# CS 405 Project Two Script Template

Bryce Cooperrider

10/23/2024

Video Link: <https://youtu.be/xAdywOYQm2Y>

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Hello, my name is Bryce Cooperrider. This is my security policy presentation for Green Pace. |
| **2** | The use of the security policy is designed to aid in the secure development of applications at Green Pace and ensure we comply with industry standards. This policy is part of our defense in depth strategy, which is a multi-layered approach to protecting our systems and data within them.  Everyone at Green Pace will need to follow the security policy, including the software developers and IT staff. The policy covers areas such as coding standards, authentication, and encryption, providing a structured approach to security risks. |
| **3** | Next up, the threat matrix outlines potential vulnerabilities I’ve identified, by categorizing them into Likely, Unlikely, Priority, and Low-Priority. You can view the matrix to see how I’ve categorized them.  These standards aren’t just ranked by these categories, they all use the likely/unlikely and priority scale. Some items may be unlikely but high in priority due to their nature. Just because something is low priority doesn’t mean it shouldn’t be taken care of, just lower on the list. |
| **4** | Here we can see the 10 core security Principles (List them off)  1: Validating data input into a system helps mitigate and prevent vulnerabilities caused by malicious inputs.  2: Developers need to understand and fix compiler warnings before code goes into production.  3: Create an architecture that considers potential security risks and design the system with security in mind.  4: Simple code reduces complexity and increases understanding of the code.  5: Denying access by default reduces the risk of unauthorized users accessing the system.  6: Users should receive bare minimum level of access that is required to do their tasks.  7: Sanitize all data passed into complex subsystems to ensure it can’t be exploited by malicious hackers.  8: Multiple layers of security protect against different types of attack.  9: Effective quality assurance techniques like code review and testing identify potentially hazardous code.  10: Set a coding standard, which is a set of guidelines and practices that all developers are to follow. |
| **5** | Next, we have our 10 coding standards. (List them off)  1: Validating input data helps mitigate certain types of attacks like SQL injection. Satisfies principles 1 and 7.  2: Casting out of range values can result in bugs or corrupted information appearing. Satisfies principle 2.  3: Returning a null pointer can lead to undefined behavior in the program, potentially returning sensitive information. Satisfies principles 2 and 4.  4: Attempting to modify a string literal is undefined behavior and can result in SQL Injection. Satisfies principle 3.  5: Freeing memory when no longer needed helps reduce undefined behavior. Satisfies principle 9.  6: Incorporating diagnostic tests is done to ensure working code functionality, and to reduce the possibility of denial-of-service attacks. Satisfies principles 4 and 9.  7: Handling all exceptions is a basic standard that must always be completed as fixing these exceptions is a top priority. Satisfies principle 2.  8: Writing constructor member initializers in canonical order is an important style the team must follow. Initializing someVal before initializing dependsOnSomeVal will cause the code to fail. Satisfies principles 4 and 10.  9: Closing files when they are no longer needed is part of a good resource management strategy. Satisfies principle 9.  10: Detecting errors when converting strings to numbers is vital so the values stored in variables aren’t unexpected. Satisfies principle 2. |
| **6** | Encryption at rest refers to any data that is in an encrypted state while it is in storage. Even if access to the data is gained, it will be unreadable without having the key to decrypt it. This is an important policy that provides a layer of defense in case a data breach happens. The data will be out but unreadable without the key.  Encryption in flight refers to any data while it is in transit from one place to another. This ensures that if the data is intercepted while in transit, it will be unreadable because the interceptor doesn’t have the key to decode it.  Encryption in use refers to the encryption of data while it is being used by a system. It is used to keep the data secure even as it is changing. It is important to encrypt data while in use so there isn’t a weak link between it being at rest and in flight. |
| **7** | Authentication is the process of confirming a user is who they say they are. The most common use of authentication is when a user attempts to login to a system. This policy applies whenever a user attempts to gain access to a system or something within a system. The typical way authentication is implemented is by using usernames and passwords.  Authorization is the process of confirming a user is supposed to have access to the system they are requesting. It confirms the level of access a user has within a system. Once a user is authenticated, they are free to access parts of a system they have access to. It helps keep users out of places that they shouldn’t be.  Accounting is the process of tracking and logging all requests made by a user. Once example would be logging the changes a user makes to a database or files. It is important to keep this log to determine what happened if something goes wrong. |
| **8** | Here are examples of unit testing for memory and data storage. |
| **9** | This test assures that the max size of the collection is greater than or equal to its size, allowing room for expansion. |
| **10** | This test ensures that the size of the collection can be decreased to zero. Further testing would be testing memory addresses to see if information is still held in them. |
| **11** | This test assures us that we can increase the size of the collection by resizing it with a larger number. Further testing could include writing information to the newly reserved space. |
| **12** | This test assures that shrink to fit removes additional space from the collection. Further testing could include resizing to different amounts and shrinking to fit a collection with size 0. |
| **13** | A brief description of the DevSecOps pipeline may look like this.  The pre-production phase starts with assess and plan. We start with security risk assessments to help identify vulnerabilities early on.  In the design phase, we enforce secure coding standards, often done through automated checks, and follow best practices such as those from OWASP.  During the build phase we secure our code and components, checking for vulnerabilities that may be brought upon by external libraries.  The verify and test phase includes adding automated security testing, which is vital to catch vulnerabilities before the code is released.  Moving onto the production phase, we start with transition and health check. This check includes conducting penetration tests to make sure our system adheres to security policies.  The monitor and detect phase involve continuous monitoring to identify potential security incidents.  The respond phase involves automated incident responses to reduce the impact of any security breaches.  Lastly, the maintain and stabilize is where we conduct audits to assess our security and ensure it remains secure over time. |
| **14** | There are benefits and downsides of acting to security concerns in the moment versus waiting.  Acting in the now can mitigate any expected attacks, as well as decrease the long-term costs. Having stronger initial security is not a bad thing to have.  Risks include building up resources against the wrong type of attack, as well as misused resources due to protecting against the wrong thing.  Benefits of acting later include knowing exactly what the company needs to defend against.  It comes with much more risk though due to the potential cost to the company, potentially long development time to produce a security patch, as well as the risk of damaging company reputation. |
| **15** | What steps can we take to increase security?  Implement security automation. Having automation tools for constant security testing.  Invest in staff training. Providing adequate training for the development team on best practices ensures security is always in their minds.  Conduct a risk assessment on the current policies. Our policies are always able to be changed and should be changed when new information or better techniques come to light.  Enable Cross Team Communication. This ensures each team is working together to mitigate security concerns and keeps them on the same page. |
| **16** | To conclude, DevSecOps is necessary in the evolution of maintaining secure and reliable software. By always having security in the front of most of the development teams’ minds, we can reduce risks, detect threats more reliable, and respond to incidents in a timely manner.  The transition will require investment but is in the companies’ best benefit to adopt these security policies in order for the team to deliver reliable and secure systems. |
|  |  |
|  |  |