Week 2 Progress Report on password Manager

UpSkill Campus, UniConverge Technologies Pvt.Ltd.

Submitted by: Bindala Kavya

# Correspondence Address:

Bindala Kavya

Madanapalle Institute of Technology and Science,

Department of Computer Science and Engineering,

Angallu, Andhra Pradesh, 517326,

India.

Mail: [bindalakavyareddy06@gmail.com](mailto:bindalakavyareddy06@gmail.com)

Phone: +91 7416206921

**Overview**

During the second week,I took my step towards the source code of my project work,I just completed one by third part of my project as it came so succesfull it really motivates me.

Tasks Completed

As the implementation have three stages

* Configure
* Add new entries
* Get entries

Configure:

* Master password is first inputted while configuring,and the hash of it is saved in a file
* Device password is generated randomly,also stored in a file
* Master password+Device password ispassed into a hashing function (pbkdf)to create a valid key for AES-256,this is called Master key
* The Master key s then used to encrypt/decrypt new entreis
* Encrypted Fields : email,username,password
* Plain Fields : sitename,url

The configure task has been completed

Challenges Faced

* This week has been really tough as I started the coding.there are so many error when I am doing the coding ,but the errors really motivated me
* This week time management has been really tough so that I scheduled time for the further weeks too

Lessons Learned

Overall, the lessons learned from the "Password Manger" project highlighted the importance of data security and it is a real-world application. These insights will guide us in future projects and contribute to our professional growth.

Code progress

|  |
| --- |
| import os |
|  | import sys |
|  | import string |
|  | import random |
|  | import hashlib |
|  | import sys |
|  | from getpass import getpass |
|  |  |
|  | from utils.dbconfig import dbconfig |
|  |  |
|  | from rich import print as printc |
|  | from rich.console import Console |
|  |  |
|  | console = Console() |
|  |  |
|  | def checkConfig(): |
|  | db = dbconfig() |
|  | cursor = db.cursor() |
|  | query = "SELECT SCHEMA\_NAME FROM INFORMATION\_SCHEMA.SCHEMATA WHERE SCHEMA\_NAME = 'pm'" |
|  | cursor.execute(query) |
|  | results = cursor.fetchall() |
|  | db.close() |
|  | if len(results)!=0: |
|  | return True |
|  | return False |
|  |  |
|  |  |
|  | def generateDeviceSecret(length=10): |
|  | return ''.join(random.choices(string.ascii\_uppercase + string.digits, k = length)) |
|  |  |
|  |  |
|  | def make(): |
|  | if checkConfig(): |
|  | printc("[red][!] Already Configured! [/red]") |
|  | return |
|  |  |
|  | printc("[green][+] Creating new config [/green]") |
|  |  |
|  | # Create database |
|  | db = dbconfig() |
|  | cursor = db.cursor() |
|  | try: |
|  | cursor.execute("CREATE DATABASE pm") |
|  | except Exception as e: |
|  | printc("[red][!] An error occurred while trying to create db. Check if database with name 'pm' already exists - if it does, delete it and try again.") |
|  | console.print\_exception(show\_locals=True) |
|  | sys.exit(0) |
|  |  |
|  | printc("[green][+][/green] Database 'pm' created") |
|  |  |
|  | # Create tables |
|  | query = "CREATE TABLE pm.secrets (masterkey\_hash TEXT NOT NULL, device\_secret TEXT NOT NULL)" |
|  | res = cursor.execute(query) |
|  | printc("[green][+][/green] Table 'secrets' created ") |
|  |  |
|  | query = "CREATE TABLE pm.entries (sitename TEXT NOT NULL, siteurl TEXT NOT NULL, email TEXT, username TEXT, password TEXT NOT NULL)" |
|  | res = cursor.execute(query) |
|  | printc("[green][+][/green] Table 'entries' created ") |
|  |  |
|  |  |
|  | mp = "" |
|  | printc("[green][+] A [bold]MASTER PASSWORD[/bold] is the only password you will need to remember in-order to access all your other passwords. Choosing a strong [bold]MASTER PASSWORD[/bold] is essential because all your other passwords will be [bold]encrypted[/bold] with a key that is derived from your [bold]MASTER PASSWORD[/bold]. Therefore, please choose a strong one that has upper and lower case characters, numbers and also special characters. Remember your [bold]MASTER PASSWORD[/bold] because it won't be stored anywhere by this program, and you also cannot change it once chosen. [/green]\n") |
|  |  |
|  | while 1: |
|  | mp = getpass("Choose a MASTER PASSWORD: ") |
|  | if mp == getpass("Re-type: ") and mp!="": |
|  | break |
|  | printc("[yellow][-] Please try again.[/yellow]") |
|  |  |
|  | # Hash the MASTER PASSWORD |
|  | hashed\_mp = hashlib.sha256(mp.encode()).hexdigest() |
|  | printc("[green][+][/green] Generated hash of MASTER PASSWORD") |
|  |  |
|  |  |
|  | # Generate a device secret |
|  | ds = generateDeviceSecret() |
|  | printc("[green][+][/green] Device Secret generated") |
|  |  |
|  | # Add them to db |
|  | query = "INSERT INTO pm.secrets (masterkey\_hash, device\_secret) values (%s, %s)" |
|  | val = (hashed\_mp, ds) |
|  | cursor.execute(query, val) |
|  | db.commit() |
|  |  |
|  | printc("[green][+][/green] Added to the database") |
|  |  |
|  | printc("[green][+] Configuration done![/green]") |
|  |  |
|  | db.close() |
|  |  |
|  |  |
|  | def delete(): |
|  | printc("[red][-] Deleting a config clears the device secret and all your entries from the database. This means you will loose access to all your passwords that you have added into the password manager until now. Only do this if you truly want to 'destroy' all your entries. This action cannot be undone. [/red]") |
|  |  |
|  | while 1: |
|  | op = input("So are you sure you want to continue? (y/N): ") |
|  | if op.upper() == "Y": |
|  | break |
|  | if op.upper() == "N" or op.upper == "": |
|  | sys.exit(0) |
|  | else: |
|  | continue |
|  |  |
|  | printc("[green][-][/green] Deleting config") |
|  |  |
|  |  |
|  | if not checkConfig(): |
|  | printc("[yellow][-][/yellow] No configuration exists to delete!") |
|  | return |
|  |  |
|  | db = dbconfig() |
|  | cursor = db.cursor() |
|  | query = "DROP DATABASE pm" |
|  | cursor.execute(query) |
|  | db.commit() |
|  | db.close() |
|  | printc("[green][+] Config deleted![/green]") |
|  |  |
|  | def remake(): |
|  | printc("[green][+][/green] Remaking config") |
|  | delete() |
|  | make() |
|  |  |
|  |  |
|  | if \_\_name\_\_ == "\_\_main\_\_": |
|  |  |
|  | if len(sys.argv)!=2: |
|  | print("Usage: python config.py <make/delete/remake>") |
|  | sys.exit(0) |
|  |  |
|  | if sys.argv[1] == "make": |
|  | make() |
|  | elif sys.argv[1] == "delete": |
|  | delete() |
|  | elif sys.argv[1] == "remake": |
|  | remake() |
|  | else: |
|  | print("Usage: python config.py <make/delete/remake>") |