https://youtu.be/WZLuDbK7FnQ

Software development at Green Pace requires consistent implementation of secure principles to all developed applications. Consistent approaches and methodologies must be maintained through all policies that are uniformly defined, implemented, governed, and maintained over time. This policy defines the core security principles; C/C++ coding standards; authorization, authentication, and auditing standards; and data encryption standards.

This threat matrix chart shows how to prioritize levels of vulnerability based on the impact of the secure coding standard.

* Validate Input Data – Coding Standard 1 STD-001-CPP
* Heed Compiler Warnings – Coding Standard 2 CON50-CPP
* Architect and Design for Security Policies – Coding Standard 3 OOP50-CPP
* Keep It Simple – Coding Standard 4 ERR60-CPP
* Default Deny – Coding Standard 5 FIO50-CPP
* Adhere to the Principle of Least Privilege – Coding Standard 6 MEM50-CPP
* Sanitize Data Sent to Other Systems – Coding Standard 7 STR50-CPP
* Practice Defense in Depth – Coding Standard 8 CTR50-CPP
* Use Effective Quality Assurance Techniques – Coding Standard 9 INT50-CPP
* Adopt a Secure Coding Standard – Coding Standard 10 EXP50-CPP

The ranking system I have utilized one which prioritizes preventing memory related issues that could cause undefined behaviors. The priorities are based on potential severity

* Do not access freed memory
* Guarantee that storage for strings has sufficient space for character data and the null terminator
* Guarantee that container indices and iterators are within the valid range
* Do not destroy a mutex while it is locked
* Do not depend on the order of evaluation for side effects
* Do not invoke virtual functions from constructors or destructors
* Do not alternately input and output from a file stream without an intervening positioning call
* Do not cast to an out-of-range enumeration value
* Do not abruptly terminate the program

|  |  |
| --- | --- |
| Encryption in rest | This policy ensures that all data transmitted over the network is encrypted. It ensures that even if the storage medium is compromised, the data remains unreadable without the proper decryption key. |
| Encryption at flight | In-flight encryption, also known as data in transit encryption, is used to secure data as it travels over a network. This is achieved by encrypting the data packets during transmission, preventing unauthorized access during transit. |
| Encryption in use | In-use encryption, also known as runtime encryption, protects data while it is being processed or used by applications or services. It ensures that sensitive information remains encrypted during computation and is only decrypted when needed. |

In summary, the DevSecOps pipeline involves a combination of planning, coding, building, testing, deployment, monitoring, incident response, and compliance/governance, with various external tools employed at each stage to automate security checks and ensure a secure development and deployment process. One tool that could be used during the coding stage would be CPPCheck. I was introduced to this tool recently and have discovered the incredible power it holds.

In summary, the DevSecOps pipeline involves a combination of planning, coding, building, testing, deployment, monitoring, incident response, and compliance/governance, with various external tools employed at each stage to automate security checks and ensure a secure development and deployment process. One tool that could be used during the coding stage would be CPPCheck. I was introduced to this tool recently and have discovered the incredible power it holds.

Waiting before the implementation of security policies can be detrimental to the success of a project. By waiting, you run the risk of unsecure code being added to your project. Having to return to previous parts of the project to fix vulnerabilities is a waste of time, money and resources. If you implement changes midway, there will likely be changes that will have to take place for your code to comply with the new policies. The solution is to create strong policies from the start.

As technology improves, security becomes ever harder to guarantee. Current know vulnerabilities are not the only vulnerabilities out there. There will continue to be new vulnerabilities that appear and must be dealt with. The introduction of quantum computers has the capability to turn everything we know about security upside down. The best way to stay safe, is to stay up to date on current vulnerabilities.

* As mentioned before, quantum computers pose a massive threat to cyber security today. One standard that should likely be included is the addition of quantum sage encryptions. While not necessary yet, it will be in just a few short years. Currently, AES 256 is considered quantum-safe due to the variable key size.