SUBVERSION ACL AUTOMATION

HARSHA KUMAR
USN: 1BM13IS024, Dept. of Information Science and Engineering
BMS College of Engineering, Bangalore, India

ABSTRACT

To restrict individuals' access to libraries and packages, Access Control List came into existence. ACL file consists of various groups and repositories. ACL defines which group has access to which repository. A group consists of users. If a group has access to a repository, then that group will be present along with its access right under the repository in ACL. In order to add a user to a repository, the user must be added to a group and that group must be added to the repository specified. So, people should manually go through a lot of groups in ACL to find a specific group where the user must be added and again search for the repository in the ACL and add the group under that repository. Along with that, people should also verify for duplicate repositories or groups before adding. This process is time consuming and very cumbersome. The problem with this approach is that it requires a lot of time to go through the Access Control List file to make changes and is not secure. Developing an application for managing the Access Control List file provides an alternative way in this direction. 'Access Control List Automation' application which handles most of the change patterns that occur commonly to the Access Control List file. This application not only makes it a lot easier to edit the Access Control List file, but also provides rich set of features such as security, notification and also saves a lot of time.

Keywords—*Acess +Control List, Discretionary Access Control*

---

1. INTRODUCTION

The Access Control List, known as 'ACL' is a list of access rights to a file or any other subject . An ACL specifies which users has access to file or an object, as well as what operations are allowed on given objects. Each entry in a typical ACL specifies a subject and an operation. For instance, if a file object has an ACL that contains (User A: read; User B: read, write), this would give user B permission to read and write the file and User A to only read it.

ACL file consists of various groups and repositories. A group consists of users. If a group has access to a repository, then that group must be under the repository in ACL file. An ACL file is length depends on the number of objects associated with it. Since the number of repositories in the Software Library is large, the ACL file is also huge and it grows day by day as new repositories are added to the Software Library. Because of this, manually editing ACL file is time taking and cumbersome.

The editing of ACL file can be made a lot easier by developing an application that handles majority of change patterns that occur to the ACL file. By developing a frontend application, users can easily choose what to edit in the ACL directly in the application instead of going through the ACL file manually and make the changes. Along with that the application should also provide user authentication, maintain logs, preserve previous configurations for restoration and many other features.

2. LITERATURE SURVEY

Khan et.al showed access control mechanism in the hospital/clinical management system wherein sensitive patient information should be available only to the appropriate doctors/users to ensure the safety and privacy of the patient [1].

Min-A Jeong et.al showed access control in hospitals where every patient record (PRn) is guarded against illegal access by deploying access policies [2].

BAI Qing-hai et.al showed DAC (Discretionary Access Control) type defined by Trusted Computer Evaluation Criteria. In DAC Access Control List (ACL) consists of a list of subjects with their permission to access the file on that operating system [3].

Yanfang Fan et.al showed that the lower lever DAC in contrast with MAC, does not allow resource owner to assign access control and to prepare their own policies. DAC is "need to know" access model. It helps realize the principle of least privilege wherein the user is allowed to access just the right amount of information (nothing more, nothing less) based upon his credentials [4].

Wenrong Zeng et.al showed CBAC (Content Based Access Control) which is an innovative access control model designed for content centric information sharing. It is applied where RBAC (Rule Based Access Control) will give more access right, on top of such model CBAC is deployed [5].

3. PROPOSED SYSTEM

In order to solve the problem, developing a program that handles the changes to the ACL is required. By analyzing the recent changes made to ACL, Few proposed features are :

- Interactive frontend atop SVN repository to allow structuring of access to any level of any of the repositories.
- LDAP authentication and provide validation of AD users and groups. For example validate if a requestor is already part of an AD group that already has access to the repository requested.
- Automatically revoke access upon expiration of the stipulated time for which the access was granted.
- Manage ACL for multiple repositories. Multiple repositories might include among the Fossware/Library/Packages/ Artifactory.
- Maintain logs of requests and changes for auditing and debugging.
- Validation before changes are incorporated and in case of errors ability to revert to previous configuration.

4. DESIGN

Access Control Lists change patterns that has to be observed first. Access Control Lists are stored in SVN repository. Since SVN stores what changes are made to the ACL along with the timestamp and revision number of the edit, going through the SVN commit comments and observing the changes made to the ACL gives a good idea about what the application must do.
The Access Control Lists Automation project can be broadly divided into 2 parts—

- Backend script to edit the ACL
- Frontend web application

Backend Script

The backend script is written in Python. The reason for choosing Python as the language for writing backend script is because –

- Python has a greater range of libraries and packages.
- Python is good in file manipulation which is more important for ACL file editing.
- Python has a good community support.

Frontend web Application

The frontend web application is developed using Ruby on Rails web framework. Ruby on Rails, or simply Rails, is a server-side web application framework written in Ruby. Rails is a model–view–controller (MVC) framework, providing default structures for a database, a web service, and web pages. It encourages and facilitates the use of web standards such as JSON or XML for data transfer, and HTML, CSS and JavaScript for display and user interfacing.

The frontend application connects backend python script through a web page. The user can choose the option displayed in the web page and enter the required information such as group name or user name and click the 'Submit' button. The web app accepts information from the user through the graphical user interface and runs the python script in the background and render the result in the HTML page.

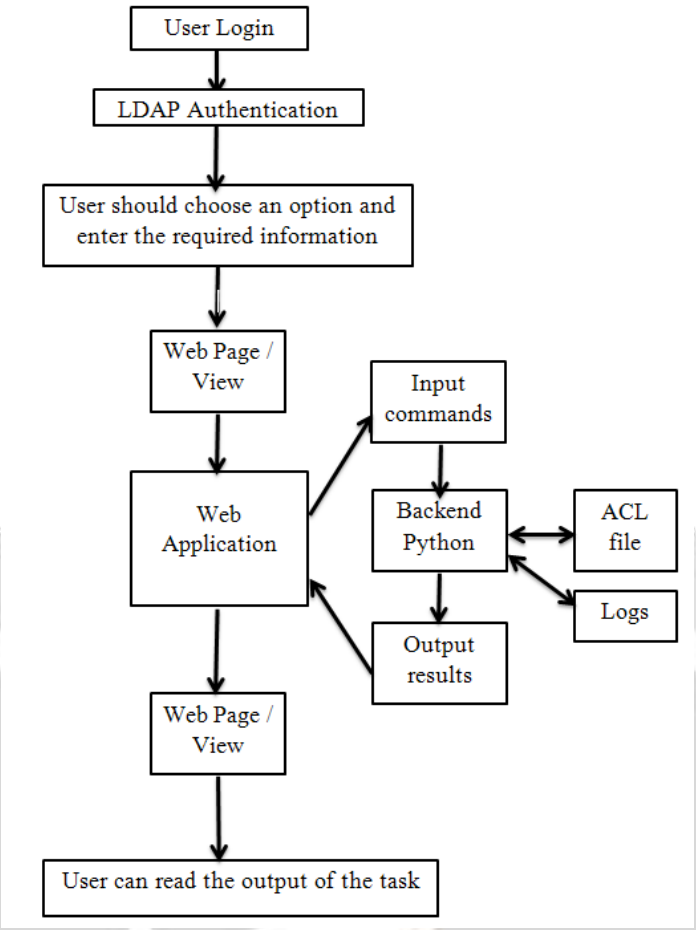

Fig-1:Architecture of ACL

5. METHODOLOGY

Methodology is a process of analysis of methods. The backend Python script should be able to connect to the server and open the specified ACL file and edit it. The script should accept the commands from the user and make changes to the ACL file as requested by the user and send back the response. The script should handle the ACL file properly without any errors. Whenever user runs the script, the script gives an option to the user to enter the command. The following commands are accepted by the program –

Add user to group – When the user enters this command, the program should connect to the server and open the ACL file and go through it to check if the group is present, then it should check if the user is already present in that group. If the group is not present, then the program should display that the specified group is not present. must add the user to the group in the ACL. This command also accepts multiple users to be added at once to a group.

Delete user from group – When the user enters this command, the program should connect to the server and open the ACL file and go through it to check if the specified group is present. If the group is not present then the program must display that the group is not present. On the other hand if the group is present, then the program must check if the given user is present in the group. If the given user is not present, then the program must display the same.

Display all groups that user belongs – When the user enters this command, the program goes through the ACL file and searches the user. If the user is found, then it gets the group user belongs to. Then the program displays all groups the user belongs to.

Create new group – When the user enters this command, the program connects to the server and opens the ACL file to edit. It goes through the ACL to check if the group is already present. If the group is already present, program displays the same to the user. If not, the program creates new group.

Delete group -- When the user enters this command, the program connects to the server and opens the ACL file to edit. It goes through the ACL to check of the group is present. If the group is not present, the program informs the same to the user. If the group is found, the program deletes the group and all its users.

6. IMPLEMENTATION

The web application contains web pages that is, the Graphical User Interface (GUI) for editing the Access Control Lists. The commands are directly available to the user. The following process must be followed by user to work with the application When the user clicks 'submit', the application first builds the appropriate command for the python script and then after the build is successful, the application calls the python script and passes the command to the script. The script makes changes to the ACL and sends the result of the operation back to the application. The application then reads the response and displays it to the user. The user can also see the log file to verify that the operation has completed successfully or not. By clicking 'Show Log' option, the system displays the log file directly without calling python script.

Some of the snapshots of the application are shown below:

Fig-2:ACL editor

This is the Edit Users page. As we can see, there are options such as 'add user to group' , 'delete user from the group' , 'display all groups that user belongs' . The user must enter the proper username or group name in the fields and click submit.

Fig-3: ACL Menu

7. TESTING AND RESULT

Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output. Here unit test cases are written for backend python script where each module performs specific tasks.

Table-1: Result

Test Cases	**Expected Outcome**	**Result**
User enters 'user' and 'group' in 'add (user) to (group)' box and clicks 'Submit'	The application must call backend script and pass the command 'add (user) to (group)' and display the output.	Success
User enters 'user' and 'group' in 'remove (user) to (group)' box and clicks 'Submit'	The application must call backend script and pass the command 'remove (user) to (group)' and display the output.	Success
User enters 'user' in 'display all groups that (user) belongs' and clicks 'Submit'	The application must call backend script and pass the command 'display all groups that (user) belongs' and display the output.	Success
User enters 'group' in 'create new (group)' and clicks 'Submit'	The application must call backend script and pass the command 'create (group)' and display the output.	Success
User enters 'group' in 'delete (group)' and clicks 'Submit'	The application must call backend script and pass the command 'delete (group)' and display the output.	Success
User enters 'group' , 'right' and 'repository' in 'modify (group) add (right) to (repository)' and clicks 'Submit'	The application must call backend script and pass the command 'modify (group) for (repository) add 'right' ' and display the output.	Success
User enters 'group' , 'right' and 'repository' in 'modify (group) remove (right) to (repository)' and clicks 'Submit'	The application must call backend script and pass the command 'modify (group) for (repository) remove 'right' ' and display the output.	Success
User enters 'group' in 'display repositories the (group) has access' and clicks 'Submit'	The application must call backend script and pass the command 'display all repositories the (group) has access ' and display the output.	Success
User enters 'group' and 'AD-Ent file' in 'Add (group) to (AD-Ent file)' and clicks 'Submit'	The application must call backend script and pass the command 'add (group) to metadata (AD-Ent file) ' and display the output.	Success
User enters 'group' and 'AD-Ent file' in 'Remove (group) from (AD-Ent file)' and clicks 'Submit'	The application must call backend script and pass the command ' delete (group) from metadata (AD-Ent file) ' and display the output.	Success
User enters 'repository' and 'repository new name/path' in 'Rename (repository) to (repository new name/path)' and clicks 'Submit'	The application must call backend script and pass the command ' rename (repository) to (repository new name/path)' and display the output.	Success

8. CONCLUSION

The Subversion ACL Automation application developed fully meets the objectives of the system which it has been developed. The application has reached a steady state where most of the bugs have been eliminated. The system is operated at a high level of efficiency and all the users associated with the system understand its advantage. The system solves the problems present in the existing system.

Since nothing in this world is perfect, there might exist few hidden defects which an user might come across while using the application, the future task of this project would be to rectify those errors and correct it. Also there might come further functionalities that the user would want to be available in the application, the future work would be adding those functionality and further enhancing the application.

9. ACKNOWLEDGEMENT

I wish to express my deep sense of gratitude to Prof. B S Mahalakshmi, Assistant Professor, Dept. of Information Science and Engineering, BMSCE, for her continuous, tireless support and advice.

10. REFERENCES

[1] Khan, M.F.F.; Sakamura, K.," Context-aware access control for clinical information systems", International Conference on Innovations in Information Technology (IIT), IEEE, pp.123 – 128, 2012.

[2] Min-A Jeong, Jung-Ja Kim, Yonggwan Won, " A flexible database security system using multiple access control policies", International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE, pp-236-240, 2003.

[3] BAI Qing-hai, ZHENGYing, "Study on the Access Control Model in Information Security", Cross Strait Quad-Regional Radio Science and Wireless Technology Conference, IEEE, pp. 830-834, 2011.

[4] Yanfang Fan, Zhen Han, Jiqiang Liu, Yong Zhao, "A Mandatory Access Control Model with Enhanced Flexibility", International Conference on Multimedia Information Networking and Security, IEEE, pp.236- 240, , 2009.

[5] Wenrong Zeng, Yuhao Yang, and Bo Luo. "Content Based Access Control: Use Data Content to Assist Access Control for Large-Scale Content-Centric Databases," IEEE International Conference on Big Data, IEEE, pp. 701-710, 2014