Kryptographische Dateisysteme genauer betrachtet

Stefan Schumacher

Magdeburger Institut für Sicherheitsforschung

Chemnitzer Linux-Tage 2011



Über Mich

- selbständiger IT-Sicherheitsberater mit Schwerpunkt auf Sicherheitsschulungen im Bereich Social Engineering und Security Awareness
- Geschäftsführender Direktor des Magdeburger Instituts für Sicherheitsforschung
- Mit-Herausgeber des Magdeburger Journals zur Sicherheitsforschung
- Bildungswissenschaftler und Psychologe



Table of Contents

1 Einführung

2 Details

3 Betriebsarten von Algorithmen

4 Schlüsselverwaltung

5 Kryptographische Dateisysteme



Warum das alles?

- Kryptographie sichert Vertraulichkeit, Integrität, Authentizität von Daten
- kryptographische Dateisysteme verschlüsseln relativ bequem Dateien
- der einzig brauchbare Weg um Daten vor Verlust zu schützen, Zugriffsrechte im Betriebssystem lassen sich relativ einfach aushebeln
- kryptographische Dateisysteme sind wenig benutzerfreundlich
- Administration lässt zu wünschen übrig
- vor allem in großen Installationen (viele Benutzer)
- Portabilität? Was ist das?
- Zeit für einen Rant und Antworten auf Standardfragen



Table of Contents

1 Einführung

2 Details

3 Betriebsarten von Algorithmen

4 Schlüsselverwaltung

5 Kryptographische Dateisysteme

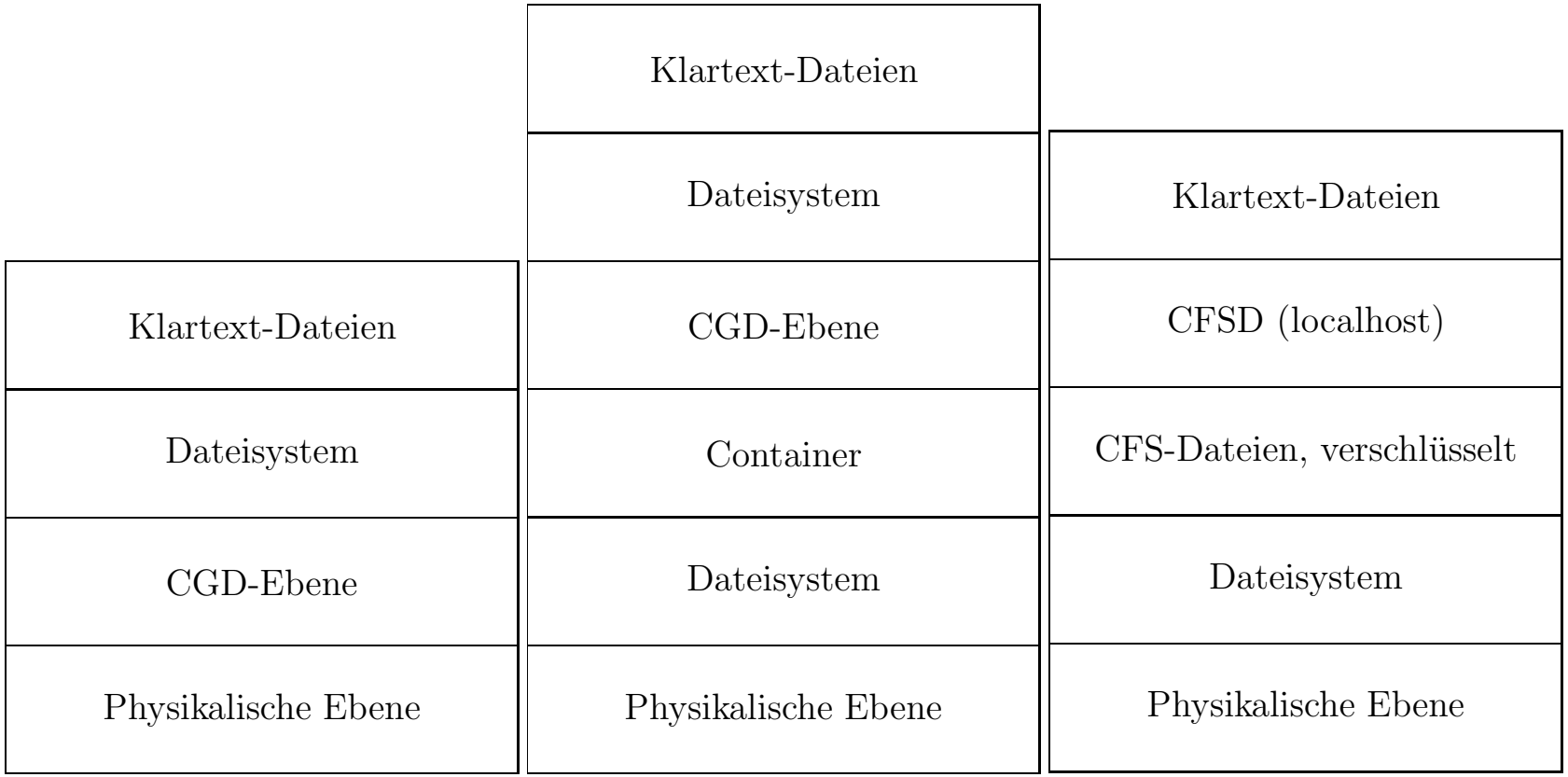


Arten von kFS

- jede Datei einzeln verschlüsseln
- Container oder Verzeichnisse im normalen Dateisystem via Dæmon verschlüsseln
- Dateisysteme/Blockgeräte verschlüsseln



Arten von kFS



(a) CGD auf Blockebene einer Partition

(b) CGD auf Pseudo-Blockebene eines Container

(c) CFS im Dateisystem

Arten von kFS

Vorteile/Nachteile

Dateien einzeln verschlüsseln

- einfach zu bewerkstelligen (mcrypt Brief.txt/mdecrypt Brief.txt.nc)
- portabel (GnuPG, mycrypt, crypt, OpenSSL)
- netzwerktauglich (NFS, Samba ...)
- hohe Granularität auf Dateiebene und Benutzerebene
- geringer Einrichtungs-Aufwand
- hoher Gebrauchs-Aufwand
- sicheres Datenlöschen
- einfaches Backup möglich (Datenmenge=1)
- Meta-Daten sichtbar!



Arten von kFS

Vorteile/Nachteile

Container

- Einrichtungs-Aufwand etwas höher
- Gebrauch transparent
- Portabilität eingeschränkt (OS ./. Anwendungsprogramm)
- netzwerktauglich
- geringe Granularität auf Dateiebene und Benutzerebene, feste Größe (!)
- Container kann gesichert werden, aber: Platzbedarf, Passwortdatei nicht vergessen, inkrementelles/differentielles Backup geht nicht ohne eingebundenen Dateien, Datenmenge >= 1, idR >> 1
- keine Metadaten sichtbar!



Arten von kFS

Vorteile/Nachteile

Dæmon

- Einrichtungs-Aufwand etwas höher
- Gebrauch transparent
- Portabilität eingeschränkt (OS ./. Anwendungsprogramm)
- netzwerktauglich
- hohe Granularität auf Dateiebene und Benutzerebene
- verschl. Verz. kann problemlos gesichert werden und ist dann schon geschützt, inkrementelles/differentielles Backup funktioniert (Datenmenge= 1)
- Metadaten sichtbar!



Arten von kFS

Vorteile/Nachteile

Blockebene/Dateisystemebene

- Einrichtungs-Aufwand hoch
- Gebrauch transparent
- Portabilität eingeschränkt (OS ./. Anwendungsprogramm)
- nicht netzwerktauglich
- null Granularität auf Dateiebene und Benutzerebene: entweder an oder aus
- Device kann mit dd(1) gesichert werden, aber: Platzbedarf, Passwortdatei nicht vergessen, inkrementelles/differentielles Backup geht nicht ohne eingebundenen Dateien, Datenmenge >= 1, idR »1
- keine Metadaten sichtbar!



Swap und Temp

- Auslagerung von RAM bzw. temporäre Inhalte
- d.h. Swap und Temp können eigentlich verschlüsselte Inhalte unverschlüsselt enthalten
- müssen verschlüsselt werden (oder RAM-Disk)
- idR nur auf Blockebene praktikabel



Table of Contents

1 Einführung

2 Details

3 Betriebsarten von Algorithmen

4 Schlüsselverwaltung

5 Kryptographische Dateisysteme



Block und Strom

- Stromchiffre: Daten und Schlüssel als Strom, die miteinander verknüpft werden, bspw. Sprachdaten im Telefon
- Blockchiffre: Daten und Schlüssel werden in gleichgroße Blöcke zerlegt und blockweise verknüpft
- Datenträger sind in Blöcken organisiert $\rightsquigarrow$ Blockmodi einsetzen

```
Rindflei schettik etierung süberwac hungserg
PASSWORT PASSWORT PASSWORT PASSWORT PASSWORT
änzungsg esetz
PASSWORT PASSW
```



Electronic Code Book (ECB)

- jeder Block einzeln verschlüsselt
- Blöcke werden nicht gekoppelt
- wahlfreier Zugriff auf bestimmten Block möglich
- Nachteil: statistische Kryptanalyse möglich, da:
- zusammenhängende Klartextblöcke $\rightsquigarrow$ zusammenhängende Chiffrate
- selber Klartextblock $\rightsquigarrow$ selbes Chiffrat
- Watermarking möglich (Nachweis, dass eine bestimmte Datei verschlüsselt wurde)



Cipher Block Chaining Mode (CBC)

Definition (CBC Verschlüsselung (rekursiv))

$$\begin{aligned} C_0 &= E_K(P_0 \oplus \mathrm{IV}) \\ \forall (i \in \mathbb{N}^+) : (\quad C_i &= E_K(P_i \oplus C_{i-1}) \quad) \end{aligned}$$

verkettet die Blöcke in dem er den Klartextblock vor der Verschlüsselung mit dem vorhergehenden Chiffreblock per XOR als sog. Initialisierungsvektor (IV) verknüpft (Klartextblock n + Chiffrat $n-1$ + Passwort)



Cipher Block Chaining Mode (CBC)

- Lawineneffekt: ändert sich der n.-Block, ändert sich der IV für $n+1$, ändert sich der $n-1$.-Block, ändert sich der IV für $n+2$, ... Einteilung der Blöcke in Gruppen
- 1. Block hat keinen IV $\rightsquigarrow$ einen Ausdenken, bei schlechter Wahl (Vorhersagbarkeit wie Zeitstempel) ist eine Watermarking-Attacke möglich
- Lösung ESSIV: IV aus der verschlüsselten Sektornummer und einem Hash aus dem Passwort, Passwort ist nicht vorhersagbar



Cipher Block Chaining Mode (CBC)

Definition (Entschlüsselung (nicht-rekursiv))

$$\begin{aligned} P_0 &= D_K(C_0) \oplus \mathrm{IV} \\ \forall (i \in \mathbb{N}^+) : (\quad P_i &= D_K(C_i) \oplus C_{i-1} \quad) \end{aligned}$$

benötigt: Chiffrat-Block n, Chiffrat-Block $n-1$, Schlüssel
Problem: Chiffrat-Block $n-1$ defekt $\rightsquigarrow$ Chiffrat-Block n
verloren aber: ab $n+1$ funktioniert es wieder



Table of Contents

1 Einführung

2 Details

3 Betriebsarten von Algorithmen

4 Schlüsselverwaltung

5 Kryptographische Dateisysteme



Schlüsselverwaltung

- Problem in großen Installationen
- 500 Mitarbeiter, davon 150 Außendienstler mit Laptop
- KryptFS ausrollen praktisch unmöglich
- Portabilität? Was ist das?
- Backups problematisch
- Kompression als Layer
- Key-Escrow? Kann der Sysadmin, Chef, ... noch auf die Daten zugreifen?
- Key-Evocation? Kann ich Benutzer Alice aus dem System werfen?



Schlüsselverwaltung

- mehrere Rollen mit Zugriff auf die verschlüsselten Daten (root/wheel, ACL, etc.pp.)
- symmetrische Verschlüsselung der Daten mit symmetrischem Schlüssel
- symmetrischen Schlüssel asymmetrisch je Benutzer verschlüsseln
- zentrale Key-Revocation



Schlüsselverwaltung

- Das eine »perfekte« System gibt es nicht.
- Will es jemand entwickeln?
- Ich sag ihm auch wie ;-)



Table of Contents

1 Einführung

2 Details

3 Betriebsarten von Algorithmen

4 Schlüsselverwaltung

5 Kryptographische Dateisysteme



kFS Kurzüberblick

CFS: Cryptographic File System von Matt Blaze (1993)

- CFS-Dæmon ähnlich dem NFS-Dæmon, auf localhost
- Dateien liegen verschlüsselt im Dateisystem
- werden bei Zugriff durch cfsd geschleift
- für nahezu jedes Unix verfügbar, portabel
- Wechseldatenträger
- alt, langsam, DES, 3DES, Blowfish, MacGuffin

CGD: Cryptographic Disk Driver, ab NetBSD 2.0

- verschlüsselt Partitionen auf Blockebene, Container via vnd(4)
- kann Zufallschlüssel generieren und swap und temp verschlüsseln
- AES, 3DES, Blowfish
- schnell, nicht-portabel,



kFS Kurzüberblick

Trucrypt

- Container oder Blockebene
- Windows, MacOS X, Linux,
- kann Kryptalgorithmen verketten

GEOM/GBDE: FreeBSD

- Blockebene, jeder Block bekommt eigenen Zufallsschlüssel

GELI/GBDE: FreeBSD

- Blockebene
- Zufallsschlüssel und mehrere Schlüssel für eine Partition
- Prüfsummen für Daten zur Integritätssicherung



kFS Kurzüberblick

eCryptFS

- ab Linux-Kernel 2.6.19
- Metadaten *in* der Datei $\rightsquigarrow$ Portabilität
- mehrere Schlüssel für eine Datei oder ein Verzeichnis



kFS Kurzüberblick

EncFS

- FUSE-basiert
- unterstützt die im Linux-Kernel vorhandenen Kryptalgorithmen
- kann jede Datei mit eigenem IV verschlüsseln
- 8-Bit-Prüfsumme für jede Datei
- Linux, MacOS X, NetBSD (mit Einschränkungen)

CryptoFS

- FUSE-basiert
- verschlüsselt jede Datei einzeln im Dateisystem



Fragen?

Stefan.Schumacher@Magdeburger-Institut-fuer-Sicherheitsforschung.de

Verschlüsselte Dateisysteme für NetBSD, GUUG UpTimes 4/2006

`http://kaishakunin.com/publ/guug-uptimes-cgd_cfs.pdf`

