Assignment 4: CryptoFS: Design Doc

Authors: Robert Sato(captain), Keerthi Krishnan, Pranav Nampoothiri
Cruzid's: rssato, kvkrishn, pnampoot

Due Date: June 2nd, 2019

Note: This is a very basic layout of what we planned to do. The details as well as full explanations of the code implemented is included in the WRITEUP.pdf

In this assignment, we were required to implement a simple cryptography file system in the FreeBSD kernel at the VFS layer. Basically, the file system is supposed to encrypt on a per-file basis. We use the AES algorithm to encrypt and decrypt on a per-file basis, when the write and read system calls are called, respectively. We must implement this by adding a new stackable layer using the VFS interface that abstracts the low-level file systems to the upper levels of the operating system.

This breaks down into three parts: A system call that adds a particular encryption/decryption key for a particular user ID, Operating system code that applies the key to a file, and a program that encrypts/decrypts the file and sets the encryption bit accordingly. The specifics are listed below:

- System call
    - Adds a particular encryption/decryption key for a particular user ID
- Operating System Code that applies the key to a file based on the following:
    - The key is set for the current user
    - The file has its encryption(sticky) bit set
- Program that encrypts/decrypts a file
    - Sets the encryption bit accordingly

Ideas/Algorithm Design for Each Part(Rough Outline):

Syscall Setkey(takes in two unsigned ints k0 and k1):

What we Need:

- Static Table to keep track of all user ids and the key associated with user id
    - Create array of structs with userid and key as parameter
    - Define it such that it only takes in 16 elements
- Create a struct to use both k0 and k1

Algorithm:

- Check if both the unsigned ints are 0
    - If both are 0, then cannot make the key
    - Return 1
- If they are both not 0
    - Create key using k0 and k1 and set it equal to a variable(type long long-64 bits or more)

- Create a loop where we traverse through the array
    - For each element in the array of structs, set the userid of the element in the array equal to the userid of the thread
    - For each element in the array of structs, set the key of the element equal to the key we generate
    - If the number of elements goes over 16
        - Return 1
    - Return 0

FUSE:

- Use FUSE to add a new layer below the VFS layer in FreeBSD
- Make changes to either fuse_vfsops.c, fuse_vnops.c, or fusexmp.c, to add encryption and decryption for read and write system calls.
- We connect the VFS layer with the underlying FreeBSD file systems, with the mounted file system from FUSE installation.

Algorithm Design:

- Create 2 functions, read and write
    - Read→ decrypts
    - Write → encrypts
- Read:
    - Read data in file
    - If sticky bit is set to 1, then decrypt
    - If sticky bit is set to 0, then do not decrypt
    - In while loop for decryption:
        - Read in 16 byte blocks of data from file
        - XOR with encryption value
        - Write back to decrypt file→ aka original text
        - End while loop when all 16 byte blocks have been processed
- Write
    - Read data from file
    - If sticky bit is set to 0, then encrypt
    - If sticky bit is set to 1, then do not encrypt
    - In while loop for encryption:
        - Read in 16 byte blocks of data from file
        - XOR with encryption value
        - Write back to encrypt file → aka encrypted text
        - End while loop when all 16 byte blocks have been processed

Protectfile:

- Same idea as FUSE, encrypt/decrypt file and turn sticky bit on/off depending

- Might have to play around with permissions and privileges→ use chmod as well as syscalls to open, close, read and write files.
- Make sure that sticky bit is off, so that no encryption/decryption happens unnecessarily