Trevor Hornsby
CS5472 Final Report

For my final project, I chose to create a personal security program capable of filling three main requirements. I wanted this program to be fully transparent to the user, use gamification principles to promote good security, and allow users to teach themselves about security, overall putting users of this program in charge of their own security. I was motivated to work on this project for two reasons, the main reason being that main-stream security programs that Avast or AVG give incredibly vague information with regard to what they are scanning and what they are protecting. This lack of visibility is very concerning as if I am trusting this program to keep my computer and data safe, it should be able to tell me where my security gaps are and what it's going to do to fill these gaps. The second main motivation for me is that as threats like ransomware continue to become more widespread, people without technical backgrounds are going to need to start learning how to be proactive with their security to avoid getting caught in a compromise.

For past work on the topic of security education, I read a paper written by M. McNulty and H. Kettani which addressed the growing concern to make security education more accessible to people without technical backgrounds. [1] To summarize the work done by M. McNulty and H. Kettani, existing research into security education mainly focuses on education for people who already work in IT, how non-technical people have a very pressing need to educate themselves about security as attacks become more advanced, and everything we use becomes more and more interconnected. Another area of past research that I looked into was gamification. To familiarize myself with the benefits and applications of gamification, I read a paper written by Dirk Basten, which explains how gamification is used in products for a number of reasons. [2] The main takeaway from this paper that applies to my project is using gamification to reinforce good behaviors. I chose to use a points system, almost like a high score to encourage good security practices. If a user has a low score, my hopes are that this will incentivize change in their behavior towards better security practices.

The main focus of my design revolves around usability. I made sure to incorporate elements of gamification like points to encourage good behavior and discourage bad behavior, and I implemented scoring in multiple formats (fraction, percentage, letter grade, and pie chart) so that the message is clear for any user reading the report to understand what their current security health is. I also tried to make the report look as aesthetically pleasing as possible, but this is a shortcoming of my project as I am not very experienced with web-design and human interface design. For future works, however, this could easily be improved by anyone who has a comprehensive understanding of HTML and CSS. I also wanted my program to be easy to run so I designed it so that it takes no arguments, and automatically saves/opens the report in a browser. This ease of use means that it could easily be scaled into automation and adapted to look for compliance purposes, like CMMC for example. Lastly, I designed this program to be readable and editable by anyone with a basic understanding of Python. I have uploaded my code and an example report output to a GitHub repository [3]. Doing this means that anyone in the world can feasibly make a branch and tune or tailor this code to their needs, as it basically serves as a proof of concept for a security policy auditing framework.

The overall implementation for my project was somewhat simple and can be broken down into a few main components. For context, everything was written in Python, and the report is displayed in HTML. For the technical requirements of the code, the program needed to collect data either from HTTP requests, OS commands, or other sources (like a port scan). From here the data is parsed into variables and evaluated against known best practices. If that data collected from the user’s system is equal to or better than the known best practices, then points are added to both the maximum score and the user’s score, and a message reinforcing their positive security behavior is added to a table object. When a user is not following best practices, points are added to the maximum score, but not the user score (creating a points deficit), and a message reflecting their negative behavior and how to fix the issue is appended to the negative behaviors table. The main idea with these text outputs is to communicate the pros and cons for a specific behavior and offer some level of insight or knowledge into the behavior so that the user can make an informed decision as to whether or not they want to fix a problem or leave it be, or on the other side, they can comprehend why a certain behavior is being considered as good behavior, and why it’s important for their security posture to continue this behavior. From here, the data is collected from the appended list and converted into an HTML format. This HTML report has three main sections, where scores and general information are displayed at the top, positive practices are displayed in the middle section, and negative practices are displayed towards the bottom. The report is somewhat color coded so that good actions are denoted by green text and bad actions are denoted by red text. The report is opened automatically in the user’s browser to avoid the need for a user to find where it was saved on their system.

Future work is needed for this project in several capacities. I think reworking the scoring system to be more dynamic and case-by-case with weights would be really helpful for intuitively communicating which issues are most pressing in the report and need immediate user attention. It would also be a significant improvement to the code to generate the text outputs more efficiently, currently, the code uses a system of if-else statements and manually written text outputs. Adding some level of autonomy to this task would make this program a lot more scalable. To add to this, it would be a good idea in future implementations to add checks for Adblock to prevent malvertising and 2FA or Windows Hello for increased authentication security for the system. Also, there isn’t much that would prevent someone from having the code check the Windows registry for specific registry values. I do not have an extensive background in working with human interface design, but the report output could be greatly improved by anyone with this type of experience. Finally, the biggest area of future work would be to have one-click solutions to implement recommended policy changes. This would single-handedly make the program more usable to the vast majority of potential users. With one-click solutions, a user could see that maybe port 22 is open, and after reading the text output realize that they do not use ssh or host an ssh server. Currently, this user would need to click the URL next to the text description and follow a guide for closing a port on Windows, in the future, I would like to have a button that the user can click that says ‘REMEDIATE’ or ‘FIX’, and clicking that button would run some commands, and close the port, thus fixing the issue identified in the report.

In conclusion, I can say that I learned a lot about how difficult it is to balance out technical features and usability features. While I felt that I could excel at writing the basic code for this project, creating intuitive text descriptions was tedious, and attempting to make the report as universally readable and aesthetically pleasing as possible was a very difficult task. The one element of my project that I struggled with the most was actually getting the matplotlib pie chart to properly generate and show up in the report to display the user’s score and score deficit. I do however feel that, while difficult, the experience of designing something intended to be used by non-technical people was incredibly valuable. For some other takeaways, I also now have a much deeper appreciation for the developers of applications I use every day, and the amount of effort and thought that went into making the software usable by everyday people. A surprising key takeaway from this project is that Microsoft does not actually implement its own best practices on Windows systems when it comes to minimum password length and lockout threshold. I thought this was really interesting since I've never configured these policy settings, and found that they were disabled by default on three of my systems.

References

[1] M. McNulty and H. Kettani, "On Cybersecurity Education for Non-technical Learners," *2020 3rd International Conference on Information and Computer Technologies (ICICT)*, San Jose, CA, USA, 2020, pp. 413-416, doi: 10.1109/ICICT50521.2020.00072.

[2] D. Basten, "Gamification," in *IEEE Software*, vol. 34, no. 5, pp. 76-81, 2017, doi: 10.1109/MS.2017.3571581.

[3] https://github.com/tjhornsb/CS5472_Final_Projec