# Bucklin - CS 405 Project Two Script

Link to presentation recroding hosted on YouTube:

**https://youtu.be/jBbhasGQM44**

| **Slide Number** | **Narrative** |
| --- | --- |
| **1** | Green Pace  Security Policy Presentation  Developer: *Sean Bucklin* |
| **2** | Overview: Defense in Depth  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   * The principles defined in this policy provide guidance on Green Pace’s values, objectives, and goals * This policy details how Green Pace with achieve these objectives and what is expected of every employee |
| **3** | Threats Matrix   * Buffer Overflow, high priority, one of the most common critical vulnerabilities * Validate input, high priority, never trust outside data * Injection Attack * Occurs when an outside actor feeds malicious commands into a subsystem * Integer Overflow/Underflow * Occurs when a value exceeds the limits of the integer data type * Memory Leaks * Occurs when a user allocates memory but does not use it |
| **4** | 10 Principles   * Validate Input Data   + STD-004-CPP * Heed Compiler Warnings   + STD-007 * Architect and Design for Security Polices   + STD-001-CPP, STD-003-CPP, STD-009-CPP, STD-010-CPP * Keep It Simple   + STD-003-CPP, STD-005-CPP * Default Deny   + STD-004-CPP * Adhere to Principle of Least Privilege * Sanitize Data Send to Other Systems   + STD-004-CPP * Practice Defense in Depth * Use Effect Quality Assurance Techniques   + STD-002-CPP, STD-006-CPP * Adopt a Secure Coding Standard   + STD-001-CPP, STD-003-CPP, STD-008-CPP |
| **5** | Coding Standards   * Data Type [STD-001-CPP]   + Copying data to a buffer that is not large enough to hold that data results in a buffer overflow * Data Value [STD-002-CPP]   + Prevent the leak of or unauthorized access of sensitive data * String Correctness [STD-002-CPP]   + Standard library functions such as ‘std:string’ is better than c-style styrings * SQL Injection [STD-004-CPP]   + String data passed to complex subsystems may contain special characters that can trigger commands * Memory Protection [STD-005-CPP]   + Accessing a pointer that has been deallocated by memory management is undefined behavior * Assertions [STD-006-CPP]   + should not be used as a security measure * Exceptions [STD-007-CPP]   + exceptions thrown are processed by the nearest handler with types that match the exception thrown * Input – Output [STD-008-CPP]   + Release the system resources used by the file when they are no longer needed * Characters & Strings [STD-009-CPP]   + Index operators are unchecked and could lead to range errors * Expressions [STD-010-CPP]   + Variables assume unexpected values when accessed before initialization |
| **6** | Encryption Policies   * Encryption in Rest   + This is designed to ensure that if hackers access data on disk, it will be encrypted and unusable   + Strong encryption standards should be adhered to, such as implementing AES or RSA encryption methods   + Encryption keys should never be stored in the same location as the data * Encryption at Flight   + This represents data being transferred over a network. Secure transfer protocols should always be used   + SFTP should be used over the unsecure FTP   + For web servers TLS / HTTPS should be used for all connections   + Wherever possible avoid using self-signed certificates * Encryption in Use   + In-use encryption helps keep data encrypted through the data’s entire lifecycle   + All sensitive data remains encrypted with AES-256   + Additional benefit of in-use encryption is that it can detect anomalies by analyzing requests and blocking suspicious activity. |
| **7** | Triple A Policies   * Authentication   + Authentication is the first layer of Triple-A and it’s the way that the application identifies a user   + Authentication may be completed using username & passwords, single sign on (SSO), biometrics, digital certificates, and public key infrastructure   + Good practice is to enforce two-factor authentication to add more defense in depth to authentication * Authorization   + Administrators grant each user a certain level of access   + Apply the principle of least privilege   + Only users with administration level of access will be granted the ability to add new users and determine their level of access * Accounting   + Allows administrators to audit the system     - See what files were accessed by users within the system     - After a user is logged in to the system all tasks performed by the user are logged to that user |
| **8** | Unit Testing:  Unit testing is a way to scrutinize each small unit of software for proper operation  Each unit can be tested to confirm conformance to security standards  Attached screenshot of the results from Google Test unit testing. |
| **9** | Automation  Test automation is the practice of running tests automatically, managing test data, and utilizing results to improve software quality. Testing occurs during bothe pre-production and production phases. |
| **10** | Tools   * + DevSecOps pipeline focus heavily on integrating tools and practices into the process of planning, building, testing, deploying, and monitoring software     - Threat modelling during the design phase     - Security Scanning and testing throughout development     - Security analysis during the maintenance phase     - Remediation addressing vulnerabilities discovered after deployment     - Monitoring deployed systems   + Static Testing Tools     - Such as CPPCheck can be used during development to help identify vulnerabilities early   + Source composition anylysis [SCA]     - Scan dependencies and libraries for vulnerabilites |
| **11** | Risk And Benefits   * Leaving security to the end does not have any benefits   + One could wrongly argue that it speeds up development * Keeping security involved early ensures a more cost effective and secure software   + By keeping security in mind throughout the DevSecOps lifecyle you inhertently build in a defense in depth strategy   + Many defense in depth tactics such as Triple-A or Zero trust require a certain architecture that would be impossible to add later without great cost |
| **12** | Recommendations   * This policy could benefit from the addition of more standards * In mapping standards to principles it became clear there aren’t enough standards for complete coverage of the principles * Review more common vulnerabilities, and identify more standards to prevent them * Review more DevSecOp tools and detail stricter guidelines on when and how they are to be used. |
| **13** | Conclusions   * The proposed security policy should be adopted at Green Pace * The identified principles provide guidance on security goals * The identified standards provide clear steps to help achieve those goals * Adopting security early is crucial to successful software   + Its very difficult to do so retroactively, and incurs great cost * This policy can provide clear guidance to all members of Green Pace and what is expected   + Get everyone moving together in the same direction on security   + Provide Defense In Depth |
| **14** | References  -Checkpoint was used as a reference for the DevSecOps pipeline and tools used  -GuardRails was a good reference used for common CPP vulnerabilities. |