We spent this course coming up with secure coding standards and a policy to follow. If we follow the 10 core principles of security along with the standards we created, it will help keep our code secure form hackers, and data leaks. Just a review of those 10 security principles we should follow:

1. ValidateInput Data
2. Heed Compiler Warnings
3. Architect and Design for Security Policies
4. Keep It Simple
5. Default Deny
6. Adhere to the Principle of Least Privilege
7. Sanitize Data Sent to Other Systems
8. Practice Defense in Depth
9. Use Effective Quality Assurance Techniques
10. Adopt a Secure Coding Standard

We focus on security at the beginning and not just at the end of the process. You want to have several layers of defense so that you have extra defense past the first line. Some types of Defense in Depth would be firewalls, encryption, safe language use, password protection and encryption, using community resources, and staying up to date on threats and research. All software is prone to be comprised as one could say there is no such thing as secure software, which is why we focus on it throughout the whole process.

The word hacker tends to evoke negative connotations when referred to, it is important to remember that all hackers are not created equal. If we didn’t have white hat hackers diligently seeking out threats and vulnerabilities before the black hats can find them, then there would probably be a lot more activity involving cybercriminals exploiting vulnerabilities and collecting sensitive data than there is now. Companies may actually employ white-hat hackers to make their software more secure. This can benefit the company.

In computer security things have changed, where we should have to implement this literal zero trust policy in most systems. It does add a level of difficulty with permissions to access and monitoring, bet especially in a business aspect, it is very important to do so. As a user this zero-trust method makes me feel good, and that if more people implemented this so many more companies would have secure networks and secure software.

Most companies now store personal data on servers and in databases that they can't afford to have anyone unintended to access it. Heavily implementing software security could cost more so is it worth the extra cost? This is one reason we use automation tools. They are quick and don’t take to long to test portions of code for vulnerabilities. The little bit of extra cost for buying automation tools and allowing developers to incorporate things like unit tests, pay off in the long run.

I recommend following the standards and policies listed in my final project and also recommend using resources such as the **CERT C coding standard.** Link below:https://wiki.sei.cmu.edu/confluence/display/cplusplus/Relation+to+the+CERT+C+Coding+Standard