INTERNATIONAL JOURNAL NATURAL AND APPLIED SCIENCES (IJNAS), VOL. 4, NOS.1& 2 (2009); P. 93 – 100, 2 FIGS.

Analysis of the UNIX operating system and improvement of the password authentication technique

I. E. Eteng*[1], F. U. Ogban[1] and H. Bassey[1]

ABSTRACT

The UNIX operating system is an operating system that safeguards against illegal access and other threats to the computer system. In this paper, the UNIX file system is analyzed, the security weaknesses are x-rayed, an improved one-time password authentication technique is presented, and the underlying model used for the design is described. Moreover, a password authentication programme was designed which implements an improvement of the general one-time password technique. Passwords, which are individually selected by users from a codebook are now randomly selected by the system for the user in the improved programme. Real-data entries into the programme demonstrate an enhancement of the security of the system even on the event of the loss of the codebook.

INTRODUCTION

An operating system presents the computer user with an equivalent of an extended machine or virtual machine that makes it a lot easier to programme and make general use of the computer. This set of manual and automatic procedures also enable a group of people to share a computer installation efficiently. Most times people compete for use of physical resources such as processor time, storage space and peripheral devices; at other times people can co-operate by exchanging programmes and data on the same installation. The operating system makes these activities tolerable.

An operating system must have a policy for choosing the order in which competing users are served and for resolving the conflicts of simultaneous requests for the same resources; it must also have a way of enforcing this policy in spite of the presence of erroneous or malicious user programmes and access(Per, 1990). The simultaneous presence of data and programmes belonging to different users requires that an operating system protect users against each other. This task the operating system must perform automatically.

The UNIX operating system is a multitasking, multi-user and highly portable operating system that provides a powerful and hospitable program development environment. It controls the computer resources and provides a base upon which the application program runs. The UNIX operating system uses a hierarchical file system that is organized as a tree with the root node called "The Root" and represented by a single '/' (slash). The hierarchical file system has the 'root' file system at the top of the hierarchy of files and this file system is the key to the UNIX operating system. File systems often contain information that is highly valuable to their users. Therefore protecting this information against unauthorized usage is a major concern of all file systems with UNIX inclusive.

One of the most important security features used today are passwords. It is important to have secure, ungues sable passwords.

However secure and unguessable the password may seem, it is pertinent to have in place a system that can authenticate the password whenever it is being used to log on the system..

However, the problem with passwords is that they are easily transferable with the owner's connivance and most times unfortunately, without owner's permission. Therefore, though passwords have been and are used widely in the computer world, it is the easiest to compromise. However, the real problem with the use of passwords is that they are transferable and substantially static. Users with or without agreements can end up transferring the password to a third party who can then pretend to be the genuine user. Indeed the worst feature of all is the inability after the event to prove what really happened. There exists no simple way to determine who actually gave the password. The vulnerability of passwords is due entirely to their predictability. The adoption of procedures (authentication techniques) that reduce this must form the basis of any security strategy.

The UNIX operating system was the fallout of the quest by MIT (MASSACHUSETTS Institute of Technology), Bell laboratories and The UNIX operating system's main thrust was providing a convenient working environment for programming. In addition to gaining wide acceptance, particularly in the academic world UNIX has influenced the design of many modern Operating Systems.

UNIX has a long history as an open development environment. UNIX performs the typical operating system task, but also includes a standard set of commands and library interfaces. The building block approach of UNIX makes it an ideal system for creating new applications. The traditional operating system consists of a small kernel that runs processes such as user applications and services.

* Corresponding author
Manuscript received by the Editor August 3, 2007; revised manuscript accepted July 21, 2008.
[1]Department of Mathematics/Statistics and Computer Science University of Calabar, Calabar, Nigeria

The UNIX kernel is a solid core that changes little from system to system, while processes are added at the user's discretion. This makes upgrades easier since the entire operating system does not need to be recompiled.(Thomas, et al 1996)

Motivation

The major motivation for this work was that in spite of the fact that the UNIX Operating System is over thirty seven (37) years old, the UNIX operating system has continued to attain wide spread popularity. Though traditionally used on minicomputers and workstations in the academic community. UNIX is now available in personal computers. Previous PC and mainframe users are now looking to UNIX as their operating system solution.

Another feature of UNIX that motivated this work is that the UNIX implementation now includes TCP/IP and support for Ethernet. UNIX therefore provides in one package the ability to install a powerful operating system on a computer that lets user's computers through one of the most common and powerful networking protocols in the industry.

Our contribution

Our major contributions in the work include the following;

(1) Knowledge of the security flaws in the UNIX operating system has been highlighted.

(2) Knowledge of the counter measures against security weakness of the UNIX operating system has been gained.

(3) The UNIX password system has been improved upon.

(4) An intrinsic knowledge of the UNIX file system has been provided.

BACKGROUND AND LITERATURE REVIEW

UNIX is a trademark of AT and T Bell laboratories now known as Lucent Technologies (Bourne, 1983). According to the designers, the file system is the key to UNIX. It offers compatible devices, file and inter-process input/output. In essence, the user simply sends and receives data. All data are treated as strings of bytes and no physical structure is imposed by the system, instead by the data. The result is a considerable freedom from any concern for physical input/output (Davies, 1992).

UNIX is a multiple-user operating system in which commands are processed by a shell that lies between the user and the resident operating system (fig. 2.1). The shell is not really part of the operating system. The idea of a command processor that is independent from the operating system was an important UNIX innovation (Sobell, 1989).

A UNIX user communicates with the system through a shell. Essentially through a command interpreter, the shell is treated much like an application programme and is technically not part of the operating system (Sobell, 1989). This allows a user to replace the standard shell with a custom shell. The Bourne shell can be replaced with a custom shell having a graphic user interface (GUI) with icons or menus replacing traditional commands (Manger, 1992).

Among its resident modules, UNIX contains an input/output control system, a file system, and routines to swap segments, handles interrupts, schedules the processor's time, manages memory space and allocates peripherals device (Bourne, 1983). Additionally, the operating system maintains several tables to track the system's status. Routines that communicate directly with the hardware are concentrated in a relatively small kernel (Fig2.1). The kernel is hardware dependent and varies from system to system. However, the interface to the kernel is generally consistent across implementation. UNIX is a time-sharing system with programme segments swapped in an out of memory as required (Bourne, 1983). To ensure reasonable response tune, processor access is limited by time slicing. Segmentations is the most addressing scheme, and most UNIX systems implement virtual memory techniques (Sobell, 1989).

The key to the UNIX operating system is the file. UNIX handles files as it has done since its inception in 1969, namely by allowing users' access to them from the command line. Secondly the new graphical user interface that sits atop the UNIX allows the handling of files graphically through the use of icons (Southerton., 1993).

General feature of UNIX file system

The UNIX file system manages all the data stored on the System's mass strong devices. UNIX provides a built-in protection system against unauthorized access to file by allowing the root or super-user to assign permission to file (Southerton, 1993). From the command line, the user can determine the permission level of a file by using the " is " command (list command) to list the file and examine a coded format (Andrew, 1990).

Hierarchical structure: The UNIX system organizes the file using an upside-down hierarchical tree structure. All files will have a 'parent' file, apart from a directory called the 'root' directory, which is the parent of all file on the system. The hierarchical component also adds to the dynamic flexibility of the file-system.

Structure less files: Files are also said to be structure-less, since the utility that creates the file normally dictates the internal format of that file.

Dynamic file expression: The file-system structure is dynamic. Its size is not determined by any rule other than the amount of disk storage that is available on the system. A file size can be changed at will by the user at any time.

Security: Files are protected using file ownership mechanism. This allows only a specific class of users to access certain files.

The shell and shell scripts

The UNIX Shell is a customized command line interpreter. UNIX commands are processed by a shell, usually, the shell starts a command as soon as it is entered and then waits for it to terminate before displaying the next prompt. The idea of a custom shell was an important UNIX innovation (Sobell, 1989). When a user logs on, the shell overlays the login, process text and data segment, but the system data segment is not affected. Thus the shell standard input, output and error files are open. The result is that the user can begin issuing commands without opening these standard files (Bourne, 1983).

Many data processing applications are run daily, weekly or at other regular intends. Some are repeated many times for example a programme treat. When such applications are run, a set of command must issued repeating the application means repeating the commands. Retyping the same commands repeatedly can be frustrating and error prone. The option is to write a shell script a shell scripts is to write a shell script is to write a shell script. A shell script is a file that consists of series of commands. The shell is actually a highly sophisticated interpretive programming language, with its now variables, expressions, sequence, decision and repetitive structure
(Southerton, 1993).

Security issues

In the ever-changing world of global data communication, inexpensive Internet connection and fast paced software development, security is becoming more and more an issue. Security is now a basic requirement because global computing is inherently insecure. Unintended individuals may gain access to a computer and maliciously intercept, alter or transform data into something not intended.

Additionally, unauthorized access to the system may be obtained by an intruder, known as crackers, who then use advanced knowledge to impersonate, steal information or even deny the legitimate user access. Through the need to ensure security of a system is worthy, however it should be noted that no computer system can ever be completely secure. All that can be done is to make it increasingly difficult for a security compromise to occur.

Unfortunately, UNIX was not designed with security is mind, but a UNIX system can be made secure if the correct procedures are adopted. The problem of Security is UNIX's flexibility as an operating system (Pfleeger, 1989). Its versatile file-system structure allows users to browse extensively through many of the systems file. It is also commonly found that non-privileged user have access to administrative tools simply because the correct access permissions is not set on the relevant file. The issues is UNIX security can be viewed as two categories; Issues related to protecting the system from the user/owner/ any multi-user system requires real security among other things protect user from greenhorns.

The most important way to safeguard a system is to limit access to dangerous functions (Wood, et al., 1983). This can be achieved by login as root only as root only when absolutely necessary and by creating administrative logins for each of the system administration functions (Southerton, 1993). Equally important is the need to ensure that the root password is known to only trusted persons.

File system security

Despite very good efforts at establishing and implementing a good security strategy, the operating system can still be broken into. A cracker's goal is to ensure continued access once access has been gained by breaking a user's password it could be charged to something more secure. Another way the cracker might ensure continued access is to install new accounts on the computer. If access is gained by breaking a user's password it could be changed to something more secure. A good file system security helps prevent or detect these modifications and discovery from a break in (Lane, 1993). System configuration files may be writable by users other than the root. Also device files may have insecure file permission and programmes, furthermore, configuration files may even be owned by user other than root.

Configuration files writable by non-root account may allow a cracker to alter or changes system memory to gain more privileges, snoop terminals, or by pass the normal UNIX files protection to read files from or alter information on deal or tape storage (Smith, 1994). A cracker can alter account. This is one of the reasons most breaches of UNIX security take place at the file level because access permission settings are not set correctly when the system is installed or because file permission settings are inevitable changes a time for various reasons (Manger, 1992). The failure to reset the permission correctly leads to all kinds of security breaches. For instance the "chimed" (change mode) command is used to alter access permission settings of a particular file or group of files. The syntax is

Chimed [options] <mode> <file…..> (*).

It is ideal that a UNIX system mountains proper file system security (intension prevention), and also a means to detect unauthorized file system changes (intrusion detection).

File permissions

Security in UNIX is centered on the UNIX unified file system concept. This is a concept, which treats device drivers, text files, and communications channels are being streams of data accessible via a file mane in the directory. Each UNIX file has a set of three permissions. Via:

- Owner (U) Rights the owner has to access the file.
- Group (g) Rights the owner's group have to access the file.

- World (o) Rights other users have to access file.

This information is stored as a series of flags together with the numeric ID of the owner and group of the files in a structure known as the "inode" associated with each other.

User authentication

Many protection schemes are based on the assumption that the system knows the identity of each user. The problems of identifying users when they log in is called User Authentication, most authentication methods are based on identifying something the user knows, something the user has, or something the user is.

One of the most important security features used today are passwords. Passwords are widely used form of authentication in which the user is require to type a set of alphanumeric characters which is then mapped by the system before login is permitted. Password protection is easy to understand and easy to implement. In UNIX it works like this. The login programme asks the user his name and password. The passwords are immediately encrypted. The login programme then reads the password file until it finds the line containing the user's login name. if the encrypted password contained in this line matches the encrypted password just computed, the logins is permitted, otherwise it is refused.

Bourne (1983) made a study of passwords on UNIX systems. They compiled a list of likely password: First names, Last names, street names, city names, words from a moderate – sized dictionary, valid license plate numbers, words spelled backwards and short strings of random characters.

Each of them were then encrypted using know password encryption algorithm and checked to see if any of the encrypted passwords matched entries in their list. Order 86 percent of all passwords turned up in their list. Therefore it is important that passwords should be as secure and unguessable as possible one way this can be achieved is to require / encourage users to pick better passwords and by having the computer offer advice. Some computer have a programme that generate random easy to pronounce nonsense words that can be used as passwords. Eg fotally, garbuNgy or BipItry (some with upper case and special characters). The most extreme form of password security measures is the one-time password. When one-time passwords, are used, the user gets a book containing a list of passwords. Each login uses the next password in the list. If an intruder ever discovers a password, it will not do him any good, since next time a different password must be used. The real problem with the use of password is that they are transferable and substantially static. User with or without agreements can end up transferring the password to a third party who can then purport to be the genuine user.

It goes almost without saying that while a password is being type in, computer should not display the typed in, the computer should not display the typed characters to keep them from prying eyes near the terminal, unencrypted in the computer and even computer center management should not have unencrypted password copies.

Hence, we distinguish this paper by the following contributions.

The standard login command improves password security in two ways:

- Incorrect login name response does not cause immediate errors, thus preventing a remote hacker from rapidly determining that a certain login name is valid on the machine.
- Password entries are not echoed (printed) by UNIX.
  UNIX.

UNIX password are stored in encrypted from in the / etc / password file. While it is difficult to invert the cipher and procedure the plaintext version of a password, it is comparatively easy to encrypt a selection of possible password and compare them against the encrypted string in / etc / passwd, a favourite crackers ploy. UNIX attempt to lesson the severity of this attack by using a seed value produced at the time of password change to modify the standard DES algorithm to frustrate the use of hardware DES chipsets. This seed is stored with the encrypted password in / etc / password. Thus two users may have identical passwords, but due to differences in seeds may have different encrypted forms. Hence plaintext search may break one user's password but give no due to the fact that the other user's password is identical.

On recent version of UNIX, the programme / etc / pwck (password validation) checks the password file for any inconsistencies (Ferbrache, et al, 1992). The check include validation of the number of fields, login name, user ID, and whether the login directory and the programme / etc / grpck checks entries in the group files. The checks include validation of the number of fields; group name, ID and whether all login names appear in the password file. These programmes should be run whenever a change is made to the password or group file (Ferbrache, et al, 1992).

METHOD OF STUDY

General one-time password aging

The general one time password aging mechanism requires the user to access the system with a new password during login depending on the password life span. Usually the user is issued with a codebook containing a list of password that is used serially and each used password crossed off the list. The next password is then used at the next login. The major drawback of this technique is the requirement to use the password list serially and cross off. This makes it easy for password sniffers, hacker (in the case of loss of the codebook) to determine which password is to be used next.

A model to imitate the implementation of an improved administrative technique, which protects against this drawback, is created.

Simulation models

In our work, we created models which formed the basis of the design used for this work.

This one-time password aging technique modelled in fig.2 is an improvement on the general one-time password aging mechanism. In this model the new user (A) is made to undergo an identification procedure with the system administrator (B). after the identification procedure is completed, a list of serially numbered passwords is generated and printed out for the new authorized user (C) in the form of a codebook. The user (C) uses the first randomly chosen password in his codebook at first long-in. simultaneously the user account (D) demands the serial number for the next login password from the system administrator randomly picks a password serial number, which it sends to the user account (D). As user (C) log's out, the user account (D) gives a prompt, which displays the next password login serial number before log-out is completed.

As an added security measure, the user who owns the codebook is advised not to cross off used passwords. This feature is employed so that if the book falls into the hands of unauthorized persons, they will have a hard time guessing which password has been used or which hasn't. The system is programmed to shut down if a wrong password is entered up to three times.

System design

The one-time password aging mechanism was originally designed in a way that authorized users serially picked their passwords from a list of numbered passwords were crossed off.

In this system designed, passwords are not picked by the individual user, but instead they are randomly chosen by the system administrator for the user's use. Coding of this mechanism would require the in-corporation of a number of modules.

Password generator

This module generates passwords using a combination of permissible characters e.g. upper and lower case letters, digits, punctuation characters, control to form a password list of a specified length. Password generated have keys i.e. serially numbered, generated passwords are stored is a codebook.

Random number generator

This generator generates passwords characters randomly to from passwords which make up the password list. This generator also randomly picks or generates password keys, which belong to specific passwords. Passwords generated are allocated to user for subsequent logins.

Allocator

This module makes use if the random number generator to generate a password key by which belongs to a password from the password list. Chosen passwords are screened to verify if passwords have already been used. If yes, another password key is generated until an unused password is gotten. The password is then allocated to the user.

Merits of the improved one-time password

The improved one-time password has a number of merits and advantages over the general one time password. In the improved one time password mechanism passwords are randomly chosen or selected by the system from the codebook for the user to use, whereas in the general one time password mechanism the passwords are chosen by the user himself for use. After use, the user crosses off the used password from the list of serially number passwords in the codebook. The loophole in this technique is that if a user misplaces his codebook or the codebook happens to be stolen, a malicious user or even a hacker can easily log into the system easily without any sweat by simply following the password sequence-using password next to the last crossed off password. The hacker gains continued access to the system since he now has the codebook and the system sees him as a legitimate user.

However, in the improved one time password technique, the password sequence is not known although it is serially numbered. Users are also advised not to cross off used passwords as an added advantage. The legitimate user or a hacker does not know the order in which the passwords are chosen since they are randomly selected by the system from the codebook for use.

Note: The system also has a copy of the codebook. If by chance the codebook is misplaced or stolen, the hacker will have a big problem deciding which passwords have been used and which passwords have not been used as well as not knowing which is the next valid password for login.

If the hacker keys in the wrong password thrice in an attempt to grind out the password selection sequence or the next valid password, the system automatically logs out and deactivates that particular account. As an added advantage when the system randomly picks and gives out a password, it does not give out the password itself but the password number or key. The user knows which number of key belongs to which password using his codebook. Onlookers or spies will find it difficult knowing which is the actual password attached to the number seen since they do not have the codebook.

Simulation of the password generator

The password generator was simulated using the C++ programming language. The major modules of this simulated programme were;

1) A codebook generator
2) A sample log in test utility

3) An exit option

Using the codebook generator

The codebook generator creates a user codebook consisting of five passwords. To generate a codebook, select the 'Generate new codebook' option from the main menu by entering a 1 at the menu prompt. A codebook is created for the specified user. The codebook created for the user is named in the following format; (user name) codebktxt, and is stored in the current working directory (usually the directory in which the programme was execute), where the system administrator can access it.

Using the 'login' test utility

The 'Log in' test utility is part of the password generator programme, it provides the user with a platform to test the access control operations of the password generator making use of the passwords generated for a given user by the book generator.

To utilize these features, select the "Log in" option from the programme main menu by entering a 2 at the menu prompt. The test utility prompts for a username, which should be the name of a user with an existing codebook. Following the user password to be provided must come from the codebook that was generated for that user.

Note: Once a password is used, it cannot be reused (For this reason, they are called one-time passwords).

The test utility allocates a password key to the user through which the user determines the next password to use in the next "Login" session. Also, if all password needs to be generated for that user.

The "Long in" test utility allows a maximum of three login attempts before a user with an invalid password is denied access.

To exit the password generator programme

Select the 'Exit' option from the main menu display and a user is automatically logged out.

RESULTS AND DISCUSSION

Creating a codebook for a user

When a user name has been specified, the generate subroutines of the password generator invoke the create Random Password routine repeatedly to create a list of unique passwords. This list of passwords is passed on to the SavePasswords routine of the File Handler module which saves the passwords to a codebook and then invokes the encode routine of the File Handler module to encrypt the codebook. An unencrypted copy of the codebook is also stored for use by the system administrator.

Logging in a user

When a user attempts to log in by providing a user name and password. The Allocator module is given the user name, which it uses to locate the user's codebook and this loads the password in the user's codebook through a call to the File Handler module's getPasswords routine. The checkPassword routine of the Allocator module is invoked with the user password as an argument. This routine (that is, the checkPassword routine) invokes the finPassword routine of the searcher module to locate the specified password from the list of passwords in the user codebook. On successful location of the password, the deAllocate routine of the masker module is invoked to mark the password as used.

The allocatePassword routine of the Allocator module is invoked to allocate a new password key to the user. This routine starts by getting a count of valid passwords left in the codebook through a call to the search module's gatecount routine. If any valid passwords are available, the random index pointing to any password in the user codebook. The crosscheck routine of the searcher module is called to confirm that the password at the randomly picked index is valid, if not valid, the process is repeated by invoking the random number generate again. On matching the index with a valid password, the allocatPassword routine assigns the index value to the user as the key to the next password.

CONCLUSION

The UNIX operating system is a portable multiuser and rugged operating system that provides a powerful programme development environment. In spite of the ruggedness of the UNIX operating system, this paper highlights the glairs of its password authentication system, gives an overview of the UNIX file systems, proposes on improved password authentication technique, provides a framework for the implementation of this technique and demonstrates a simulation of this technique.

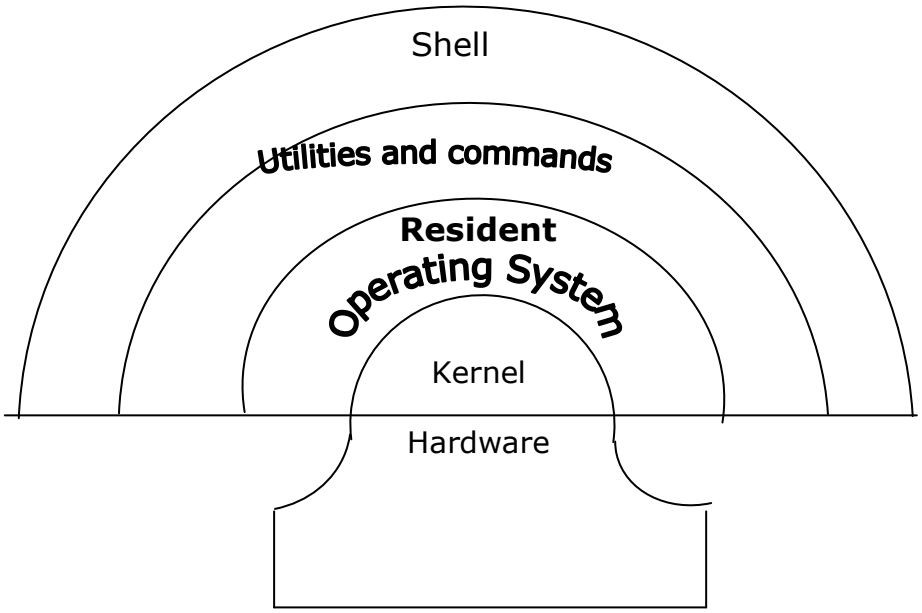

Fig. 1. Shell between the user and the resident Operating System

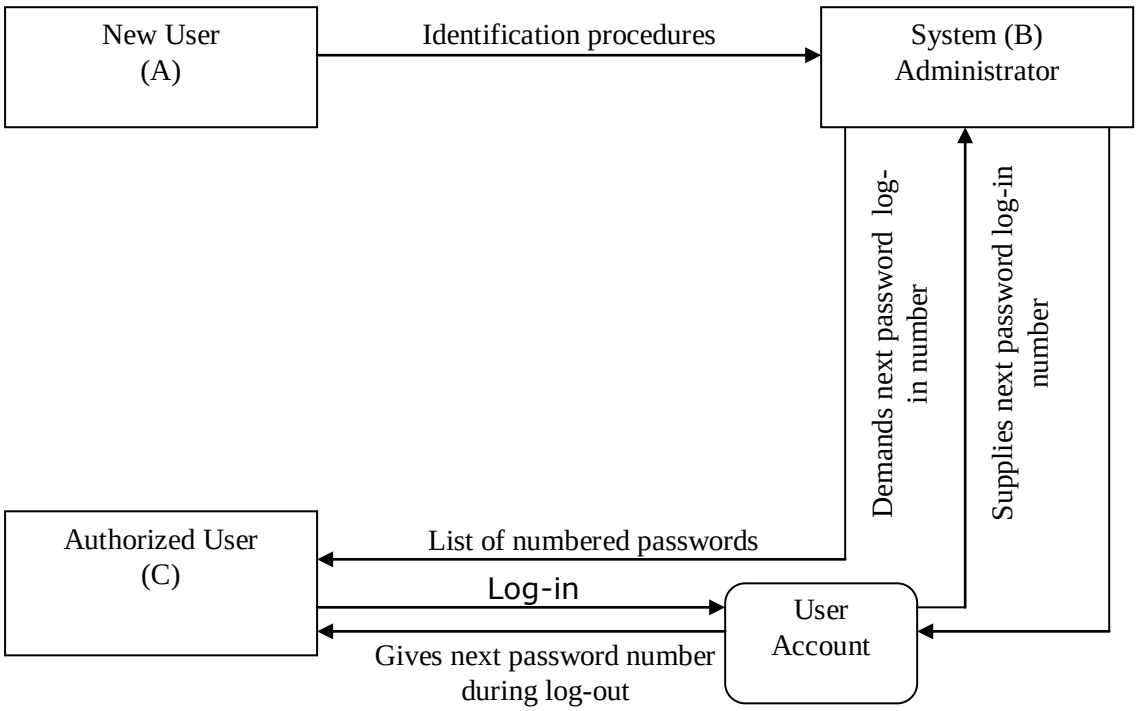

Fig. 2. Simulation model of the one-time password aging with an improved authentication technique

REFERENCES

Andrew, S. T. (1990). Operating systems: *design and implementation*, Prentice – Hall International inc., London.

Bourne, S. R. (1983). *The UNIX system*, Addison Wesley Publishing, New York.

Davies, D. W. and Price, W. L. (1989). *Security for computer networks*,(Second edition). John Wiley and Sons, Canada.

Davies, S. W. (1992). Operating systems: *a systematic view*, The Benjamin/Cumming Publishing Company, New York.

Ferbrache, D. and Shearer, G. (1992). UNIX system security, Butterworth-Heinemann Ltd., Oxford, London.

Manager, J. J. (1992). UNIX™ – *the complete book guide for the professional user*, Sogma Press. Wilmslow, England.

Per, B. H. (1990). *Operating system principles*, Prentice-Hall, International Inc., London.

Pfleeger, C. P. (1989). *Security in computing*, Prentice-Hall International Inc., New York.

Sobell, M. G. (1989). *A practical guide to the UNIX system*. The Benjamin/Comings Publishing Company,California.

Southerton, A. (1993). *Modern UNIX*, John Wiley and Sons Inc., Canada.

Thomas, R. and Yates, J .(1996.). *User Guide to Unix*, 2nd Edn. McGraw Hill (Asian Edition).