Universität Bremen

03. Übungsblatt : Betriebssysteme

Dr. rer. nat. Bernhard J. Berger

Ausgabe: Mittwoch, 07. Januar 2026
Bearbeitungszeitraum: 07.01. bis 22.02.2026
Abgabe: 22.02.2026

Wintersemester 2025/2026

Auf diesem Übungsblatt werden wir uns mit der Implementierung von Dateisystemen unter Linux mithilfe von FUSE auseinandersetzen.
Hinweis: Es kann sein, dass Details der folgenden Spezifikation sich über die Wochen noch ändern, wenn die Spezifikation ungenau und missverständlich ist. Solche Änderungen werden im Dokument entsprechend farbig markiert.

Aufgabe 1: Verschlüsselndes Dateisystem

Hinweis: Änderungen gegenüber dem Originaldokument sind in blau hervorgehoben.

Aufgabe: Implementieren Sie das im Folgenden beschriebene Dateisystem mit Hilfe von FUSE
Als Senior-Softwareentwickler des Unternehmen *ExtremeSecureIT* haben Sie den Auftrag bekommen, ein Linux-Dateisystem zu entwickeln, das Daten mit dem *extrem sicheren*™ Verschlüsselungsverfahren *xor* verschlüsselt.[1]

1 Speicherlayout

Die Struktur des Dateisystems ist an das Minix-Dateisystem angelehnt und besteht aus mehreren Bereichen. Jeder Block auf diesem Dateisystem hat die Größe von $1k = 1024$ *Byte* und die ganzzahligen Werte der in diesem Dokument spezifizierten Strukturen werden als *unsigned*-Werte gespeichert. Beachten Sie die Speicherung als Little-Endian, damit es nicht zu Interpretationsunstimmigkeiten kommt. Die grundlegende Struktur des Dateisystems ist die folgende:

SecFS – Speicherlayout		
	Größe (Blöcke)	Beschreibung
Super Block	1	Informationen über das Dateisystem
Inode Map (iMap)	$\#Inodes/(1024 \times 8)$	Welche Inodes belegt sind (bitcodiert)
Zone Map (zMap)	$\#DataZones/(1024 \times *8)$	Welche Data Zones belegt sind (bitcodiert)
Inodes	$(\#Inodes \times 32)/1024$	Liste aller Inodes
Data Zones	Rest	Liste aller Daten

Die *iMap* und *zMap* speichern als Bit-Wert, für jede einzelne Inode (bzw. jede einzelne Zone), ob diese belegt ist oder nicht. Ist das Bit auf 0 gesetzt, dann ist der Eintrag leer. Ist dieser auf 1 gesetzt, so ist er belegt.

2 Superblock

Der Superblock enhält Informationen über die Struktur des Dateisystems. Der Mount-Status wird beim mounten des Dateisystems auf den Wert 1 gesetzt und beim Aushängen des Dateisystems auf den Wert 0. Das Dateisystem soll sicherstellen, ob ein Dateisystem, das gemounted wird, in diesem Eintrag den Wert 0 stehen hat. Der Schlüssel für die root-Inode wird bei der Erstellung des Dateisystems ausgesucht und in den Superblock geschrieben. Die Struktur des Suberblocks sieht wie folgt aus:

[1] Dieses Verfahren wurde von der Tutorin für Kryptographie als sicherer eingestuft.

SecFS – Superblock			
Start		Ende	Inhalt
00	-	01	Anzahl der Inodes
02	-	03	Anzahl der Zonen
04	-	05	Größe der iMap (in Blöcken)
06	-	07	Größe der zMap (in Blöcken)
08	-	09	Startblock der Zonenblöcke
0A	-	0B	Größe einer Zone (in Blöcken)
0C	-	0F	Maximale Dateigröße (in Bytes)
10	-	11	Magic Number 57005
12	-	12	Mount-Status (0 oder 1)
13	-	13	Schlüssel für die root Inode

3 Inode

Eine Inode enthält alle Informationen für einen Dateieintrag. Dieses kann entweder eine Datei oder ein Verzeichnis sein. Der Modus-Einträg der Inode wird mit den Werten aus `<linux/types.h>` gesetzt, zum Beispiel `S_IFREG` und `S_IFDIR`. Die Datenzonen sind ihrerseits wieder mit dem oben genannten Verfahren verschlüsselt. Der Zonenschlüssel wird jeweils vor dem zugehörigen Zonenindex gespeichert und wird beim Erzeugen eines Eintrags zufällig erzeugt. Um einen Zonen-Index-Eintrag als ungültig zu markieren benutzen Sie den Zonen-Index `0xFFFF`. Die root-Inode wird immer in Inode 0 gespeichert und der zugehörige dentry-Eintrag in Zone 0.

SecFS – Inode			
Start		Ende	Inhalt
00	-	01	Modus
02	-	03	User ID
04	-	07	Dateigröße
08	-	0B	Zugriffszeit
0C	-	0D	Gruppen ID
0E	-	0E	Anzahl an Links
0F	-	0F	Schlüssel für Zone 0
10	-	11	Zonenindex für Zone 0
12	-	12	Schlüssel für Zone 1
13	-	14	Zonenindex für Zone 1
15	-	15	Schlüssel für Zone 2
16	-	17	Zonenindex für Zone 2
18	-	18	Schlüssel für Zone 3
19	-	1A	Zonenindex für Zone 3
1B	-	1B	Schlüssel für Zone 4
2C	-	1D	Zonenindex für Zone 4

Wichtiger Hinweis: Wir implementieren in dieser Version des Dateisystems nur direkte Zonenverweise. Dadurch ist die obere Dateigrenze stark limitiert ($5 * 1024 << block\_size$).

4 Dentry

Dentrys werden in den Zonen abgelegt, die über eine Verzeichnis-Inode verlinkt sind. Ein einzelner Eintrag in einem Dentry ist 32 Byte lang. Die ersten beiden Bytes geben die Inode des Verzeichniseintrags an und die verbleibenden 30 Zeichen sind für den Namen des Eintrags reserviert. Ist der Name kürzer als die genannten 30 Zeichen, so wird dieser mit einem Null-Byte (\0) abgeschlossen.

5 Implementierungshinweis

Am besten starten Sie mit einem C-Programm (`mkfs.secfs`)zum Erzeugen eines leeren Dateisystems. Hierbei können Sie Dateisystemdatenstrukturen anlegen und das grundlegende Konzept des Dateisystems verstehen. Versuchen Sie sich erst anschließend an der Implementierung des Kernel-Moduls. Das Kommandzeilenprogramm `hexdump` ermöglicht ihnen, dass sie sich die geschriebene Datei byteweise anschauen.

6 Evaluation

Um eure Implementierung zu prüfen bekommt jede Gruppe eine individuell erstellte Datei. Diese Datei müsst ihr mit eurem Dateisystemtreiber mounten und den Inhalt der Datei `/bs/challenge.txt` (unverschlüsselt) auslesen. Diesen Inhalt sollt ihr nun in einer neuen Datei unter `/abgabe/winter2526/response.txt` schreiben. Darüber hinaus sollt ihr in dieser Datei ein Gedicht, Limerick oder Haiku ablegen.