ICS 491 Assignment 5: Release

Authors: Brandon Arakaki, Justin Ho, Elliot Ito

Date: August 5, 2016

Abstract:

1.0 Establish Security Requirements

Due to the nature of the data that a password manager deals with, this application must have strict security requirements. To minimize the attack surface, the application is required to be a standalone application. This will ensure that the application only communicates with the user and nothing more. Applications that communicate with other processes are more vulnerable to the possibility of a man-in-the-middle attack. The password manager must utilize a secure non volatile datastore. The datastore must be kept secure so that if the datastore is stolen the data will still be secure. The application is required to use strong up to date (with present security standards) encryption on the datastore. To unlock the password manager the user must enter a password, this password will be required to meet standards of complexity to ensure that a password won't be cracked easily to attain the users passwords contained in the datastore.

To ensure users the most amount of privacy the application will be required to only allow a single specific user to use the application on a user account. This will ensure that the users data is kept private to only that specific user. As stated earlier our application will be required to use strong up to date encryption on the datastore. This will help secure the privacy of a user's data.

To track security flaws during the development process we will be using the built in GitHub issue tracker. For critical security related issues we will be using custom labels in the issue tracker, this will allow us to understand the importance and allot our time and attention to critical security flaws before other issues or feature requests.

1.1 Create Quality Gates/Bug Bars

- Critical
    - Privilege Escalation
    - Information Disclosure
        - Weak access control for a resource
    - Spoofing the password manager process
    - Data flow sniffing
- Important
    - Spoofing
        - Human user external entity
    - Denial Of Service
        - Potential process crash or stop
        - Data flow interruption
        - Encrypted Database corruption
    - Input Validation
- Moderate
    - Spoofing of Encrypted Database
    - Denial of Service

        - Encrypted Database Inaccessible
- Low
    - Potential Data Repudiation

1.2 Perform Security and Privacy Risk Assessments

To determine which parts of the program would need security reviews and threat modeling access, creation of PII and the risk of PII being leaked are used as the baseline. This would ensure that the program would have proper assessments to prevent data loss of PII. In addition to preventing data loss, the assessments would also ensure that the security measures that are taken would prevent any unauthorized access to the PII. At the point of the “creation” of the PII, the assessments would ensure that other PII cannot be accessed or leaked.

2.0 Establish Design Requirements

The design of our password manager is going to be a terminal based design. The user will open our application and it will ask the user to verify themselves. From there our system will either display or send the password they want to retrieve from our application. When the user wants to store their data, they will be asked to verify themselves. From there they can send their password through our application and it will be stored and encrypted in our system.

The application is going to be a standalone application. This ensures that the user is interacting with our application and not something else. When the user sends us their password to save, we will encrypt it to ensure that if something were to happen and someone got ahold of our data that they would not get anything out of it.

For privacy, only the programmers, and the user will have access to the user's information. Only a single user can can retrieve password and only the programmers will have access to the encrypted files. We will create a code that will take the user's password and encrypt it into a file where it is then stored. When the user requests the password, the program will decrypt the file and give the password to the user.

Stated earlier in the paper, 1.0 establish security requirements, we will be using github to track security flaws in our application. This will ensure that as we program, we are building a stronger security system.

2.1 Perform Attack Surface Analysis/Reduction

The only way a user is allowed to get the information they want is through verification. Only the programmers, our team, are the only ones that will be able to see what is going on inside the application. This includes receiving of data, encryption, and sending data. Some vulnerabilities with password manager applications that other companies faced are:

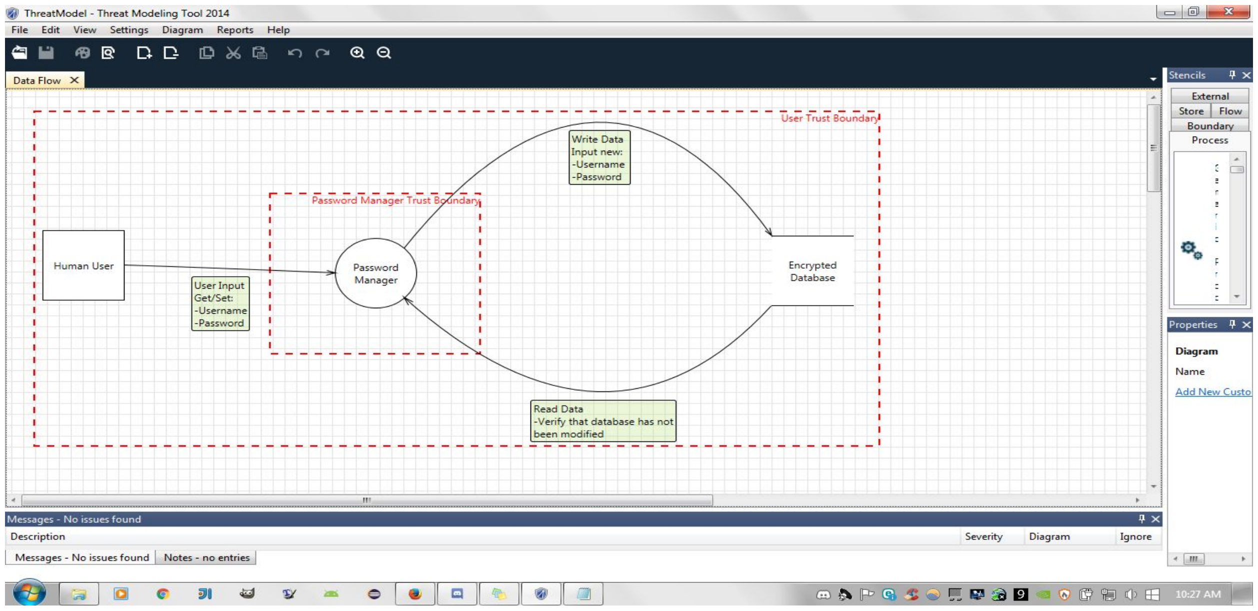
- Open sockets

- Fake software updates
- Empty string password
- Incorrect encryption

These are some attack surface analysis vulnerabilities found in password managers. With these in mind, precautions will be taken to avoid such situations and make the application more secure.

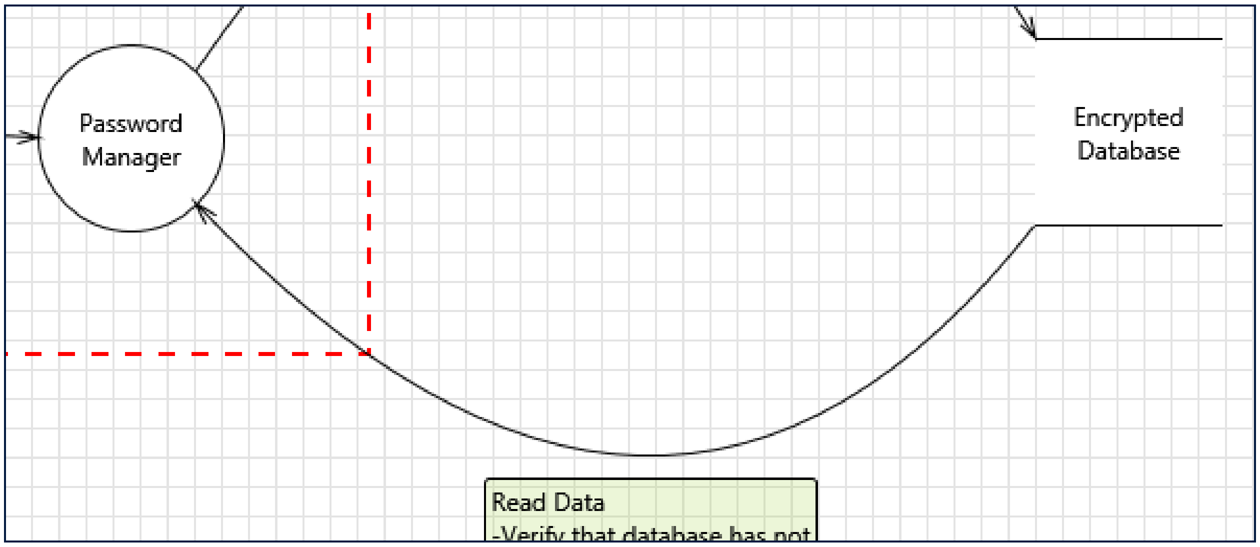
2.2 Use Threat Modeling

The following is the threat model for the program:


ThreatModel - Threat Modeling Tool 2014
File Edit View Settings Diagram Reports Help
Data Flow
User Trust Boundary
Password Manager Trust Boundary
Human User
User Input Get/Set: -Username -Password
Password Manager
Write Data Input new: -Username -Password
Encrypted Database
Read Data -Verify that database has not been modified
Stencils
External
Store
Flow
Boundary
Process
Properties
Diagram
Name
Add New Custo
Messages - No issues found
Description
Severity
Diagram
Ignore
Notes - no entries
10:27 AM

The threats for the program are as follows. The threats described are modified from the report generated from the SDL threat modeling tool. Each threat is from the data flows and individual threats presented by each entity in the flow diagram:

Interaction: Read Data - Verify that database has not been modified



1. Spoofing the Password Manager Process [Category: Spoofing]

Category:	Spoofing is when a process or entity is something other than its claimed identity. Examples include substituting a process, a file, website or a network address.
Description:	Password Manager may be spoofed by an attacker and this may lead to information disclosure by Encrypted Database.

2. Spoofing of Encrypted Database [Category: Spoofing]

Category:	Spoofing is when a process or entity is something other than its claimed identity. Examples include substituting a process, a file, website or a network address.

Description:	Encrypted Database may be spoofed by an attacker and this may lead to incorrect data delivered to Password Manager.

3. Potential Data Repudiation by Password Manager [Category: Repudiation]

Category:	Repudiation threats involve an adversary denying that something happened.
Description:	Password Manager claims that it did not receive data from a source outside the trust boundary.

4. Weak Access Control for a Resource [Category: Information Disclosure]

Category:	Information disclosure happens when the information can be read by an unauthorized party.
Description:	Improper data protection of Encrypted Database can allow an attacker to read information not intended for disclosure.

5. Potential Process Crash or Stop for Password Manager [Category: Denial of Service]

Category:	Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.
Description:	Password Manager crashes, halts, stops or runs slowly; in all cases violating an availability metric.

6. Data Flow Is Potentially Interrupted [Category: Denial of Service]

Category:	Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.
Description:	An external agent interrupts data flowing across a trust boundary in either direction.

7. Encrypted Database Inaccessible [Category: Denial of Service]

Category:	Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.
Description:	An external agent prevents access to the encrypted database on the other side of the trust boundary.

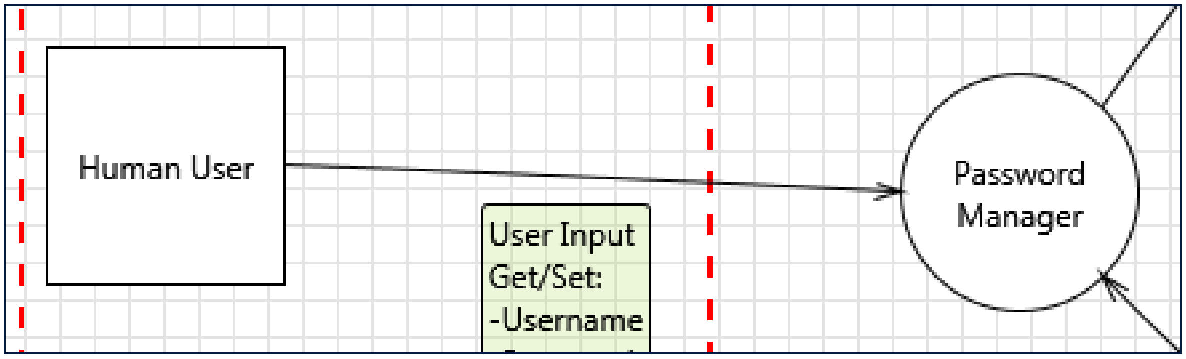
8. Password Manager May be Subject to Elevation of Privilege Using Remote Code Execution [Category: Elevation Of Privilege]

Category:	A user subject gains increased capability or privilege by taking advantage of an implementation bug.
Description:	Encrypted Database may be able to remotely execute code for Password Manager.

9. Elevation by Changing the Execution Flow in Password Manager [Category: Elevation Of Privilege]

Category:	A user subject gains increased capability or privilege by taking advantage of an implementation bug.
Description:	An attacker may pass data into Password Manager in order to change the flow of program execution within Password Manager to the attacker's choosing.

Interaction: User Input (Get/Set: Username and Password)



10. Spoofing the Password Manager Process [Category: Spoofing]

Category:	Spoofing is when a process or entity is something other than its claimed identity. Examples include substituting a process, a file, website or a network address.
Description:	Password Manager may be spoofed by an attacker and this may lead to information disclosure by Human User.

11. Spoofing the Human User External Entity [Category: Spoofing]

Category:	Spoofing is when a process or entity is something other than its claimed identity. Examples include substituting a process, a file, website or a network address.
Description:	Human User may be spoofed by an attacker and this may lead to unauthorized access to Password Manager.

12. Potential Lack of Input Validation for Password Manager [Category: Tampering]

Category:	Tampering is the act of altering the bits. Tampering with a process involves changing bits in the running process. Similarly, Tampering with a data flow involves changing bits on the wire or between two running processes.

Description:	Data flowing across User Input (Get/Set: Username and Password) may be tampered with by an attacker. This may lead to a denial of service attack against Password Manager or an elevation of privilege attack against Password Manager or an information disclosure by Password Manager. Failure to verify that input is as expected is a root cause of a very large number of exploitable issues.

13. Potential Data Repudiation by Password Manager [Category: Repudiation]

Category:	Repudiation threats involve an adversary denying that something happened.
Description:	Password Manager claims that it did not receive data from a source outside the trust boundary.

14. Data Flow Sniffing [Category: Information Disclosure]

Category:	Information disclosure happens when the information can be read by an unauthorized party.
Description:	Data flowing across User Input (Get/Set: Username and Password) may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations.

15. Potential Process Crash or Stop for Password Manager [Category: Denial of Service]

Category:	Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.
Description:	Password Manager crashes, halts, stops or runs slowly; in all cases violating an availability metric.

16. Data Flow Generic Data Flow Is Potentially Interrupted [Category: Denial of Service]

Category:	Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.

Description:	An external agent interrupts data flowing across a trust boundary in either direction.

17. Elevation Using Impersonation [Category: Elevation Of Privilege]

Category:	A user subject gains increased capability or privilege by taking advantage of an implementation bug.
Description:	Password Manager may be able to impersonate the context of Human User in order to gain additional privilege.

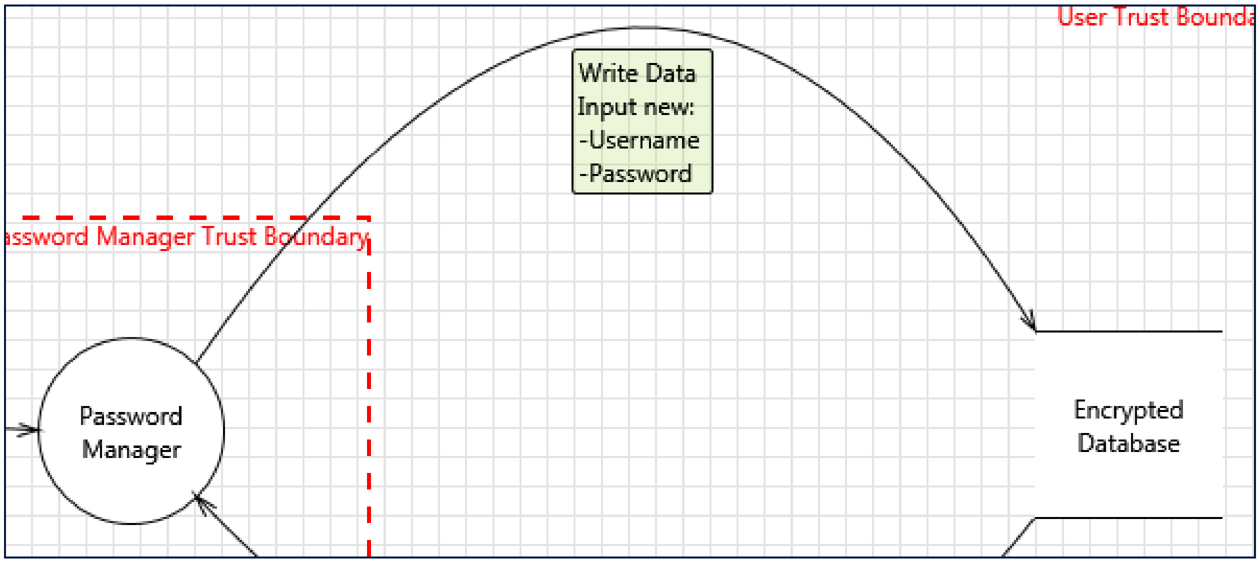
18. Password Manager May be Subject to Elevation of Privilege Using Remote Code Execution [Category: Elevation Of Privilege]

Category:	A user subject gains increased capability or privilege by taking advantage of an implementation bug.
Description:	Human User may be able to remotely execute code for Password Manager.

19. Elevation by Changing the Execution Flow in Password Manager [Category: Elevation Of Privilege]

Category:	A user subject gains increased capability or privilege by taking advantage of an implementation bug.
Description:	An attacker may pass data into Password Manager in order to change the flow of program execution within Password Manager to the attacker's choosing.

Interaction: Write Data (Input new: Username and Password)



20. Spoofing the Password Manager Process [Category: Spoofing]

Category:	Spoofing is when a process or entity is something other than its claimed identity. Examples include substituting a process, a file, website or a network address.
Description:	Password Manager may be spoofed by an attacker and this may lead to unauthorized access to Encrypted Database.

21. Spoofing of Destination Encrypted Database [Category: Spoofing]

Category:	Spoofing is when a process or entity is something other than its claimed identity. Examples include substituting a process, a file, website or a network address.

Description:	Encrypted Database may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of Encrypted Database.

22. The Encrypted Database Could Be Corrupted [Category: Denial of Service]

Category:	Tampering is the act of altering the bits. Tampering with a process involves changing bits in the running process. Similarly, Tampering with a data flow involves changing bits on the wire or between two running processes.
Description:	Data flowing across Write Data (Input new: Username and Password) may be tampered with by an attacker. This may lead to corruption of Encrypted Database.

23. Denying Encrypted Database From Potentially Writing Data [Category: Repudiation]

Category:	Repudiation threats involve an adversary denying that something happened.
Description:	Encrypted Database claims that it did not write data received from an entity on the other side of the trust boundary.

24. Data Flow Sniffing [Category: Information Disclosure]

Category:	Information disclosure happens when the information can be read by an unauthorized party.
Description:	Data flowing across Write Data (Input new: Username and Password) may be sniffed by an attacker. Depending on what type of data an attacker can read, it may be used to attack other parts of the system or simply be a disclosure of information leading to compliance violations.

25. Potential Excessive Resource Consumption for Password Manager or Encrypted Database [Category: Denial of Service]

Category:	Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.
Description:	An adversary uses the Password Manager or Encrypted Database to use resources. Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job.

26. Data Flow Generic Data Flow Is Potentially Interrupted [Category: Denial of Service]

Category:	Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.
Description:	An external agent interrupts data flowing across a trust boundary in either direction.

27. Data Store Inaccessible [Category: Denial of Service]

Category:	Denial of Service happens when the process or a datastore is not able to service incoming requests or perform up to spec.
Description:	An external agent prevents access to a data store on the other side of the trust boundary.

3.0 Approved Tools

Tool	Minimum Required Version	Recommended Version	Comments
Python Interpreter	2.7.0	2.7.12	
Git	2.1.4	2.7.4	
PyCrypto	2.6.1	2.6.1	
PyCharm	2016.1.4	2016.1.4	

4.0 Deprecate Unsafe Functions

File io

Character and string manipulation

Packages

5.0 Perform Static Analysis

Pycharm is an IDE (integrated development environment) used for programming in python. It includes a style guide for coding in python and has version control software integration.

5.1 Brandon's Experience with Pycharm

I have little knowledge on python. University of Hawaii at Hilo only teaches a little about the basics of python, but they don't teach us in depth in the programming language. This is my first hands on experience with python, and it was easier than I thought. It was tricky at first because the orientation of the code matters to how the code runs, so a majority of my errors were because I forgot to tab a line of my code. Another tricky thing that I found out was that unlike C++, you don't need to use brackets or semicolons. Python is unique in that it uses colons and indentions to identify which part of code runs in which functions. Overall, my experience with Pycharm was good. It was a good learning experience and Pycharm is a good IDE to use because it is easy to use.

5.2 Elliot’s Experience with Pycharm

While I have worked on python before for other activities and school assignments, this was my first school project with python. I had not taken any course for learning python, so I found it nice to have the PyCharm’s integrated python styling. I had used PyCharm before, but I did not use version control software alongside the PyCharm project. With my previous usage of PyCharm, I was used to the UI coloring for the styling and errors. The color choices presented a visible method of knowing what changes were needed with the code (if any). The on the fly code checking with the color choices allows for quick changes and notifications. This is a feature that I utilize often and depend on to ensure that my code is presentable in a manner that is easy to read, but also follow the styling guidelines. An issue that I ran into PyCharm was that it included extra information when pushing to GitHub (with the git integration). The extra information included the PyCharm version as well as the localhost information of the computer. I found this out during one of the group meetings and saw it after doing a push to the repository. As a result, I ended up having to use the git bash terminal to do the pushing.

5.3 Justin’s Experience with Pycharm

This was my first python project for a school assignment. I’ve used python for other small python scripts before, but this was the first python based project that I’ve had. I had no formal teaching of python so a lot of what I knew was based on me taking what I knew from other programming languages and using that with python syntax and also teaching myself using tutorials online. I’ve never really managed a project in PyCharm before, but I just started using IntelliJ for some Java projects recently and so some of the tools and the layouts of the IDE were similar and easy to use for me. The keyboard shortcuts are definitely helpful to know as well. The static analysis tool that PyCharm was very useful to me, analyzing code on the fly saves a lot of time in the development process and can prevent you from making errors in your development before compilation or in this case before the script is run. PyCharm achieves this by alerting you with warnings when something might be wrong. PyCharm can also help you to keep your code looking clean by suggesting where to put new lines and how many new lines to put in order to make your code look cleaner. You can also take advantage of the auto formatting to keep your code clean. Overall PyCharm was a very useful tool to use.

1.0 Dynamic Analysis

1.1 PyUnit

The tool that we have chosen to perform dynamic analysis with is the PyUnit. PyUnit is a standard unit testing framework for the python language. It is similar to the JUnit framework for Java. PyUnit can be used for test automation and similarly to JUnit PyUnit contains mechanisms for set up and shutdown code which will be run before and after each test.

1.2 Brandon's Experience

PyUnit is a very helpful dynamic analysis tool to use in Python. All you need to do to use pyunit is to import the unittest library in python. From there all you need to do is use the functions in the unittest library to compare what you want your functions does to what output you expect. There were lots of documents that explained what pyunit is and how to use the unittest library. This made it easy to use. The only downside to pyunit was that it is very redundant because you have to call your class then you have to make different scenarios and keep writing the same code but with different parts to check. Other than the redundancy, pyunit is a very good dynamic analysis tool in python.

1.3 Elliot's Experience

It has been a while since I had done unit testing, but I had done JUnit tests before. Python’s version of unit testing was similar so it did not take me long to get adjusted to what was needed. The only issue then was working and creating the testing for the right functions. Besides from this, PyUnit provides a JUnit style testing for ensuring that unit tests are done in a manner that is reasonably easy to work with.

1.4 Justin’s Experience

I’ve written JUnit tests before so PyUnit tests were very similar in structure which made it very easy to use since it was something I was familiar with already. Like JUnit, PyUnit is a great tool to use for dynamic analysis. PyUnit can be great for regression testing as well, the tests are all automated which makes things easy and fast to run. Overall my experience with PyUnit was positive and even though it was my first time using PyUnit, the similarities made PyUnit easy and fast to use.

2.0 Fuzz Testing

Random input or edge case input can find bugs that would otherwise be left open. During testing we found a couple interesting cases with wrong or other types of input. The following are found issues and their noted GitHub Issue (if any):

- Passing the application bad input during the get credentials option in the identifier field and got back a python Traceback of calls due to an IndexError. (GitHub Issue #11)
- Passing the application an identifier with a space, program accepted the account and credentials but when trying to get the identifier with a space it doesn't work (GitHub issue #13)
- Quitting the application by sending an interrupt (ctrl + c) causing python to respond with errors containing lines of code due to the keyboard interrupt (GitHub issue #15)

3.0 Attack Surface Review

We have not made any changes to the tools that we have been using. However we did find an issue where our application couldn't run properly using PyCharms Run tool. The issue is due to the way PyCharms Run tool handles input and output to and from the user. This causes our application to have unexpected results in the way our user interface is displayed. These issues do not exist when the application is run from a terminal or command prompt. Luckily PyCharm also has a tool to run a terminal from within PyCharm so we are currently using PyCharms terminal tool instead of the Run tool to run our application.

1.0 Incident Response Plan

Justin will be our Escalation Manager, it will be his job to ensure the project is stable and secure for a release. He will be responsible for managing privacy, security and business experts to ensure the safety of user information which our application handles. Elliot will be our Legal Representative, he will be responsible for handling any legal matters that deal with our application. He will handle any legal concerns, advising, or lawsuits that concern the application. In the case of a data breach, the legal representative will also be responsible for finding or giving legal aid to victim users. Brandon will be our Public Relations Manager, it will be the PR Manager's responsibility to represent the organization in a timely professional manner through public facing media.

In case of emergency, we will be creating a group email (etaPass@gmail.com) that users can contact us with. For other non essential matters, our contact information can be found in the wiki pages of our repository.

Incident response procedures:

- Determine the priority of the incident reported (using security bar or privacy bar depending on the case)
- Determine what resources are needed in order to resolve the conflict (steps to reproduce, OS, and other types of information that would be useful to debugging)
    - If resources are needed gather the resources from the reporting party
- Determine the impact of the situation
- Construct a timeline of expectations
- Disperse the information to the appropriate parties to work on resolving the incident
    - Dev team

- Legal Representative
- PR Representative
- HR Representative

2.0 Final Security Review

Our team has decided to give our password manager application a grade of passed FSR(final security review). We have decided to give our application a passing FSR because our application is secure. We have implemented input validations, and encrypted the data. Input validation ensures that injection codes cannot be passed into the code by making sure that the input follows specific rules. We had to encrypt the data being received because if someone were to get the data, it would be encrypted and only a specific key can decrypt the data. We also have been testing our code with our dynamic analysis tools and our code passed each test. Our team has also ran our program multiple times and made sure no problems showed up.

3.0 Certify Release and Archive

GitHub Repository: https://github.com/justin-ho/passwd-mng/tree/master

Assignment 5 Release: https://github.com/justin-ho/passwd-mng/releases/tag/v1.0