Not leaving security to the end is one of the most critical things to remember as a coder. If you leave security to the end, you are a step behind everyone else already. Simply by coding with proper standards, coding with security architecture and principles in mind, and running unit tests, you can ensure that your code is as strong as it can get right off the plate. You can always add security strength to the code later, but you can only do that if you have built the right foundation for secure coding.

One thing that you will have to access when you are coding is whether the extra time spent on security is worth the cost of the time, the hassle to access the data, and so on. Simply put, if I just wanted to play a game with some friends online, I likely wouldn't put strong securities for them to pass since it is small-scale, unimportant, and would be a general hassle for something that doesn't matter that much. Would I really need to spend the money and time to develop Fort Knox-level security for a shopping list that I save online? I wouldn't think so. Thus, you should assess the importance of what it is that you have to protect, assess what risks there are with giving it only as much security as you plan to give it, and also assess whether or not you are giving any potential hackers some motive to hack whatever it is that you are protecting.

One of the strongest and most important ways to code is with "zero trust." Zero trust is the idea that everyone is banned from accessing your things unless you give them explicit permissions for explicitly what they plan to do for only as long as they plan to do it. Like the name implies, you trust no one and only validate and authenticate those that you know can be trusted or you otherwise have to let it.

The most important security policies to enact in your coding follow basic principles that can be summed up like this: When you code, make sure you validate input data and ensure that it is cleaned before you send or store it anywhere. Ensure that you take a "Defense in Depth" approach and apply layers of security to project you. Always write your code with the architecture of security policies in mind, adopting a secure coding standard for everything you write -- and make sure you look for compiler messages during and unit test it afterward! Be sure to keep your code as simple as possible and to only authorize users to access your code and functions when necessary, denying everyone else by default and ensuring that you are only giving people the least amount of privilege you can possibly grant them to do the task they are assigned to do. If you follow all of these principles I am recommending, your code will surely be secure.