Semester Project Milestone

MSCS 630L-711 - Spring 2022

Jarett Sutula

Jarett.Sutula1@Marist.edu

Due on April 17th, 2022.

1 Abstract

While user password security has increased over time, users now maintain larger and larger amounts of online accounts than ever before. In order to ensure the compromise of one account does not include others, user password protocols typically advocate for the use of distinctly different passwords for different accounts. While this helps isolate credentials from each other, it also makes credentials harder for users to remember. Password managers exist to bridge this gap by storing a user's credentials for them in a secure and encrypted way where they can access them from any device while still maintaining strong, secure passwords without forgetting them.

2 Introduction

With an ever-growing presence of data on the internet, it is important for one to protect their identity and ensure the security of their information. With virtually every online service requiring an account, credentials have become the target of hackers and data vendors alike. Password managers were built to help users safely store and remember their important credentials without having to store them physically offline. This convenience puts our credentials in the hands of third-party companies and many have already been attacked and had their data leaked into the hands of others. Password manager leaks are not uncommon, and there are many instances where passwords are being saved in databases with outdated encryption methods or simply in plaintext by large companies.

This paper serves as a further look into creating a secure password manager with various levels of encryption that can prove to be trustworthy to users on the web or offline. This includes encryption of credentials before they are placed into a database as well as additional hashing and salting of the encrypted credentials after they are placed into the database to ensure the greatest security.

3 Background

A further look into password managers shows many different approaches for credential encryption. Previously, password managers used algorithms like MD5 and SHA1 for encryption but these algorithms have not withstood the test of time. MD5 is no longer safe, SHA1 has had collisions, and both were deprecated by the large tech giant Google and many companies followed suit. AES is still a government standard for encryption and therefore enjoys usage by password managers from both browser-based and standalone apps, although they typically are using at least 128-bit encryption. This could surely be expanded to 192+ or even a consistent 256-bit encryption to improve quality. Software companies also have taken Google's recommendations and many use SHA-256 and the newer SHA-3 hashing algorithms to further secure their credentials.

Using a combination of high-bit AES encryption (256-bits if project time allows it) with a combination of the popular python hashing and salting library bcrypt should prove a strong encryption process that guarantees the data's security not only when the user enters them, but also as they stored in the database with an extra layer of hashing/salting. This means, unlike some of the most infamous data breaches, credentials will not be sitting in low-protection databases without that extra layer of security. This app will also offer offline support for those who do not want their data being stored anywhere online but will come at the added cost of not being able to access their passwords from every device and will prove to be vulnerable to device destruction or loss of physical access to the offline app.

4 Progress

So far, I have researched and decided on a data encryption plan. I will be using python as a programming language for both the offline and online applications, which will make use of the familiar AES encryption algorithm we have created in Java. This has been mostly translated into python already and only needs to jump from Java-based MixColumns operations into python to be finished. I've also researched and tested bcrypt, a python library for hashing and salting credentials to get them out of plaintext in use of the app and database. Together, I am confident that encryption will be strong and safe - even if someone were to gain access to the database, they should only find hashed and salted credentials that would be impossible

to break. I've also decided on using MongoDB, specifically their online variant Atlas, to store the encrypted credentials. Mongo has become an industry standard in software for NoSQL databases, and further usage of Atlas has proven to give me quick, easy access and sight of the database's materials to ensure they are being encrypted and hashed behind-the-scenes from the user. Using the python library pymongo, I've connected my local code to the mongo database and tested how to push code to the Atlas cluster.

From here, the rest of AES encryption will be wrapped up and work will begin on creating an interface with Django for users to access the online version of their password manager. Other work will include further research on offline-based user interfaces in python as that is not something I am familiar with. I think with the added security of hashing and salting in databases, along with the added luxury of obtaining passwords from any device you can log into, the online variant of my app will be the larger focus for me at the time being, as opposed to my original mindset. Talking to fellow software developers has proven that the majority use multiple computers as well as smartphones, and having access to user accounts across all of our devices is a convenience that many want. If time becomes an issue, the offline variant will be an extra piece, as I do not believe it applies to a large group of users nor does it seem to be an available option for most third-party offers in my research. If time allows and the rest of the interface and database connection goes smoothly, I would like to fit in an offline version for those who perhaps do not want their information stored in a database.