



SecureDrop

Noah Fay



Overview

User Registration

- If a user does not exist, the registration module will be presented to the user
- Credentials are required to register user

User Login

- Following successful registration, the login module is presented immediately

SecureDrop

- Upon Successful user login, the user is given a few options within the newly started shell
- The user can add contacts and list mutual online contacts as well.
- SecureDrop also allows a user to send a file securely to a mutual contact

User Registration

- By validating the presence of a private_key, SecureDrop will know recognize if new user registration is required
- User registration requires full name, email, and password
- Credentials for files are held on an encrypted user.json file for user confidentiality

```
12 def main():
13     # check if client have been registered
14     # using private key to check
15     if not file_exist(Helper.private_key_file) and not file_exist(Helper.public_key_file):
16         print("No users are registered with this client.")
17         # ask user to register
18         answer = input("Do you want to register a new user (y/n)? ")
19         if answer == "y" or answer == "Y":
20             # register a new user
21             register()
22         else:
23             # exit
24             sys.exit(0)
```




Private_key validation

Collecting and securely storing
user credentials →

```
27 def register():
28     # get username
29     username = input("Enter Full Name: ")
30     # call fuction to take email and validate
31     email = Helper.validate_email()
32     # get user's password
33     password = getpass.getpass("Enter password: ")
34     # get user's password again
35     password2 = getpass.getpass("Re-enter password: ")
36
37     # check if password match
38     if password != password2:
39         print("Password doesn't match")
40         # go back to register()
41         register()
42     else:
43         print("\nPassword match")
44         # hash the password
45         hashed_password = Helper.hash_data(password)
46         hashed_email = Helper.hash_data(email)
47         # save user's information to a file
48         # create a dictionary
49         user = {
50             "username": username.lower(),
51             "email": hashed_email,
52             "password": hashed_password
53         }
54         # save to a file
55         with open(Helper.user_file, "w") as f:
56             json.dump(user, f, indent=2)
```

User Login

- Logging in a previously registered user requires the email and password that was user for registration
- A successful login starts a shell and gives the user access to certain commands
- The command list can be shown by simply typing “help” 

```
65 def login():
66     try:
67         global User_email
68         #ask user for email, password
69         loginEmail = Helper.validate_email()
70         User_email = loginEmail
71         loginPass = getpass.getpass("Enter Password: ")
72         #opens user.json to compare values, not sure if better way to do this
73         Userinfo = json.load(open(Helper.user_file, "r"))
74         #checks login info against user.json info
75         if Helper.check_hashed_data(Userinfo["email"], loginEmail) and Helper.check_hashed_data(Userinfo["password"], loginPass):
76             print("\n\nAuthentication successful, Welcome to SecureDrop")
77             # decrypt the contact.json file so system can access
78             if file.exists(Helper.contact_file):
79                 Helper.decrypt_textfile(Helper.contact_file, Helper.contact_file, Helper.private_key_file)
80             startShell()
81         else:
82             print("\n\nAuthentication failed, try again")
83             login()
84     except:
```


```
Enter Email Address: noah_fay@student.uml.edu
Enter Password:

Authentication successful, Welcome to SecureDrop

Welcome to SecureDrop, type help for command list.
SecureDrop>> help

Documented commands (type help <topic>):
=====
add exit help list send

SecureDrop>> |
```

 Code for handling user authentication

Adding Contacts

- The “add” command allows users to add contacts to their own contact list
- Upon using the command, a name and email is required for each contact
- All contacts are held on an encrypted contact.json to ensure user confidentiality

```
121 # append contact.json everytime add new contact in
122 contact_list.append(contact)
123 if not file_exist(contact_file):
124     with open(contact_file, "w") as f:
125         # json.dump(list, f, indent=4)
126         json.dump(contact_list, f, indent=4)
127 else:
128     with open(contact_file, "r") as f:
129         data = json.load(f)
130         data.append(contact)
131
132 # remove duplicate contact
133 unique_entries = []
134
135 for entry in data:
136     if entry not in unique_entries:
137         unique_entries.append(entry)
138     else:
139         Duplicate = True
140
141 # write back to json file with non duplicate key
142 with open(contact_file, "w") as write_json:
143     # json.dump(unique_entries, write_json, indent=4)
144     json.dump(unique_entries, write_json)
145
146 # tell user that contact have been added
147 if Duplicate:
148     print("Duplicate contact, nothing added")
149 else:
150     print("Contact added")
```

```
Welcome to SecureDrop, type help for command list.
SecureDrop>> help
```

```
Documented commands (type help <topic>):
```

```
=====
```

```
add exit help list send
```

```
SecureDrop>> add
```

```
Enter Full Name: john doe
```

```
Enter Email Address: john_doe@student.uml.edu
```

```
Contact added
```

```
SecureDrop>> █
```

Listing Contacts

- Users can list mutual contacts that are online by using the “list” command
- By communicating through a TCP server, SecureDrop can identify active contacts and display them effectively

```
237 # Continuously accept connections and handle them until a timeout occurs
238 while True:
239     try:
240         Contactinfo = json.load(open(contact_file, "r"))
241         # establish a connection
242         # receiving email from other client
243         clientsocket,addr = serversocket.accept()
244         client_message = clientsocket.recv(1024)
245         client_msg_email_str = client_message.decode("utf-8")
246         # search client email within contact file
247         contact_email = search_email(client_msg_email_str, Contactinfo)
248         contact_name = search_user(client_msg_email_str, Contactinfo)
249         # if client email exist then send server email back
250         if contact_email == client_msg_email_str:
251             clientsocket.send(email.encode('utf-8'))
252             # receiving another message on handshake whether
253             # both client have added each other contact
254             client_handshake = clientsocket.recv(1024)
255             client_handshake_str = client_handshake.decode("utf-8")
256             if client_handshake_str == "yes":
257                 print("* " + contact_name + " <" + contact_email + ">")
258                 connect = True
259             else:
260                 clientsocket.send(msg.encode("utf-8"))
261                 clientsocket.close()
262         except socket.timeout:
263             break
264     serversocket.close()
265     return connect
```

SecureDrop>> list

The following contacts are online:

No contact online

SecureDrop>> █

Secure File Transfer

- The most crucial feature “send” allows users to send a file to mutual contacts that are online
- This secure file change only requires the user to enter the recipients email and the file name you would like to send

```
219 # File transfer receiving modules
220 if(incoming_msg_str == "sendrequest"):
221     contactemail = client_socket.recv(1024) #recieve Identity of sender and filename
222     contactemail_str = contactemail.decode('utf-8')
223     filename = client_socket.recv(1024)
224     filename_str = filename.decode('utf-8')
225     filesize = client_socket.recv(1024)
226     filesize_str = filesize.decode('utf-8')
227     print("\nContact <= > contactemail_str + > would like to send you <= > filename_str + >\n--> Accept? y/n: ", end=" ", flush = True) #Accept file
228     fileAccept = input()
229     if fileAccept == "y":
230         client_socket.send(fileAccept.encode('utf-8')) #send acceptance/refusal
231         with open(filename_str, 'wb') as file:
232             c = 0
233             while c <= int(filesize_str):
234                 data = client_socket.recv(1024)
235                 if not (data):
236                     break
237                 file.write(data)
238                 c += len(data)
239             print(filename_str + " successfully written.")
240     else:
241         msg = "n"
242         client_socket.send(msg.encode('utf-8')) #send acceptance/refusal
243     client_socket.close()
244
```

```
347 while True:
348     try:
349         # establish a connection
350         client_socket = serversocket.accept()
351         # wrap the socket in an ssl layer
352         clientsocket = context.wrap_socket(client_socket, server_side=True)
353
354         # receiving email from other client
355         client_message = clientsocket.recv(1024)
356         client_msg_email_str = client_message.decode('utf-8')
357
358         if(recipient_email == client_msg_email_str): #if email sent by client is recipient email
359             connect = True
360             clientsocket.send(sendmsg.encode('utf-8')) #send request to send a file
361             clientsocket.send(email.encode())
362             clientsocket.send(filename.encode()) #Additionally send identity and filename
363             clientsocket.send(str(file_size).encode())
364
365             ready_to_accept = clientsocket.recv(1024) #receive acceptance/refusal from client
366             ready_to_accept_str = ready_to_accept.decode('utf-8')
367
368             # read data and send data byte by byte
369             if ready_to_accept_str == "y":
370                 with open(filename, "rb") as file:
371                     c = 0
372                     while c <= file_size:
373                         data = file.read(1024)
374                         if not (data):
375                             break
376                         clientsocket.sendall(data)
377                         c += len(data)
378                     print("answer was yes, file has been sent")
379             else:
380                 print("Receiver refuse to accept file")
381             clientsocket.close()
382         except socket.timeout:
383             break
384
```