Lab 8: Security and Cipher Tools Documentation

Jade Pearl

SDEV 300 Building Secure Python Applications

Prof. Craig Poma

Pylint Score: Overall 10/10.0

My score was a 10/10.0 on pylint with no needed changes.

Test Case Table

|  |  |  |
| --- | --- | --- |
| Test Case (Expectation) | Input/Output as shown on Terminal Console | Pass/Fail |
| 1a: User login with invalid username | Used a username that did not exist | Pass |
| 1b: User login with incorrect password | Used the correct username but invalid password | Pass |
| 2: failed-logins.txt (failed login log for this lab) is connected and gets filled | The two failed login attempts I did above were written in the failed-logins.txt file I had | Pass |
| 3a: Update password with common password | Common password used from CommonPassword.txt: free    As a bonus, I added the NIST functionality to the User Registration page so that users could not register with a common password on there as well | Pass |
| 3b: Update password with invalid complexity | I used my current password without the uppercase letters | Pass |
| 3c: Update password works with next login | When I input proper credentials, it brought me to the home page: | Pass |

Log File and NIST Password Complexity

For this lab, the log file proved to be an interesting addition and I can understand why we implemented it. I think a log file such as the one we used for this lab is particularly useful for detecting patterns of abuse because if the moderator/administrator of a website sees that someone has a lot of failed login attempts for an account in a short period of time, they may have reason to believe that someone is trying to break into that account. They can also keep track of previous failed login attempts and compare IP addresses from the failed login attempts. If they detect a different IP address from before is trying to get into the account, it could be a hacker. Some websites and platforms allow a certain number of failed login attempts before they ask the user to change their password from the email associated with that account in order to throw off hackers and to encourage the users add additional protection to their account such as multifactor authentication. Other sites may lock the user out of the account for a certain amount of time before they can try logging in again. Lastly, they can send users notifications that their account had a failed login attempt from an unrecognized device not associated with their account. All of the above methods could be helped with a log file.

For the NIST password complexities for the lab, I made sure to first cover the aspect of common passwords. When giving users the option to update their password or when a new user registers for an account, I referred to a list of common passwords that was given and prevented the user from being able to use one of those common passwords. Another NIST factor I considered from last lab and this lab was password complexity of at least 12 characters containing at least one uppercase letter, lowercase letter, number, and special character. This was covered in my last lab and the user is forced to meet that criteria whenever trying to register update their password.

Decrypting Secret Messages

1. - .... .. ... / ... -.. . ...- / ...-- ----- ----- / -.-. .-.. .- ... ... / .... .- ... / ... --- -- . / ... - .-. .- -. --. . / .-. . --.- ..- . ... - ... .-.-.-

This message is written in morse code and decrypts into “this sdev 300 class has some strange requests.”

1. U28gdGhpcyBpcyBiYXNlNjQuIE5vdyBJIGtub3cu

This message is a base64 message that says “So this is base 64. Now I know.”

1. --- Psuwb Ysm ---- W oa gc qzsjsf. Bc cbs qcizr dcggwpzm twuifs hvwg cih. --- Sbr Ysm ---

This is a plain cryptogram message that translates into “--- Begin Key ---- I am so clever. No one could possibly figure this out. --- End Key ---”. I used a simple cryptogram solver from the rumkin.com website to decode this message.