“ALEXANDRU IOAN CUZA” UNIVERSITY OF IAȘI

FACULTY OF COMPUTER SCIENCE



THESIS

Interactive web application-based learning for the

Operating Systems course

proposed by

Dorin Haloca

Session: iulie, 2019

Scientific Coordinator

Lect. dr. Vidrascu Cristian

“ALEXANDRU IOAN CUZA” UNIVERSITY OF IAȘI

FACULTY OF COMPUTER SCIENCE

Interactive web application-based learning for the

Operating Systems course

Dorin Haloca

Sesiunea: iulie, 2019

Scientific Coordinator

Lect. dr. Vidrascu Cristian

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul „Titlul complet al lucrării” este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau din străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei; - reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă; - codul sursă, imaginile etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise; - rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi, data

Absolvent Dorin Haloca

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original)

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „Titlul complet al lucrării”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi, data

Absolvent Dorin Haloca

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original)

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil şi sursă, să aparţină autorului prezentei lucrări, Prenume Nume.

Încheierea acestui acord este necesară din următoarele motive:

[Se explică de ce este necesar un acord, se descriu originile resurselor utilizate în realizarea produsului-program (personal, tehnologii, fonduri) şi aportul adus de fiecare resursă.]

Iaşi, data

Decan Adrian Iftene Absolvent Dorin Haloca \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -----(semnătura în original) (semnătura în original)

INDEX

[Configuring the Linux machines 10](#_Toc7457547)

[1. General description 10](#_Toc7457548)

[2. Configuring the Linux machine used for execution of code 10](#_Toc7457549)

[3. Configuring the Linux machine used for hosting the application 15](#_Toc7457550)

[4. Configuring the Apache server 16](#_Toc7457551)

[5. Configuring the MySQL server 17](#_Toc7457552)

[Types of users in the web application 18](#_Toc7457553)

[Authentication process 20](#_Toc7457554)

[1. Components 20](#_Toc7457555)

[1.1. The MySQL database server 20](#_Toc7457556)

[2. Scenarios 20](#_Toc7457557)

[Back-end mechanisms 23](#_Toc7457558)

[Interface nagivation 24](#_Toc7457559)

[Deployment of the application 25](#_Toc7457560)

[Bugs 26](#_Toc7457561)

INTRODUCTION

I have chosen to create this web application for the Operating Systems course because the Databases course has one and from my point of view it aids students. Instead of solving problems just by yourself, you can interact with your colleagues through the application and check your knowledge or learn through trial and error. For sure, the teacher wouldn’t have the necessary time, energy or imagination to create all the interesting questions students would post there.

The Databases course was the first time I experienced an interactive web application at the faculty and I enjoyed it. Of course, it had its flaws but I tried to fix some of them in my application. Unfortunately, I don’t think it is truly reliable method of grading the students due to the complexity of the Linux operating system. As far as SQL is concerned, it’s way of working is simple and straightforward to assess. While some types of problems can be assessed without getting too deep into the Linux Kernel mechanisms, others require it.

My application consists of an Apache web server which the student will interact with, a Linux machine which will handle the execution of code and a MySQL Database server. Of course, the web server will limit the extent to which a student will have access to the Linux environment.

CONTRIBUTIONS

aceasta va avea cel mult o pagină şi va descrie schematic principalele contribuţii ale absolventului în realizarea lucrării.

# I. Configuring the server/servers

## General description

Initially, I thought about using the faculty’s server so the application would run the code on the students’ accounts. However, on further discussion with my scientific coordinator, I realised that uploading the question’s author code on the students’ account could lead to unwanted leaks and possible exploits. The web interface limits the actions a student can perform. This is why I need to configure my own Linux machine, with its own students’ accounts, which will communicate via SSH with the Apache server.

## Configuring the Linux machine used for execution of code

First we make sure that the Linux distribution is up-to-date

$ sudo apt-get update

$ sudo apt-get upgrade

Now we install and start the SSH service to connect remotely to the Linux machine and execute code.

$ sudo apt-get install openssh-server openssh-client

$ sudo service ssh start

### 2. 1. How the Apache server will access it with php

First we make use the ssh2\_connect() method to connect to a machine using a given $host and $port, with a default value of 22. It returns a resource which we will use in the authentication process.

ssh2\_connect ( string $host [, int $port = 22 [, array $methods [, array $callbacks ]]] ) : resource

In order to authenticate a user though SSH, we use the ssh2\_auth\_password() which takes as parameters the previous resource, a username and a password. In case the process was successful, it returns true. Otherwise, it return false.

ssh2\_auth\_password ( resource $session , string $username , string $password ) : bool

The ssh2\_exec() method is used to send a bash command for execution through SSH. It takes as arguments the resource returned by the ssh2\_connect() method and the command.

ssh2\_exec ( resource $session , string $command [, string $pty [, array $env [, int $width = 80 [, int $height = 25 [, int $width\_height\_type = SSH2\_TERM\_UNIT\_CHARS ]]]]] ) : resource

In order to get the output or errors we need the ssh2\_fetch\_stream() method. The stream id can be SSH2\_STREAM\_STDIO or SSH2\_STREAM\_ERR. The stream\_get\_contents() returns the string.

ssh2\_fetch\_stream ( resource $channel , int $streamid ) : resource

stream\_get\_contents ( resource $handle [, int $maxlength = -1 [, int $offset = -1 ]] ) : string

This method (ssh2\_scp\_send()) is able to send a $lolca\_file to the Linux machine identified by the $session variable using the Secure Copy Protocol. If the operation is completed successfully, it will return true and false otherwise.

ssh2\_scp\_send ( resource $session , string $local\_file , string $remote\_file [, int $create\_mode = 0644 ] ) : bool

### 2.2. How the Apache server limits the execution time of a command

Before sending a command to the Linux to execute via ssh2\_exec(), the Apache server will call the following method. It’s purpose is to stop students from wasting CPU resources. Moreover, without this time limit, the php lock (I will cover this topic later) would produce a deadlock for that particular student.

ssh2\_exec($this->connection, "sleep " . $timeout\_seconds . " ; pkill --signal SIGKILL -u " . $this->execution\_user);

This mothod will trigger a pkill bash command for the Linux machine’s user account ($this->execution\_user) used for executing the command after a given number of seconds ($timeout\_seconds).

At first, I tried to used the $ timeout bash command with the -k argument, which makes the command to send the KILL signal which cannot be bypassed. However, a student could inject another bash command which would bypass the timeout. For example, in this case $ timeout -k 0 1 ls ; ls the second ls would run without a time restriction.

### 2.3. How a student’s account is created on the Linux machine

After modifying the web application to work with containers, multiple student accounts is no longer needed. Instead, it only needs a Linux user account who can access docker.

Bellow I explain how I used to create an account for each student:

The scprit CreateUser.sh contains the commands used to create a new user. It has the following parameters:

$ ./CreateUser.sh <password> <username> <quota\_limit>

In order to simply create a new user on the Linux machine I use the following command:

$ useradd -m --password <password> <password>

I tried to use the adduser command, but it would not let me create users with usernames longer than 8 chars. For example, my faculty’s webmail username is dorin.haloca and it’s too long for adduser command.

The password used for authenticating on this Linux machine will be generated randomly and stored in the MySQL database server.

The setquota bash command is what limits the amount of storage memory a user can use.

The line echo "@"$2 "hard" "nrproc" $4 | sudo tee –append /etc/security/limits.conf > /dev/null is used to modify the maximum numbers of processes a user account can have running at the same time.

In order to avoid a student peeking the files of another student, I used the following two chmod commands to allow only the root to change the permissions and to allow only the student to read/write it:

$ chmod 0770 /home/$2

$ chown root /home/$2

When a student submits an answer, the author’s solution is uploaded, if it is the case, for execution in the current student’s home folder. The current student cannot steal the author’s solution, but another student could if he or she could access the home folder. However, after the permissions change of the script, such an attack becomes impossible.

In the end, if the user account process was successful, a proper response would be returned.

echo "User created!"

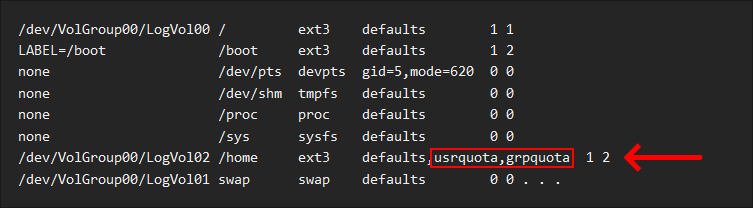
### 2.4. Configuring quotas[[1]](#footnote-1)

After modifying the web application to work with containers, I no longer need to limit the amount of storage memory a student can use. Bellow I explain how I used to limit a student from using more storage memory than allowed:

Quotas are useful because they prevent a user from using more storage memory than the system administrator allowed.

$ sudo apt install quota

$ sudo nano /etc/fstab



$ sudo mount -o remount /

$ quotacheck -cug /

$ sudo quotacheck -m -avug

Note: Quotas don’t work for users with only a number as username. They need at least one letter!

## Configuring the machine used for hosting the web application

### 3.1. General description

This machine can run any operating system which supports the Apache server and the required libraries for my web application. For example, I couldn’t properly install the ssh2 php library on Windows, but I managed it instantly on Linux.

$ sudo apt-get update

$ sudo apt-get upgrade

My web application used to need to be albe to execute sudo commands without needing to input the sudo password[[2]](#footnote-2), mainly for creating a new user. The reason for this decision is the fact that when you first use a sudo command, you are asked for the password, but afterwards you are not asked for the password anymore. <VERIFICAT ASTA> Now, it only needs permissions to use Docker containers, which can be done using the following commands:

$ sudo groupadd docker

$ sudo gpasswd -a $USER docker

After this, restart the computer and test if you can you can create a Docker container without root privileges.

< DE CE AM PUS ASTA AICI? – PUS LA CONFIGURING DOCKER la punctul 2 + adaugat si cum se intaleaza>

Moreover, my web application used to need a proper permissions configuration such that students could not access the web application directory (“mvc/app/”), in case the Apache server was hosted on the Linux machine used for executing code. The group “www-data” needs at least read permission for the “mvc/app/” to read the php scripts and write permission for the “mvc/app/questions/” and “mvc/app/scp\_cache/” in order to create and modify files. The owner of the application directory (“mvc/”) should be a sudo account. These rights adjustments can be done using the ConfigureRights.sh. It has the following syntax: ./ConfigureRights.sh <a sudo user>. Place it in the same directory with the mvc folder of the application and execute it.

### 3.2. Configuring the Apache server

$ sudo apt-get install apache2

$ sudo apache2ctl configtest

$ sudo ufw allow in "Apache Full"

$ sudo apt-get install mysql-server

$ sudo mysql\_secure\_installation

$ sudo apt-get install apache2 php libapache2-mod-php php-mysqli

$ sudo systemctl restart apache2

$ sudo systemctl status apache2

$ sudo apt-get install libssh2-1 php-ssh2 -y

$ sudo a2enmod rewrite

$ sudo service apache2 restart

If the apache version is 2.4 then you have to go to /etc/apache2/. There will be a file named apache2.conf. You have to edit that one (you should have root permission). Change directory text like this:

<Directory /var/www/>

Options Indexes FollowSymLinks

AllowOverride All

Require all granted

</Directory>

$ service apache2 reload

### 3.3. The JSON configuration file

This file contains some configuration parameters for the web application which are necessary to be correctly set within the deployment process.

First of all, the “app/local\_path” parameter refers to the location of the “mvc” directory, which contain all the web application components.

The “external\_shh” entry refers for example to the Faculty of Computer Science serer which will be checked via SSH to make sure that a user who attempts to connect is indeed a student. The “check” parameter MUST be se to “true” when the application is deployed. In case of development purposes, it can be set to “false”. The other two parameters (“host” and “port”) are intuitive.

The “ssh” entry refers to the Linux machine used to execute code. The first two parameters (“host” and “port”) are intuitive. The next two parameters (“sudo\_user” and “sudo\_pass”) are credentials for a user who can use the sudo command on that Linux machine. The “timeout\_seconds” paramtere refers to the maximum amount in seconds a code can executed before being forcefully terminated.

The “db” entry refers to the MySQL database server. The first parameter (“host”) is intuitive. The next two parameters (“user” and “pass”) are credentials for a user who can select,update and insert into the database. The last parameter (“name”) refers to the database name on the database server.

## Configuring the MySQL server

### 5.1 Creating a new user[[3]](#footnote-3)

It is a known fact that you don’t use the root user to access a database. Therefore, we need to create a normal user and to give it privileges.

$ sudo mysql -u root

mysql> USE mysql;

mysql> CREATE USER 'YOUR\_SYSTEM\_USER'@’localhost’ IDENTIFIED BY 'YOUR\_PASSWORD';

mysql> GRANT ALL PRIVILEGES ON \*.\* TO 'YOUR\_SYSTEM\_USER'@'localhost';

mysql> UPDATE user SET plugin='mysql\_native\_password' WHERE User='YOUR\_SYSTEM\_USER';

mysql> FLUSH PRIVILEGES;

mysql> exit;

$ service mysql restart

Is may be possible that the ‘mysql’ user has a quota limit set. If it is the case, it will eventually run out of disk space and not function properly.

# II. Types of users in the web application

1. Guest user
   1. Can access the authentication page
   2. Can authenticate
2. Normal user (student)
   1. Can do anything the above user can do
   2. Can view announcements
   3. Can solve questions
   4. Can report questions
   5. Can post questions according to set criteria
   6. Can delete his or her own questions according to set criteria
   7. Can view own posted questions in “My Questions” page
   8. Can view the message and the date of the report for those questions
   9. Can de-authenticate
3. Administrator (professor)
   1. Can do anything the above user can do
   2. Can post announcements
   3. Can delete announcements
   4. Can post questions unconditionally
   5. Can view all posted questions in “All Questions” page
   6. Can view all deleted questions in “All Questions” page

The deleted questions are not removed from the system, but marked as “Deleted”. I chose to keep the deleted questions in case of a system error, unintentional operation or a conflict (i.e. maybe the student posted an inappropriate question and he or she deleted it to hide any proof).

* 1. Can access the administrator page (url: <ip>/mvc/public/admin)
  2. Can add administrator by username
  3. Can remove administrators
  4. Can post chapters
  5. Can unpost chapters

# III. Authentication process

## Components

### 1.1. The MySQL database server

The username used for authenticating on my web application will be the one used to authenticate on an external SSH connection. It will be stored in the MySQL database server.

The password used for authenticating on my web application will be the one used to authenticate on an external SSH connection. Its hash, generated using the PHP default hashing algorithm[[4]](#footnote-4), will be stored in the MySQL database server.

### 1.2. The external SSH connection

For example, this is the server of the Faculty of Computer Science. This connection can be configured on the JSON configuration file. It is used to verify that a user is indeed a student of the faculty.

Even if this server is not online, the student will still be able to authenticate in most cases (see below scenarios) because the credentials are stored in the MySQL database server. The only thing that becomes impossible is the account creation process, if the application is configured to use this server, due to the impossibility to verify if a user is indeed a student.

## Scenarios

### 2.1. The student doesn’t have an account on my web application and the flag to check the external SSH connection is set to true.

My web application checks the MySQL database server in case there exists an account for that username. It will not find anything, so it initiates the account creation process:

My web application checks the external SSH connection to verify the credentials. If they are correct, a new account is created in my web application and on the Linux machine used to execute commands. The student is then automatically logged in.

### 2.2. The student doesn’t have an account on my web application and the flag to check the external SSH connection is set to false.

My web application checks the MySQL database server in case there exists an account for that username. It will not find anything, so it initiates the account creation process:

My web application doesn’t check the external SSH connection to verify the credentials. A new account is created in my web application and on the Linux machine used to execute commands. The student is automatically logged in.

### 2.3. The student does have an account on my web application, the flag to check the external SSH connection is set to true and the password is incorrect

My web application checks the MySQL database server in case there exists an account for that username. It will find the account and it initiates the authentication process:

My web application checks the password’s hash stored in the MySQL database to verify the credentials. The password is incorrect, so the application verifies the credentials using the external SSH connection. If the credentials are correct, a new hash is stored in the MySQL database and the student is logged in. Otherwise, an error message is displayed.

### 2.4. The student does have an account on my web application, the flag to check the external SSH connection is the to false and the password in incorrect

My web application checks the MySQL database server in case there exists an account for that username. It will find the account and it initiates the authentication process:

My web application checks the password’s hash stored in the MySQL database to verify the credentials. The password is incorrect, but the application doesn’t verify the credentials using the external SSH connection. A new hash is stored in the MySQL database and the student is logged in.

### 2.5. The student does have an account on my web application, the flag to check the external SSH connection is the to true and the password is correct

My web application checks the MySQL database server in case there exists an account for that username. It will find the account and it initiates the authentication process:

My web application checks the password’s hash stored in the MySQL database to verify the credentials. They are correct, so the student is logged in.

# IV. Back-end mechanisms

# V. Interface navigation

# VI. Deployment of the application

1. General information

The web application consists of 3 main components:

- Apache server

- Linux machine used for execution of code

- MySQL database server

These components can be placed on a single Linux machine without any problems. However, in case there is too much load for a single server, the application can be split on 2 or 3 separate servers.

1. How it will be done in the Faculty of Computer Science
   1. Database server

<https://students.info.uaic.ro/db/>

# VII. Bugs

1. PHP semaphores in combination with the sleep x instruction in Linux makes the header() function in PHP misbehave. If we have two tabs opened with the chapter solving page, waiting for response from the Apache server, the instruction header() will make both tabs stop waiting, giving them the same response, even though one tab was waiting for a response that would arrive later. Therefore, if we execute a sleep 5 and a sleep 10 for the same chapter simultaneously, only the tab with sleep 5 will receive a response while the other one will just stop waiting and receive nothing.

2. Sometimes, when pressing ‘Execute’ or ‘Submit’ for chapters based on C, an error message like “Output cannout be empy!” or “Could not send file execution” may appear. This is because of the SSH connection. Press the button again and it should work fine.

CONCLUSIONS

în această parte a lucrării de licenţă se regăsesc cele mai importante concluzii din lucrare, opinia personală privind rezultatele obţinute în lucrare, precum şi potenţiale direcţii viitoare de cercetare legate de tema abordată. Concluziile lucrării nu se numerotează ca şi capitol.

BIBLIOGRAPHY

1. <https://www.php.net>

2. <https://stackoverflow.com>

3. <https://linux.die.net>

4. <https://validator.w3.org>

5. <https://brianscode.com>

<PUS SI CART DE LA CURSURII>

1. <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/deployment_guide/ch-disk-quotas> [↑](#footnote-ref-1)
2. <https://www.cyberciti.biz/faq/linux-unix-running-sudo-command-without-a-password/> [↑](#footnote-ref-2)
3. <https://stackoverflow.com/questions/39281594/error-1698-28000-access-denied-for-user-rootlocalhost> [↑](#footnote-ref-3)
4. The default PHP hashing algorithm will change in the future according to the latest security requirements [↑](#footnote-ref-4)