

**The University of Queensland
School of Information Technology and Electrical Engineering
Semester 2, 2009**

COMP3301 COMP7308

ASSIGNMENT 3 – Encrypted File Systems

Version 1.1

Due: **5pm Friday 30th October, 2009**

Weighting: **25% (25 marks)**

Objective

As part of the assessment for this course, you are required to complete three assignments which will assess your understanding of the Minix operating system and ability to program at a low level in C. This particular assignment will assess your ability to add functionality to the file system which is run as a user-level server in Minix 3.

The Assignment

The Minix 3 filesystem is currently not secure. Any system can mount the filesystem (if root access is provided) and is able to access files in the system. In this assignment, you are required to encrypt certain files within the filesystem, so that the disk is only readable by a kernel with the appropriate encryption key in the filesystem code.

The Details

The contents of all files created with the naming format “filename.encrypt” (i.e. ending with or is named “**.encrypt**”) are to be stored in encrypted form. The encryption to be applied is trivial where each byte of the file should be XOR-ed with a single byte key – 0xDB (or in binary 1101 1011). When a file, satisfying the naming requirement is written, the **contents of the file on disk must be encrypted**. The contents of the file in memory (RAM) may be encrypted or not (this is up to you to decide). For testing and simplicity reasons, encrypted files **must not** be of a different file type to regular files and they must be operable on a default clean Minix image where their contents shall appear as “garbage”. Apart from these differences the operation of the system should be transparent such that user applications should not need to know anything about the encryption status of the file.

Links, movement, and copying are to be supported in the following manner:

- Creating hard links to encrypted files shall result in errors.
- Symbolic links shall be supported for encrypted files however they may only extract the encrypted “garbage” data regardless of whether they satisfy the naming requirement.
- An encrypted file shall be converted to a regular file if its filename is changed such that it no longer satisfies the naming requirement.

- If the filename of a non-encrypted file is changed such that it satisfies the naming requirement for encrypted files, the file must then become encrypted if it possesses no additional hard links, if otherwise, the renaming operation shall fail.
- Movement of encrypted files across partitions (e.g. from */usr* to */home*) is not required to be supported. The displayed error messages in any of the described failures may be anything as long as the failures occur (i.e. you may use existing error codes).

Marking

Your submission will include the following files:

- `s123456_ass3_final.patch` (where 123456 is replaced with your student number)
- README documenting your changes and justifying any assumptions you have made in your implementation.
- `build.sh` (optional), which builds your modified source code

To mark your assignment, we will use a default Minix image (similar to previous assignments – this is provided on the course website for your own testing purposes). Unless alternative instructions are given in your README file, your changes will be applied in the following manner. The patch file will be applied in the `/usr/src/` directory with the following command:

```
patch -p0 < s123456_ass3_final.patch
```

The source code will then be built using “make clean world” at the `/usr/src` directory and then rebooted. **Ensure that your submission will work correctly according to this procedure – particularly ensure that the patch file applies cleanly to the provided image.** We will create encrypted files in multiple places on the filesystem and check that operation is correct (i.e. no changes should be observable). We will then attempt to create symbolic links to these files and verify that we are able to extract the raw, encrypted data from them. We will also attempt to rename the files so they are decrypted and re-encrypted. Finally we will boot the system using the original kernel. Reading the encrypted files should then result in “garbage” data which when manually decrypted, XOR-ed with the key in a normal program, produces the original data. If this process does not produce the original data the encryption system did not work correctly.

The README file should detail your approach and any limitations specific to your implementation. This will assist the marking process.

Hints

- The function “`rw_chunk()`” in `/servers/fs/read.c` handles most IO related operations. It may be worth your time to investigate how it works and how it is used
- Look back to prac 1 and how system calls are implemented in minix, it may be helpful to trace through existing system calls that performs functions which you must modify.

Help sources

Help for this assignment is available from a number of sources. Help in understanding the Minix code is best obtained either from the textbook, or from the course newsgroup, uq.itee.comp3301. As a last resort, you can ask the tutor or lecturer for help.

Submission Details

The due date for the assignment is 5pm Friday 30th October, 2009. The assignment must be submitted via the ITEE online submission system. Assignments may be resubmitted numerous times, but only the last submission will be marked.

Late Submission Penalty

Late submissions (without penalty) will not be accepted except with a doctor's note and even then by permission of the course coordinator.

Otherwise, a 10% penalty applied for each day your submission is late. The weekend shall count as a single day. Thus the following scheme:

Submit	Penalty
Friday 30th Oct - 5pm	0%
Monday 2nd Nov - 5pm	10%
Tuesday 3rd Nov - 5pm	20%

and so on. The late penalties are not cumulative.

Allocation of Marks

Students will be marked on their ability to implement changes in the filesystem code and the demonstration of how those changes can be seen in the operation of the system.

The following marking scheme will apply:

	Range applicable
Coding and testing: (18 marks)	
System fails to boots to login screen	0-5
System boots but is unusable	6-8
Files created correctly, but not handled correctly	9-12
Files read and written mostly correctly	13-15
Files read and written correctly	16-18
Documentation: comments (7 marks)	
Nothing	0
Some, seriously lacking detail	1
Reasonable, but limited details	2-3
Clear, lacks justification for assumptions	4-5
Very clear - no omissions	6-7
Deductions	
Obscure code	2.5
Inefficient code	1

Academic Merit, Plagiarism, Collusion and Other Misconduct

You should read and understand the statement on academic merit, plagiarism, collusion and other misconduct contained within the course profile and the document referenced in that course profile. You should note that this is an **individual assignment**.

Notification of Results

You will be notified of your results via the course homepage. You will be able to view your individual results only.