|  |  |
| --- | --- |
| **COMP260 – Distributed Programming** | **Worksheet 9** |
| **Creating Secure User Accounts with Hashing and Salting** | |

**Introduction**

The goal of this worksheet is to explore user accounts and making them secure for use in your MUD servers for assignment 2. To do this, there are three activities described below that will allow you to isolate the complexity of secure user accounts into functional sandboxes that will build your understanding and confidence. Finally, you can engage what you have learnt in the activities to create a secure user account system for your MUD services, which covers the ‘user security’ and ‘data persistence’ criteria of assignment 2.

**The command line password tool**

As a starting point, take the hashing code from this week’s slides and use this to create a simple command line application in Python that will let you explore hashing and salting functionality. Ideally, you command line tool should allow a user to create new password hash it and add a salt to it. The user can then attempt to re-enter their password using the input() class, their entry will have the salt applied, will then be hashed and the two hashes can be compared.

**The command line password tool with persistent storage**

Take what you learnt from last week’s workshop to create an SQL database using sqlite3 which will store a username and associated password hash and salts to create a persistent user account database. When a new username is added, check to make use that it’s not in the database already.

**The PyQt5 password tool**

Extend your existing command line tool to support an appropriate UI/UX for account creation and account login. If you fancy using a modal dialog to manage this, PyQt’s QDialog class has a setModal() method for making dialogs act in a modal way.

**Next Steps**

Now apply all of this to your MUD at get your colleagues to create accounts, log in and play on your MUD service.