import json

import os

import hashlib

import re

import random

from cryptography.fernet import Fernet

import logging

import socket

# Configuration for logging

logging.basicConfig(filename='voting\_audit.log', level=logging.INFO)

# Key file path

KEY\_FILE = 'key.key'

# Data file path

DATA\_FILE = 'voting\_data.json'

def load\_key():

"""Load the encryption key from a file or generate a new one if it doesn't exist."""

try:

if not os.path.exists(KEY\_FILE):

key = Fernet.generate\_key()

with open(KEY\_FILE, 'wb') as key\_file:

key\_file.write(key)

else:

with open(KEY\_FILE, 'rb') as key\_file:

key = key\_file.read()

return key

except Exception as e:

print(f"Error loading/generating key: {e}")

return None

# Load the encryption key

key = load\_key()

if key:

cipher\_suite = Fernet(key)

else:

raise Exception("Encryption key could not be loaded or generated.")

def hash\_password(password):

"""Hash the password using SHA-256."""

return hashlib.sha256(password.encode()).hexdigest()

def encrypt\_data(data):

"""Encrypt JSON data."""

try:

json\_data = json.dumps(data).encode() # Convert data to JSON and then to bytes

return cipher\_suite.encrypt(json\_data) # Encrypt the JSON bytes

except Exception as e:

print(f"Error encrypting data: {e}")

return None

def decrypt\_data(encrypted\_data):

"""Decrypt data and return it as a Python dictionary."""

try:

decrypted\_data = cipher\_suite.decrypt(encrypted\_data) # Decrypt the data

return json.loads(decrypted\_data.decode()) # Decode bytes to string, then parse JSON

except Exception as e:

print(f"Error decrypting data: {e}")

return None

def save\_data(data):

"""Save encrypted data to file."""

encrypted\_data = encrypt\_data(data) # Encrypt the voting data

if encrypted\_data is None:

print("Failed to encrypt data. Data will not be saved.")

return

try:

with open(DATA\_FILE, 'wb') as f: # Write in binary mode

f.write(encrypted\_data)

except Exception as e:

print(f"Error saving data to file: {e}")

def load\_data():

"""Load encrypted data from file."""

if not os.path.exists(DATA\_FILE):

return {'users': {}, 'candidates': {'cand 1': 0, 'cand 2': 0, 'Cand 3': 0}}

try:

with open(DATA\_FILE, 'rb') as f: # Read in binary mode

encrypted\_data = f.read()

return decrypt\_data(encrypted\_data)

except Exception as e:

print(f"Error loading data from file: {e}")

return {'users': {}, 'candidates': {'cand 1': 0, 'cand 2': 0, 'Cand 3': 0}}

def is\_valid\_username(username):

"""Validate usernames (alphanumeric and underscores allowed)."""

return re.match("^[a-zA-Z0-9\_]+$", username)

def is\_valid\_candidate(candidate):

"""Validate candidate names (alphanumeric, spaces, and underscores allowed)."""

return re.match("^[a-zA-Z0-9\_ ]+$", candidate)

def generate\_otp():

"""Generate a 6-digit OTP for two-factor authentication."""

return str(random.randint(100000, 999999))

def register\_user(data):

"""Register a new user with a hashed password."""

username = input("Enter your username: ")

if not is\_valid\_username(username):

print("Invalid username. Only alphanumeric characters and underscores are allowed.")

return

if username not in data['users']:

password = input("Enter your password: ")

hashed\_password = hash\_password(password)

data['users'][username] = {'password': hashed\_password, 'voted': False, 'attempts': 0}

save\_data(data)

logging.info(f"User '{username}' registered.")

print(f"User '{username}' registered successfully!")

else:

logging.warning(f"Attempt to register an existing username '{username}'.")

print("Username already exists.")

def login\_user(data):

"""Login a user by verifying their password and implementing 2FA."""

username = input("Enter your username: ")

if username in data['users']:

if data['users'][username]['attempts'] >= 3:

logging.warning(f"Account locked for username '{username}'.")

print("Account locked due to too many failed login attempts.")

return None

password = input("Enter your password: ")

hashed\_password = hash\_password(password)

if data['users'][username]['password'] == hashed\_password:

otp = generate\_otp()

print(f"OTP sent to your email/phone (Simulated): {otp}") # Simulating OTP sent

entered\_otp = input("Enter the OTP: ")

if entered\_otp == otp:

data['users'][username]['attempts'] = 0 # Reset attempts after successful login

save\_data(data)

logging.info(f"User '{username}' logged in successfully with IP: {get\_ip\_address()}")

return username

else:

print("Incorrect OTP.")

return None

else:

data['users'][username]['attempts'] += 1

save\_data(data)

logging.warning(f"Failed login attempt for '{username}'. {3 - data['users'][username]['attempts']} attempts left.")

print(f"Incorrect password. {3 - data['users'][username]['attempts']} attempts left.")

return None

else:

logging.warning(f"Login attempt with non-existing username '{username}'.")

print("Username not found.")

return None

def vote(data, username):

"""Allow a user to cast their vote for a valid candidate."""

if data['users'][username]['voted']:

print("You have already voted.")

return

print("Candidates:")

for candidate in data['candidates']:

print(candidate)

candidate = input("Enter the name of the candidate you want to vote for: ")

if not is\_valid\_candidate(candidate):

print("Invalid candidate name.")

return

if candidate in data['candidates']:

data['candidates'][candidate] += 1

data['users'][username]['voted'] = True

save\_data(data)

logging.info(f"User '{username}' voted for '{candidate}'. IP: {get\_ip\_address()}")

print(f"Vote cast for {candidate}.")

else:

print("Invalid candidate.")

def view\_results(data):

"""Display the current voting results."""

print("Voting Results:")

for candidate, votes in data['candidates'].items():

print(f"{candidate}: {votes} votes")

def get\_ip\_address():

"""Get the IP address of the user's machine (for logging)."""

hostname = socket.gethostname()

return socket.gethostbyname(hostname)

def main():

data = load\_data() # Load the data, either new or from the file

while True:

print("\n--- Command-Line Voting System ---")

print("1. Register")

print("2. Login and Vote")

print("3. View Results")

print("4. Exit")

choice = input("Choose an option: ")

if choice == '1':

register\_user(data)

elif choice == '2':

username = login\_user(data)

if username:

vote(data, username)

elif choice == '3':

view\_results(data)

elif choice == '4':

break

else:

print("Invalid choice. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()