

Reference Monitor
CS681 & CS392 Computer Security
Fall 2005
DUE 10/20/2005
October 10, 2005

1 Objective

The objective of this lab is to design and develop a reference monitor for FEAU. The reference monitor is an independent process that is responsible for making decision related to access control.

2 FEAU Reference Monitor (FRM)

2.1 Reference Monitor

Design and implement a reference monitor as separate process to handle access control for FEAU. Figure 1 depicts a reference monitor. In the figure actions are inputs and decisions (usually yes or no) are outputs.

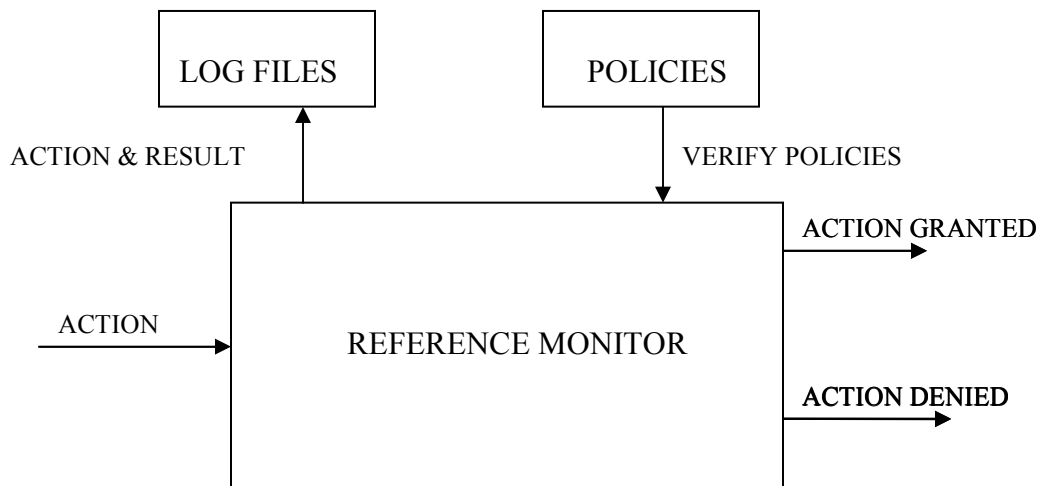


Figure 1: Reference Monitor

Actions are basically commands supported by FEAU. When a user wants to execute a command the reference monitor decides whether the command should be executed or not. The decisions are based on the policies, which are set by the administrator(s), and the credentials of the user who executed the command.

To do the above you must also design a suitable language to encode action such that the reference monitor could make accurate decisions. Also develop security policies for FEAU and some method to encode them. You may consider the Reference monitor as a trusted back end process that keep track of all users and group information independent from FEAU.

Policies that could be supported include but are not limited to

- 1) A User must at least be a member of one group.
- 2) A group cannot be the owner of a file.
- 3) A file can only be owned by one user
- 4) A user can be deleted if s/he exist and does not own any file.
- 5) A group can be deleted if it exists and there are no users in that group.
- 6) A user can transfer ownership of a file to another user if he is the current owner.
- 7) A user can set group decrypt permission to his files.
- 8) A user can only set group decrypt permission if he belongs to that group.
- 9) A user can only encrypt any file he owns, but not encrypt an encrypted file
- 10) A user can always decrypt any file he owns, but not decrypt a decrypted file.
- 11) A user can only decrypt a file owned by other user in the same group if allowed by the owner and if the file is in encrypted state.

Think about the policies that you want to support. You should have a justification for the policies that you want to support.

Actions that could be supported include but are not limited to

1) addgroup(<groupname>)

Description: Add a group named <groupname>

Pre-Condition: The group should not exist, if not generate "Action Denied".

Post-Condition: The group is remembered by the reference monitor, and generates "Action Granted"

2) adduser(<username>, <g_1>,, <g_n>)

Description: Add a user named <username> and make him/her a member of groups specified in <g_1>,, <g_n>

Pre-Condition: The user should not exist, if not generate "Action Denied".

Post-Condition: The user remembered, and generates "Action Granted"

3) own(<username>, <file>)

Description: make the user <username> the owner of file <file>

Pre-Condition: The user should exist and the file should not be owned by another user, if not generate "Action Denied"

Post-Condition: The file/user ownership action should be remembered and generates "Action Granted"

4) encrypt(<file>)

Description: encrypt file <file>

Pre-Condition: Check policies, if necessary generate "Action Denied".

Post-Condition: The reference monitor should remember that the file is in encrypted state and generate "Action Granted".

5) decrypt(<file>)

Description: decrypt file <file>

Pre-Condition: Check policies, if necessary generate "Action Denied".

Post-Condition: The reference monitor should remember that the file is in decrypted state.

6) set_group_decrypt_permission(<user>, <group>, <file>)

Description: set group decrypt permission for file <file>

Pre-Condition: Check policies, if necessary generate "Action Denied".

Post-Condition: Check policies, if necessary generate "Action Granted".

7) unset_group_decrypt_permission(<user>, <group>, <file>)

Description: set group decrypt permission for file <file>

Pre-Condition: Check policies, if necessary generate "Action Denied".

Post-Condition: Check policies, if necessary generate "Action Granted".

8) del_user(<username>)

Description: delete user <username>

Pre-Condition: Check policies, if necessary generate "Action Denied".

Post-Condition: Check policies, if necessary generate "Action Granted".

9) del_group(<groupname>)

Description: delete group <groupname>

Pre-Condition: Check policies, if necessary generate "Action Denied".

Post-Condition: Check policies, if necessary generate "Action Granted".

10) trfr_ownership(<user1>,<user2>,<file>)

Description: <user1> transfer his file <file> ownership to <user2>

Pre-Condition: Check policies, if necessary generate "Action Denied".

Post-Condition: The new association should be remembers and generate "Action Granted"

Like policies the actions that you support should be justifiable.

For this section you must design and implement the reference monitor. It is advisable to develop and test it independently. To do this you can put all the actions in a file and read it and produce a decision file as result.

2.3 Integrating FEAU and Reference Monitor

For this section design and implement a mechanism that integrates FEAU and the reference monitor. You can use, but are not limited to, the following inter process communication mechanism:

- Message queues: <http://www.cs.cf.ac.uk/Dave/C/node25.html>
- Shared Memory: <http://www.cs.cf.ac.uk/Dave/C/node27.html>
- PIPES: <http://www.cs.cf.ac.uk/Dave/C/node23.html>

3 What and how to Handin

Handin: You should submit assignment through my.poly drop box no later then 12 AM midnight on the due date.

You must zip all files related to the assignments and use the following convention to name the zip file:

<First Name>_<Last Name>_<Lab#>.zip

Submit using the name of only one of the members in your group.

REMEMBER IF YOU DO NOT USE THIS NAMING CONVENTION YOUR ASSIGNMENT WILL NOT BE GRADED AND YOU WILL NOT RECEIVE ANY CREDIT.
PLEASE SUBMIT HOMEWORK ON TIME.