**Capstone Report**

Sensitive Data Storage

Garrett McDonnell

The inspiration for my project is that despite exhaustive measures taken to keep data safe, major companies still continue to get hacked. Although very elaborate hashing algorithms are used, there is still always a possibility for thousands of users to lose their sensitive data all at once. Also, thanks to resources like "rainbow tables", easier passwords in plain text have already been matched to their hashes, and can be stolen with ease. I have thought for a long while that, perhaps, this problem could be entirely avoided.

I tried to approach this problem in such a way that it could conceivably be used in the real world. Originally, I was trying to do this from a web application, as if, let's say, you were trying to log into something like Chase Bank from your browser. So, I tried simply writing some C++ code and compiling into an executable, making it downloadable from the web application, and run it when the user wants to create an account or login. The problem is that I needed some dll's in order to run the executable, which means I would have to have zipped up the files and have the user download them all together. It became painfully obvious that this would never be a realistic solution for the end user.

What I alternatively did was write a C# Windows Form to simulate using a desktop app such as, again, Chase Bank. Ultimately, because of the problem of allowing an operating system to collaborate with a web app, my idea could only really be implemented in real life by allowing the user to use a desktop/mobile app, as opposed to a website. In addition, I found a free package installer online, called Inno Setup, which allows me to wrap my C# executable and associated dll's in a single executable, which will install my program on your computer using an installer wizard. This was not really necessary, but was done for fun to simulate what downloading a real software app on a desktop or mobile device would be like. I did everything I could to make things feel like they could be used in real life, as I would love to hear feedback from the teacher and classmates when the time comes as to whether or not the idea is useful.

When you run the program, all you have is a simple Windows Form with 2 buttons, a Create Account button and Login button. I chose not to add features and make it look like a real program, because I felt that it would distract from what the program is supposed to convey. It is merely an alternate method of information storage, when you think about it. So, to explain how it works, it is pretty simple, but a lot is happening behind the scenes. First, when you create an account, after you provide a username and password, the username is stored in a database. The password, however, is not. A password protected .zip folder is created locally on your computer in your program files where my program is stored. The password to open the .zip folder is stored in my database along with your username (instead of the password you chose). Only my database has access to that .zip folder password, nobody else, not even the owner of that account. Inside the .zip folder sits a simple .txt file containing your password in plaintext. To reemphasize, nobody has access to open that .txt file containing your password, because you need the .zip password to open any of the files in the .zip folder, and only the database administrators have access to your .zip password. So in summation, when you create an account, my system generates a .zip folder with a randomly generated password wherever my program is installed on your computer. Your chosen password is stored in the .txt file in that .zip folder, and the .zip password and your username are stored in my database. Now, let's talk about the Login function. When you enter your username and password, the database looks for your username and pulls out the .zip password associated with it. My program searches for the .zip folder in the current directory, tries to open the password protected .txt file using the .zip password, and pulls out your chosen account password. Your input password during login is compared against the password that is pulled out (assuming all steps are successful), and if they match, you are successfully logged in. If any of these steps fail, an error message for unknown username, incorrect password, or no .zip folder/.txt file found on your computer is displayed, depending on which step of the process fails.

Now that the program has been thoroughly explained, I'd like to address some of the issues you may see. I did everything in plaintext for representation purposes, but there is nothing stopping me from hashing passwords (both .zip and regular passwords). There is probably a more secure method of storing information on your computer, as opposed to a .zip folder and .txt file, but the principal is still there. If I were to do things differently, and work on this project further using my own free time, I would try to look for a more sophisticated way of doing what I am trying to do in the most secure way possible. However, the experience was useful, as I forced myself to learn new technology like C# Windows Forms and Inno Setup. Also, I love that I was able to challenge the status quo to try and see if I could come up with an alternative software solution for the security problem that we face in real life, and I will feel that my solution holds up as a real possibility to be a viable security option.