“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

Session: 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 the learning process. Instead of solving problems just by yourself, you can interact with your colleagues through the application and check your knowledge or simply 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. Therefore, I configure my own Linux machine for the application’s particular needs. I also added here instructions on how to configure the database server as well as the web server.

In previous versions, my application created a separate account on the Linux machine for every student using it to prevent write and execution conflicts. It was pretty costly to do this and also adjustments to users’ rights were required. However, after implementing Docker containers, one account is enough because I run the code in separate containers. By treating the code as plain text prior to mounting it in Docker, the overall security of the application improved drastically.

After modifying the web application to work with containers, I also no longer need to limit the amount of storage memory a student can use through quotas. This is a good thing because it didn’t work for users with only a number as username. They need at least one letter.

Docker was not the first virtualization solution I tried in order to improve the stability and security of the system. Before this, I tried the “chroot” command.

## 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

Now we install Docker.

$ sudo snap install docker

### Configuring Docker

First, we create a group called “docker” so running a container can be done without root privileges. The $USER represents the account on the Linux machine which will be used through SSH.

$ sudo groupadd docker

$ sudo gpasswd -a $USER docker

We can user the “my\_ubuntu” image which I preconfigured and attached to the application package. Or we can configure a vanilla ubuntu image for our needs:

$ docker pull ubuntu

$ docker run --name my\_ubuntu -it ubuntu bash

Now that we have an interactive shell opened, we install the “strace” command and thte gcc compiler. Note that we don’t need to use “sudo” anymore because we are logged in as root.

$ apt-get update

$ apt-get install strace

$ apt-get install gcc

$ exit

After installing those packages, we need to create an image out of that container.

$ docker commit my\_ubuntu

$ docker images

Copy the IMAGE ID of the first image from the list, with the “<none>” tag. We will change it’s name to “my\_ubuntu”.

$ docker tag $IMAGE\_ID my\_ubuntu

We don’t need anymore the container “my\_ubuntu” so we remove it. Note that the newly created image “my\_ubuntu” will be kept.

$ docker rm my\_ubuntu

Note that the web application may not be able to find Docker location. To solve this, you need to add its location in “/etc/environment”. The location of Docker and be found using $ which docker. In my case, I added “:/snap/bin” (ignore “”) at the end of string.

## Configuring the Apache server

This can run any operating system as long as it has the required libraries for my web application. Unfortunately, I couldn’t properly install the ssh2 php library on Windows, but I managed it instantly on Linux. Below I explain how I did it.

### 3.1. Installation

$ sudo apt-get install apache2

$ sudo apache2ctl configtest

$ sudo ufw allow in "Apache Full"

$ 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

Note that the group “www-data” needs read and write permissions for the entire “mvc/app/” directory.

### 3.2. Adding the web application

On Linux, my web application will be placed the “html” directory of the Apacher server. It is located in /var/www/html. You only need the “mvc” directory there.

### 3.3. The JSON configuration file

It is located in mvc/app. 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 server 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

### 4.1. Installation

It can be installed on any operating system. I describe bellow how we install in on Linux.

$ sudo apt-get install mysql-server

$ sudo mysql\_secure\_installation

### 4.2. Creating a new user[[1]](#footnote-1)

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. I describe bellow how we create a new user on Linux.

$ 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

### 4.3. Importing the database

For the import I use the phpMyAdmin’s web interface. Go to Import>File to import and choose AplicatieSO.sql.

# 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
   7. Can access the administrator page (url: <ip>/mvc/public/admin)
   8. Can add administrator by username
   9. Can remove administrators
   10. Can post chapters
   11. 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[[2]](#footnote-2), 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 the 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

## Administrator panel

As the name implies, it can only be accessed by administrators at “public/admin”.

## Managing questions

### 2.1. Answering questions

To answer questions, got to Choose Chapter and select one. While solving questions, you won’t come across any questions of yours or and question marked as invalid by an administrator. Even though the questions for that particular chapter are selected at random, you won’t get the same question twice in a row.

After you hit “Submit”, you will get to see the arguments, input file and the keyboard input of the author where it is the case. In case you answered correctly to the question, you will also get to see the author’s code. Otherwise, you can spend 3 correct answers to reveal it. This is useful especially if the problem seems impossible to you.

### 2.2. Reporting questions

After you submit an answer, you have the choice to report a mistake in the question. This can mean the text or the code. The message of the report will be visibible to the question’s author, but not the user who reported it. So a report is anonymous, unless an administrator checks it.

### 2.2. Posting questions

The process of posting questions is divided per chapter. The default criteria to post the first questions is to have answered correctly to at least 10 questions from that chapter. The second question posted requires the same number of corrects answers. After that, the cost gets multiplied by 2. In other words, 20 correct answers for the third question, 40 for the fourth one etc.

To change the default formula, change the code from mvc/app/models/Forumlas.php. Be careful to update the other formulas as well.

Questions wich can’t get a consistent output after two consecutive runs won’t be accepted. An error message regarding the non-determinism of the code will be displayed.

The metadata of a question, such as the chapter it belongs to, author, times attempted, times answered correctly, reports, date uploaded etc. are stored in the database. The text, code, arguments, input file, keyboard input of the question are stored in files located in mvc/app/questions. They have as name their id in the database. Their extensions are .code (what needs to be executed), .text (question’s requirements), .args (arguments), .keybd (keyboard input), .input (input file contents).

The uploading process has 3 steps. The first one consists in making an insert in the questions table with the status “pending”. Then, that entry’s id is read and the question’s files are created. When this is done, the status of the question in the database is changed to “posted”.

### 2.3. Deleting questions

The deleted questions are not removed from the system, but marked as “deleted” in the database. I chose to keep the deleted questions in case of a system error or unintentional operation. An administrator can restore any deleted question. This will update the number of posted questions and the number of available correct answers of the user.

If a question is deleted, the number of posted questions in the particular chapter for the user decrements and he or she gets back the spent right answers to post it. In other words, the user can immediately post again a question with only a penalty of 10 right answers. This measure greatly lowers the risk of reposting spam, thus, wasting storage space in the system.

### 2.4. Validating questions

Only the administrator can validate questions. This mechanism ensures a filtering of inappropriate questions which otherwise would get shown to students in the application. The validation status of a question can be updated as many times as necessary.

## Code execution on the Linux machine

### Preparation

The web application first writes the code and other necessary information in dedicated files with specific extensions. They are located in “mvc/app/scp\_cache” and their unique names correspond to the user id who initiated the operation.

### Establishing the connection and sending the files

First we make use the ssh2\_connect() method to connect to the Linux 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

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

### Executing the code

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

On top of that, I use the Docker functionalities, namely containers. The bash command I send for execution looks like this:

docker run --name " . $this->session\_user . " -v $(pwd)/" . $this->session\_user . ".sh:/code.sh -v $(pwd)/" . $this->session\_user . ".keybd:/code.keybd:ro -v $(pwd)/" . $this->session\_user . ".input:/code.input -v $(pwd)/" . $this->session\_user . ".output:/code.output -v $(pwd)/" . $this->session\_user . ".run:/code.run:ro --rm my\_ubuntu bash ./code.run

The “--name” argument means that the container will also have a name, that of the user initiating the code execution operation. The “-v” argument is used to mount a file in the container. The path before “/” represent the host’s files path and the path after “/” represents where it will be mounted in the container. Adding “:ro” makes the mounted file read-only. I change the unique name of the file stored on the host system to a generic “code.<extension>” so that it is easier to identify the files in container. The argument “--rm” tells the container to delete itself after executing the code. Otherwise, it would store a copy of the changes made. Instead of using a default ubuntu image, I use the “my\_ubuntu” image. The .run file I execute contains the following code:

chmod +x code.sh && ./code.sh " . $args . " < code.keybd

First, I give the script file (.sh) the right to execute. If it was done successfully, I run it using the arguments from $args if there are any. I am also able to feed the script the keyboard input stored in “code.keybd”. If I wouldn’t have a used an intermediate file (“code.run”), there would exist the risk of code injection through the $args variable.

In order to get the output or errors after execution 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

### Checking system calls

For this I use the “strace” command. The “.run” file for the C programs which need to use forking looks this:

gcc code.c -o code.out && (strace -e trace=clone ./code.out " . $args . " < code.keybd)

The strace commands looks for system calls involving process cloning. If no system call of this kind if detected during execution, an error message will be displayed indicating that the student did not use fork()in his/her code. To use the “strace” command in docker you need additional parameters:

docker run --cap-add=SYS\_PTRACE --security-opt seccomp=unconfined --security-opt apparmor=unconfined …

Another solution is to simply add --privileged, but it is recommended to avoid it.

### How the application limits the execution time

I use the “timeout” command to limit the execution of the “docker” command. Like this, students will not be wasting CPU resources on purpose or by mistake though an infinite loop. Moreover, without this time limit, the php lock (I will cover this topic later) would produce a deadlock for that particular student. Below is an example from php:

ssh2\_exec($this->connection, "timeout --signal=SIGKILL " . $ssh\_timeout\_seconds . " " . $docker\_command);

This mothod will trigger a pkill bash command after a given number of seconds ($ssh\_timeout\_seconds). The KILL signal which cannot be bypassed, unlike TERM.

## Php semaphores

# 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.

3. Because I use containers, the “ps” command will be pretty much unusable because it output will be almost empty. The same can be said about “who”. Even by modifying /var/run/utmp I couldn’t make it output something unless actual users are connected.

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://stackoverflow.com/questions/39281594/error-1698-28000-access-denied-for-user-rootlocalhost> [↑](#footnote-ref-1)
2. The default PHP hashing algorithm will change in the future according to the latest security requirements [↑](#footnote-ref-2)