Access Authorization and Monitoring for Web Based Resources for e-Learning System for Secondary Schools in Tanzania

K. Mkocha, M. M. Kissaka and B. M. M. Mwinyiwiwa

Faculty of Electrical and Computer Systems Engineering
College of Engineering and Technology
University of Dar es Salaam

***Abstract*–This paper presents the development of a mechanism for the authorization and monitoring of data for a web based e-learning system. System data as well as the different user groups accessing the data have been defined. Constraints to use the LAMP solution stack, the UML models are coded in PHP which is embedded in the HTML. These interfaces are used as doorways through which different users, as per their role, can access and work on the system data according to their clearance. On the other hand, the administrator can see different reports, from system logs by simply pressing a button.**

***Index Terms*–access authorization, system monitoring, web based resources, e-learning system, SQL, HTML, moderator, administrator, illustrator, student, guest, user ID.**

I. INTRODUCTION

The growth in ICT has spreads to all facets of our lives, not leaving the education sector behind. We, however, have to take a ride on this technology carefully, not compromising our privacy and the integrity of the content on one hand but ensuring a graceful adaptation on the other hand.

The profile of the stakeholders of secondary education in Tanzania includes 95% students; 4.8%, teachers and the remaining few, the administrators and system supervisors [1]. Within this profile, the students, which are the majority, are mainly at the receiving end of knowledge. The teachers create the knowledge whereas the management team regulates the knowledge exchange process. There is a need to maintain this division of roles with the introduction of new technologies in education so as to reduce training requirements hence making technology a means to ease the learning process rather than a burden.

On the other hand, it has happened in many cases that only after a problem has occurred one discovers that there had been signs of the impending trouble for a long time back. In some cases this has happened as a result of one not knowing what danger a certain hazard can lead to, but, in other cases, it has been due to one not even knowing whether a certain situation is a security hazard.

In the first case, the mere presence of a monitoring mechanism could solve the problem. This implies that, by just watching what is happening in a system, one can tell when something is not right. The problem with this simple solution is that the problem makers have become tactical and to keep up with them, one has to always find ways around not only the first scenario but the second as well [2,3].

The situation thus necessitates for a monitoring that can detect policy violations and go a step further to assess threats to the system security. In addition, a monitoring mechanism should be able to detect malicious attacks, to check the system performance, to maintain the system configuration, to keep system accounts (useful for forensic analysis) and to allow for timely fault detection in a computing environment [4-6]. In this way, the corresponding system administrator can find problems and resolve them before serious damage occurs while at the same time being able to detect unauthorized accesses and deal with them accordingly.

II. WEB BASED REOSURCES AUTHORIZATION

Process Description

The aim is to have a system with mutual exclusive rights among roles. This is so as to prevent one super user having the clearance to do whatever one wishes to the system. Furthermore, security by obscurity is intended where any user, unauthorized to perform a certain task, can not access even an interface to the task. This is different from systems where the interface is always present but inactive for unauthorized entities or displaying messages of un-authorization.

There is, therefore, a PHP script in place that will probe the user for the authentication credentials and use this information to define the role of that particular user according to the information in SQL databases. Upon role definition, a customized HTML interface, according to one's role, should be presented to the user. In addition, each customized interface will have to verify the session (the authentication credentials and the role) just in case the user bypassed the normal entry point to the interface. There were five roles defined, namely, Administrator, Moderator, Illustrator, Student and Guest.

Administrator: System configuration, creation of Guest account, responsible for logs: generates reports from logs, rotates logs and archives them. Can modify the Administrator account information or delete the account

This research is supported by sida/SAREC under Engineering Capacity Building and sida/SAREC Extended Support Projects.

altogether. This is a new post that schools have to create to manage the non– academic aspects of the system.

Moderator: Student information management, overall in charge of system data except for the logs. The Moderator can create Administrator account if none exists. Creates Illustrator and Student accounts. Creates the Moderator account once and this account can not be deleted, only modified. Manages the rest of the databases. This role reflects the school administration team. Takes care of staff records. Creates announcements for staff and students.

Illustrator: This role is equivalent to that played by the academic staff in a secondary school. The Moderator creates learning materials, produces academic reports, exams, tests, quizzes. Takes care of student records. May create announcements. Can read announcements meant for staff.

Student: Learning Materials, Announcements, student information, student. As the name implies, this is where the student interfaces to the system. The student can access learning materials as well as the relevant student announcements and/or information. Can create one's own account and modify account profile.

Guest: This is the role of a visitor to an academic institution. The tasks are also in five groups, depending on the category of data in question. These are as tabulated in table 1.

TABLE 1: THE TASK PROFILE

Data group	Tasks
User Accounts	Create, Modify, Delete
Account information	Read, Modify, Delete
Lessons / Student reports	Upload, Read, Modify, Delete
Databases	Read, Append, Modify, Delete
Logs	Read, Append, Modify, Delete
Web documents	View, Edit, Delete

Assumptions

There are several assumptions that were made to ease the design of the interfaces. These are as described in the following paragraphs:

(i) All soft documents will be internally produced.
(ii) Hard documents will first be scanned before storage.
(iii) Identity spoofing is under control.
(iv) Requirements Definition

(a) System requirements

Input Output Requirements
An administrator should get the Administrator interface.
A moderator should get the Moderator interface.
An illustrator should get the Illustrator interface.
A student should get the Student interface.
A guest should get the Guest interface.
One wrong ID entry should get another chance to log in.
Two wrong ID entries should get another chance to log in.
If there have been three wrong ID entries, the login window should be closed and the script exited.

Technology Constraints: The software components of the system should be Linux (the operating system), Apache (the web server), PHP (the scripting language) and MySQL (the Database Management System).

Project Management: User documentation is required
Safety Requirements: There should be a session check on each page.
Legacy Requirements: All sessions should be timed. All sessions should be logged.

(b) Interface Requirements

The home page:
Should be visible to the public.
Should have a log in facility.
Should allow an existing user to a maximum of three wrong log in attempts.
Should allow for a visitor to create a Guest account.
Should allow for a visitor to create a Student account.
The Administrator interface
Should verify that the session accessing it is of administrator type.
Should allow the administrator to access system log.
Should allow the administrator to generate reports from the system logs.
Should allow the administrator to view the server logs.
Should allow the administrator to generate reports from the server logs.
Should allow the administrator to rotate and archive logs.
Should allow the administrator to interact with the logs database.
Should allow the administrator to create the guest account.
Should allow the administrator to edit the administrator account profile.
Should allow the administrator to view allowed web documents.

The Moderator interface:
Should verify that the session accessing it is of moderator type.
Should allow the moderator to create an administrator account if none exists yet.
Should allow the moderator to create Illustrator and Student accounts.
Should allow the moderator to modify the moderator account profile.
Should allow the moderator to interact with all databases except for the logs.
Should allow the moderator to view all learning materials.
Should allow the moderator to view allowed web documents.

The Illustrator interface:
Should verify that the session accessing it is of illustrator type.

Should allow the illustrator to edit the Illustrator account profile.
Should allow the Illustrator to delete (terminate) the Illustrator account.
Should allow the Illustrator to create Student account.
Should allow the Illustrator to register Students for the learning material.
Should allow the Illustrator to create lessons
Should allow the Illustrator to edit own lessons.
Should allow the Illustrator to interact with learning materials storage.
Should allow the Illustrator to view allowed web documents.

The Student interface:
Should verify that the session accessing it is of student type.
Should allow for a Student to edit the Student account profile.
Should allow the Student to view learning materials registered for.
Should allow the Student to view allowed web documents.

The Guest interface:
Should verify that the session accessing it is of guest type.
Should allow the guest to edit Guest account profile.

(c) Applications Requirements

The CONTROLLER
Should verify that authentication credentials provided by a user exist in the database.
Should not be accessible by any user.
Should prevent some characters in the place of username and password to prevent fatal commands to the system.
Should prevent passwords less than 6 characters long.
Should restrict passwords to the alphabet and numerals only.
Should allow not more than three login attempts by a user.
Should provide a user interface that corresponds to the role of a user.

The DATABASE MANAGER:
Should be accessed directly by the Administrator as an interface to the logs.
Should be accessed directly by the moderator as an interface to the rest of the databases.
Should be accessed by the LOGGER for log entry.
Should be accessed by the all other roles via script interfaces; particularly, the PROFILER for registering new users and the STUD_ILL_INT for interfacing Student and Illustrator to learning materials in the database.

System Design
The system will have three tiers, namely the user-interface layer, the application layer and the databases. The uppermost, user-interface layer comprises the role specific HTML pages through which the users can access various allowed data items in the system. This access is however not direct but via the PHP application layer, which is made up of several scripts for user validation and interaction with the databases. Figure 1 shows a simplified block diagram of the system.

The database layer contains the actual data items, systematically arranged for fast and easy retrieval.

III. IMPLEMENTATION OF WEB BASED REOSURCES AUTHORIZATION

Upon entering the main page, a user is given an option to log in if he or she already has an account. The screen would contain that interface as shown in figure 2, at the left side of the page.

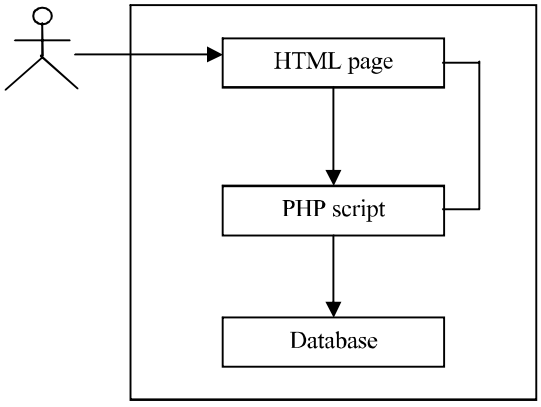


Fig. 1: Block Diagram of the Web Based resource Authorisation System.

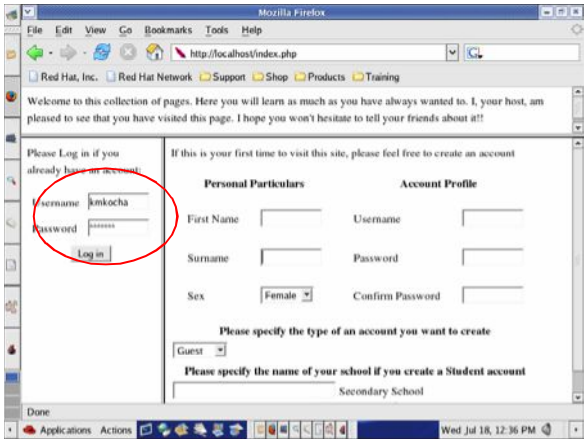


Fig. 2: User Interface for Login

On the other hand, if one has no an account in the system, he or she would, on the same page, be prompted to create, either a permanent Student account or a one session guest account as depicted in figure 3.

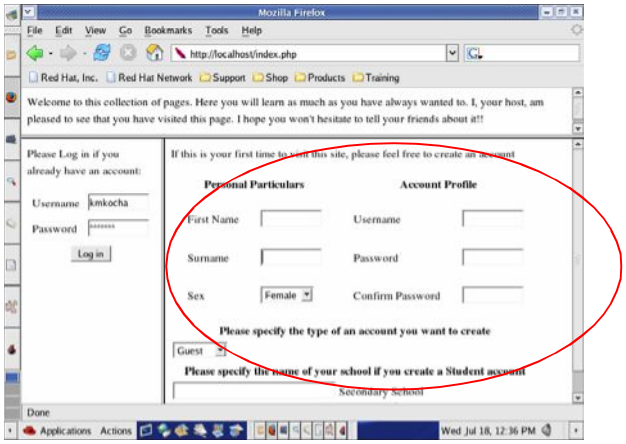


Fig. 3: User interface to create a New Account.

Upon successful identification, the user is taken to an interface that is customized per his/her predetermined role as in shown in figure 4 for the database administrator. On the other hand, if the identity validation phase fails, the user is taken back to the main page to start over, with a message that the first attempt was unsuccessful.

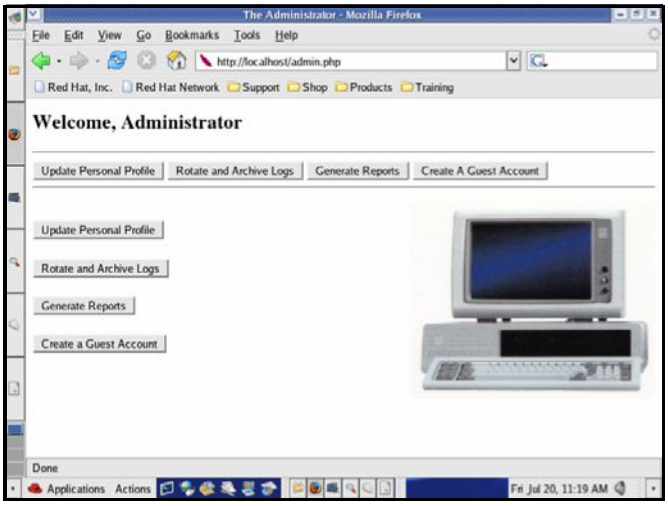


Fig. 4: User Interface for Database Administrator.

IV. DEVELOPMENT OF MONITORING MECHANISM

With the Apache web server, the only way to access monitoring information is by reading the log-files and in most cases the data in these files is meaningless; an analysis needs to be done on this data to get useful information from the logs and generating reports accordingly. Currently, most open source platforms do not provide this service.

The monitoring report generation program will enhance the administrators' awareness of network usage practices. In this way, the administrator will be able to promptly know what is going on in the system, where and when it is happening while being able to easily take the necessary action to prevent or stop it if undesirable.

The Apache web server utilizes the powerful standard logging server that comes with UNIX, namely Syslogd. Syslogd which reads and forwards system messages to the appropriate log files or users, depending upon the priority of a message and the system facility from which it originates.

Syslogd reads from the STREAMS log driver, /dev/log, and from any transport provider specified in /etc/netconfig, /etc/net/transport/hosts, and /etc/net/transport/services. It is also possible for syslogd, configured via syslog.conf, to point to a pipe instead of the messages file.

By default, apache has two log files which are access_log and error_log. To tell the logging facility what exactly should be recorded in these files, the LogFormat directive is used. By default the two log formats in use are the Common log format and the Combined log format. One can however define another format and call it whatever one wishes. Besides, one could create log files in addition to the default files or even direct the logs to an external program via the TrasferLog or the newer, CustomLog directives.

The monitoring mechanism employs the MONITOR, a PHP script which is to be accessible via the Moderator interface. The main purpose of its design is to produce relevant reports from database stored logs. This therefore necessitated the presence of the LOGGER, which would extract Apache logged data, parse it and load it accordingly to a relevant table. Furthermore, Log rotating and archiving were added to satisfy the legacy requirements of the current monitoring mechanism.

The first step was, however, to configure Apache such that the information logged was that of interest as per requirements. This information included: the remote host name (%h), time in common log format time format (%t), the first line of request ("%r"), the returned response status (%>s), bytes sent excluding the HTTP header (%b), the process ID of the child that serviced the request (%P) and the time taken, in seconds, to service the request (%T). It could be seen that this is the Common Log Format with additional two last fields. An example of such an entry would look as follows:

127.0.0.1 [18/Apr/2007:16:05:50 +0300] \"GET/ alumni.php HTTP/1.1\" 200 - \"-\" \"Mozilla/5.0 (X11; U; Linux i686; rv:1.7.3) Gecko/20041020 Firefox/0.10.1\"

Information in the standard log files is kept as a back up.

Requirements Definition

Legacy Requirements:

The error log should be monitored continuously using the command "tail -f error_log".

No one should have write access to the directory the logs are stored in without being purposely intended so.

Raw logs should be processed carefully since log files may contain information supplied directly by the client, without escaping making it possible for malicious clients to insert control-characters in the log files.

Security:

Logger program must be kept simple.

Code must be audited for vulnerabilities like buffer overflow.
Directory where the program resides must be owned by root, and non-root users should not have write permissions
Log size must be kept in control by rotating logs periodically.
Archived logs should be compressed to save disc space.
Raw log entries should be filtered and sanitized before storage to get separate fields for the log table.

As it was pointed out previously, all monitoring activities are the responsibility of the Administrator and hence all logs and monitoring scripts would be accessible only by the Administrator. The LOGGER would read from Apache log files, parse the lines into distinct fields and load the data into the Logs table.
The MONITOR would produce reports on system usage depending on:
The users: All, the Administrator, the Moderator(s), the Illustrator(s), the Student(s), and the Guest(s)
The Web documents: Creation, Modification, Deletion, Read / Normal Access.
Time frame: Hourly, Daily, Weekly, Monthly.
Format: Printable and/ or Graphical.

The flowcharts for the LOGGER and the MONITOR are as shown in figures 5 and 6.

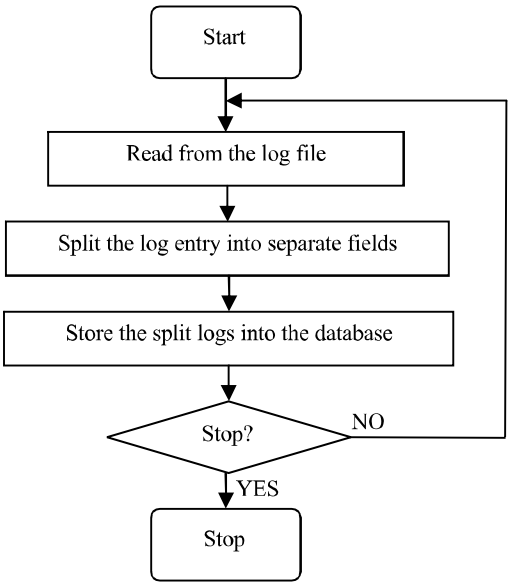


Fig. 5: Flowchart of the Logger.

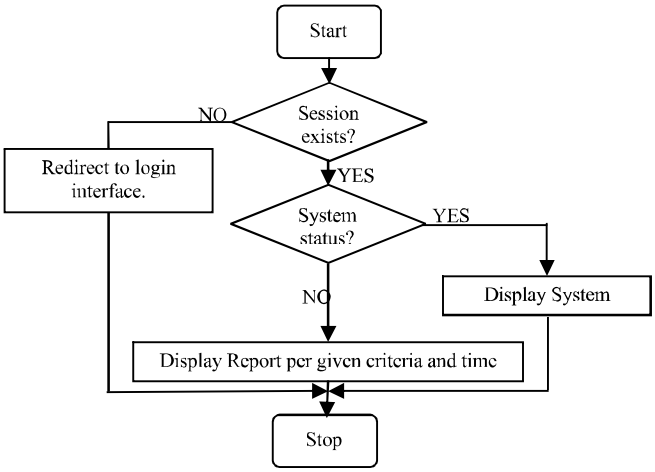


Fig. 6: Flowchart of the Monitor.

V. SYSTEM TEST RESULTS

To recall, the system is kept in place to make sure that the right people access the right information. In addition, the system keeps an eye on all that is happening within the server and report it to the administrator at his or her discretion. As an example, if the administrator wants to generate weekly access statistics within the server, a brief graphical summary would pop up in a new browser page and give some brief information as shown in figure 7.

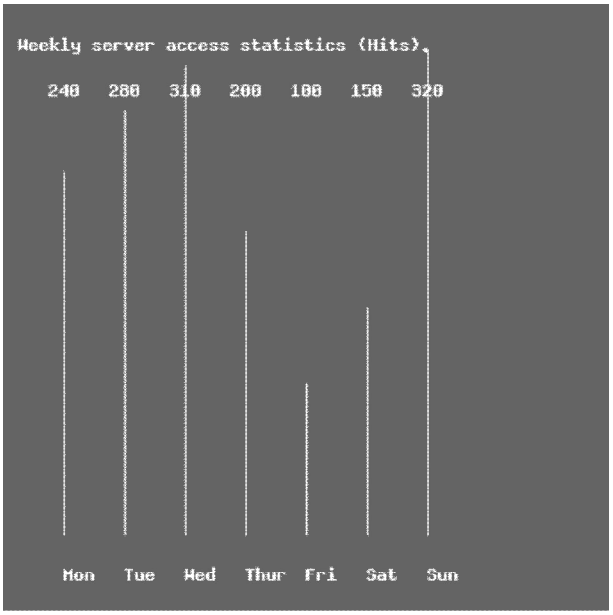


Fig. 7: A graphical report on server access statistics over a week.

VI. RELATED WORK

With regard to the need for authorization and monitoring in e–learning systems, the researchers have unanimously stressed on its necessity [8, 9, 10, and 12]. A proper authorization mechanism will ensure data integrity,

reliability [10] and above all accountability of the source of the data, which are important for an educational institution [8]. Nevertheless, authors suggested different mechanisms to solve the problem, including biometrics, digital certificates and access control lists [14]. The suggested mechanisms have been proposed to cater for a particular need in a given case study since there is a trade off between authorization and access controls in general, on one hand and the wide dissemination and availability of materials for information sharing on the other hand.

Furthermore, with the exception of biometrics, the data authorization implementation could be at the discretion of the user or mandatory according to some preconfigured rules [13]. It is almost a standard nowadays to have mandatory access control in place [10]. In this particular implementation, grouping of users according to their institutional responsibilities and the differences in data access needs has proven to be most effective in terms of resource usage efficiency [7, 11]. This is exactly the motivation behind role based authorization mechanisms in various institutions. In this particular case, the web based system was made to emulate the traditional education system and hence make it easier for the new system users to assimilate to the technology. This way, technology will have enhanced traditional duties thus improving performance [9, 10] on one hand while reducing the time required for training the users on the other hand.

VII. CONCLUSIONS

In this paper, the development of a mechanism for the authorization and monitoring of data for a web based e-learning system has been presented. System data as well as the different user groups accessing the data have been defined. Constraints to use the LAMP solution stack, the UML models have been coded in PHP which is embedded in the HTML. These interfaces are used as doorways through which different users, as per their role, can access and work on the system data according to their clearance. On the other hand, the administrator can see different reports, from system logs by using user-friendly interface.

The developed system is currently under tests. The test results will be included in the full paper.

VIII. REFERENCES

[1] Ministry of Education & Vocational Training "Basic Education Statistics in Tanzania (BEST); 2000-2006", (2006),. Available at http://www.moe.go.tz/zip/National%202006.zip.

[2] Tony Mobily, "Logging in Apache – Remote Logging", *Appress Publishing*, 2005.

[3] Prevelakis, V., Spinellis, D. (2007), "The Athens Affair", *IEEE Spectrum,* Vol. 44, no. 7 (INT), July 2007, pp. 26-33.

[4] Derek Feichtinger and Andreas J. Peters "Authorization of Data Access in Distributed Storage Systems", *Proceedings of 2005 Grid Computing Workshop*, Seattle, Washington, November 13-14, 2005, pp. 172-178.

[5] Michał Kosiedowski , Paweł Słowikowski, "Authentication and Access Control in Portals: The Progress Grid Access Environment', *Proceedings of 2003 Sun HPC Consortium*, June 20-21, 2003, Heidelberg, Germany.

[6] Steven, A., "The Security Monitoring and Attack Detection Planning Guide", Microsoft TechNet, 2005.

[7] Dow Jones and Reuters Company "How Role-based Integration Strategies Increase Content Management Utilization" *Factiva®*, October, 2005.

[8] Dowens, S., Mourad, M., Piccarielo, H. and Robson, R., "Digital Rights Management in E-learning: Problem Statement and Terms of Reference", *Association for the Advancement of Computing in Education (AACE)*, November, 2003.

[9] Sandhu, R., S., Coynek, E., J., Feinsteink, H., L. and Youmank, C., E. (1996), "Role-Based Access Control Models", *IEEE Computer*, Volume 29, Number 2, February 1996, pages 38-47.

[10] Schaad, A., Moffett, J., Jacob, J., "The Role-Based Access Control System of a European Bank: A Case Study and Discussion", *6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, Chantilly, VA, USA, ACM, 2001.

[11] Kuhn, D., R., "Mutual exclusion of Roles as a Means of Implementing Separation of Duty in Role – Based Access Control Systems", *National Institute of Standards and Technology*, 1997.

[12] Evered, M. and Bögeholz, S. "A Case Study in Access Control Requirements for a Health Information System", *School of Mathematics, Statistics and Computer Science - University of New England*, 2003.

[13] Hilchenbach, B., "Observations on the Real-World Implementation of Role-Based Access Control" *Schumann Security Software, Inc*, 1997.

[14] Fernandez, R., "Enterprise Dynamic Access Control (EDAC) Overview", *Prepared for Commander, U.S. Pacific Fleet Pearl Harbor, HI 96860*, 2005.