das beste System nützt nichts, wenn es nicht benutzt oder umgangen wird, solange es für den Benutzer umständlich ist ein sicheres Verfahren zu benutzen. Der Benutzer soll sich also maximal einmal authentifizieren müssen, die Verschlüsselung sollte damit auskommen und alle Daten bereitstellen.
- Natürlich muss auch betrachtet werden, wovor eigentlich geschützt wird und welche Angriffe überhaupt vorstellbar sind. Im Fall von Dateisystem kommen hier die zwei Fälle in Frage: wenn Hardware entwendet wird, also z.B. ein Notebook gestohlen oder wenn das Netzwerk abgehört wird.
- Sind die Daten so wichtig, dass man sie speziell schützen will, sollen diese meistens auch mit einem Backup System vor versehentlichem Löschen etc. geschützt werden. Da hier viele mögliche Angriffspunkte vorhanden sind, wäre es sinnvoll das Backup nur in verschlüsselten Form abzulegen. Ein Backup nützt natürlich nur, wenn im Falle eines Falles wirklich eine Datei wieder in einen früheren Zustand zurückversetzt werden kann, ohne großen Aufwand und ohne die Sicherheitsregeln zu verletzen.

- Aus modernen Büros sind Netzwerke nicht mehr wegzudenken. Die Dateien der Mitarbeiten liegen zentral auf wenigen Servern, die alle Clients gleichzeitig bedienen müssen. Da die Rechenleistung im Server begrenzt ist und kryptografische Algorithmen von Natur aus sehr rechenintensiv sind, ist hier eine clientseitige Verschlüsselung angebracht. Dies hat auch den Vorteil, dass die Kommunikation im internen Netz nicht zusätzlich abgesichert werden müsste.
- Kein Algorithmus schützt perfekt, es wird immer Möglichkeiten geben diesen zu knacken, die Frage ist nur, wie lange ein potentieller Angreifen dafür braucht um an die Daten zu kommen, und ob sie im dann noch was nützen. Um dies zu vermeiden wollen wir nur effektive Algorithmen benutzen. Metainformationen wie Dateinamen, Bearbeitungszeit, etc. sollen bestenfalls auch mit verschlüsselt werden.

2. CFS

Ein Lösungsansatz auf die in 1 erwähnte Vorgaben ist die Implementierung eines kryptografischen Dateisystems von Matt Blaze, dem er den Namen CFS (Cryptographic File System) gegeben hat.

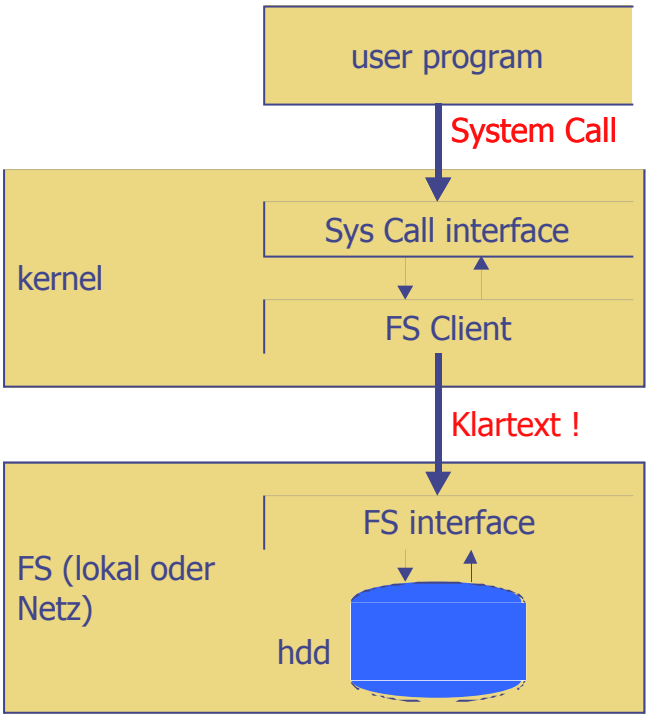
2.1. Eigenschaften

Bei CFS werden die Dateien separat codiert, mitsamt ihrem Namen. Die anderen Metadaten wie Zeitpunkt der Erstellung, die ungefähre Dateigröße sowie Benutzer und auch die Tatsache, dass überhaupt verschlüsselt wurde bleibt dabei erhalten. Dies hat den ungemeinen Vorteil, dass Dateioperation vom Betriebssystem aus, wie z.B. ein filesystem check, eine quota oder auch ein Backup ohne weiteres durchgeführt werden kann und dabei dem Operator kein Vertrauen geschenkt werden muss.
Das Verschlüsselungssystem präsentiert den Programmen die Dateien dabei in einem lokalen Verzeichnis, auf das den Benutzer bzw. dessen Programme zugreifen können ohne dass sie auf die Tatsache achten müssen, dass die Daten am Ende verschlüsselt auf dem physischen Datenträger landen. Eine einzige Einschränkung muss hier jedoch gemacht werden, da die Performance naturgemäß durch die Verschlüsselung sinkt. Dieser Effekt wird in einigen Spezialfällen deutlich, wenn etwa ein Programm eine Datei nicht sequentiell liest, sondern viele verstreute Zugriffe auf eine Datei macht.
Die Verschlüsselung wird in Sitzungen (sessions) gehandhabt, für die jeweils ein eigener Schlüssel existiert, der aus einem Passwort erstellt wird.
Der Benutzer startet eine Session mit einem Attach, dabei wird das Verzeichnis, auf dem die verschlüsselten Dateien liegen einem Sessionnamen zugewiesen und ein Passwort abgefragt, unter dem dann die Ver-/Entschlüsselung läuft. Unter dem CFS Verzeichnis (typischerweise /crypt) existiert nun ein Unterverzeichnis mit dem Sessionnamen, auf das nur der Benutzer Zugriff hat und dass sogar vor root unsichtbar gemacht werden kann. Root könnte sich hier also Zugriff verschaffen, was ein generelles Problem des Unix Benutzerschutzes ist.
Wollen mehrere Benutzer mit den gleichen Daten arbeiten können sie in ein gemeinsames Verzeichnis erstellen, in das sie Daten mit einem gemeinsamen Passwort verschlüsselt ablegen.
Nach dem Arbeiten sollte das Verzeichnis wieder gedetached werden, um die Gefahr eines Angriffes zu minimieren .

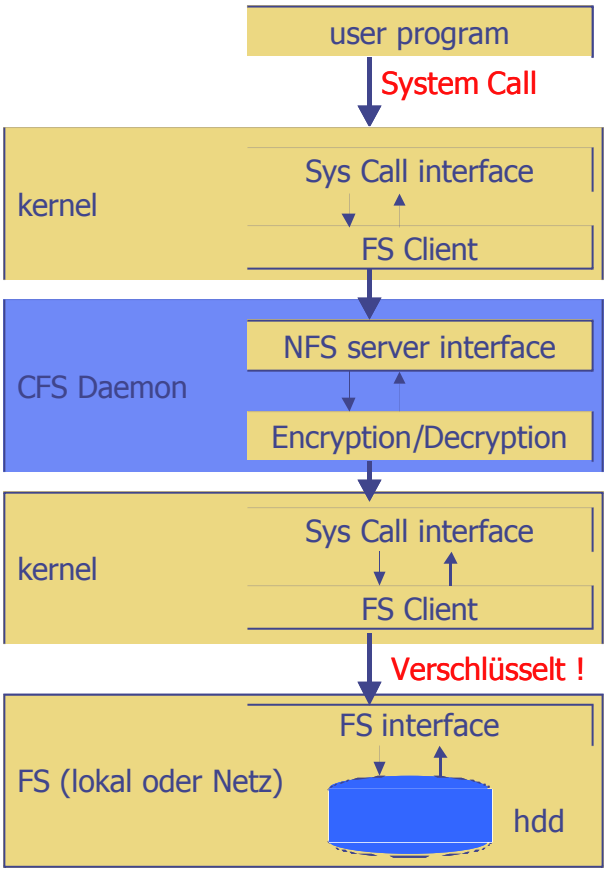
2.2. Systemarchitektur

Das Cryptographic File System besteht aus dem Dienst cfsd und den Befehlen cmkdir, cattach und cdetach. Der cfsd Daemon ist im Prinzip ein modifizierter NFS Server, der seine Daten bevor er bevor er sie localhost auf dem speziellen Port 3049 anbietet entschlüsselt, bzw. wenn

er sie speichern soll wieder verschlüsselt. Dazu wird der einzig angebotene Pfad in den Systempfad gemountet (typischerweise auf /crypt).
Mit cmkdir lässt sich ein Verzeichnis anlegen, in dem später die Daten in verschlüsselter Form liegen, dabei wird auch gleich der Schlüssel festgelegt unter dem codiert wird und der aus dem Passwort erstellt wird.
Auf den Klartextinhalt, des in einem mit cmkdir erzeugten Verzeichnis lässt sich nach einem cattach zugreifen. Dabei wird dem cfsd (der ja grundsätzlich auf dem gleichen System läuft) welches Verzeichnis er unter welchem Schlüssel anbieten soll. Dieser bietet das Verzeichnis als Unterverzeichnis des gemounteten Verzeichnisses an.
Eine Dateioperation (z.B. auf /crypt/mp3/lied1.mp3 wird also zunächst an den kernel weitergeleitet, dessen filesystem client leitet diese mit dem NFS Protokoll an den cfsd weiter, dieser wiederum übernimmt holt sich den entsprechenden Block mit Hilfe eines erneuten SystemCalls macht die Verschlüsselung und gibt das Ergebnis zurück. (siehe Abbildung 2.1)



Standard UNIX SystemCall (open, write, close …)



SystemCall bei verschlüsselten Dateien

Abbildung 2.1

2.3. Verschlüsselung

Die Verschlüsselung erfolgt bei CFS mit dem DES Algorithmus, der inzwischen nicht mehr als sehr sicher gilt. Die neuesten Verbesserungen von CFS schließen jedoch die Sicherheitslöcher von DES aus.
Die Dateien werden einzeln verschlüsselt. Der Schlüssel wird aus dem Passwort erstellt.
Für effektive Gruppenarbeit sind leider viele Passwörter notwendig.

3. TCFS

Eine Verbesserung von CFS stellt TCFS dar, das **Transparent Cryptographic File System**.

3.1. Eigenschaften

Bei tcfs können geschützte und ungeschützte Dateien bzw. Verzeichnisse im gleichen Verzeichnis stehen. Die Eigenschaft des „Verschlüsselt-sein“ wird ähnlich einem Dateiattribut behandelt. Auch für Gruppen gibt es ein Attribut, das sofern gesetzt nur der Gruppe (gid) der Datei Zugriff gewährt. Hier wird ein weiteres Konzept eingeführt, der threshould, eine Art Mindestanzahl der Gruppenmitglieder, die eingeloggt sein muss, um überhaupt Zugriff zu bekommen. Eingeloggt bedeutet in diesem Zusammenhang der Aufruf des Kommandos tcfsputkey, der einem Benutzer Zugriff auf seine verschlüsselten Dateien (also die mit gesetztem Attribut) bekommt.
Es sind mehrere Schlüsselverwaltungsmöglichkeiten gegeben, ebenso können mehrere Algorithmen zur eigentlichen Verschlüsselung hergenommen werden. Üblicherweise werden die Schlüssel aus den login-Passwörter erzeugt.

3.2. Systemarchitektur

Im Gegensatz zu CFS läuft TCFS im kernel, also direkt im System. Das vermeidet einerseits den doppelten mount der sonst nötig wäre als auch Sicherheitsrisiken. Auch die eigentliche Kryptographie, also die Verschlüsselungsalgorithmen sind Kernel Module.
Es gibt also im System ein neues Dateisystem tcfs unter dem gemountet wird, und dass seine eigentlichen Daten von einem NFS Server bezieht, auf dem dann die Dateien in verschlüsselter Form liegen.
Ein Benutzer muss, um auf entschlüsselte Dateien zugreifen zu können seinen Schlüssel an den kernel geben, der dann damit die Dateien, die dem Benutzer gehören und ein verschlüsselt-flag tragen diesem Benutzer entschlüsselt präsentiert.
Systemcalls schauen hier folgendermaßen aus:

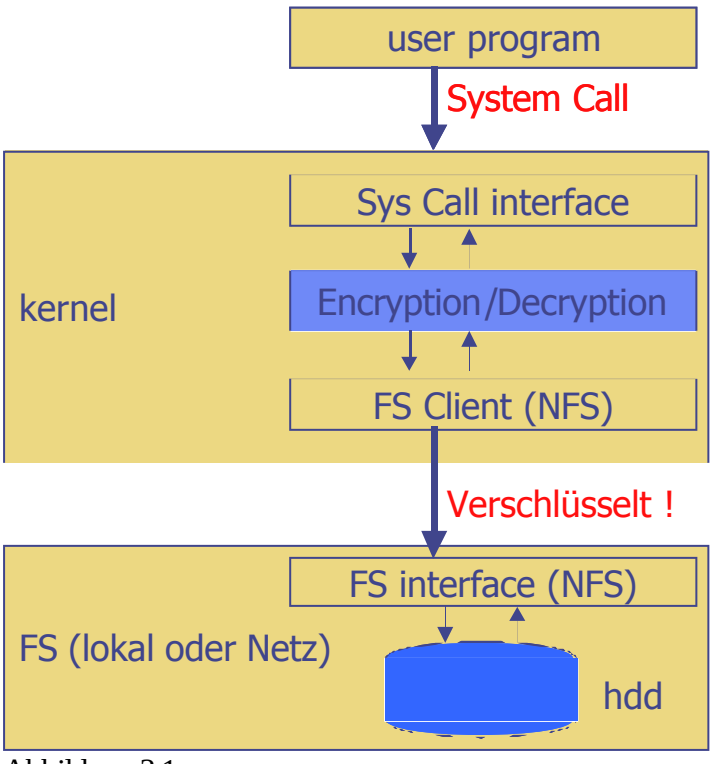


Abbildung 3.1.

3.3. Verschlüsselung

Die eigentliche Verschlüsselung Erfolg bei TCFS mit Hilfe von Kernel Modulen. Es werden Module für folgende Algorithmen unterstützt: 3DES, IDEA, RC5 und Blowfish. Dadurch, dass der Algorithmus nicht vorgeschrieben ist, kann leicht ein anderer eingesetzt werden, falls der aktuelle zu unsicher ist.
Zu erwähnen ist auch noch, dass hier lediglich die Schlüsselverwaltung mit Basic KMS behandelt wurde, es existiert auch die Möglichkeit die Schlüssel mit Kerberos zu verwalten.

4. Fazit

Ein Verschlüsselungssystem zu verwenden ist immer mit zusätzlichem Aufwand und Risiko verbunden. Wer sich dazu entschließt sollte sich genau überlegen, was er erreichen will. Ein verschlüsseltes Dateisystem kann nur Teil eines Gesamtkonzeptes sein. Nach diesem Konzept muss sich dann auch die Wahl des Systems richten, zwar scheint TCFS hier die Nase vorn zu haben, doch ist Installation tiefer im System und dadurch schwieriger.

5. Verwendete Literatur / Quellen:

- A Cryptographic File System for Unix“, Matt Blaze (ftp://ftp.research.att.com/dist/mab/cfs.ps)
- „The Design and Implementation of a Transparent Cryptographic Filesystem for UNIX“, Guiseppe Cattaneo, Luigi Catuogno, Aniello Del Sorbo, Pino Persioano (http://www.tcfs.it/docs/freenix01.ps)
- http://www.cbi.pku.edu.cn/Doc/CS/cfs/CFS/
- „Abenteuer Kryptologie“, Reinhard Wobst, Addison-Wesley Verlag, München 2001 (aus Uni-Bibliothek)
- FAQs von TCFS (http://tcfs.dia.unisa.it/tcfs-faq.html)
- ftp://ftp.research.att.com/dist/mab/cfs.announce
- http://www.tcfs.it/docs/linux-expo-2001/Diapositiva1.JPG.html