MedView Imaging User Authentication and Access Control System Prototype

In summary, the MedView Imaging user authentication and access control system prototype has 4 major parts.

The 1st part is an RBAC and ABAC access control system that takes in a role as an input argument and prints out the permissions for that role, is also able to check if the use has permission if given the object and the action as an argument.

The 2nd part is a password authentication module that is able to take in a password, generate a random salt using bcrypt, and use the hashlib's pbkdf2_hmac hashing algorithm on the password using the salt. It is also able to store the userID with the role name and the salt and the hashed password inside a passwd.txt text file.

The 3rd part is a user enrolment interface that will take in a user's input of a userID, a role, and a password, and adds it to the passwd.txt password file.

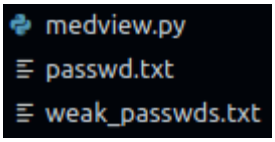
It will also check if the userID is a duplicate of another userID within the password file, and if the role is a valid role or not, and features a password checker that will check if the password is an acceptable password that adheres to the MedView Imaging password policy. It will also feature a weak_passwds.txt file for MedView to add in more vulnerable or exposed passwords without changing the source code.

The 4th part is the user login and authentication module. It will ask the user to input their userID and password, and calls the password authentication module created in the 2nd part, and once the user is authenticated, it will display the user's userID, their role, the current time, and the permissions they are able to have in the MedView system. Also, there is a user_action_interface that gives a list of all the actions the user can do, and lets the user decide which action they want to take, and prints out whether they have permissions to do the actions or not.

Running the Prototype

To run the prototype, please ensure the following files are present in the directory as shown in the figure below:

Figure 1. MedView Imaging user authentication and access control system prototype file directory



Next, type in the terminal for the same directory the following command:

python3 medview.py

For the purpose of the demo, the medical_image_interface_demo function is set up already.

The code listing for the function is listed below:

Figure 2. medview_imaging_interface_demo function

```
def medview_imaging_interface_demo():
    """
    Demo interface used to demo the program
    Can be used to access the MedView user enroller system
    and the login system
    """
    while True:
        print('-----------------------------------------------------')
        print('MedView Imaging System Demo')
        print('Please choose an action by selecting the number only:')
        print('1. MedView user enrolement system')
        print('2. MedView user login system')
        print('3. Quit Demo')
        action = input('Enter an action: ')
        if action == '1':
            user_enrolment_interface()
        elif action == '2':
            login_interface()
        elif action == '3':
            print('Thank you for using MedView Imaging System Demo, Goodbye')
            print('-----------------------------------------------------')
            break

medview_imaging_interface_demo()
```

When the medview.py file runs, this function will be automatically called.

User Enrolment Demo

For the first part of the demo, I will use the MedView user enrolment system to demonstrate how this prototype can create a new hashed password with salt to the passwd.txt file, as well as checking if the userID, the role, and the password are valid.

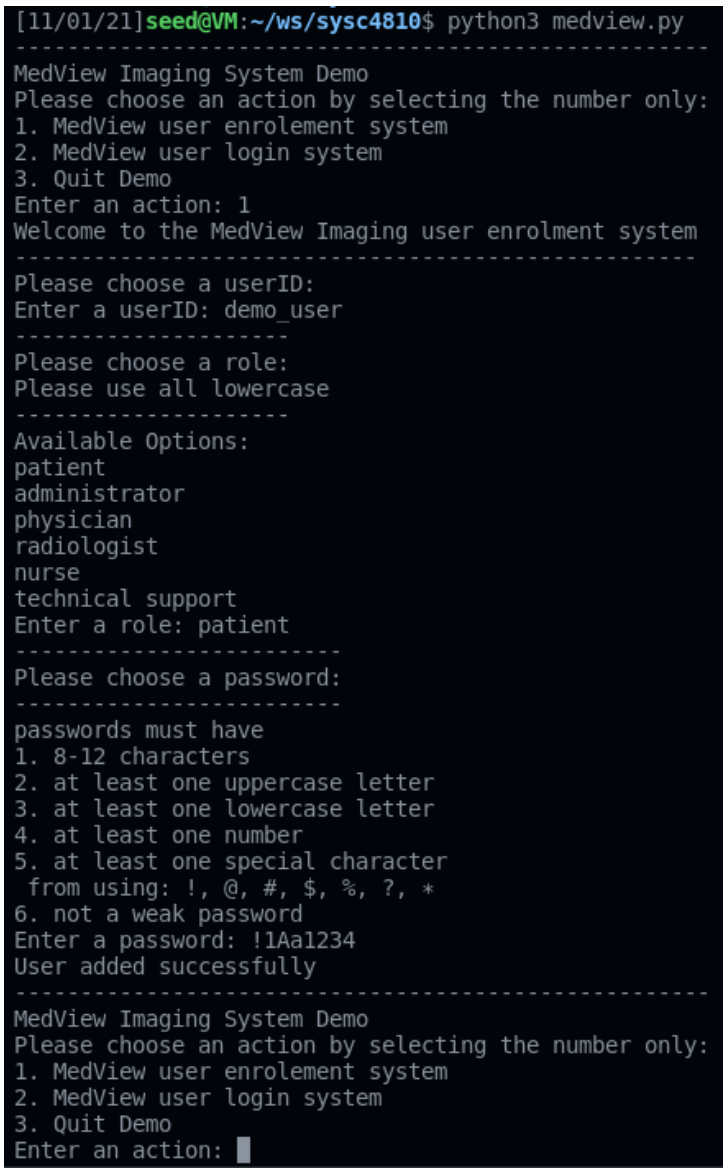
In the figure below, I chose a name of demo_user as the userID

I chose a valid role of the user, and I chose patient

I chose a valid password for the user, and I chose !1Aa1234

And after that the user was successfully added to the MedView system.

Figure 3. User enrollment flow



As shown in the figure below, the demo_user, with the role of patient, along with the salt, and the hashed password was added to the passwd.txt password file.

Figure 4. Password file showing demo_user

Login System Demo

Next, we will try out the user login system flow.

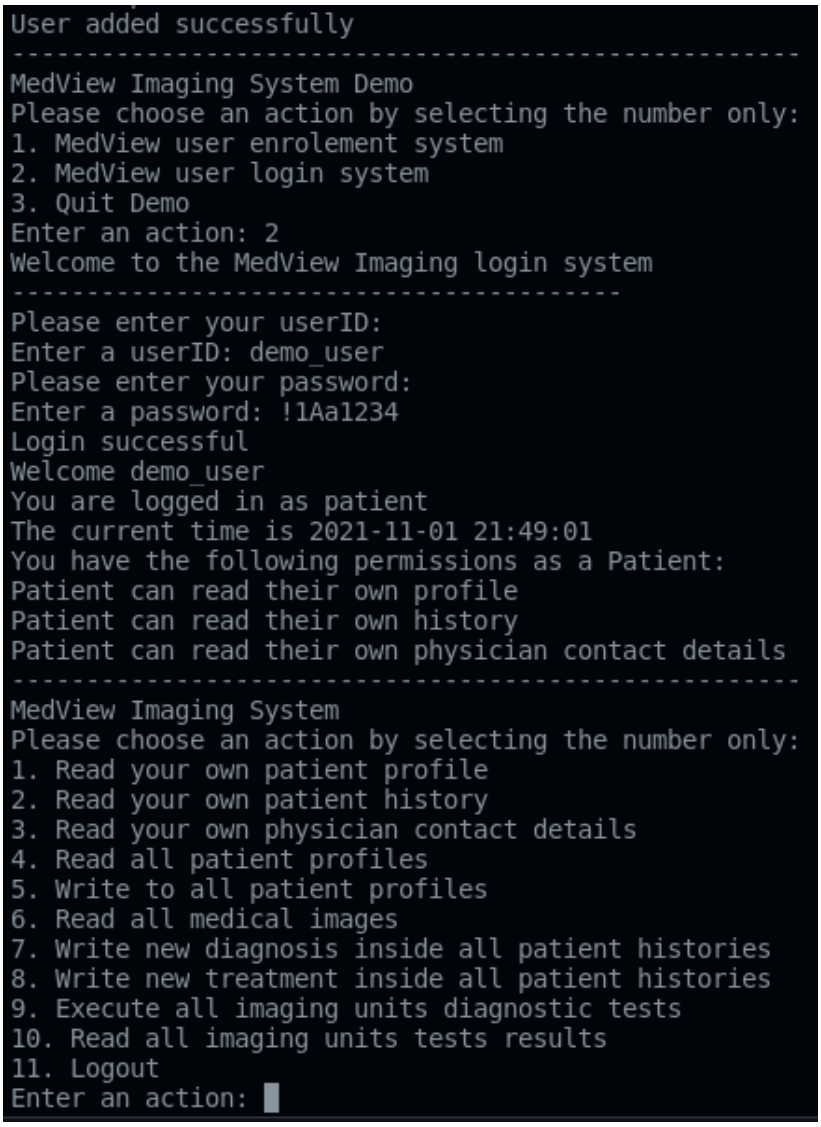
In the figure below, we continue where we left off the with demo_user and we choose the MedView user login system.

We use demo_user as the input userID.

We use !1Aa1234 as the password.

And we are logged into the system, as the MedView system says Welcome demo_user, and that we are logged in as patient, and the current time is 9:49 PM, and also lists out the permissions we have as a patient.

Figure 5. User login flow

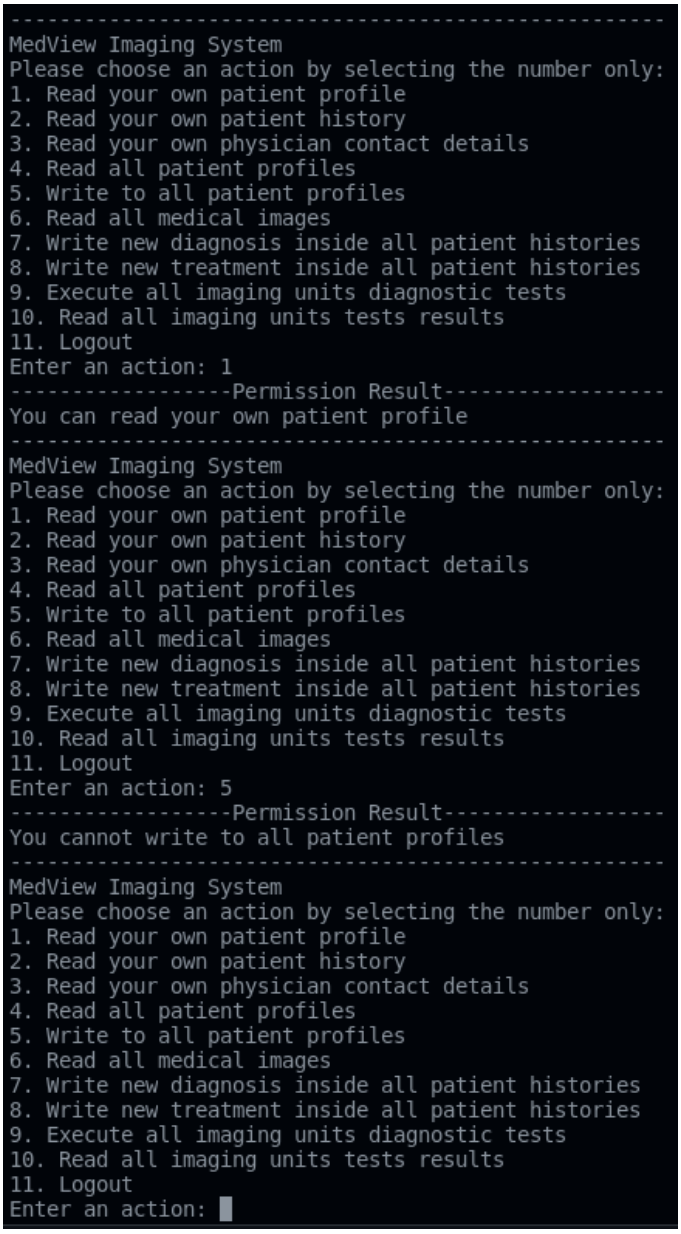


Next, we are able to interact with the MedView Imaging System by selecting actions.

We as a patient can select to view our own patient profile. Which the system permission result will say we are able to read our own patient profile.

But if we pick an action such as writing to all the patient profiles, the system permission result will say we are not able to write to all patient profiles.

Figure 6. Interacting with MedView Imaging System

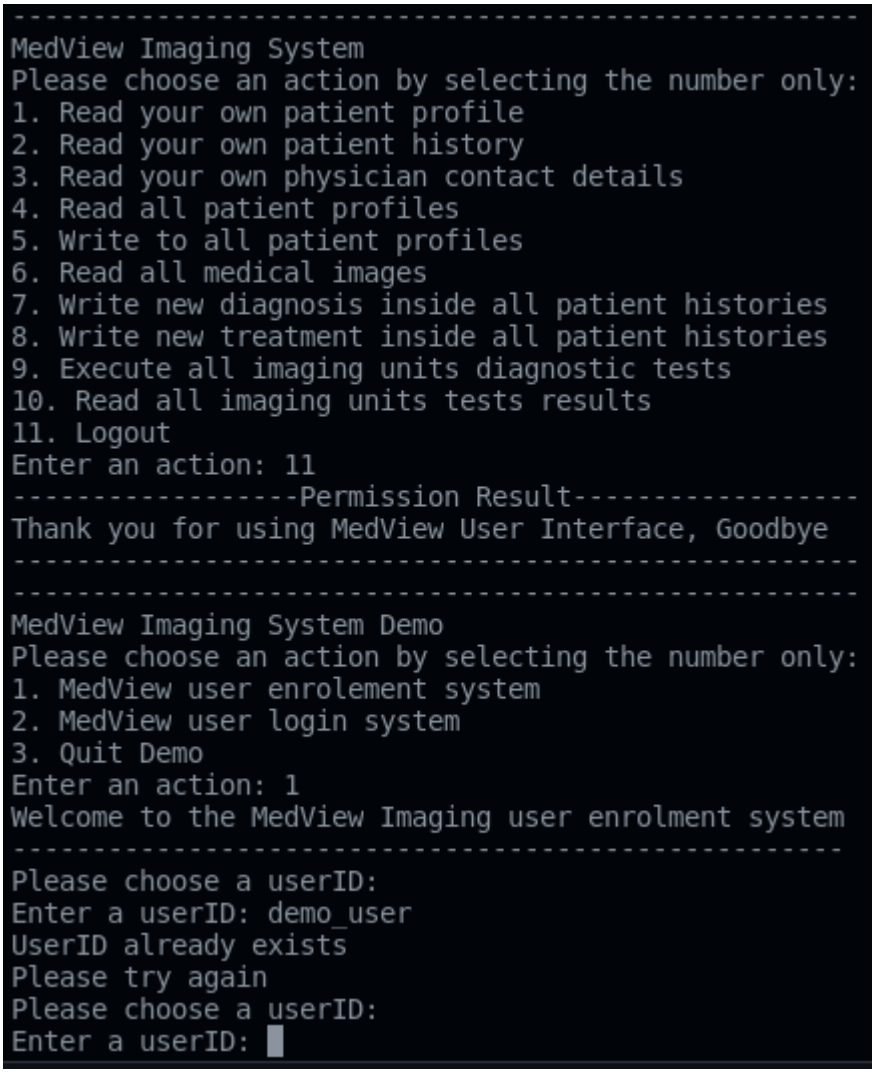


UserId and Role and Password validation

The prototype user enrolement system will also validate if the given userID is a valid userID that does not exist in the passwd.txt file.

In the figure below, continuing where we left off with demo_user, we logout of the MedView Imaging System and continue to the user enrolement system, when we pick the same name demo_user again, the system will give back the error that the UserID already exists.

Figure 7. Picking the same userID as demo_user

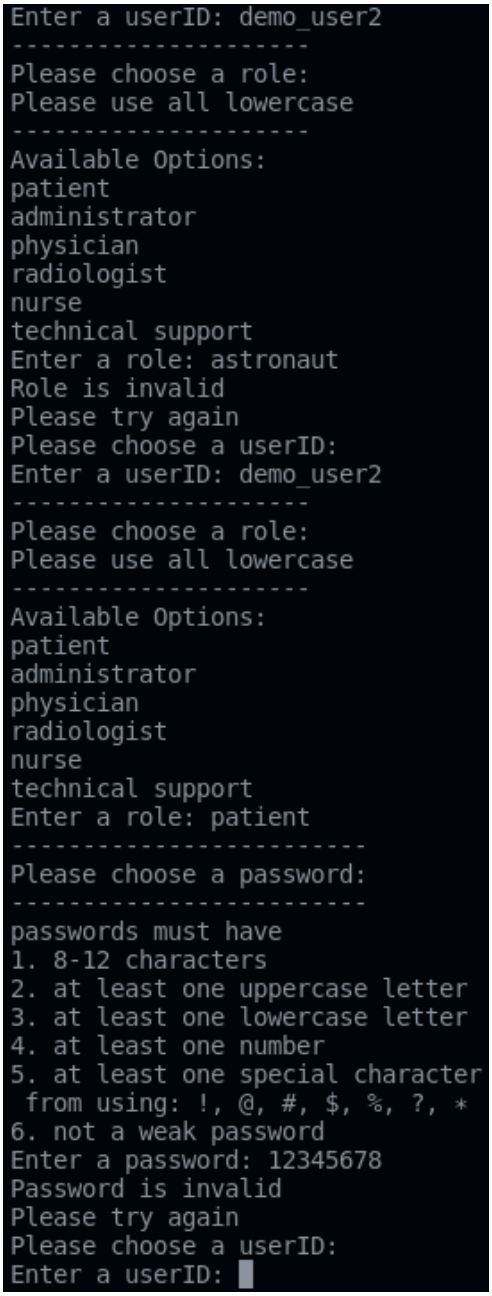


Furthermore, we continue on using demo_user2 as a valid userID.

When asked for the role, if we give a role that doesn't exist inside the MedView system such as astronaut, it will return that the Role is invalid.

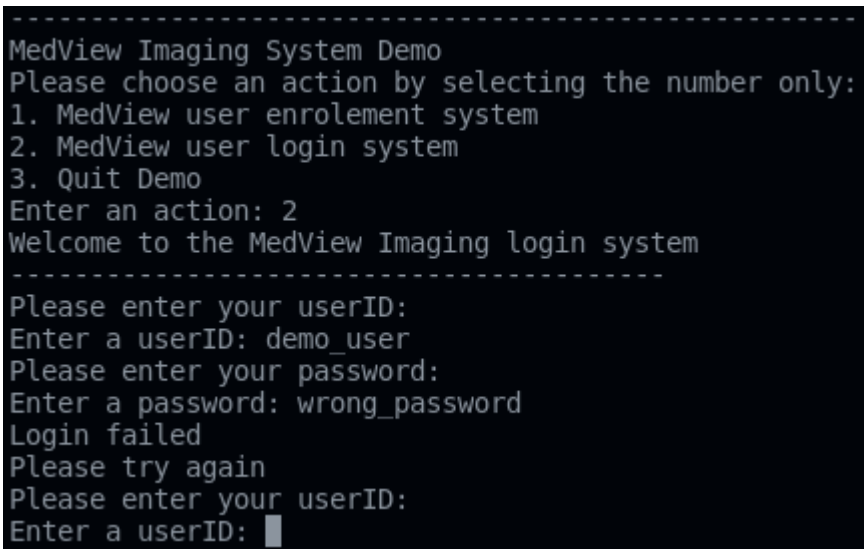
When picking a valid role, such as patient, and continuing on, if we chose a weak password such as 12345678, we will also get told by the system that the password is invalid, and we must go through the user enrolment flow again.

Figure 8. Picking astronaut as role, and 12345678 as password



In addition, for the user login system, if we give the correct userID of demo_user but with an incorrect password as wrong_password, we would also be told that login failed, and we must try the login system again as shown in the figure below.

Figure 9. Picking a wrong password for the login system



Administrator User Demo

Now, we will enroll another user to the system, named demo_administrator, and we give them the same password of !1Aa1234 as the demo_user.

Figure 10. Adding demo_administrator to the system

```
-----------------------------------------------------
MedView Imaging System Demo
Please choose an action by selecting the number only:
1. MedView user enrolement system
2. MedView user login system
3. Quit Demo
Enter an action: 1
Welcome to the MedView Imaging user enrolment system
-----------------------------------------------------
Please choose a userID:
Enter a userID: demo_administrator
--------------------
Please choose a role:
Please use all lowercase
--------------------
Available Options:
patient
administrator
physician
radiologist
nurse
technical support
Enter a role: administrator
------------------------
Please choose a password:
------------------------
passwords must have
1. 8-12 characters
2. at least one uppercase letter
3. at least one lowercase letter
4. at least one number
5. at least one special character
 from using: !, @, #, $, %, ?, *
6. not a weak password
Enter a password: !1Aa1234
User added successfully
```

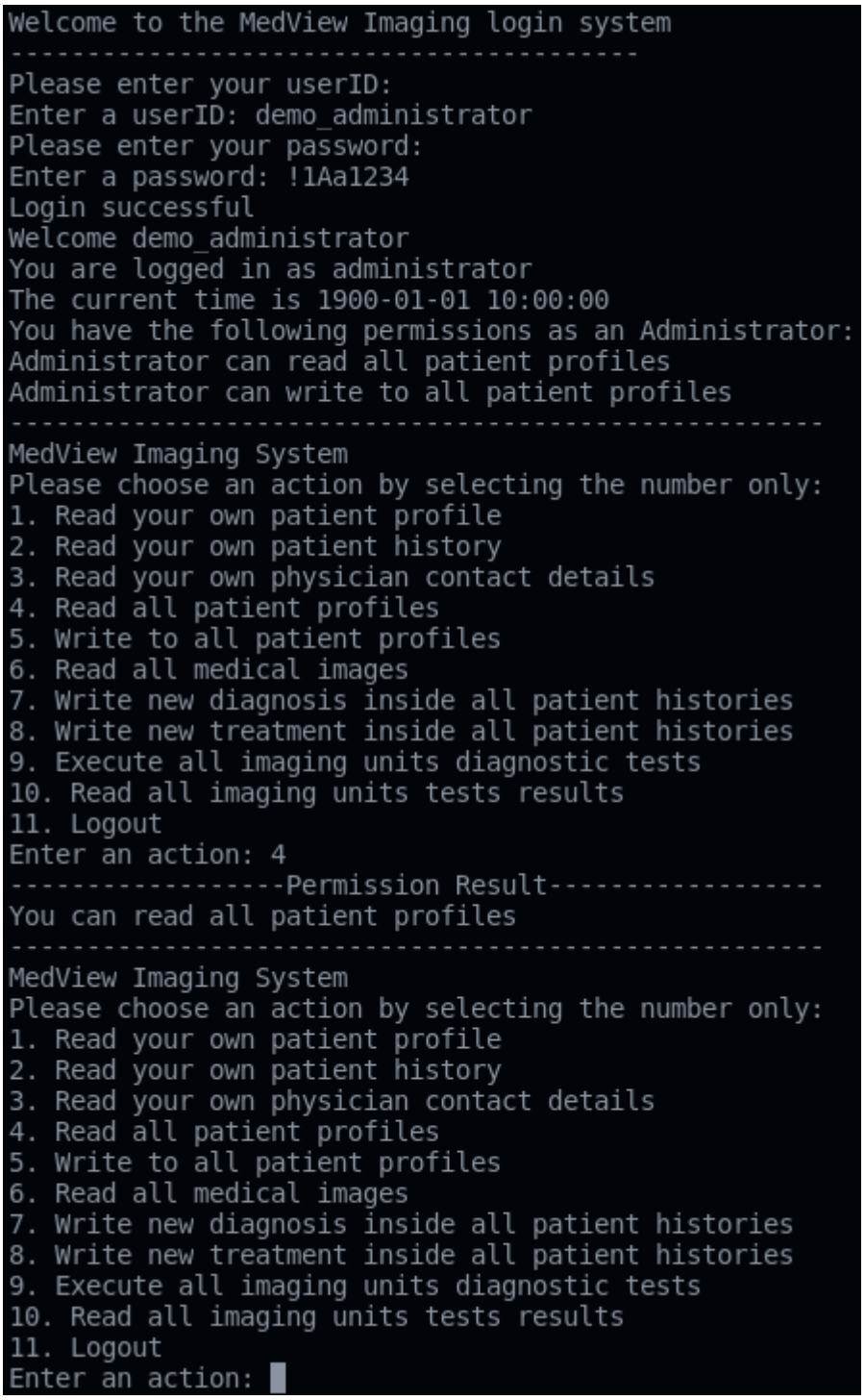
And we can see in the figure below, even though the demo_user and the demo_administrator were all generated with the same password of !1Aa1234, because of the they are salted with different salts, they are saved to have different password hash values.

Figure 11. Password file, with different hash codes

```
demo_user patient JDJiJDE2JHdoMzFUMWg5SFh5bkNHOFFxQTR5ZE8= gOP99kn6OuyH1O4+4PtDifoRqPAZUD7W5pJO/comZG8=
demo_administrator administrator JDJiJDE2JERQT2doWG5Pa2lNcmZ1Q3hWVy5VWi4= NuTbDaGVYF2rwdzoELtY2LqKJBYqMLTC4Otk4wRNI50=
```

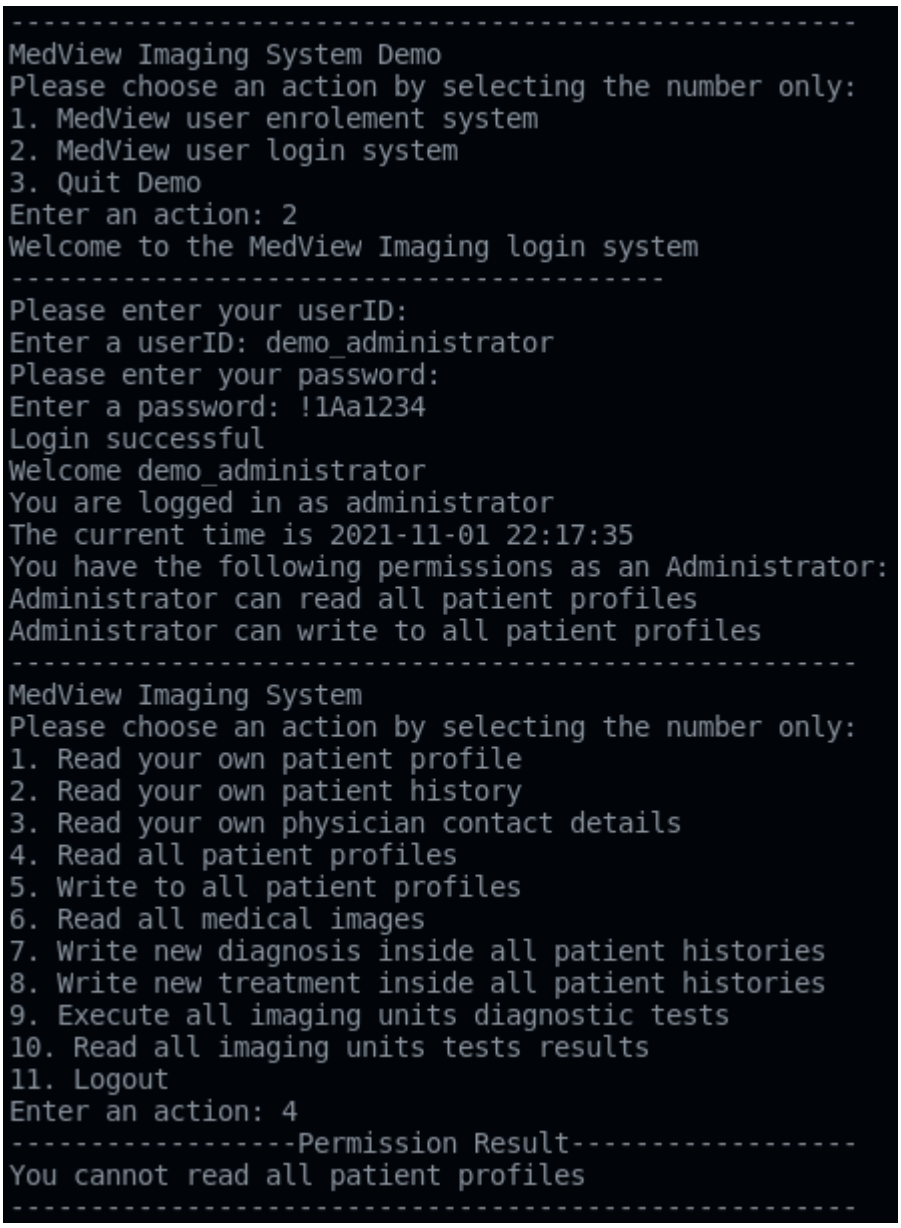
And if we login as the demo_administrator, if the current time is 10:00 AM, we are allowed to use the system as shown in the figure below, when we are trying to read all the patient profiles, we are allowed.

Figure 12. Administrator accessing the system at 10:00 AM



And if we login as the demo_administrator, if the current time is 10:17 PM, we are not allowed to use the system as shown in the figure below, when we are trying to read all the patient profiles, we are not allowed.

Figure 13. Administrator accessing the system at 10:17 PM



Weak Passwords Text File

Lastly, we have a weak_passwds.txt text file, it is used to store all the weak or compromised passwords. And it is saved as a text file in the same directory as the MedView access control and user authentication prototype file to allow it to be edited by MedView without needing to change the source code of the prototype.

The prototype will read the content of the weak_passwds.txt file each time the weak passwords checker is called and new passwords additions will be noted by the MedView access control and user authentication password checker function.

Figure 14. MedView Imaging user authentication and access control system prototype file directory

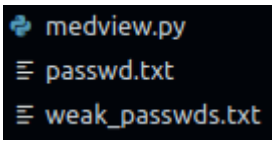


Figure 15. weak_passwords.txt text file for storing weak or compromised passwords