**Aplicatia mea pentru SO**

**Configurare Linux:**

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install openssh-server openssh-client

sudo service ssh start

sudo service ssh status

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 php libapache2-mod-php php-mcrypt php-mysql

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

To run sudo commands without sudo (for php scripts)

<https://www.cyberciti.biz/faq/linux-unix-running-sudo-command-without-a-password/>

In case the Apache server is on the same Linux machine used for executing commands/scripts/pograms, for the “mvc/app/” directory of the application, students should not have any permission over it. However, the group “www-data” needs at least read permissions for the “mvc/app/” to read the php scripts and write permissions for the “mvc/app/questions/” “mvc/app/scp\_cache/” in order to create files and modify files. The owner of the applicaton files should be a sudo account.

Those 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 application (mvc/ directory) and execute it.

**In caz ca Linux 16.04 raporteaza la infinit eroarea PCIe Bus error:**

Open terminal

$ sudo nano /etc/default/grub

Add "pci=noaer" to GRUB\_CMDLINE\_LINUX\_DEFAULT

$ sudo update-grub

**In caz ca Linux 18.04 se blocheaza la ecranul mov dupa logare:**

Enter recovery mode

Update grub bootloader

Resume boot

Open terminal

$ sudo nano /etc/default/grub

Add "nouveau.modeset=0 acpi\_rev\_override=5" to GRUB\_CMDLINE\_LINUX\_DEFAULT

$ sudo update-grub

Note: if terminal doesn’t open upon boot, then restart, select Ubuntu in grub, press e, add "nouveau.modeset=0 acpi\_rev\_override=5" at the end of the line that starts with "linux" and press Ctrl+X

**Configurare Apache**

sudo a2enmod rewrite

sudo service apache2 restart

New apache version has changed in some way. If your 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

**Configurare user MySQL server**

<https://stackoverflow.com/questions/39281594/error-1698-28000-access-denied-for-user-rootlocalhost>

$ sudo mysql -u root # I had to use "sudo" since is new installation

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

E posibil ca mqsql sa aiba quota setat. Cand va atinge limita, va refuza sa porneasca.

**Limit execution time**

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

After $timeout\_seconds, the pkill will kill all precesses for the user $this→execution\_user. This prevents students from overloading the CPU with useless tasks. I chose to use the SIGKILL signal because it cannot be ignored by any process. By default, pkill uses SIGTERM, which can be ignored.

A student cannot interfere with this method of limiting execution time because the only process that appears when the command to limit execution time is called is “sleep” and “sh”. In case the sleep process is killed, the pkill command is called immediately instead of $timeout\_seconds.

OLD:

$ timeout -k 0 x command -> makes any command run for x seconds.

If the command times out, then exit with status 124. Otherwise, exit with the status of COMMAND. If no signal is specified, send the TERM signal upon timeout. The TERM signal kills any process that does not block or catch that signal. For other processes, it may be necessary to use the *KILL (9)* signal, since this signal cannot be caught. -> <https://linux.die.net/man/1/timeout>

It can be easily bypassed using ‘;’. Exemple: sleep 10; sleep 10.

The SSH2 lib from seclib has a seTimeout(seconds) function, but it just stops waiting for the end of the processes execution for a user and doesn’t kill them.

**Creating new user**

To create a new user accoun on my Linux machine for a student first we check that he/she has an account on the external ssh connection (ex: students.info.uaic.ro), then we check that the script to create the user with the specified criteria is located in the root of the personal home directory of the sudo user (path: ~) and that it can be executed. It will echo a message in case it is successful, otherwise nothing.Make sure that students cannot access or modify that script by adjusting the rights of the sudo user’s home directory.

**Limit user max processes limit**

<https://www.tecmint.com/set-limits-on-user-processes-using-ulimit-in-linux/>

- my set limit is 20 (in the .json config life)

**Limit user storage with quotas**

<https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/deployment_guide/ch-disk-quotas>

sudo apt install quota

Add usrquota to /etc/fstab

sudo mount -o remount /

quotacheck -cug /

sudo quotacheck -m -avug

sudo setquota dorin.haloca 20M 20M 0 0 /

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

**Server pentru testare**

- Initial ma gandesam sa execut comenzile/codul pe contul fiecarui student de pe serverul facultatii numit Fenrir, dar nu este in regula ca studentul sa aiba access direct la mediul unde se va incarca si executa solutia corecta a problemei. Interfata Web restrictioneaza accesul in mai multe privinte (timp de executie, nr procese limita, semafoare)

- In final, imi voi configura eu prorpiul server cu care voi comunica prin SSH (fiecare utilizator va avea un cont de utilizator standard pe serverul Linux)

**Server pentru aplicatia web**

- Apache Linux

Nu am putut adauga libraria pentru SSH2 pe Windows.

**Limbaje**

- front-end: HTML+CSS

- back-end: PHP 7

**Autentificare pe serverul meu de Linux**

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

- voi crea un utilizator pe masina mea de Linux folosind useradd, deoarece adduser nu imi permite sa creez utilizatori care contin punct (ex: dorin.haloca)

- voi stoca hash-ul parolei de la Fenrir si voi genera o parola aleatoare secreta pentru fiecare cont de utilizator de pe masima mea Linux

**Utilizatori**

- guest user:

- poate vedea pagina de autentificare

- se poate autentifica

- normal user:

- can do anything the above user can

- can solve questions

- can post questions according to established criteria

- can logout

- can delete own questions according to established criteria

- can view own questions in My Questions page

- admin user:

- can do anything the above user can

- can post questions unconditionally

- can view all questions in All Questions page

- can mark question as valid/invalid

- can post announcements

- can delete announcements

- can access the “public/admin” page

- can add/remove admins

- can post/unpost chapters

**Capitole**

- comenzi bash

- scripturi

- programe C Linux

**Adaugare capitol**

- se adauga capitolul in tabela chapters din baza de date

- se creeaza o tabela in baza de date chapter\_x, x fiind id-ul capitolului adaugat

- se adauga in controllers fisierele chapter\_x\_solve.php, chapter\_x\_submit.php, chapter\_x\_result.php

- se adauga in views fisierele chapter\_x\_solve.php, chapter\_x\_submit.php, chapter\_x\_result.php

- se adauga in sytlesheets fisierele chapter\_x\_solve.css, chapter\_x\_submit.css, chapter\_x\_result.css

**Adaugare intrebare:**

**Evaluarea**

- comenzi bash

- voi stoca comanda celui care adauga problema

- cand cineva ii rezolva problema, prin SSH ma voi conecta la masina mea Linux (cu contul celui care rezolva problema) si voi rula pe rand solutia buna a problemei si solutia celui ce rezolva problema si voi compara rezultatle

- comenzile nu vor fi executate direct, ci dintr-un fisier script pentru a preveni combinatii precum “sleep 100; sleep 100” (timeout se aplica doar primului sleep 100 daca e rulata la linia de comanda)

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

- - la sfasit voi orice fisier/director din spatiul de lucru al utilizatorului folosind comanda “rm - rf \*”

- scripturi

- voi stoca codul scriptului celui care adauga problema

- cand cineva ii rezolva problema, prin SSH ma voi conecta la masina mea Linux (cu contul celui care rezolva problema) si il voi executa

- apoi voi trimite si voi executa solutia corecta a problemei

- voi compara cele 2 rezultate

- citirea si afisarea se vor face de la tastatura sau cu fisiere

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

- la sfasit voi orice fisier/director din spatiul de lucru al utilizatorului folosind comanda “rm - rf \*”

- programe C Linux

- voi stoca codul celui care adauga problema

- voi trimite prin SCP pe masina mea Linux (cu contul celui care adauga problema) si voi compila codul

- - nu voi stoca programele compilate in baza de date deoarece diferite compilatoare pot compila diferit

- cand cineva ii rezolva problema, prin SSH voi trimite codul programului pe masima mea Linux (cu contul celui care rezolva problema), il voi compila si il voi executa

- apoi voi trimite solutia corecta compilata si o voi executa

- citirea si afisarea se vor face cu fisiere sau de la tastatura

- voi compara cele 2 rezultate

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

- bool **ssh2\_scp\_recv** ( resource $session , string $remote\_file , string $local\_file )

- - la sfasit voi orice fisier/director din spatiul de lucru al utilizatorului folosind comanda “rm - rf \*”

**-** When user submits an answer, he will be redirected to the restults page. In case the answer was correct, he will see the author’s solutions. Otherwise, it will be hidden. The motivation behind this strategy is that the application’s role is to aid learning. Therefore, showing a different solution to the problem to the user teaches him or her another way of solving it. Of course, a student who didn’t correctly solve the problem should not get the answer.

- The user can report the question despite the correctness of their submitted answer. In addition, a short message may be added.

**Evaluation problems:**

* The output of the “who” command may vary between consecutive calls because a user may connect/disconnect in between them. However, due to the way ssh library in php is working, a user connected through it won’t be detected by the command.
* The output of the “ps” command may vary between consecutive calls because a process may die/spawn in between them. Moreover, it also displays the “ps” process which spawn every time it’s called with an different(bigger) ppid. Initially, I wanted to prohibit the submission of questions with commands whose outputs would vary between consecutive calls by calling it twice and comparing the results. A more appropriate solution was to compare the character’s frequency of the commands, the of the author and the one of the user who attempts to solve the problem, ignoring the space character, but it lacked precision when two totally different commands would give the same result. Neither using the “strace” command would help as two different commands would have different system calls.

**Security:**

- a standard user cannot change their (secret random) password without knowing it. Therefore, they can only access their standard user account on my Linux machine only though the web interface of the application with some restrictions.

- all public functions in PHP classes must call check\_login() from controller, except for home/index, login/index, login/process and logout/index

- check\_login() saves the following session data: user\_id, user (the username), password of the ssh connection (plain\_text), and the is\_admin status. At the end of the saving process, one more check is done to see whether the user is still logged in. ($\_SESSION[‘user’] is set)

- a student cannot interfere with the evaluation process after pressing submit (semaphore per user). He/she needs to wait until the process is over to interact with the Linux machine again. This prevents him/her form reading the solution of the question’s author.

- a student cannot create a child process with fork() and leave it running after killing the parent process. The application will wait for the termination of that child process.

- the script which creates a new user on the Linux machine (AplicatieSO/CreateUser.sh) prevents other users from peeking into the student’s home directory.

**Random function:**

- As of PHP 7.1.0, **rand()** uses the same random number generator as [mt\_rand()](http://php.net/manual/ro/function.mt-rand.php).

- mt\_rand — Generate a random value via the Mersenne Twister Random Number Generator

- There is no need to seed the random number generator with [srand()](http://php.net/manual/en/function.srand.php) or **mt\_srand()** as this is done automatically.

**HTML Validation:**

- All html code was validated with <https://validator.w3.org/>

**Concurency per user (student) problem:**

- solved by using semaphores per user account

- <https://brianscode.com/php-semaphore-example/>

**SQL Injection:**

- prevented by using prepared statements

**Solving questions:**

- only posted questions which weren’t marked as “Invalid” will be taken intro consideration

- the user will not get the same question twice in a row

-the user will not get his/her own questions

**Bugs:**

- Semaphores + sleep() prevent header() from executing in another tab. Header() executes when sleep() finishes execution. Ex: open two tabs of chapter\_x\_solve and click ‘Execute’ on both with a delay.

- If there is only one questions suitable for a given chapter, when the user will want to solve questions for that chapter, an infinite loop will occur (the user can’t get the same question twice in a row)

- SQL Server may have quota set which may prevent it from working properly

- some users may have a quota set, even though they don’t need it