Access Control List in Operating Systems

E. Law, S. Shek, K. Wong, and C. Zhu

***Abstract*—This paper compares the usability of Access Control List (ACL) among two operating systems, Windows XP Professional and Fedora Core 2 SELinux, based on experiments. 20 test subjects are asked to set up the ACL for a list of files according to a pre-made scenario. Results show that most participants are unable to utilize ACL correctly; therefore, they create potential data confidentiality and integrity issues. Moreover, Linux is found to be a more efficient in configuring the ACL as compared to Windows.**

***Index Terms*—Access Control List**

I. Introduction

THE PURPOSE of this paper is to examine the usability of ACL in Windows XP Professional and Fedora Core 2 SELinux operating systems. ACL is a powerful security mechanism that allows user to maintain the confidentiality and integrity of data. The implementation of ACL in operating systems is very secure since they are integrated into the operating systems at the developing stage. However, users of ACL can introduce vulnerabilities into the system by incorrectly setting the access control entries. As stated by Cranor and Garfinkel, when the objective is to maintain privacy, usability plays a very important role in the system [1]. Therefore, it is important to determine the usability of access control lists in different operating systems.

II. Experiment Methodology

The purpose of the experiment is to analyze and compare the usability of ACL on Windows XP Professional and Fedora Core 2 SELinux with respect to accuracy, task completeness, and completion times. The actual results will also compare to the intended security goal.

A. Scenario

The test subject will be a Hardware Engineer in company called ACL Group. The participant will try to configure the ACL for each file or folder to grant access to other staff within the company. The goal is to prevent any unauthorized access or modification to all your personal files.

The company consists of fourteen people and eight different positions. Hugo, the Boss of the company, hired two Project Managers, Leo and Cindy, to lead one project team each. In each project team, there is a Design Engineer, a Requirement Engineer, and a Hardware Engineer. The test subject will be the Hardware Engineer for Team 1. An Assistant Manager, Nelson, is switched between the two project teams and provides help when necessary. Jen, the Accountant of the company, is responsible for using the budget list to do accounting. Three Technicians, Peter, David, and Kosta are responsible for parts ordering for engineers from both teams.

There are a total of 7 files that the test subject needs to share with and protect from other employees. These 7 files are grouped into 3 folders: the Design Document folder, the Progress folder, and the Finance folder. The Design Document folder contains schematics and PCB layouts of the company's design. The Progress folder includes meeting summary, personal schedule, and progress reports. The Finance folder consists of both the company's budget list and part ordering list. These files are not accessible to everyone in the company, for example, the Boss, Hugo, can only view the company's design documents while Project Manager Leo can both view and make changes to them.

B. Test Subjects

A total of 20 test subjects are involved in the experiment. All test subjects are students from various faculties including Arts, Biology, Commerce, Dentistry, Engineering, and Forestry. Their ages ranged from 20 to 28. All 20 test subjects had some experience with Windows XP, and four test subjects have used Linux operating systems before, but only two are experienced. None of all 20 test subjects understood ACL prior to the experiment.

C. Test Subject's Assistance

To help test subjects become familiar with the two different operating systems, a brief tutorial is provided before the experiment is conducted. Since most participants are Windows based users, a Linux oriented tutorial is given. Windows and Linux ACL configurations differ significantly; Windows relies heavily on its user friendly graphical layout, while Linux requires command inputs from the user via the terminal. Since most participants do not have any programming background, understanding Linux commands and their syntax becomes an issue. Based on the test subject's knowledge of Linux systems, the appropriate amount of assistance on useful Linux commands such as setfacl, getfacl, --help, and man, is given.

Before the experiment begins, all rights for each file were denied (default setting). The test subjects were then given time to read through the scenario and role description documents. No assists were given during the first ten minutes. If the test subjects cannot figure out where to start after ten minutes, hints and guides were then provided to assist them. Each test subject had a one hour time frame to complete the tasks for each operating system.

After the one hour period, the modified ACL's were recorded and compared with the correct answers. They were then restored back to their default setting for the next test subject.

III. HYPOTHESIS

There are two main objectives to be achieved in the experiment. The first involves determining the usability of access control lists in general. The second objective is to compare and analyze the usability and complexity of access control lists in two operating systems, namely the Microsoft Windows XP Professional and the Fedora Core 2 SE (Security Enhanced) Linux. The data collected from the 20 test subjects will be used to reach these objectives.

Before any experiment was conducted, the following hypotheses were made:

1. Current access control lists incorporated into modern operating systems have low usability. In order to protect files within a system, one must need to grant permission levels to other users for every file. This process becomes quite cumbersome, complicated, and time consuming as the set of files and users increase.
2. Windows ACL is less prone to mistakes by users. Since the Windows operating system is more widely used by home users, participants should be more comfortable and familiar with a Windows environment. Linux ACL requires a series of command input to set up the ACL; therefore, participants may find it confusing and may be unable to complete the task within the one hour time frame.

IV. RESULTS

To determine the usability of ACL in both Windows and Linux, twenty test subjects are gathered to do an experiment based on the scenario. The results are separated into two categories: quantitative and personal comments. Using these results, the usability of ACLs in both operating systems will be measured based on accuracy, test completion time and test subjects' comments.

To increase diversity, the test subjects are chosen from different faculties. Figure 1 below shows the percentage of the test subjects in different faculties:

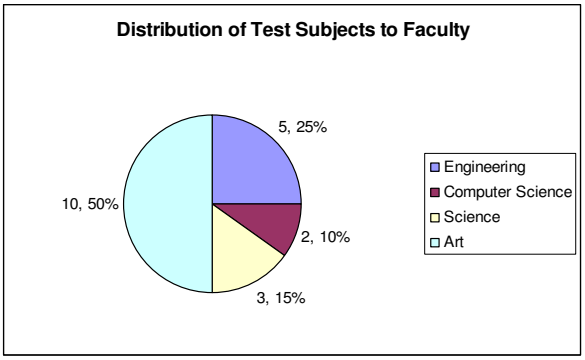

Figure 1. Distribution of Test Subjects to Faculty

The numbers beside the percentage values are the number of test subjects in that specific faculty. To truly test usability, most test subjects are chosen from the Arts faculty to reflect the large percentage of Arts students in the real world. This consideration is also taken into account when choosing students from other faculties.

Among the test subjects, Alvin and Rosita are in Computer Science and are very experienced in Linux Systems. In addition, there are two former EECE 412 students, Claudia and Ivan. They both have sufficient knowledge on computer security issues.

Quantitative results

A. *Accuracy*

In this experiment, accuracy is determined by counting the number of test subjects who can produce the ACL within 95% correctness. This shows if the test subjects are able to use the options provided by the operating systems correctly. Figure 2 below compares the correctness of test subjects in relation to the two operating systems.

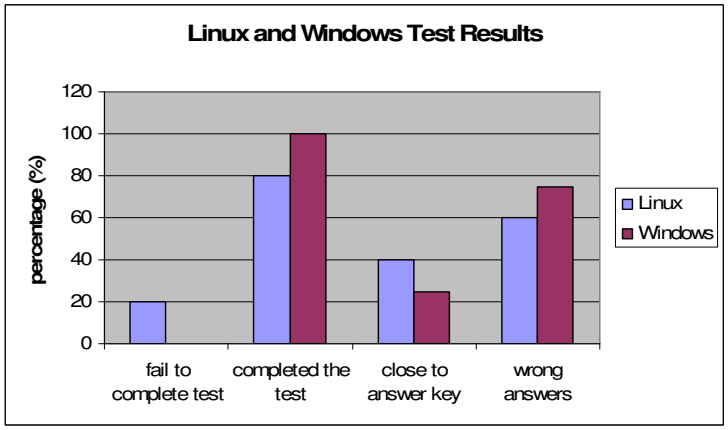

Figure 2. Linux and Windows Test Result

In the chart, the x-axis represents different categories of correctness while the y-axis represents the percentage of test subjects. The chart shows that the ACLs created on Linux are more accurate than the ACLs on Windows: 40% of the total test subjects have created the correct ACLs, while only 25% correctly configured ACLs on Windows. From these results, it is obvious that test subjects have a higher chance of configuring a correct ACL on the Linux than on the Windows system.

Figure 3 and 4 below show the distribution of the faculties of test subjects who correctly configured ACLs on Linux and Windows systems. According to the pie chart, 50% are Engineering students, 25% are Computer Science students, and 12.5% for both Art and Science students. A result of interest is that both Computer Science students with extensive Linux experience are able to complete the ACL with precision.

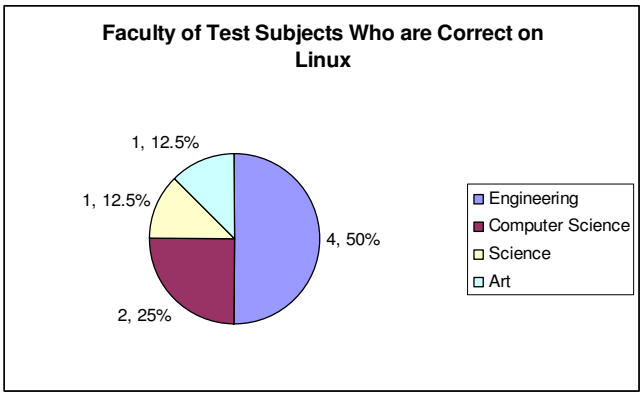

Figure 3. Faculty of Test Subjects Who are Correct on Linux

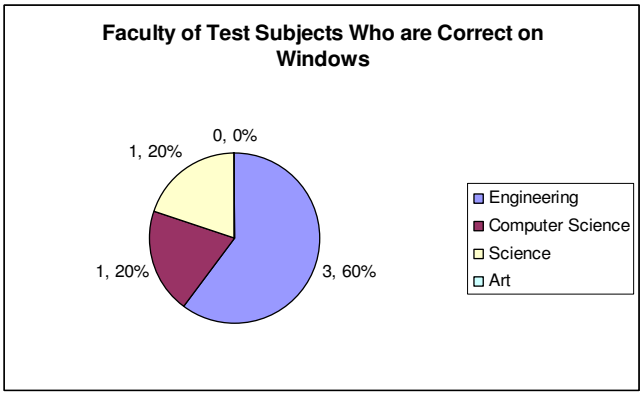

Figure 4. Faculty of Test Subjects Who are Correct on Winodws

B. *Test Completion Time*

Test completion time tests how much time it takes a participant to finish creating all ACL entries on each operating system. It reflects the user-friendliness of the user interfaces on the two operating systems. Figure 5 below shows the test completion time for both operating systems:

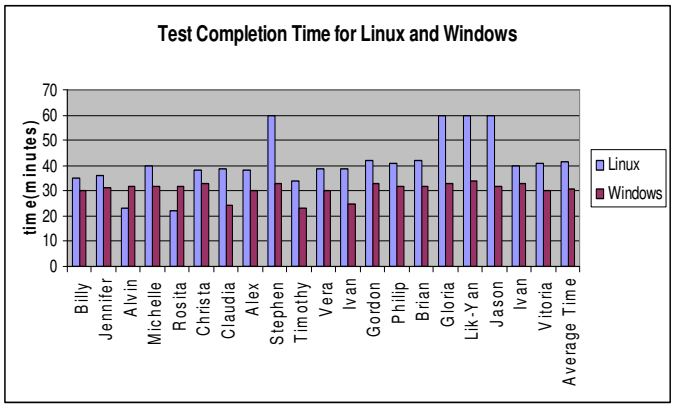

Figure 5. Test Completion Time for Linux and Windows

The x-axis is categorized into different test subjects and the y-axis represents the time in minutes. According to the graph, the test completion time of every participant on Linux is much higher. The average time for Linux is around 40 minutes, but the average time for Windows is only around 31 minutes. The difference between the two operating systems is obvious. However, for Rosita and Alvin, the two Linux experts, they only spent approximately 22 minutes on the Linux test. Because they know all the necessary commands for Linux, no time was wasted on figuring out the appropriate commands and syntax.

Test Subjects' Comments

The last category, which is test subjects' personal comments, represents test subjects' satisfactory level towards the ACL on these operating systems.

For Linux ACL, almost all the test subjects claimed that they spent most of their time on finding the correct commands to use; however, once they are familiar with the commands, the progress is much faster than before. Because most of the test subjects do not have much knowledge with Linux, they needed a sufficient amount time to get used to the controlling environment of Linux.

As for Windows, although all the test subjects are familiar with its interface, they encountered several problems when modifying the ACL. They stated that the ACL interface is confusing and complicated; setting one ACL entry for one file involves traversing through multiple windows, adding users to a list, and then granting permissions to those users. In addition, they are often confused about the distinction between some permission levels, namely "read" and "read and execute".

V. Discussion And Observation

A. *Analysis of Results and Hypotheses*

Using the results obtained from the experiments, initial hypotheses can be reassessed.

1) *Usability of ACL*

The first hypothesis states that current access control lists in modern operating systems have low usability. It can be concluded that this theory is fairly accurate, given the low rate of users successfully completing the experiment with a high degree of accuracy. Of the 40 experiments conducted, there are only 13 instances where all access control lists are configured to achieve 95% or higher correctness. This translates into a 32.5% success rate, a percentage way below the standard for a system to be recognized as usable.

There are several reasons for the low usability of access control lists. The scenario the experiment is based on is neither complex nor simple; 13 users in a company is a reasonably small number, whereas home users might never have the need for more than four accounts. However, due to the inexperience of our test subjects on access control lists, the scenario is quite complex and sometimes even difficult to comprehend.

2) *Comparison of Windows and Linux*

The second hypothesis states that Windows ACL is more usable. However, this is inconsistent with the results obtained from the experiments. Out of the 20 test subjects, 8 configured the ACL in Linux to within 95% accuracy, while only 5 test subjects were able to achieve that level of accuracy by using Windows. Considering the fact that only 10% of all test subjects are experienced at Linux, it can be seen that Linux ACL is definitely more usable than Windows. Although the average time for completion is higher for Linux, it does not mean that the access control lists in Linux systems are less usable; this is analyzed and discussed in a later section.

B. Linux

Results have shown that all test subjects on average require ten extra minutes to set up the ACL for Linux. This is due to the fact that most test subjects spend at least 20 minutes understanding and becoming comfortable with Linux commands, syntax, and the overall environment. Once the test subjects are familiar with the command, they are able to configure the ACL much more efficiently than by using Windows. The following features implemented in Linux contribute to this efficiency advantage over Windows:

1. Test subjects are able to add or remove user permissions with one simple line of command via the terminal. Compared to Windows, this approach decreases the setup time significantly by avoiding multiple levels of user interfaces.
2. The terminal is capable of storing a history of previous input commands. Test subjects can simply recall any previous input commands by pushing the up arrow button on the keyboard. In the test scenario, for setting ACL entries with multiple users and files, this feature can be put to use extensively for maximum efficiency. Setting ACL for a file is done by one command: setfacl –m u:[username]:[rwx] [filename]. Although a maximum of three parameters can be changed, typically only one parameter differs from one command to the next. This results in a vast improvement in efficiency.
3. Upon completion of the ACL setup, test subjects are able to review the results in text format by executing the getfacl [filename] command. The output is clear and simple, allowing test subjects to check for any mistakes with ease.

As the number of files to be secured in a system increase, the ACL setup duration changes dramatically. The steps required to change the Linux and Windows' ACL is linearly related to the number of files and users. Each step is defined to be typing a word or clicking a mouse button. The numbers of steps in order to complete one ACL entry can be calculated as follows:

Case 1

Assumptions:

Total number of files (n) = 5000

Total number of users (u) = 20

Linux users are not taking advantage of the history feature in the terminal.

Windows:

1. Best case scenario: One permission level granted for every user to each file

Number of steps

= minimum step per file* number of files + number of users * number of files * number of permission changes per user

= 7 * 5000 + 20 * 5000 * 1

= 135000

2. Worst case scenario: Three permission levels granted for every user to each file

Number of steps

= minimum step per file* number of files + number of users * number of files * number of permission changes per user

= 7 * 5000 + 20 * 5000 * 3

= 335000

Linux:

1. Best case scenario: Same permission granted to all users for every file

Number of steps

= (minimum step per user + number of files) * number of user

Minimum step per user includes: setfacl –m u: [user] : [permissions]

= (5 + 5000) * 20

= 100100

2. Worst case scenario: Six combinations of permissions are evenly distributed

Number of steps

Given:

There are totally six kinds of rights: r, w, x, rw, rx, rwx

Therefore, the probability of each kind of rights, P(rights) is 1/6.

= ((number of files * P(rights) + minimum step per user) * total number of rights) * total number of users

= ((5000 * (1/6) + 5) * 6 * 20)

= 100600

As shown in Figure 6 and 7, Windows requires a growing number of steps to configure ACL than Linux as the number of files increases. In both cases, the number of steps required to change the ACL on Linux is much lower than on Windows.

Best Case Scenario for Setting Up ACL in Linux and Windows with 20 users and 5000 files

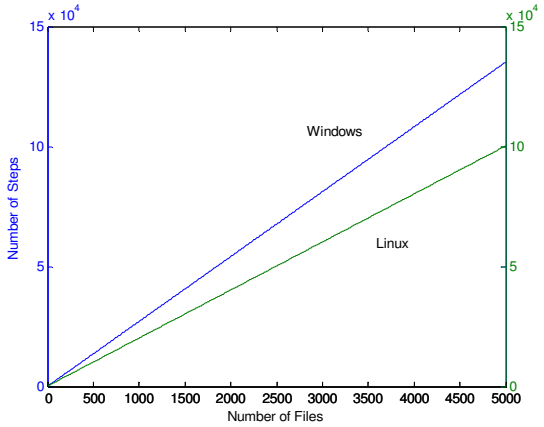

Fig. 6. Number of steps is as a function of number of files. As the number of files go up, Linux require less time to configure ACL than Windows.

Worst Case Scenario for Setting Up ACL in Linux and Windows with 20 users and 5000 files

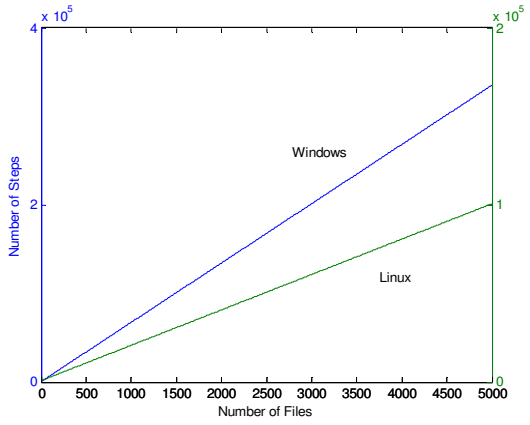

Fig. 7. Number of steps is as a function of number of files. As the number of files increases, Linux require much less time to configure ACL than Windows.

Case 2:

Assumptions:

Total number of files (n) = 5000

Total number of users (u) = 200

Calculation is the same as case one Case 2 reiterates the calculations done in Case 1, but with an increase in the number of users. Figure 8 shows that the performance of Linux and Windows are almost identical in the best case scenario. However, as shown in Figure 9, Linux is more efficient in the worst case scenario, even as the number of users increases.

Best Case Scenario for Setting Up ACL in Linux and Windows with 200 users and 5000 files

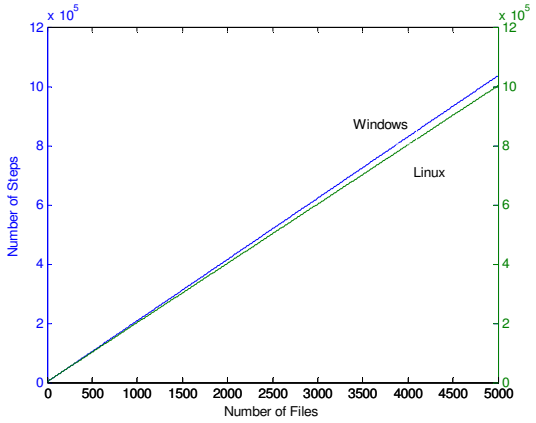

Fig. 8. Number of steps is as a function of number of files. When the number of user increases, the step requires to setup ACL is almost the same in the best case scenario.

Worst Case Scenario for Setting Up ACL in Linux and Windows with 200 users and 5000 files

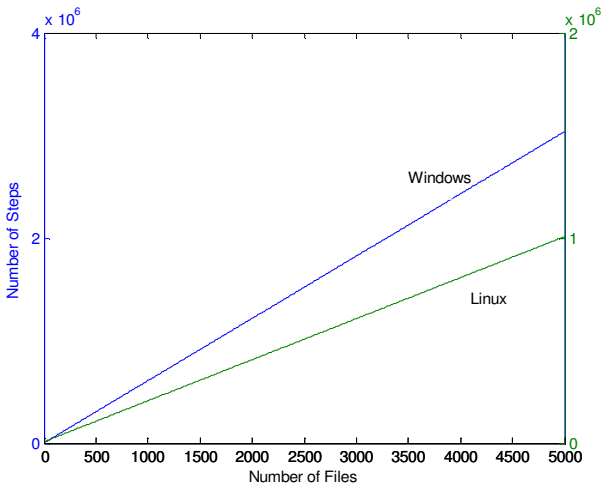

Fig. 9. Number of steps is as a function of number of files. In the worst case scenario, even when the number of user increases, Linux requires much less steps to achieve.

C. *Windows*

The experiment showed that users get acquainted with Window's ACL quicker than Linux. This is due to the fact that there are no commands to learn; all commands and actions are shown explicitly on the graphical interface. The help documents are also very well structured, with easy to navigate menus and hyperlinks.

The set of permission levels in Windows ACL is also greater than those in Linux. In simple mode, there are six permissions: modify, read, write, read and write, execute, and full control. Having these extra options allow the users more flexibility when setting the ACL entries.

Compared to Linux, setting the same permission levels of all files and subfolders in a folder can be done easily in Windows. When permissions are granted to a folder, the permissions are also applied to all files and folders in the folder hierarchy.

However, there is a major downside to the Windows ACL. The process of setting up access control lists is more complex and requires additional time for completion. Since Windows' graphical interface aims to achieve high user friendliness, a limited set of action and commands are given to each window for simplicity. This also means users must traverse through more windows before reaching the intended destination. In addition, users must manually add other users to a file before granting them permissions. Both of these factors contribute greatly to the fact that setting up ACL on a Windows machine consumes more time.

VI. Possible Development

From the results obtained from the experiment, it is safe to conclude that the usability of ACL in Linux systems is considerably higher than ACL in Windows. The implementation and user interface of ACL in Fedora Core 2 is by no means flawless; indeed, it can be said that much improvements can still be made to improve usability.

A. *Documentation*

Better documentation accessibility and layout will dramatically reduce the learning time for novice users. Linux

comes with a large set of very detailed and technical documentation [2]. However, to access these documentations, users must first be acquainted with the commands for displaying these documentations. There are two commands that are of use to novice Linux users: "--help" and "man [command]". On most modern Linux operating systems, when a user incorrectly uses a command due to syntax error, the terminal automatically displays the help menu. The help menu is quite short and uninformative; the manual contains much more variety of information, including technical details and real examples on how to use the specific commands and options associated with that command. A simple, effective way to inform users of the existence of the "man" command is to include it in the help menu, along with a short description of what "man" accomplishes [3].

The manual option for commands is very detailed and technical, but complaints about the readability and the length of the documents are frequent during the experiments. The format of the manual is usually very technical and dull, and the terminal displays the contents in a command line context. After browsing through the manual and failing to find the appropriate material, test subjects in the experiments becomes uninterested, get frustrated, or get discouraged. One method to improve the effectiveness of the manuals is by developing a better graphical layout for the manual. A full blown help and support center such as the one in Windows may also be adopted to centralize all help and manuals for easier access. Such a layout will improve usability of the manuals and help users find specific sections efficiently.

B. *User Interface*

Linux systems rely heavily on a command line interface, as opposed to a graphical user interface such as Windows machines. There are many advantages and disadvantages of operating in a command line interface environment. A command line interface presents users with more control and improved speed, and also allows users to easily create scripts to perform some specific task [4]. In contrast, the major drawbacks of such an interface are that it sacrifices ease of use and the ability to multitask.

Speed and control is achieved by having one-line commands for simple and complex actions. An advanced Linux user using the terminal is able to perform complex actions faster than an advanced Windows user performing complex actions on Windows systems. This is due to the fact that multiple mouse and keyboard actions are replaced by one line of command. The ability to create a script easily also significantly raises the usability of ACL in Linux systems; setting the same ACL's on multiple machines would be a simple matter of executing a previously created script.

The issue of multitasking is irrelevant in improving the usability of ACL. However, the ease of use becomes an important issue for novice users. Novice users often find it much more difficult to successfully operate a command line interface due to the memorization and familiarity needed to operate it. In comparison, graphical user interfaces are much easier to pick up and learn since the users do not need to memorize any commands. The actions that can be performed in a graphical unit are all laid out in the form of tabs, buttons, and checkboxes.

The analysis above poses some profound questions on how to improve the user interface on Linux systems. Both schemes of user interfaces have their own advantages and disadvantages. The main disadvantage of a command line interface is its ease of use. To improve ease of use, a graphical interface can be implemented for setting ACL entries. On the other hand, this change would completely eliminate the advantages that command line interfaces deliver. Therefore, one implementation of an improved ACL is to combine the two schemes into a hybrid model. This concept involves integrating a command line interface into a graphical unit. Based on the user's experience in command line interfaces, they may choose to use the graphical section of the unit first. The specific commands corresponding to the user's actions is shown in the command line section. In another word, the unit acts as a translator between actions performed on the GUI and specific commands.

Once an ACL entry is created by interacting with the operating system through the graphical unit, the corresponding "setfacl" command and syntax would be outputted onto the command line section. For further assistance, users may choose to select an option which explains the syntax and options of that command. Such an option will significantly reduce the learning time for novice users, since it frees user of the task of reading through the manual.

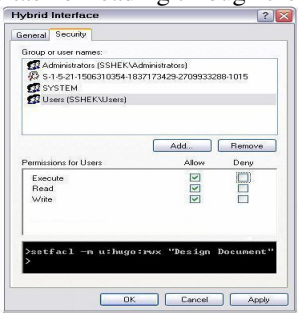

Fig. 10: Hybrid Interface for ACL

C. *Access Control Options*

On Linux machines, there are only three permission options: read, write, and execute. On Windows, there are six permissions: full control, modify, read and execute, read, and write. This unnecessarily complicates the process of configuring access control lists; the options of read, write, and execute are more than enough for most purposes.

In addition, test subjects sometimes assume that giving a user some permission to a folder will grant that user the same permission to every file and subfolders. Therefore, a useful option is to prompt the user and let the user choose to whether to set all files in the folder hierarchy to the same permission as the folder. For some applications, this option will improve usability by improving speed and efficiency.

REFERENCES

[1] L. F. Cranor and S. Garfinkel, "Security and Usability," O'Reilly, pp. 634-637, August 2005

[2] Scorpion City (2001, Feb.21) *Linux Tutorial*. [Online]. Available: http://www.scorpioncity.com/linux.html

[3] J. Valade, "Spring Into Linux," Addison Wesley Professional, pp. 245-256, April 25, 2005

[4] Computer Home (2005, Dec 1) *Command Line vs GUI*. [Online] Available: http://www.computerhope.com/issues/ch000619.htm