# Southern New Hampshire University

CS 405: Secure Coding

# Justin Vallia

Security Policy Presentation

December 08, 2021

# CS 405 Project Two Script Template

Complete this template by replacing the bracketed text with the relevant information.

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Good afternoon, my name is Justin Vallia, and I will be presenting the following security policy information which includes using external testing methods in order to identify any potential vulnerabilities which may exist. I will also be covering guidelines and methods to begin implementation of this policy. Accompanying this information will also be recommendations for maintaining and adding to this policy in the future as we see necessary. |
| **2** | I would like to begin this presentation by showing an illustration of Defense in Depth and the hierarchy of this policy will work. It is important to note the different methods within the Defense in Depth model in order to maintain multiple levels of secure coding standards and principles. Our ultimate goal here at Green Pace should be to produce the best coding services we can while maintaining the most secure environment we can also. |
| **3** | When we begin to dive into the Secure Coding methods and principles we first need to understand how a threat matrix works. Typically, it is separated into likely and unlikely threats followed by high relevancy and low relevancy threats. The diagram here represents the baseline of our policy. It also should be noted that once these standards are in place it is not a failsafe, and vulnerabilities will still exist. |
| **4** | As we move forward into the principles themselves there are 10 baseline or initial principles that we should remember and be aware of. The 10 principles are as follows:  Be sure and validate our input data, heed our compiler warnings, design and construct for security first in development, keep it simple and avoid overcomplicating things, deny access to systems as a default, adhere to the principle of least privilege, always sanitize and clean the data that is sent to other systems, practice Defense in Depth because nothing in this world is perfect or secure, never compromise on utilizing effective quality assurance techniques, and ensure we are adopting secure coding standards. |
| **5** | Out of the many coding standards that exist we again want to clarify our baseline standards that we shall always comply with:   * Do not cast an out-of-range enumeration value * Be sure to use valid references, pointers, and iterators to reference the elements of a container * Do not attempt to create a std::string from a null pointer * Do not store pointer values that are already owned in an unrelated smart pointer * Properly deallocate dynamically allocated resources * Use a static assertion to test the value of a constant expression * Handle all exceptions thrown before the main() begins executing * Do not alternate input and output from a file stream without an intervening positioning call * Do not invoke virtual functions from constructors or destructors * Value returning functions must return a value from all exit paths |
| **6** | As we continue to tear into the meat of our security policy we next want to cover our encryption policies. These encryption standards should always be in our mind when we are dealing with any type of data or information. Encryption in flight is designed to encrypt the data while it is in transit inside the system. Encryption at rest is primarily designed to prevent and deter attackers from gaining access to any unencrypted information because this information will be encrypted while it is on the disk. Finally, we look at encryption in use. This is a combination of the two previous encryption types given that it ensures that all data and information is encrypted at all times. This is the most effective method and should always be used when possible. |
| **7** | Breaking down our policy even further we want to next take a look at Triple-A policies. This entails **A**uthentication, **A**uthorization and **A**ccounting. Authentication is authenticating or confirming the user is a registered user who has access to the system. Authorization is the individual level of access each user will have within the system. Accounting is the process which tracks the users’ actions taken while they are logged into the system. |
| **8** | Unit testing plays an important role within our policy, and it is important that we always perform our unit testing as early and as often as possible. This will ensure our code is free of vulnerabilities prior to us getting to a point where it will all snowball into a very large problem. |
| **9** | This is a diagram of our automation summary. We see that we have sets of steps for both pre-production and production. |
| **10** | DevSecOps, this term may seem complex however it simply refers to the integration of security measures into the SDLC (software development life cycle).  This method takes “full circle” approach which is mainly focusing on keeping our code secure. While this method is truly an incredible tool to have at our disposal it does not eliminate our need to still utilize Defense in Depth practices and to also perform unit testing still as early and often as possible. This combination of all of these principles and methods will ensure we are able to locate and fix any vulnerabilities which may arise or exist within our code. |
| **11** | Nothing in our world is perfect or secure. This is no different inside of our code and our systems. We must be diligent in always assuming that threats and flaws exist. It falls back on us to stay up to date with the ever-changing world of software development and technology or else we risk becoming obsolete. This includes studying common threats and prevention techniques that arise. |
| **12** | Now the choice and success of our future falls onto our decisions. Do we act now and implement this policy, or should we wait? In choosing to wait we risk our company to being exposed to many different attacks and threats which are growing in frequency on a near daily basis. By choosing to implement this policy we can begin securing our data and information almost immediately.  As we discussed previously nothing is perfect and it should not be expected that this policy is perfect either. There will be speed bumps we encounter while implementing this policy and we should do our best to be prepared for them and to overcome them in a swift manner. Complexity can be a great asset to our code however the shear complexity of this policy combined with it being in its infancy within our company could lead to vulnerabilities which will need to be addressed until we work out the rhythm of the policy.  One major concern is once this policy is implemented the balance of our existing code bases could be upset and it is highly recommended that all information and data be backed up prior to beginning this implementation. |
| **13** | There are a few other recommendations to keep in mind as we move forward and progress with this policy. First we should be aware of the speed at which our development occurs could be slowed down because of the extra steps we will be taking in order to maintain a secure code. Secondly, we should have, at minimum, 2 employees to monitor the changes and we will still have the need for other tools to help automate this process altogether. Lastly, we should consider implementing two-factor authentication for our employees as we move forward with this policy. In adding this extra layer of security, we are also strengthening the protection of our system. |
| **14** | In closing, it should be noted that the list of 10 core Security Principles in combination with the 10 Coding Standards which were covered in detail for this policy should be considered the minimum baseline for our policy and should not be compromised on.  If we are diligent in ensuring this baseline is always met then at this point we can expect to build more layers onto this policy in the future as we see necessary. With any additions of new standards and principles we can ultimately add more strength to our code in development.  The beauty of this policy is the flexibility that exists within in it. We can add to our baseline anytime we see fit and each addition adds another layer of strength to our code and systems. |
| **15** | Thank you for the opportunity to write and present this policy for Green Pace. I am looking forward to any questions and feedback that anyone may have. |