Course: PROG38263

Assignment: 3-4

Member 1 Name and Student ID: <Matthew Caven and 991516698>

Member 2 Name and Student ID: <Ibraheem ABD and 991555590>

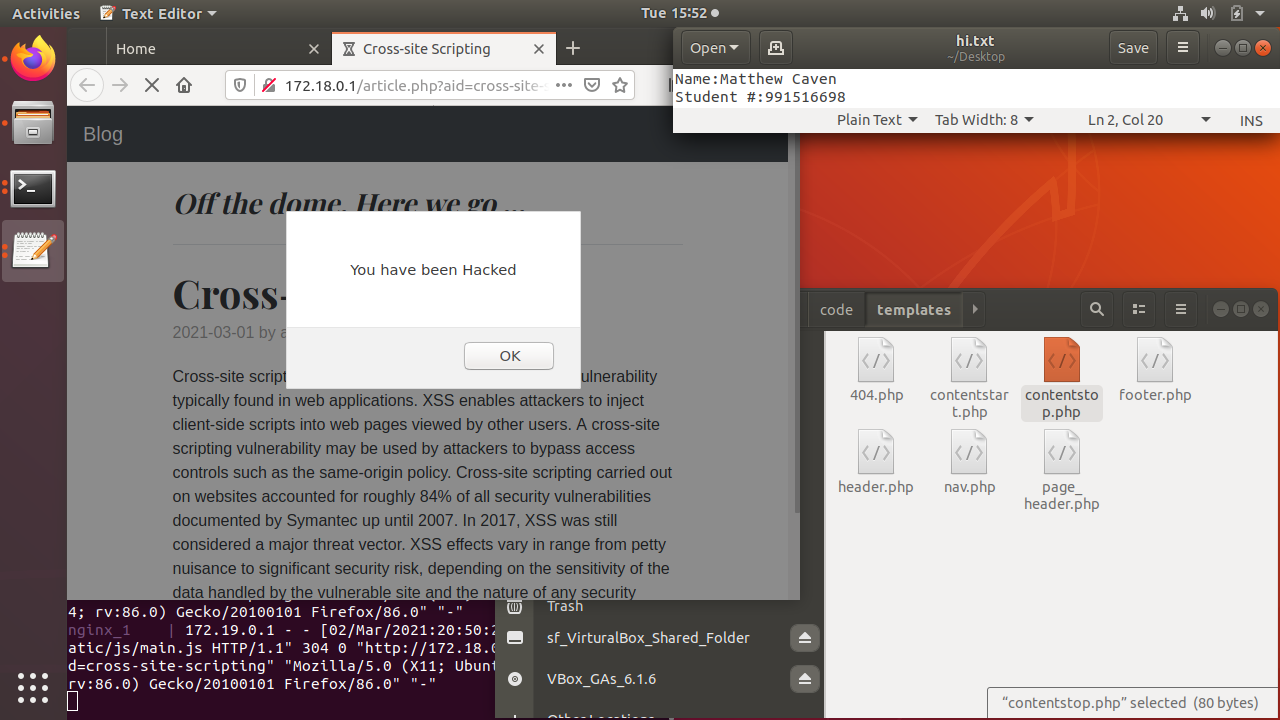
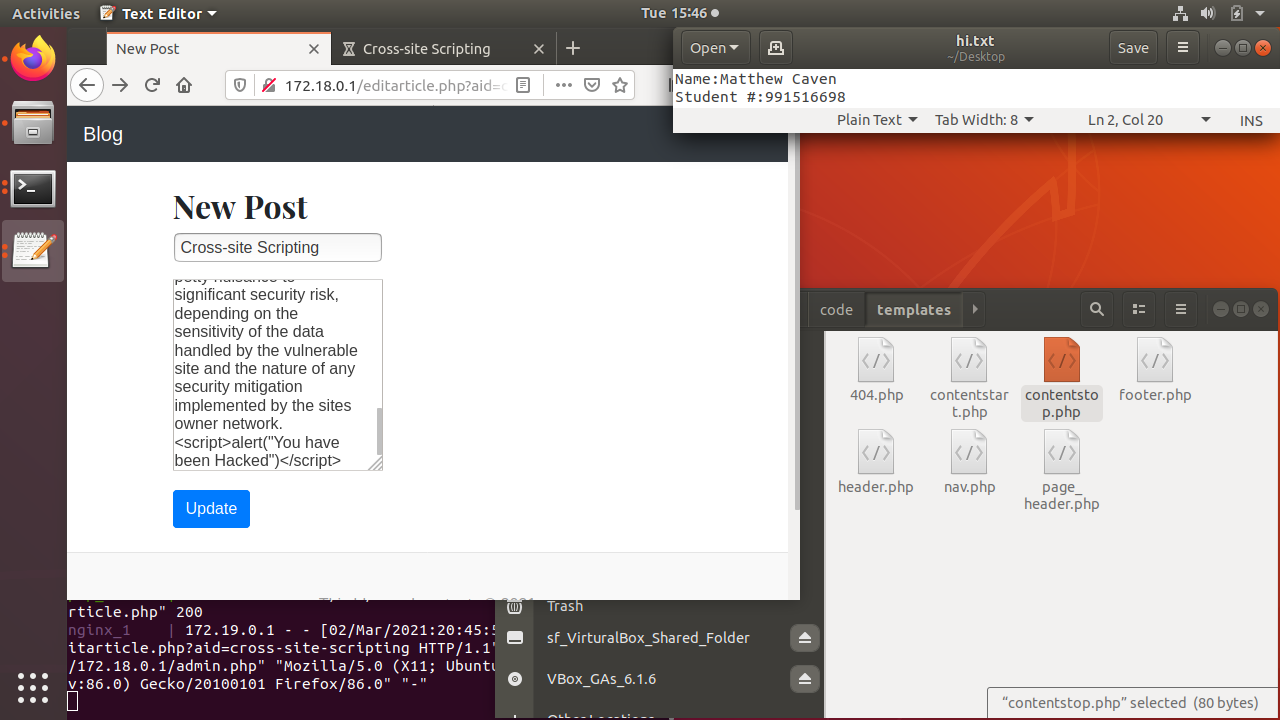
Section: 34778

Instructor: Syed Tanbeer

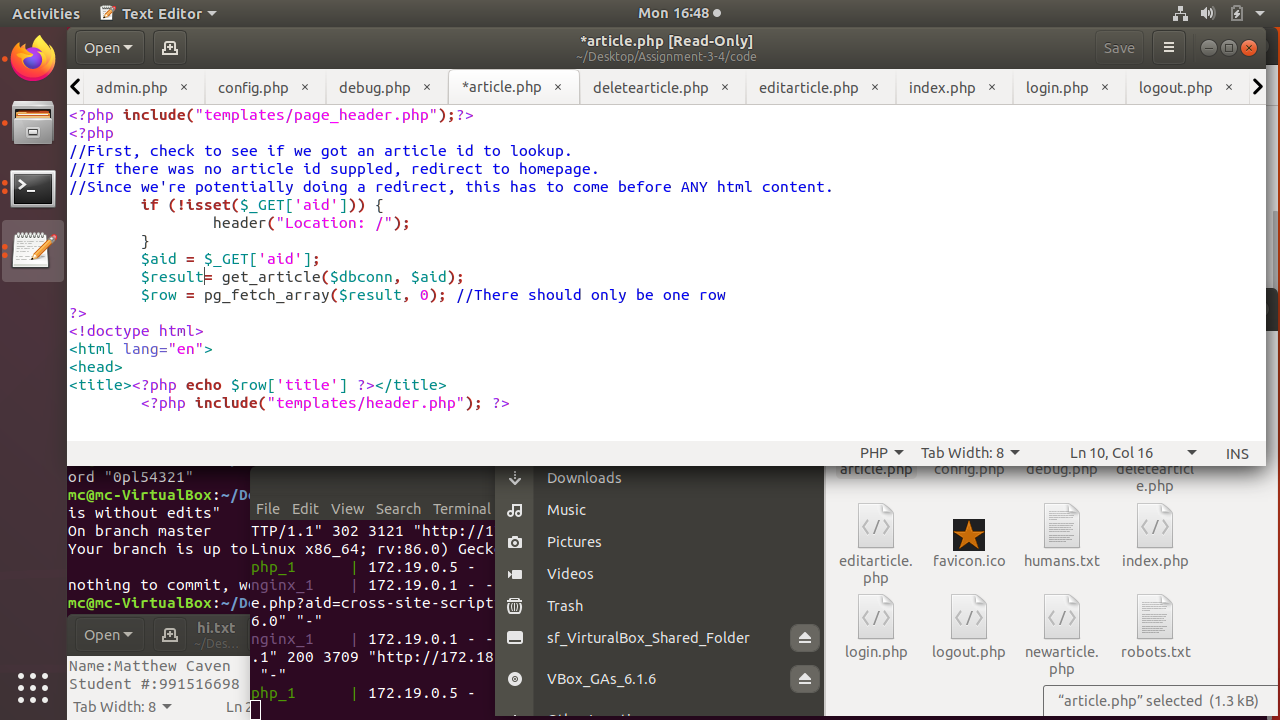
Vulnerabilities

The application has the following known vulnerabilities that you must find and exploit.

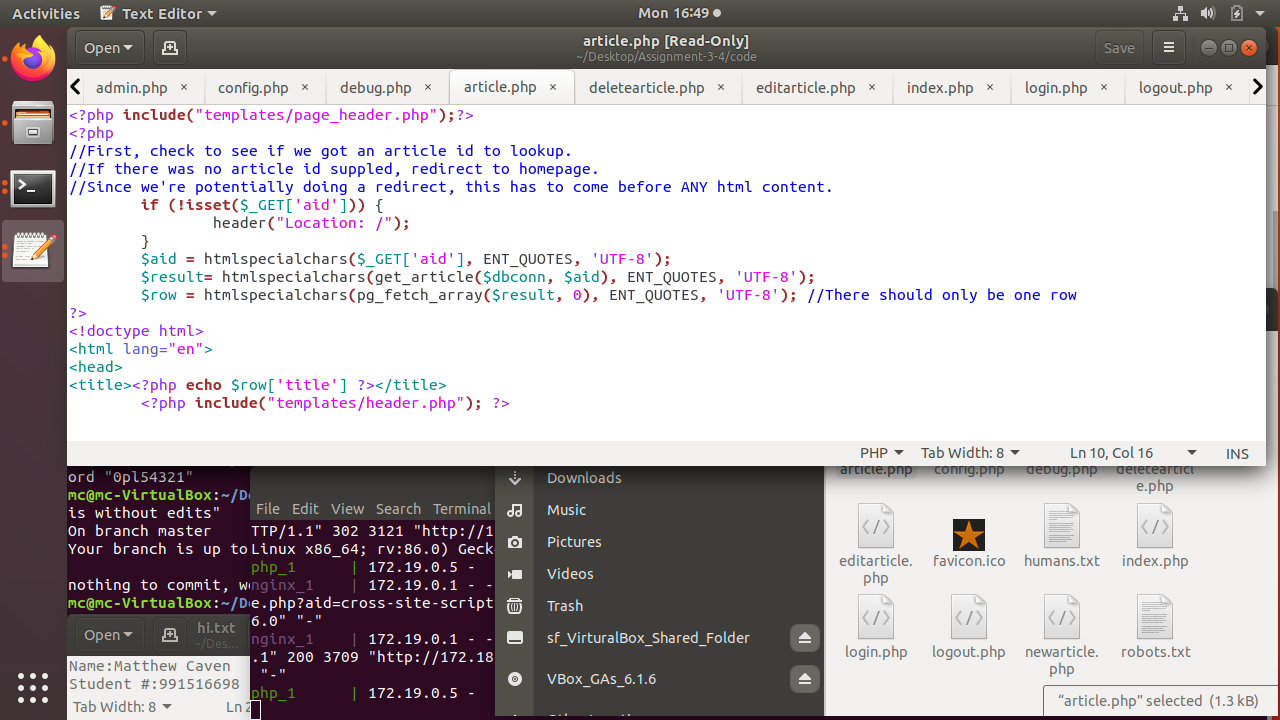
V-1. Cross-Site Scripting (XSS)



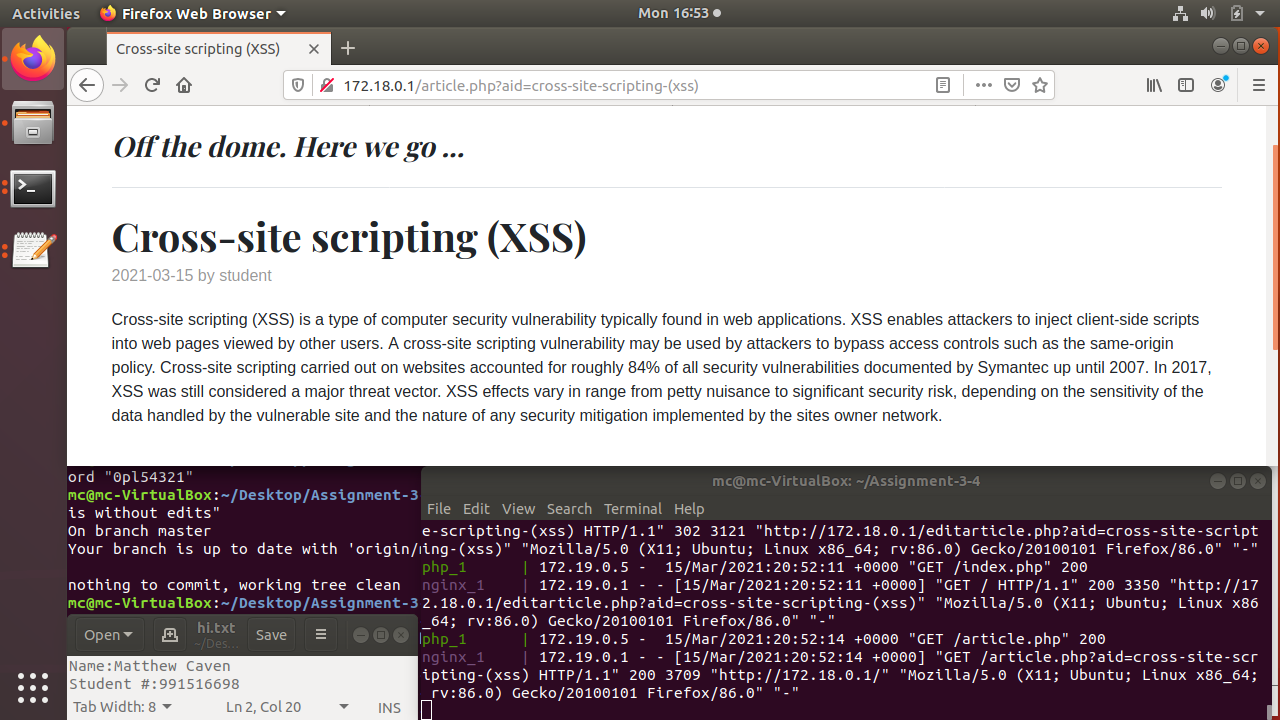
The ability to inject javascript into an articla is quight severe to to the ability to exploit anyone that open the altered page.There are ways to stop this from occuring such as the indesriminent removal of all special charicters/anything resembaling code. Then there is code librarirys that will let you issolate input so that the program dose not execute it. I used the htmlspecialchars command to interprit the text as UTF-8 encoded text. Within the ~/Assignment-3-4/code/article.php it surrounds



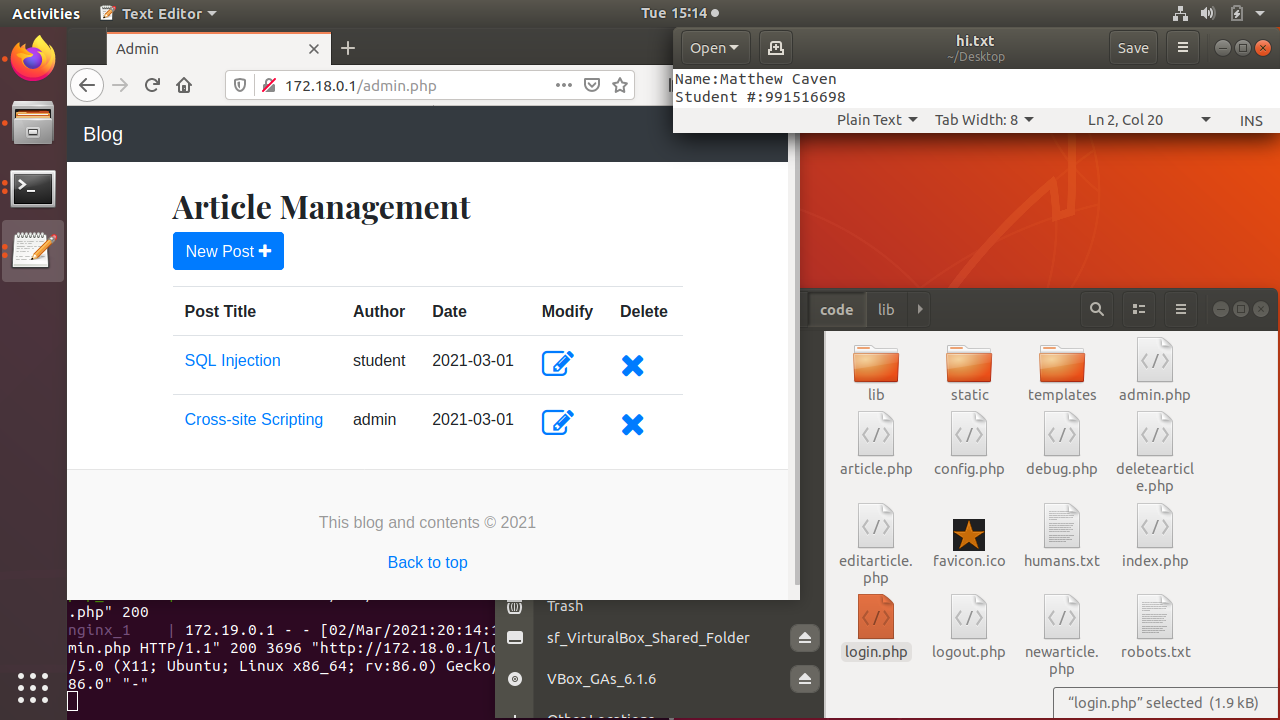
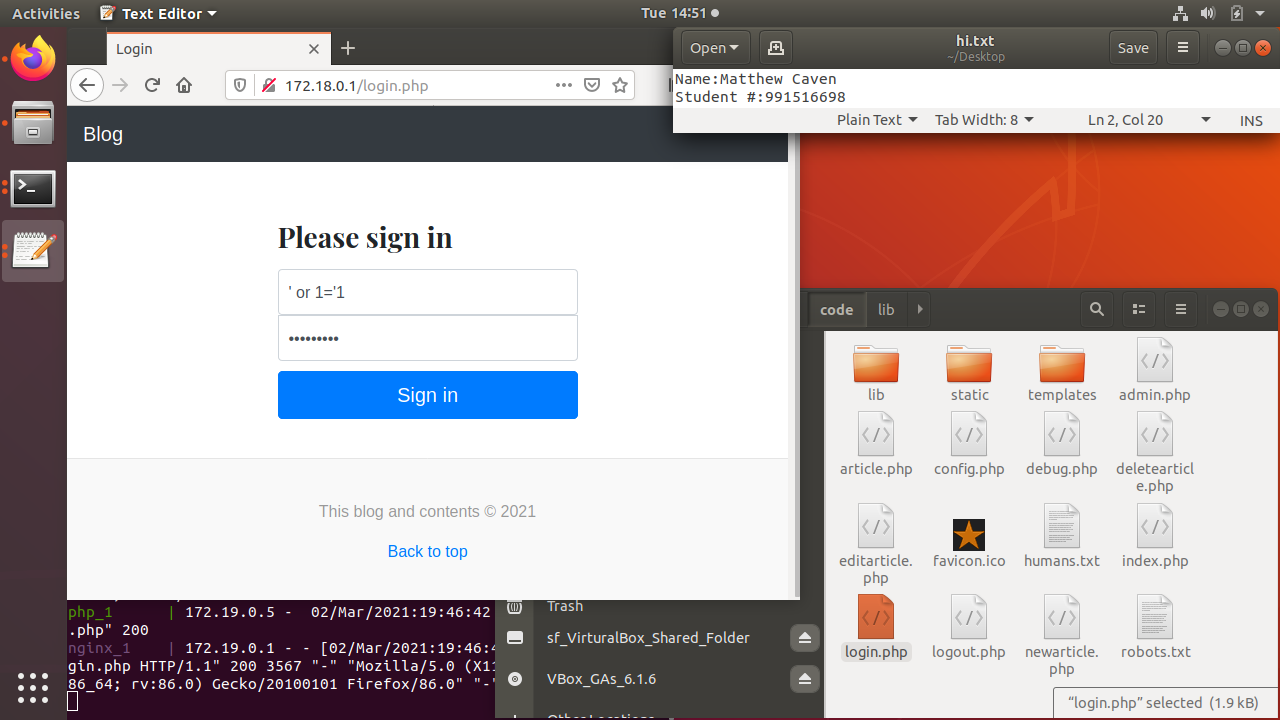
And became



This takes the Cross-site scripting (XSS) article with injected script and dose not execute the injected code.



V-2. SQL Injection

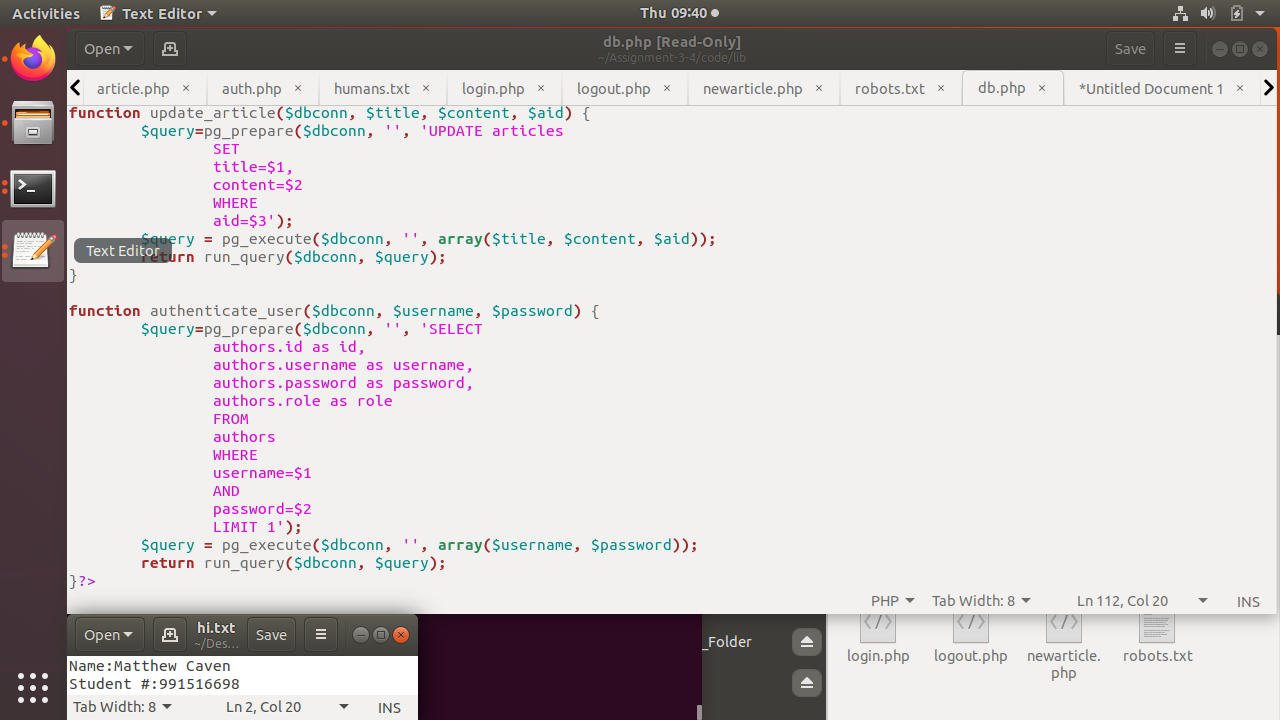


The ability to login as the admin by telling database you are the admin. This is done by inputing ‘ or 1=’1 in both username and password. Then you are logged in. This is both dangerus and will have high impact. Use prepared statments that are used for the filtering of user input and variables.

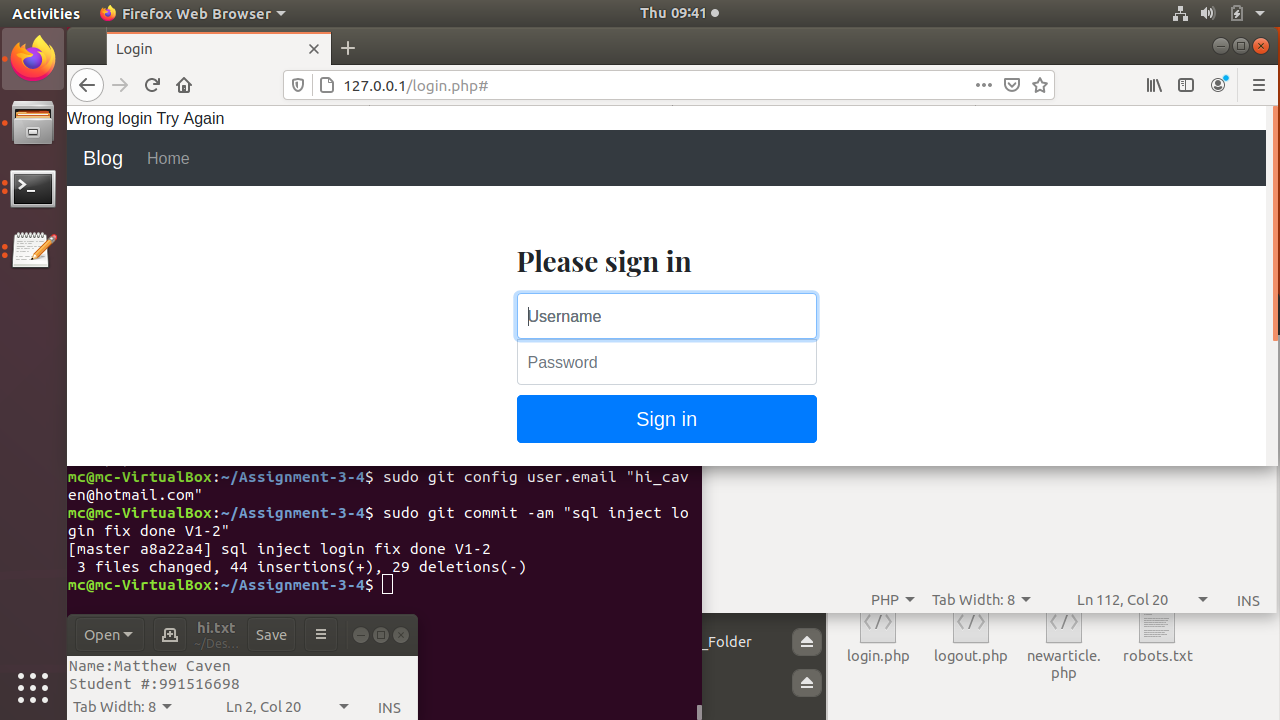
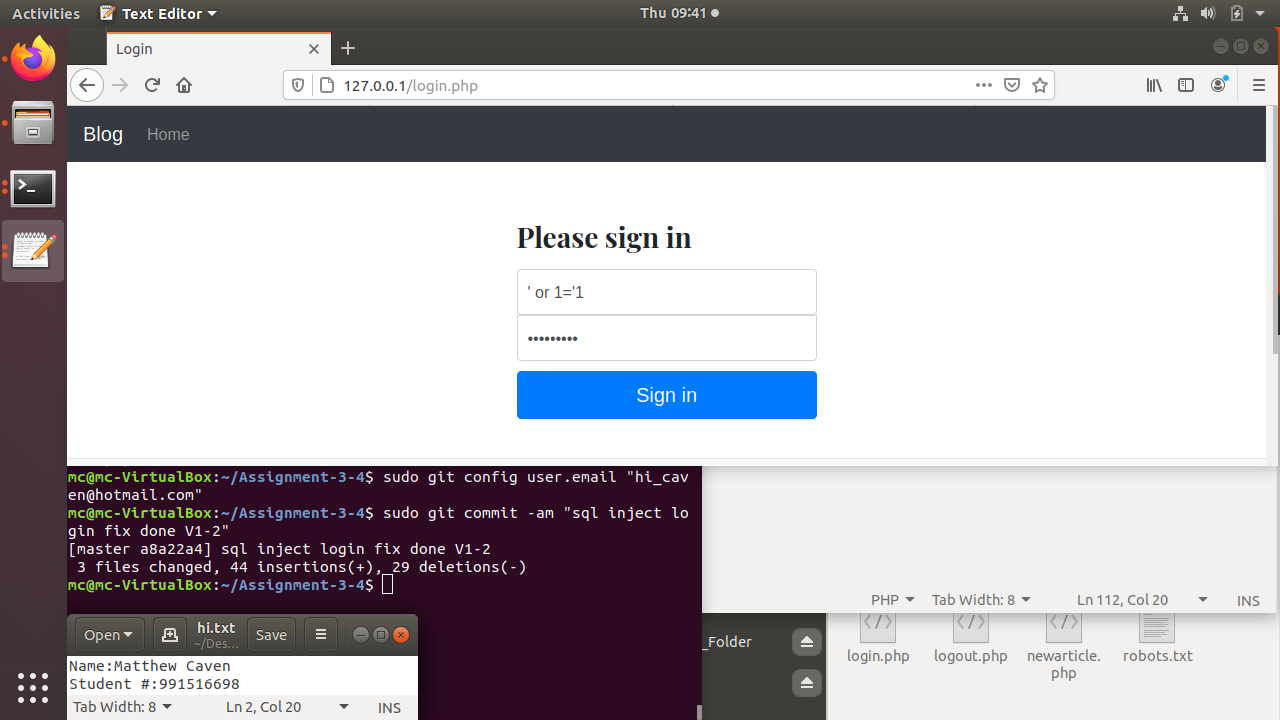
This is fixed by making everything into pg prepared statments and using pg\_excute to filter and excute for the variables.

db.php got the pripared statments

example

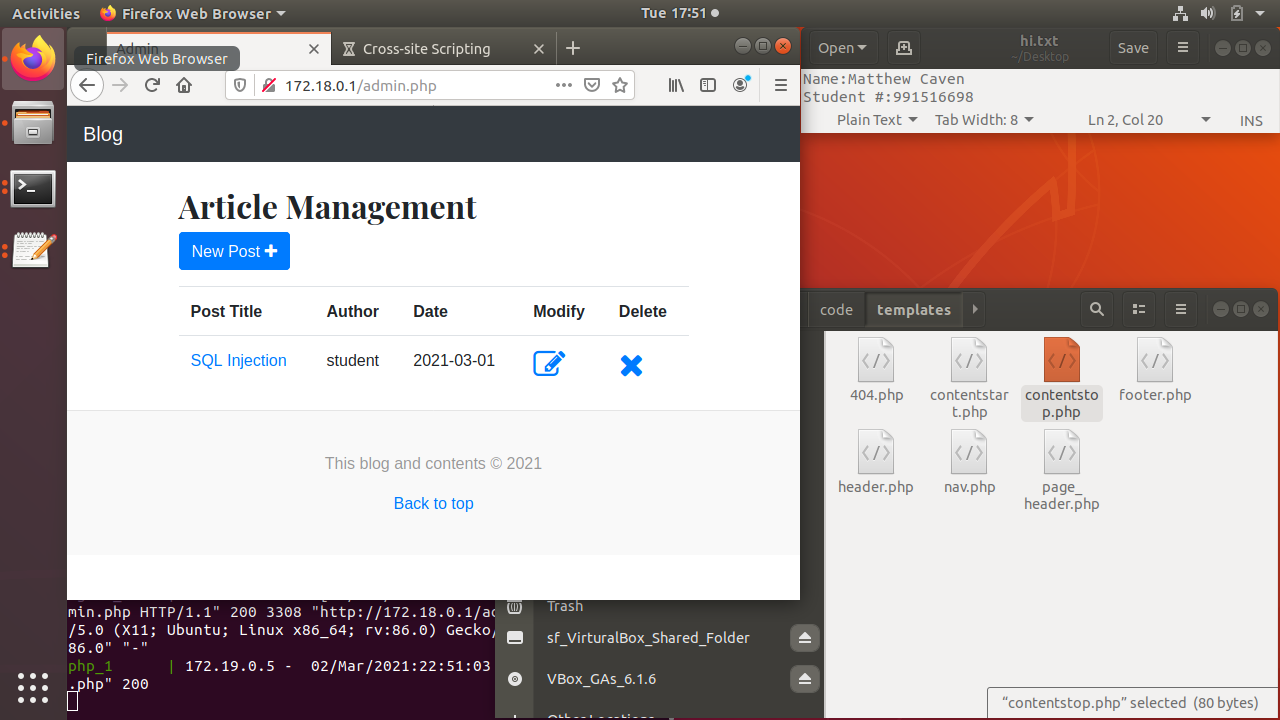
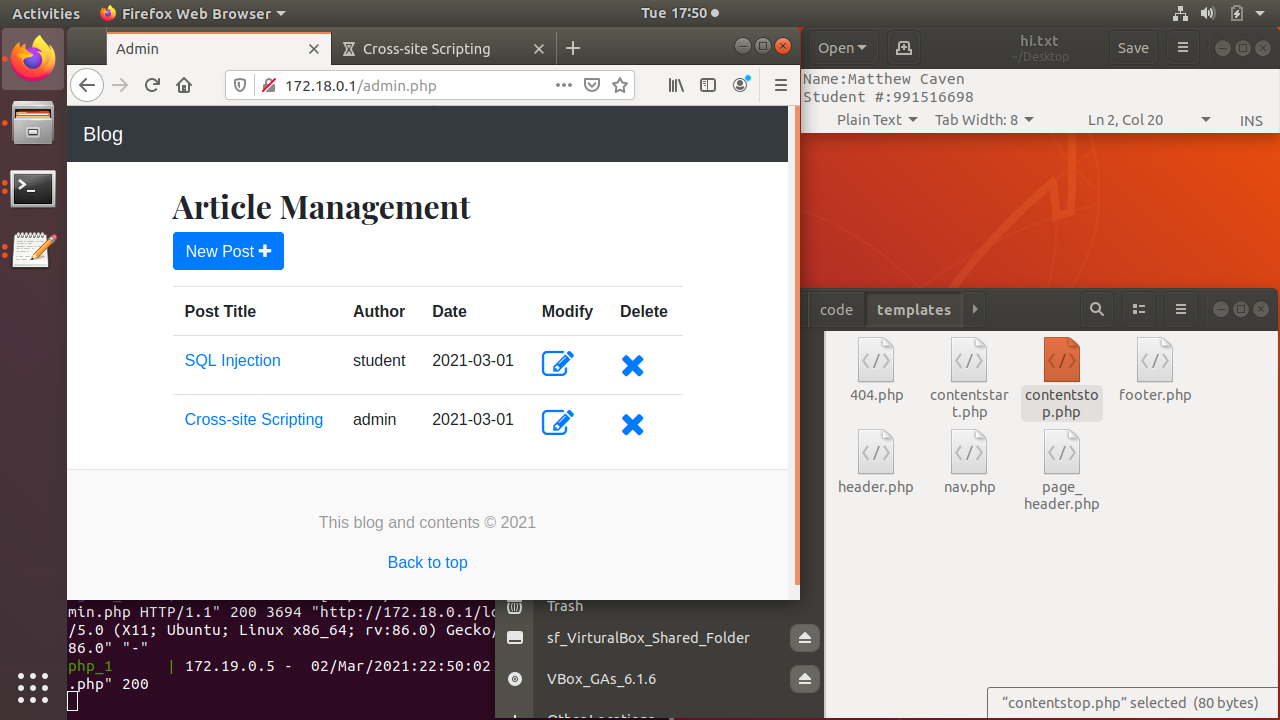
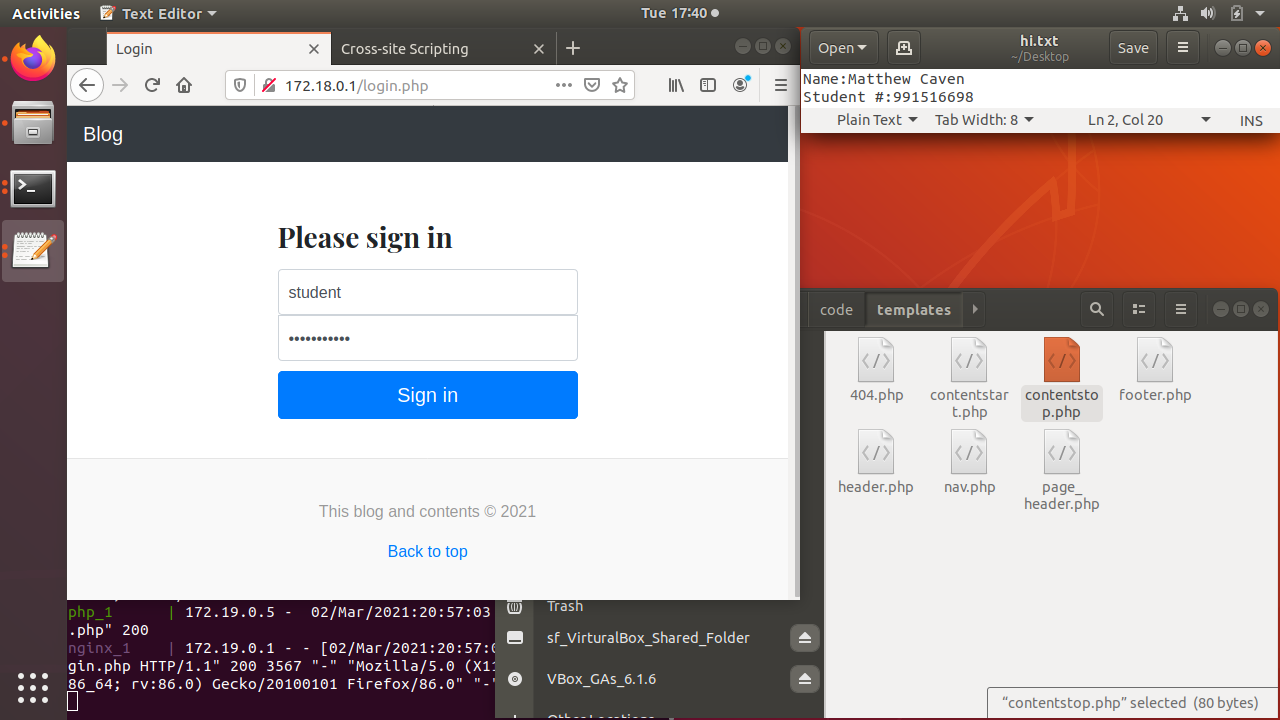


proof



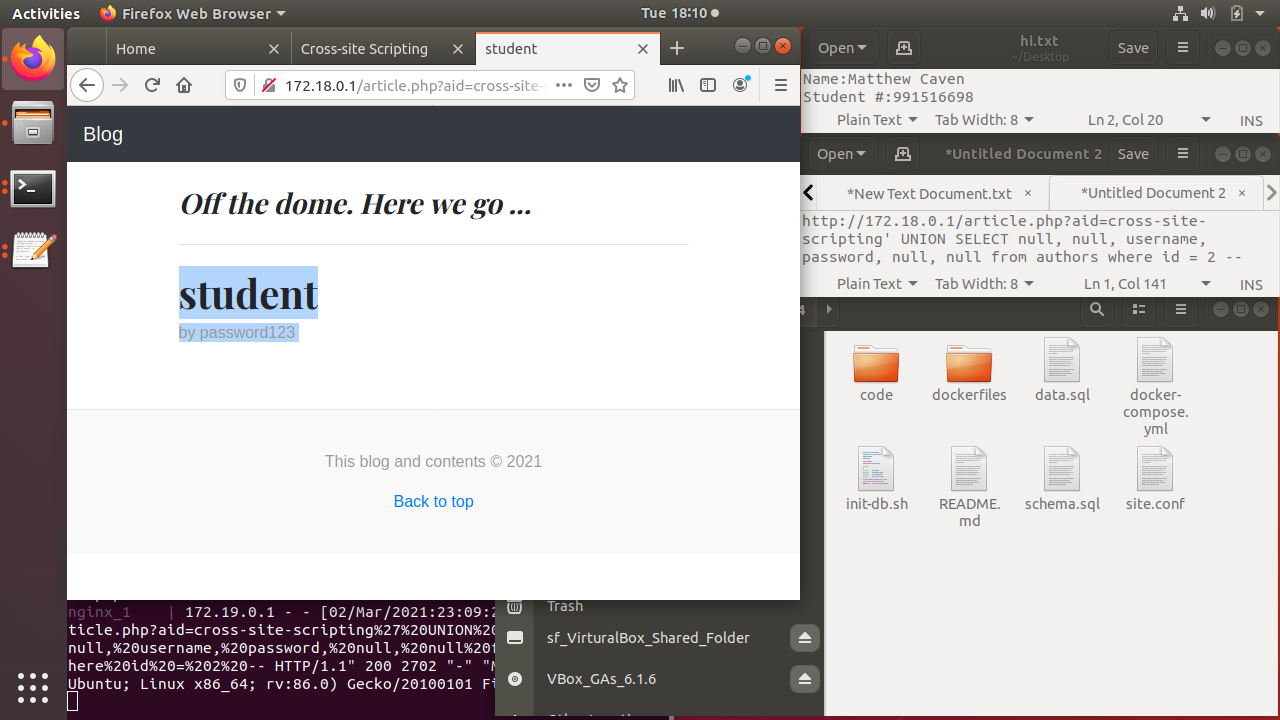
V-3. Broken Access Control (i.e. you can do things without authenticating that you should be able to do.)

V-4. Missing role-based access control enforcement and management (i.e. Blog authors should only be able to delete their posts. Admins can delete anyone's posts. There is currently no mechanism for changing or managing the roles for users. Actually, there is no mechanism for even creating or managing users without issuing manual SQL against the database).



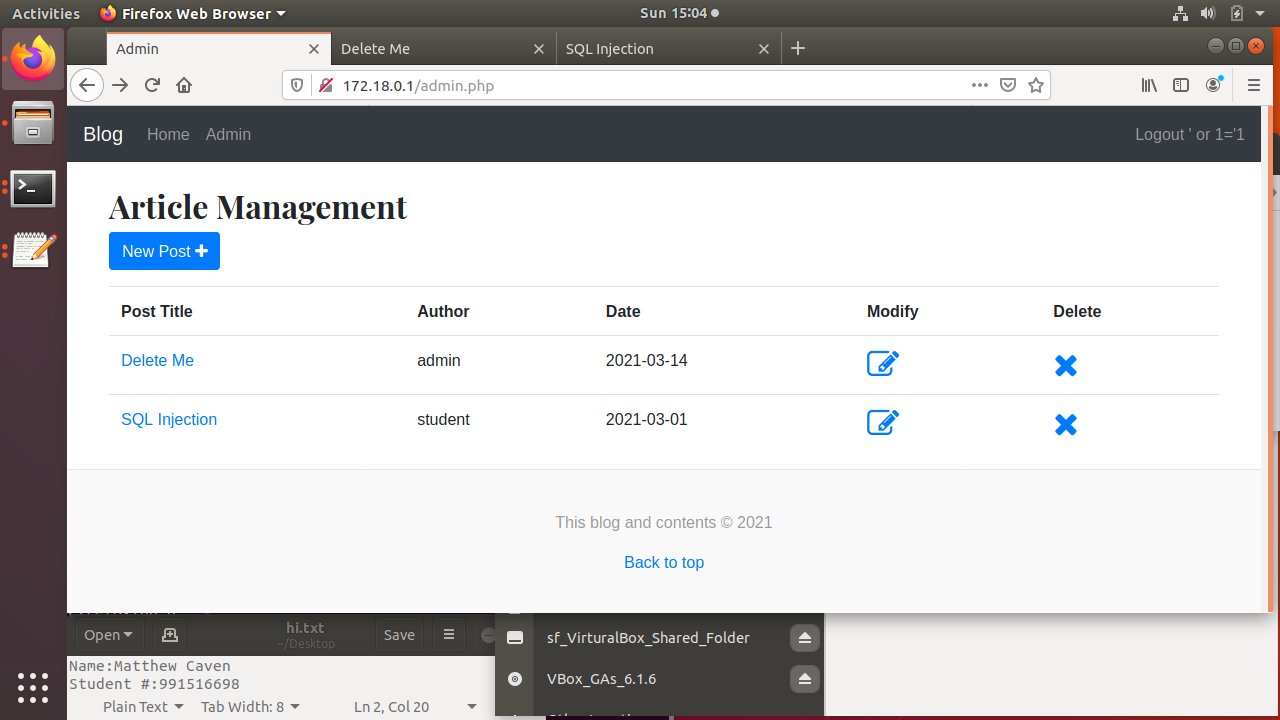
Student is able to alter and delete things by admin. This is decently severe sence the ability to change things you should not be able to is dangerus. The implametion of a permitions enforcment and check is needed.This would be done on the website ena d the sql server to stop student from being able to edit stuff in the database by admin not just no the front but backend to. You could also sto them from seeing the stuff they dont have permitions for. This could be done as a form of masking.

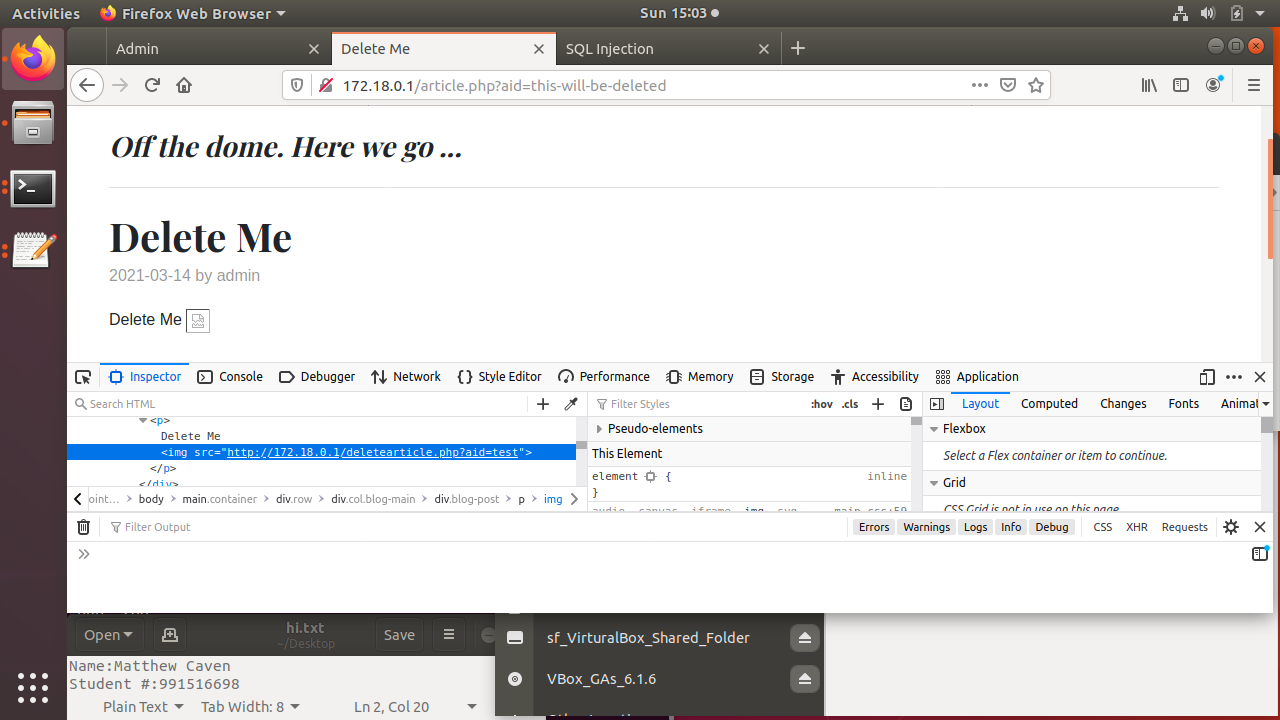
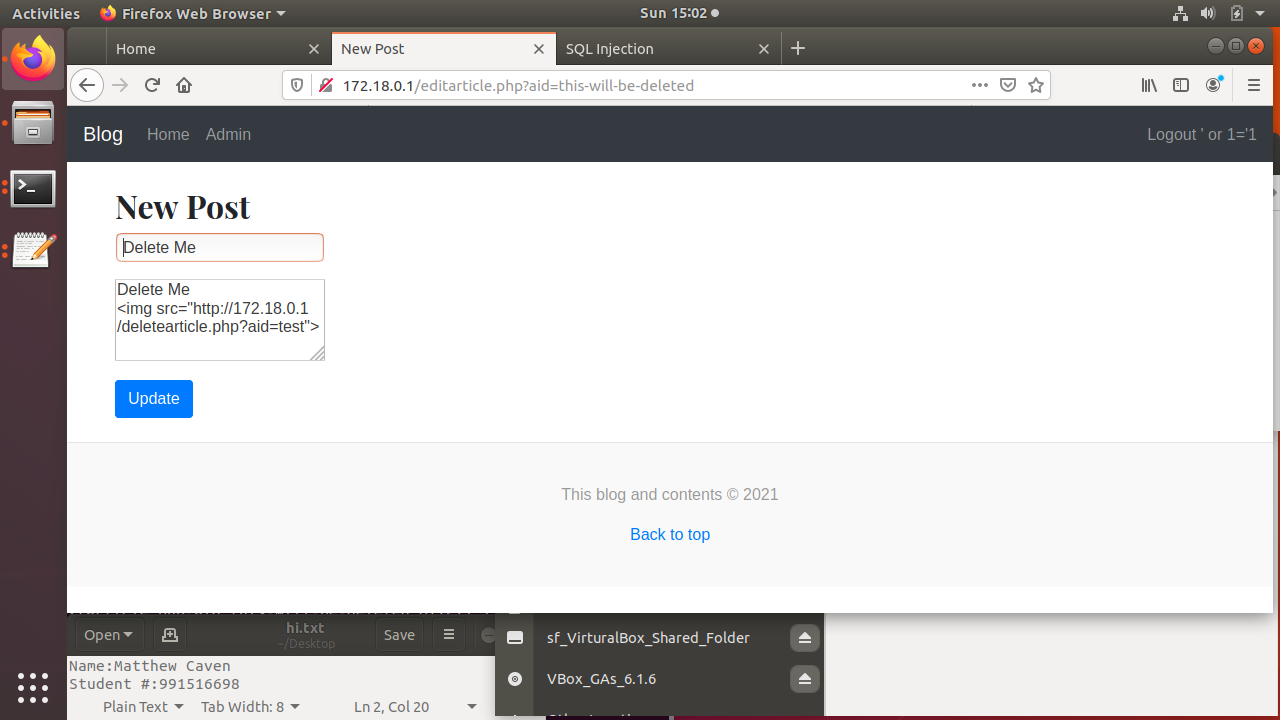
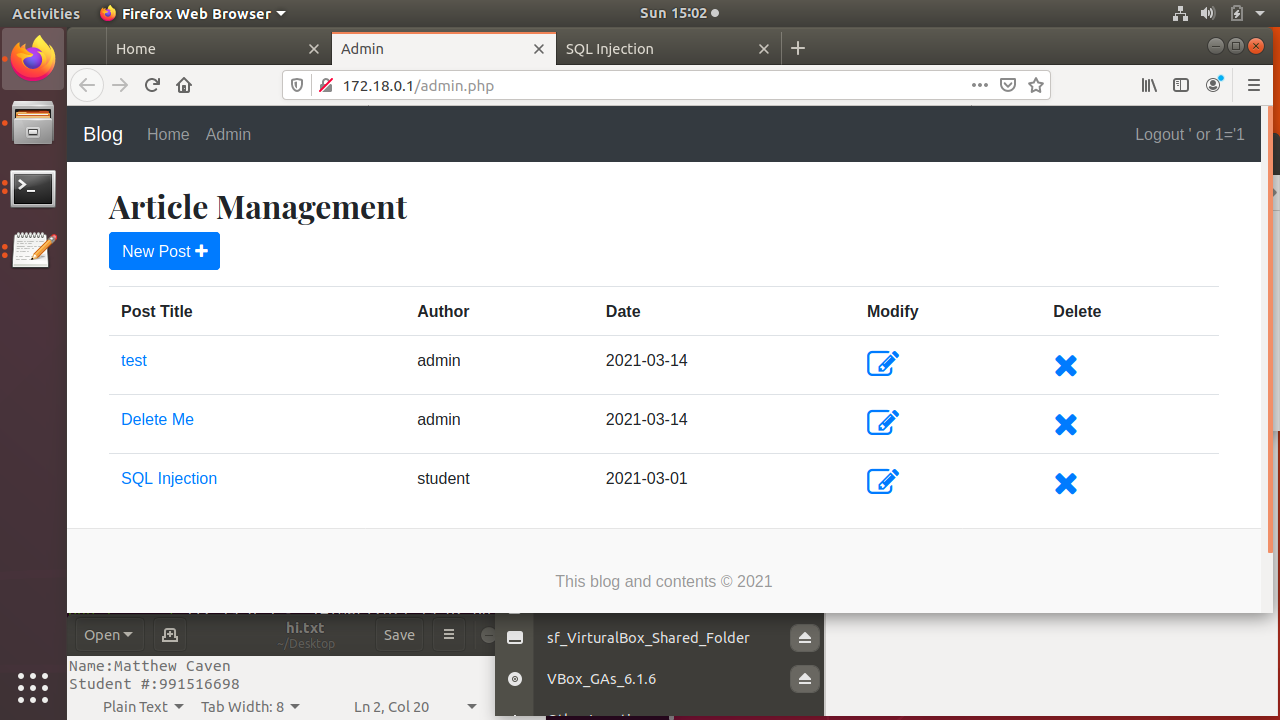
V-5. Insecure password handling and storage.



Anyone is able to both see and alter usernames and password for logging in. This is extremly dangerrus and has a large impact. The best is to mask hash passwords and maybe even usernames. Then make it so only admins can acces the tables holding the info. Encrypt the tablespaces and mask the tables.

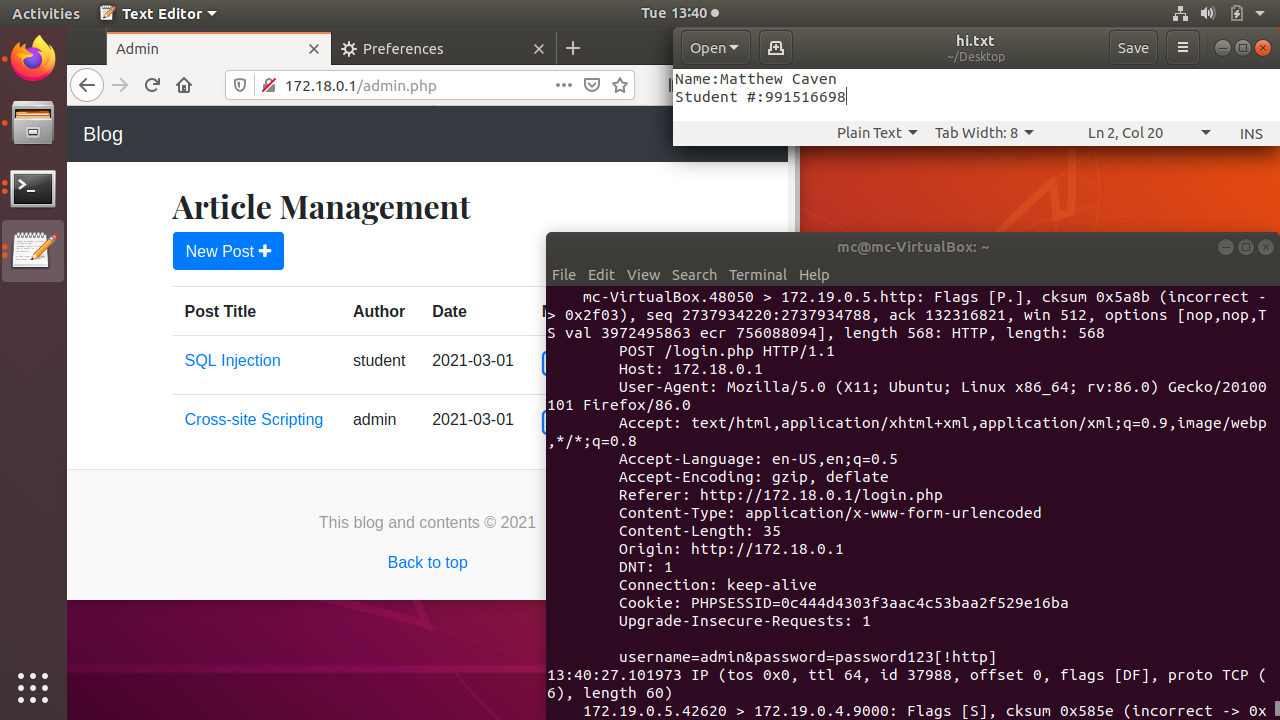
V-6. CSRF





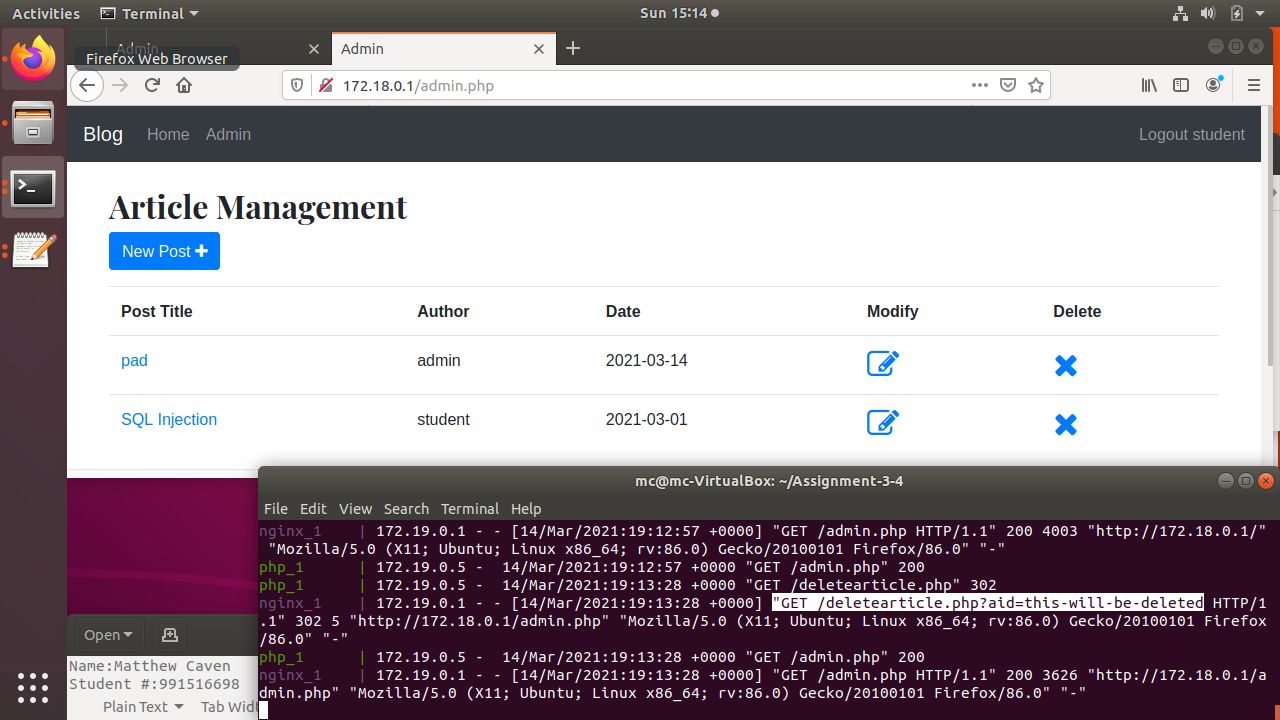
You are able to bypass security by using crosssite scripting to deelete an artical. This can have a large impact becase is bypasses all security mesures. This could bemitigated with validation to stop script injection or you might be able to imploment fixed staments.

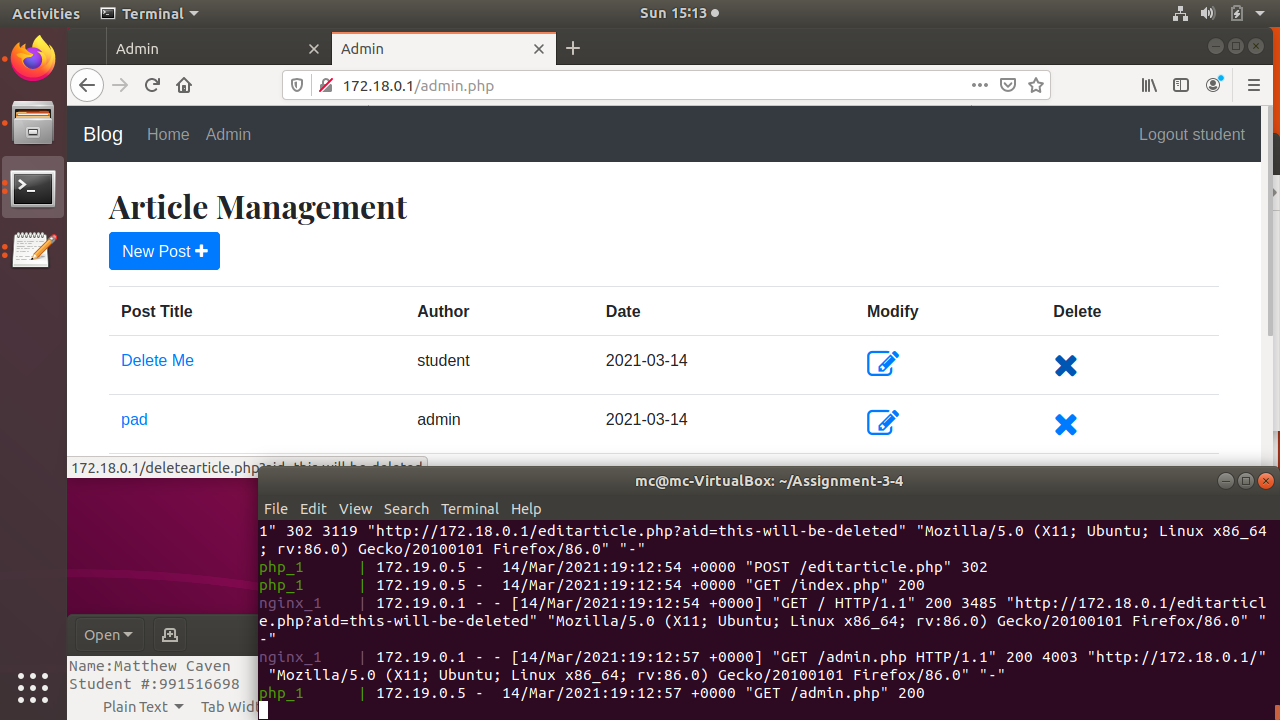
V-7. The entire website, including the login page, is served over plaintext HTTP.



All traphic and be caght and and seen in plain text including login credenchals. This is bad and will have a large impact. There sould eb some sort of encryption applyed to importent if not all trapic on the site. You could also switch to https insted of http proticals.

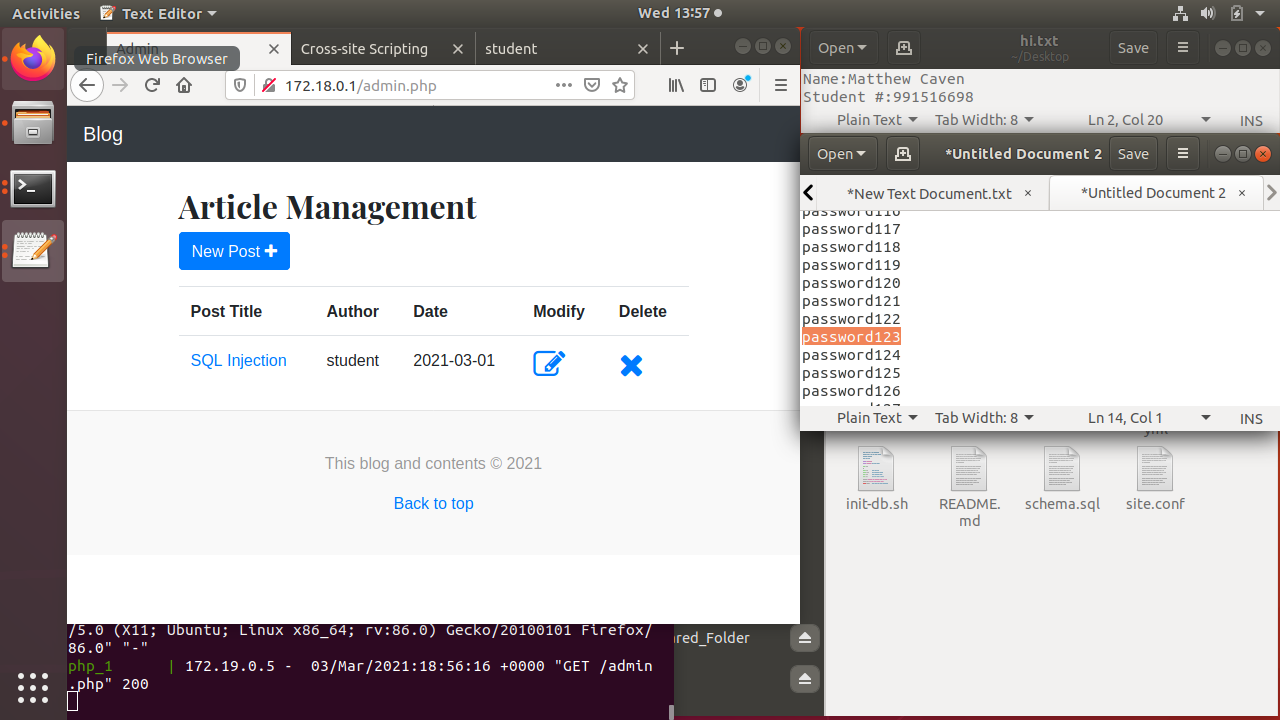
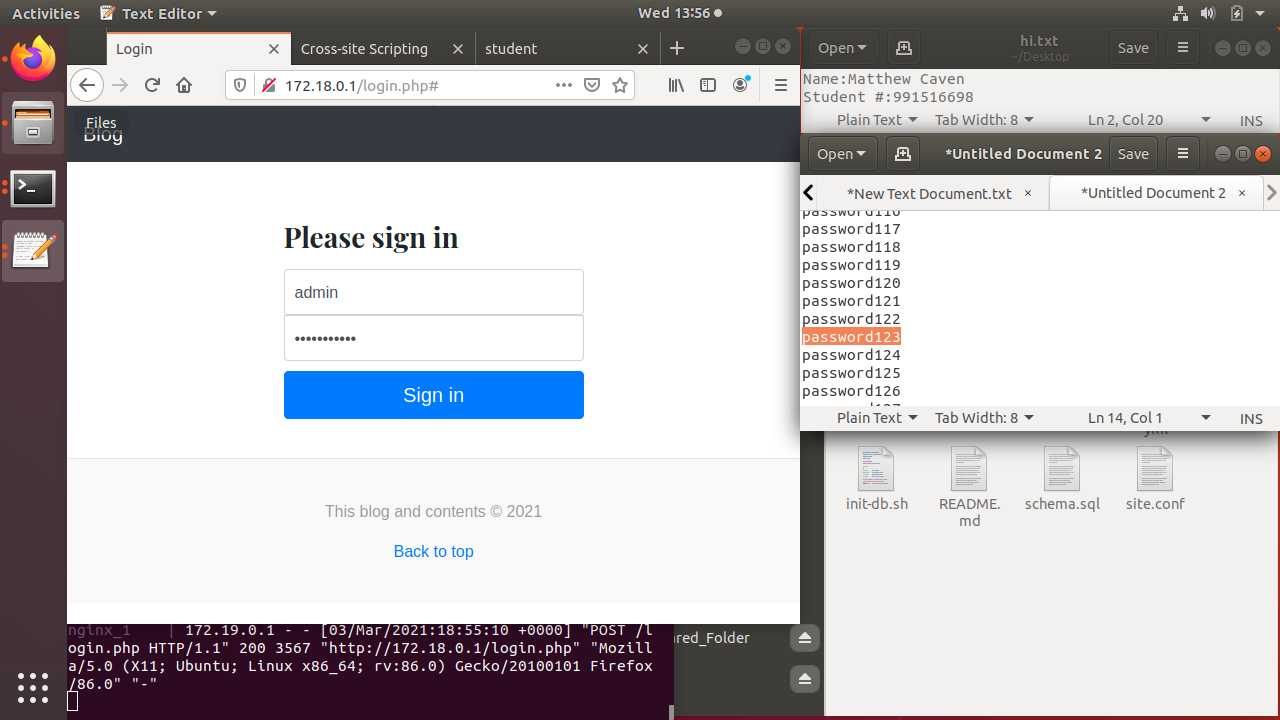
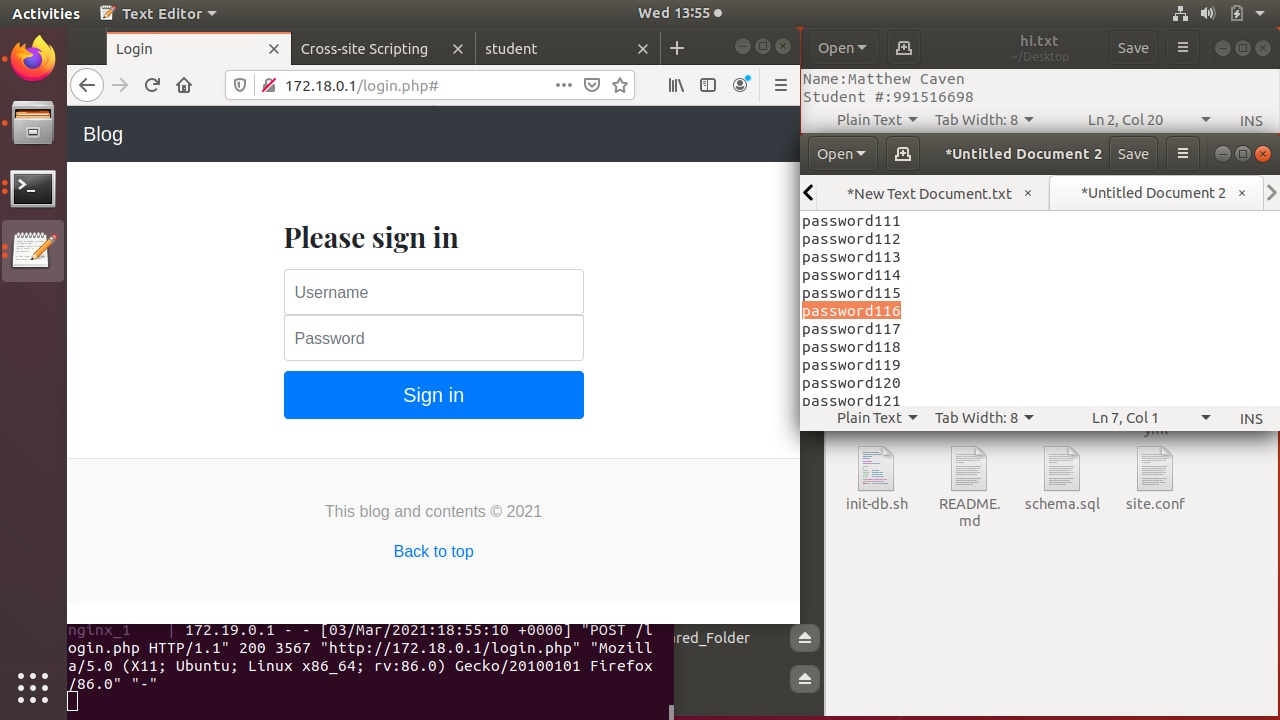
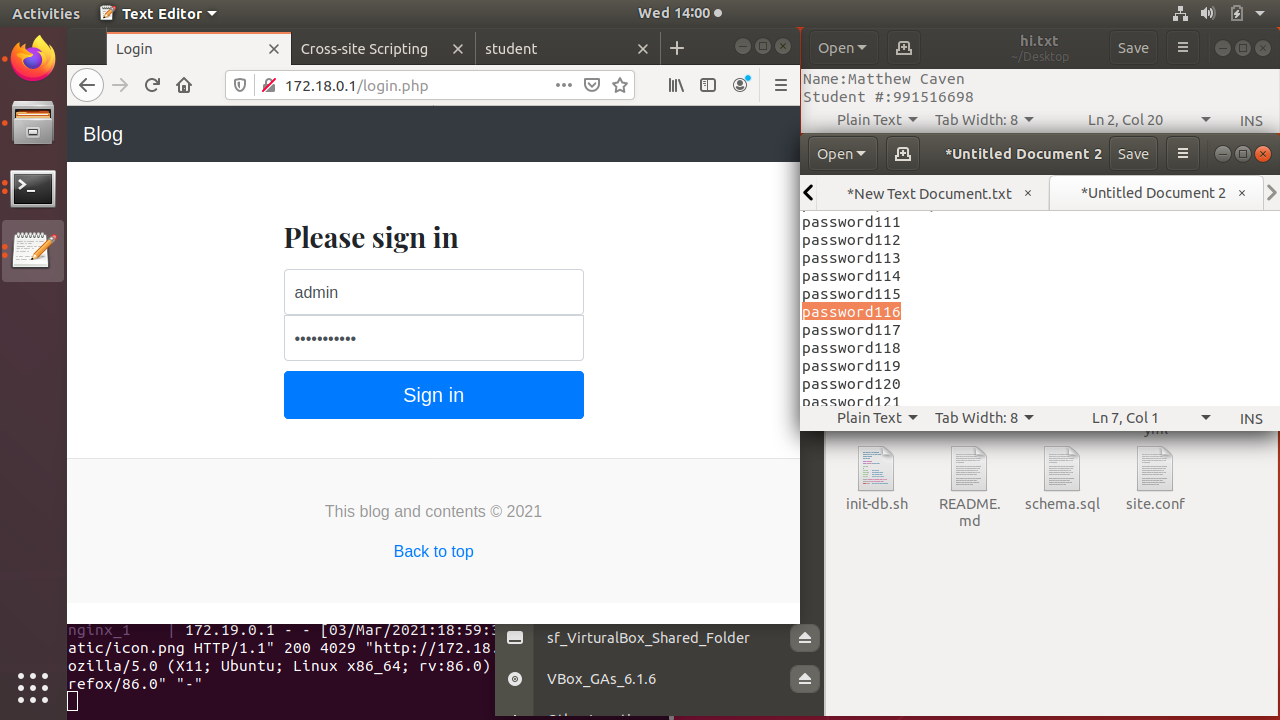
V-8. The web application has no logging except for the default logs generated by Nginx.





There is no tracking of anything but actions in the logs. It does not say who did what. This is a very bad thing but will not comeup untill needed. The lack of detales in the logs will be an issue when you want to know what has happend. The adding of a logping application, plugin or even checking advanced loging option are potenchal fixes for the issue in the future.

V-9. The website only uses single-factor authentication.



The use of only what is know is vulnerable to brute force attacks. This is not very sevear and may have some moderit amount of impact. The implemention of a second form of factor of athentication is aodius form of way to remidy the issue. Another is the use of a form of human verification to prove the person is human.

Objectives

O-1. For the vulnerabilities described above (V-1 to V-9), find and exploit at least one instance of that vulnerability.

O-2. Provide proof of your exploit through screenshots or tool output.

O-3. Write a sentence describing the severity and impact of the vulnerability.

O-4. Implement the most appropriate security control for that vulnerability.

O-5. Document the code where necessary.

O-6. Investigate if further security control could be implemented on the application.

Evaluation & Grading

E-1. 2 marks for each vulnerability: 1 mark for identifying/exploiting and providing proof. 1 mark for the description of the severity and potential impact of the vulnerability.

E-2. 2 marks for each successfully implemented and tested security control: 1 mark for the control implementation, 1 mark for testing the control (tests should be sufficiently robust).

E-3. 2 marks for using proper version control practices (i.e. git on either Github, Bitbucket, Gitlab, etc) to regularly commit and save your work. If you're working with a partner your changes must be merged regularly into the master branch.

E-4. 1 marks for documentation

E-5. 1 mark for exploring the option of further security controls and reporting the finding.

Bonus Mark!

B-1. Implement database caching using memcachd. 1 mark.

B-2. Implement persistent session management using redis. 1 mark.

B-3. Allow Articles to be written using either some simple HTML formatting tags (like <h1>, <ol>, etc.) or Markdown but still prevent the use of malicious JavaScript in articles. 1 mark.

Deliverable

D-1. For each of the vulnerability described above (V-1 to V-9), take screenshot demonstrating your security fixing (and bonus parts) (E-1 and E-2 above) and submit it (see Submission Guideline below for details).

D-2. Take screenshot showing the version control repository (E-3) and submit it.

D-3. In addition, add your instructor ([syed.tanbeer@sheridancollege.ca](mailto:syed.tanbeer@sheridancollege.ca)) as a collaborator to the git project so they can review the commits and commit messages.

D-4. Upload a complete copy of your source code to the SLATE Dropbox.

Submission Guideline

**Step 1**: Create a document file (e.g., Word file, or PDF file) with name following this format: <YourFirstName-Partner'sFirstName><A3-4>. Record following information in the first page (either at the top of the page or in a separate cover page) of the document:

o **Course:** PROG38263

o **Assignment:** 3-4

o **Member 1 Name and Student ID:** <Member 1 name and student id>

o **Member 2 Name and Student ID:** <Member 2 name and student id> o **Section:** 34777 or 34778 or 34779

o **Instructor:** Syed Tanbeer

**IMPORTANT: 2 marks deduction for missing/incorrect cover information and/or your name and id in your program (as commented)**

**Step 2**: For each vulnerability, record in the document file, the screenshot(s), required description and documentations with appropriate reference for that vulnerability, and screenshot for showing version control repository.

**Step 3**: Document the option of further security controls and reporting the finding in the file.

**Step 4**: Answer the bonus part in the file.

**Step 5**: Submit the document file, and complete copy of your version of source code as a single zip file named <YourFirstName-Partner'sFirstName><A3-4>.zip on Slate Dropbox for Assignment 3-4. You can submit the assignment multiple times, but only the last submission will be marked. Make sure the last submission is complete. For group submission, one of the members should submit the assignment.