Maalvika Bachani
CSU ID: 831112878

# DBMS - Term Paper Report

# TOPIC: Linux Policy Machine

# INTRODUCTION

Linux Policy Machine provides an infrastructure to handle concurrent applications on a single server instances. It provides an isolated runtime environment to run individual application components. These applications in isolated environments needs to share resources with each other. Therefore, LPM provides a framework for such application to access OS resources and allow controlled collaboration and coordination for service components running in disjoint containerized environments under a single Linux OS server instance

# CURRENT IMPLEMENTATION

LPM consists of following layer:

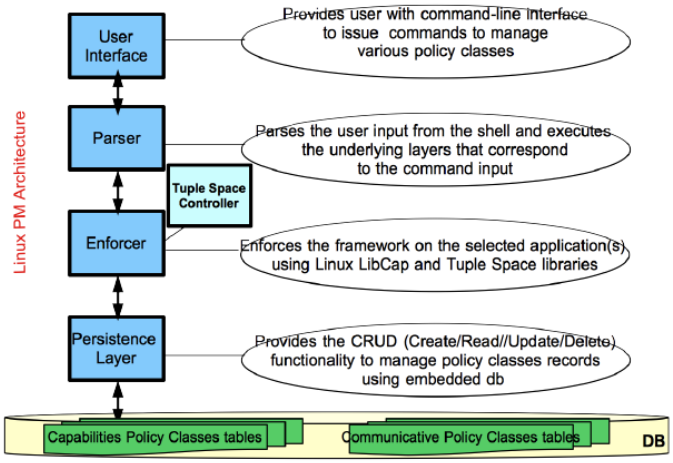
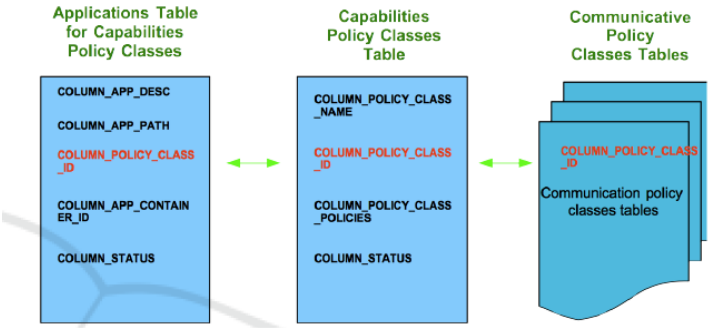



Figure 4: Linux Policy Machine (LPM) Architecture.




Current implementation consists of a prototype for the above model. These layers are implemented as Java classes and interfaces which communicate with each other to provide the suggested functionality.

# TASK ASSIGNED

TASK 1: Refactor the existing code

I worked on Parser_implement class. Initially, Parser_implement class had 1885 LOC. It contained numerous “if-else” statements.

Operation performed in the task:

1. Based upon Single Responsibility Principle, I had split the Parser_implement into 3 classes.
    a. Parser_implement - The responsibility of this class is to parse the incoming command. It accepts the command and check it against various command formats supported by LPM.
    b. Parser_validate - It validates the command received against the command supported by LPM. It checks the proper format of the command received. The various if-else statements were converted into different case statements.
    c. Parse_and_Execute - Parser_validation class uses this class. It contains the actions are performed once the command is identified.

TASK 2: Add CHANGE_PASSWORD command and authentication

CHANGE_PASSWORD command requires the interaction with Parser layer to interpret the it and check if valid arguments are present or not. If Parser parses the command successfully, then values are sent to DB layer.
In DB layer, a user authentication table exits which check whether old password is correct or not. After satisfying all conditions, the password is changed in the database.

New Classes added to accommodate this command:

1. UserAuthDAO – It is an interface that provides functionality to access the user authentication database.
2. UserAuthDAO_impl: It contains the implementation for UserAuthDAO.
3. New constants were added in DB_Constants.

TASK 3: Enable Remote Log-in

In order to work on this task, java.net.Socket and java.net.ServerSockets class are used. The server will run on port 9001. Only one client will be able to connect to server at a point of time.

# CONCLUSION

The extension done is mostly targeted to Parser Layer and Database Layer. The complexity of code is reduced by dividing it into different classes. It requires some more work to make more modular by adding more error and exception conditions. For testing remote log-in and remote access functionality, a test client needs to be developed.