Hello, welcome to the security policy presentation for Green Pace, I am Jacob Mueller associate developer.

The security policy covers a host of subtle and obvious vulnerabilities ranging in their severity and likelihood. Some of the standards will seem like common sense, I assure you that we are working to avoid that way of thinking here to stop common mistakes early.

These are the 10 principles of safe coding:

1. Validate Input Data​ – always ask who, and never trust
2. Heed Compiler Warnings​ – work each issue from the center out
3. Architect and Design for Security Policies​ – leave room and comments calling out issues
4. Keep it Simple​ – don’t be a hero
5. Default Deny​ – Principle of least privilege
6. Adhere to the Principle of Least Privilege​ – I’m serious
7. Sanitize Data Sent to Other Systems​ – that’s the information that will tank our accountability
8. Practice Defense in Depth​ – like kevlar
9. Use Effective Quality Assurance Techniques​ – don’t just trust yourself, automate
10. Adopt a Secure Coding Standard​ – stick to this like a guide

These are the coding standards we are holding each other to:

1. Do not cast to an out-of-range enumeration value. – Data Type​
2. Value-returning functions must return a value from all exit paths. – Data Value​
3. Guarantee that storage for strings has sufficient space for character data and the null terminator. – String Correctness​
4. Prevent SQL Injection – SQL Injection​
5. Do not access freed memory – Memory Protection​
6. Use a static assertion to test the value of the constant expression – Assertions​
7. Exception objects must be no-throw copy constructable. – Exceptions​
8. Do not alternately input and output from a file stream without an intervening positioning call – Input/Output​
9. Guarantee that container indices and iterators are within the valid range – Containers​
10. Copy operations must not mutate the source object – Object Oriented Programming​

Here is the matrix, 3 4 and 5 are our priority targets.

Encryption:

* Encryption at rest is a means of protection for stored data designed to prevent attackers from accessing any unencrypted data by ensuring that it is instead encrypted.  It’s used primarily to prevent attacks where a hacker would obtain access to the storage or database, but wouldn’t have the means to bypass encryption.  High priority and highly recommended. ​
* Encryption at flight is when data is encrypted when it’s being transmitted, and should be applied by using secure transfer such as SFTP (Secure File Transfer Protocol). Data is vulnerable during flight if left unencrypted or improperly secured.  VPN may offer the level of defense desired in the current climate.​
* Encryption in use is when data is accessed or used by an application or by an end user.  This is widely considered the most vulnerable and needs to be limited to users who need it, look to the rule of least power.

Triple A policies

Authentication - The process when the user wants to access the network and needs credentials such as a username and password.​

Authorization -Enforces policies on networks for the user and gives them access to only the resources that they need.​

Accounting -Holds users accountable for their actions and keeps track of who does what by monitoring or logging an user’s activities.

Testing:

Here you can see that we are testing out our clear functionality. \*read comments\*

Here you can see that we are testing out our resize and decrease functionality. \*read comments\*

Here you can see that we are testing out our resize and increase functionality. \*read comments\*

Here you can see that we are testing out our resize and decrease to zero functionality. \*read comments\*

Automation:

Automation will be used for the enforcement of and compliance to the standards defined in this policy. Green Pace already has a well-established DevOps process and infrastructure. Define guidance on where and how to modify the existing DevOps process to automate enforcement of the standards in this policy. Use the DevSecOps diagram and provide an explanation using that diagram as context.​

Automation is a necessity in the development of software and can be utilized in many areas and at almost anytime during the development cycle. That being said, it should be used within reasonable timelines, meaning it may be used from the start but might not have the most impact on the development, also it may take up too much time if over utilized. When implemented properly, automation can do a lot of heavy lifting parsing code for vulnerabilities, testing functionality, and configuring security measures.

Tools:

The devsecops pipeline is the course the project takes as it cycles through release and planning stages​

Automation helped in the code, build, and test portions of the devsecops pipeline. CPP check was used to assess coding efficiency, visual studio handles the building and can even help with testing if the right frameworks are in use.​

Depending on how the project is deployed there are automated monitoring options that will help shore up the defenses after release.

Risks and Benefits:

Hackers already have a head start, they are able to build off of the same concepts and basic logic that we referenced for this policy​

Starting early and communicating often will lead to a more comprehensive defense​

Waiting to develop defensive measures once we decide on the data structures and functionality of our project could mean missing out on vulnerabilities that take time to catch or patch.

Recommendations:

Incorporate more languages within our assessments and policies​

Or pare down the amount of languages in use to provide deeper security for a smaller coding language pool

Conclusion:

With the bulwark provided there is nowhere to go but forward into a zero trust security policy that will build upon our principles and standards.

Thank you.