Sprint 3 Retrospective

Team 19 – Sentinel Data Vault

Jiho Choi, Zhaoji Jiang, Adam Petty, Dingfu Sun, Thomas Worns

December 14, 2015

What Went Well?

1) **Making Cohesive Code Decisions**
   In the previous sprint, we had some troubles in this area. However, we worked hard to improve this time around, and were generally successful. Before beginning to code, we made better team consensus decisions on design, implementation, and code style. All members were on the same page when it came to making these commitments and had a complete understanding of them. This saved a lot of time in regards to code integration. Also, we ensured that everyone had a solid understanding of the proper program control flow, and everyone understood how to utilize the correct classes.

2) **Coordination / Communication**
   This was another area in which we faltered last sprint but were able to turn around during Sprint 3. We ensured that all progress towards the completion of any planned items was tracked carefully through our Trello board. Any needed miscellaneous fixes or design tweaks were made clear to the entire team. We added TODO comments to the code and noted any significant progress in the Git commits, which improved our coordination as well. We also paid further attention that one person's work didn't interfere or conflict with another person's code development. Each team member was aware of the tasks that were being completed by the other members.

3) **Presentation Preparation**
   Towards the backend of Sprint 3, we recognized that the appearance of our software was just as important as how well it functioned. Thus, instead of beginning implementations of a larger features, we took the time to complete and refine what we had accomplished for public demonstration. We also recognized the importance of a clean, concise, and professional presentation of our software. We took our time preparing for the final presentation and made many dry runs through our presentation outline, trying to cover as much information about the Sentinel Data Vault as possible. This process ensured that everyone presenting knew what their role was and what they needed to introduce and explain. Those who implemented certain features, spoke about them in the presentation because of their deeper understanding of its development. Overall, we were very pleased with our final presentation.

4) **Backup Implementation**
   Implementation of backup functionality for user accounts and data entries, as well as importing from backup files, went very smoothly and we are very pleased with its implementation. Early on in the sprint, we were concerned that this would be a very challenging feature to develop because we essentially had no idea how we would go about it. Fortunately, we were able to find a way to create new database files containing single user accounts and their entries, and have the user decided where to store them. And because we used an SQLite

database, this process became very straightforward once the underlying principles were understood. Additionally, because of the way backups were handled, it allowed for secure and streamlined importing from database files.

What Did Not Go Well?

1) **Early Design Decisions Hindering Development**
   Over the course of Sprint 3, we encountered a couple of unexpected roadblocks that caught us off guard. Firstly, we had to add functionality to our Cryptography class in order to facilitate our sharing functionality. Our previous design decisions in this area made the encrypting/decrypting of entries between users very difficult and not straightforward. This did not set us back significantly, but we had to take extra time retroactively fixing this implementation. We also encountered barriers regarding how the HomeView retrieves data entries. These limitations resulted in only being able to share a data entry locally between one user. A work-around was put in place, but this is far from ideal. However, a complete and proper fix would require a significant rewrite of the code that manages how the HomeView retrieves and, specifically, displays the permission-based shared entries. Lastly, on multiple occasions we found ourselves having to go back to the database design and add additional fields to the tables to service our implementation needs for sharing, cryptography, etc. This meant fixing a large portion of code that handled database accessing, for each instance that the database itself was modified.

2) **UI Issues**
   A majority of our team members developed the project using Macs, and this translated to us putting more effort in constructing the user interface to look best on OS X as opposed to Windows. Therefore, we realized we had some cross-platform user interface issues while finalizing our project. The UI of Sentinel Data Vault on Windows looks significantly less polished and even has some layout problems such as text extending beyond the boundaries of buttons and frames. In a few cases, button icons were not displaying correctly. Some of this is due to the fact that a trade-off exists between the Java UI components on different systems. Tweaking something on Windows ends up looking undesirable on Mac OS, and vice versa. Finding a good medium was often difficult. A complete fix of this issue would have taken many rounds of trial-and-error testing, which is very time consuming and was not something we could fit into this sprint.

3) **User Story (Virtual Keyboard):** As a user, I want to have a built-in virtual keyboard for text input.

   We were unable to get a working virtual keyboard during this sprint. As a team, we decided that this feature was the least important to the overall system and chose to focus on completing the other user stories, leaving the virtual keyboard as the last feature. Considering the overall goals of our program, the virtual keyboard did not contribute as much compared to the other Sprint 3 features. However, these other user stories took longer than expected to implement, and we ultimately ran out of time to integrate a virtual keyboard. We had expected

that there would exist a Java API for a virtual keyboard that we could simply integrate into the data vault. Unfortunately, this was not the case. A few virtual keyboards existed online for use, but these were a part of larger applications that did not allow us to easily extract the portions we needed. Furthermore, they did not provide the functionality we were looking for. Moving forward, if we were to create a virtual keyboard, we would have to write one ourselves for use in the data vault.

How Should We Improve?

1) **Early Design Decisions Hindering Development**
   To help mitigate unforeseen events, we can plan a small amount of extra time to deal with any unexpected implementation or functionality limitations. As always, good design and planning will serve to keep the number of surprise problems to a minimum. Planning for the future, such as anticipating new features and their requirements, is also something we need to devote more time to during the design stages of development. This would have particularly helped us avoid having to go back and add fields to the database tables, which in turn also meant revising a lot of code related to the database. More research and planning needed to go into our encryption layer of the system. This would have reduced the work necessary to implement entry sharing.

2) **UI Issues**
   The user interface is the face of our program and is also where our users get their first impression. We should obviously spend more time and effort in improving the UI than we did during this sprint. Constantly checking the look and the feel of our user interface between the two platform can be very time consuming, but it is something that must be done and have an appropriate amount of time dedicated to it. This issue can be solved by taking some extra planning steps to avoid having to fix issues at the last minute. More effort is needed to ensure that features or UI elements are all platform-neutral.