**Green Pace Developer: Security Policy Guide Template**



# Green Pace Secure Development Policy

## Contents

[Overview 2](#_Toc52464053)

[Purpose 2](#_Toc52464054)

[Scope 2](#_Toc52464055)

[Module Three Milestone 2](#_Toc52464056)

[Ten Core Security Principles 2](#_Toc52464057)

[C/C++ Ten Coding Standards 3](#_Toc52464058)

[Coding Standard 1 4](#_Toc52464059)

[Coding Standard 2 5](#_Toc52464060)

[Coding Standard 3 6](#_Toc52464061)

[Coding Standard 4 7](#_Toc52464062)

[Coding Standard 5 8](#_Toc52464063)

[Coding Standard 6 9](#_Toc52464064)

[Coding Standard 7 10](#_Toc52464065)

[Coding Standard 8 11](#_Toc52464066)

[Coding Standard 9 13](#_Toc52464067)

[Coding Standard 10 14](#_Toc52464068)

[Defense-in-Depth Illustration 15](#_Toc52464069)

[Project One 15](#_Toc52464070)

[1. Revise the C/C++ Standards 15](#_Toc52464071)

[2. Risk Assessment 15](#_Toc52464072)

[3. Automated Detection 15](#_Toc52464073)

[4. Automation 15](#_Toc52464074)

[5. Summary of Risk Assessments 16](#_Toc52464075)

[6. Create Policies for Encryption and Triple A 16](#_Toc52464076)

[7. Map the Principles 17](#_Toc52464077)

[Audit Controls and Management 18](#_Toc52464078)

[Enforcement 18](#_Toc52464079)

[Exceptions Process 18](#_Toc52464080)

[Distribution 19](#_Toc52464081)

[Policy Change Control 19](#_Toc52464082)

[Policy Version History 19](#_Toc52464083)

[Appendix A Lookups 19](#_Toc52464084)

[Approved C/C++ Language Acronyms 19](#_Toc52464085)

## Overview

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.

## Purpose

This policy defines the core security principles; C/C++ coding standards; authorization, authentication, and auditing standards; and data encryption standards. This article explains the differences between policy, standards, principles, and practices (guidelines and procedure): [Understanding the Hierarchy of Principles, Policies, Standards, Procedures, and Guidelines](https://www.linkedin.com/pulse/understanding-hierarchy-principles-policies-standards-wally-beddoe/).

## Scope

This document applies to all staff that create, deploy, or support custom software at Green Pace.

## Module Three Milestone

### Ten Core Security Principles

| **Principles** | Write a short paragraph explaining each of the 10 principles of security. |
| --- | --- |
| 1. ValidateInput Data | To ensure proper formatting of input and prevent any malicious injection, validation should be applied at both the syntactic level (correct structure, such as for currency, dates, phone numbers) and the semantic level (e.g., ensuring values are within a valid range or consist of the correct number of digits). |
| 1. Heed Compiler Warnings | Always ensure that if an error occurs during the process, a warning message is displayed so developers know what the issue is, rather than ignoring it and potentially causing a bug in the system. |
| 1. Architect and Design for Security Policies | Follow design and architecture best practices and policies. This includes defining security policies that address data protection, access control, and threat mitigation. |
| 1. Keep It Simple | Keep the code simple and easy to understand. Whenever possible, avoid complexity, as simplicity ensures higher levels of user acceptance and interaction. |
| 1. Default Deny | Set the default to "deny" for any unauthorized access to minimize vulnerability to attacks and ensure that only authorized users can gain permissions and access. |
| 1. Adhere to the Principle of Least Privilege | Provide users with the minimum/limit privileges necessary to perform their tasks, preventing unauthorized users from accessing data or performing additional functions. This reduces potential risks. |
| 1. Sanitize Data Sent to Other Systems | When transmitting data, especially sensitive information, it's important to perform a sanitization process to protect against SQL injection attacks. This ensures that only safe, validated data is sent, reducing the risk of breaches and vulnerabilities. |
| 1. Practice Defense in Depth | Implementing multiple layers of protection is a best practice for safeguarding sensitive data. For example, if an attack compromises the first layer, additional layers will still provide defense, reducing the overall risk of a successful breach. |
| 1. Use Effective Quality Assurance Techniques | Follow and implement best QA techniques, such as manual testing, unit testing, and functional testing. By using these QA methods, you can ensure that the code functions as intended and reduce the risk of defects. |
| 1. Adopt a Secure Coding Standard | Follow the rules of secure coding standards and best practices, such as writing clear comments, validating input, and using proper error handling. Additionally, ensure that sensitive data is encrypted, avoid hardcoding secrets, and regularly review and update code to address potential vulnerabilities. |

### C/C++ Ten Coding Standards

Complete the coding standards portion of the template according to the Module Three milestone requirements. In Project One, follow the instructions to add a layer of security to the existing coding standards. Please start each standard on a new page, as they may take up more than one page. The first seven coding standards are labeled by category. The last three are blank so you may choose three additional standards. Be sure to label them by category and give them a sequential number for that category. Add compliant and noncompliant sections as needed to each coding standard.

#### Coding Standard 1

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Type** | [STD-001-CPP] | Do not use floating-point variables as loop counters |

| **Noncompliant Code** |
| --- |
| In this code, using floating-point numbers as loop counters causes precision limitations and confusion due to inconsistent loop conditions. |
| For (float x = 0.0; x<=10.0; x+=1.0) |

| **Compliant Code** |
| --- |
| Solution to use the integers as loop counters |
| For (int x = 0; x<=10; x+=1) |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** 2.Head Compiler Warnings ; 9.Use Effective Quality Assurance Techniques; 10.Adopt a Secure Coding Standard |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Probable | Low | **P6** | **L2** |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/c/Clang) | 3.9 | cert-flp30-c | Checked by clang-tidy |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 24.04 | **for-loop-float** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-FLP30** | Fully implemented |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | [STD-002-CPP] | Ensure size arguments for variable length arrays are in a valid range |

| **Noncompliant Code** |
| --- |
| If the size argument is not validated, an attacker could provide a very large size, leading to excessive memory allocation, which may cause a buffer overflow or memory corruption. This can allow an attacker to overwrite critical program data, leading to security vulnerabilities. |
| #include <iostream>  int main() {  int x;  std::cout << "Enter number: ";  std::cin >> x;  int arr[x];  for (int y = 0; y < x; ++y) arr[y] = y; // Risk of overflow  for (int y = 0; y < x; ++y) std::cout << arr[y] << " ";  std::cout << std::endl;  return 0;  } |

| **Compliant Code** |
| --- |
| This code safely handles user input by validating that the entered number x is positive before allocating the array, preventing potential errors. By ensuring the input is valid, it reduces the risk of stack overflow or memory loss, making the program more safe. |
| #include <iostream>  int main() {  int x;  std::cout << "Enter number: ";  std::cin >> x;  if (x <= 0) {  std::cout << "Invalid number." << std::endl;  return 1;  }  int arr[x]; // Safe allocation  for (int y = 0; y < x; ++y) arr[y] = y;  for (int y = 0; y < x; ++y) std::cout << arr[y] << " ";  std::cout << std::endl;  return 0;  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** 1.Validate Input Data; 9.Use Effective Quality Assurance Techniques; 10.Adopt a Secure Coding Standard |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | Probable | High | **P6** | **L2** |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Cppcheck](https://wiki.sei.cmu.edu/confluence/display/c/Cppcheck) | 2.15 | **negativeArraySize** | Context sensitive analysis Will warn only if given size is negative |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.1p0 | **ALLOC.SIZE.IOFLOW** **ALLOC.SIZE.MULOFLOW** **MISC.MEM.SIZE.BAD** | Integer Overflow of Allocation Size Multiplication Overflow of Allocation Size Unreasonable Size Argument |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2023.1 | **CERT\_C-ARR32-a** | Ensure the size of the variable length array is in valid range |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **9035** | Assistance provided |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String correctness** | [STD-003-CPP] | Do not attempt to modify string literals |

| **Noncompliant Code** |
| --- |
| In this code, attempting to modify the character at index 0 to 'S' causes undefined behavior. |
| char \*str = "string literal";  str[0] = 'S'; |

| **Compliant Code** |
| --- |
| A solution is to create an array that contains the string, allowing for safe modifications since the string is stored in an array. |
| **char** str[] = "string literal";  str[0] = 'S'; |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** 2. Heed Compiler Warnings; 9.Use Effective Quality Assurance Technique; 10.Adopt a Secure Coding Standard |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Likely | Low | **P9** | **L2** |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2024.2 | **C0556, C0752, C0753, C0754**  **C++3063, C++3064, C++3605, C++3606, C++3607** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2023.1 | **CERT\_C-STR30-a** **CERT\_C-STR30-b** | A string literal shall not be modified Do not modify string literals |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **489, 1776** | Partially supported |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2024a | [CERT C: Rule STR30-C](https://www.mathworks.com/help/bugfinder/ref/certcrulestr30c.html) | Checks for writing to const qualified object (rule fully covered) |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | [STD-004-CPP] | Do not concatenate user input into SQL queries |

| **Noncompliant Code** |
| --- |
| In this code, user input is directly concatenated into an SQL query, making it vulnerable to SQL injection attacks. |
| std::string email = getRequestString("email");  std::string query = "SELECT \* FROM Users WHERE Email = '" + email + "'"; |

| **Compliant Code** |
| --- |
| In this compliant code, a prepared statement is used to safely insert user input into the SQL query, preventing SQL injection attacks. |
| PreparedStatement\* stmt = conn->prepareStatement("SELECT \* FROM Users WHERE Email = ?"); stmt->setString(1, email); |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** 1.Validate Input Data; 8.Practice Defense in Depth; 10.Adopt a Secure Coding Standard |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | Medium | Medium | **P12** | **L1** |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| CppCheck | 2.15 | Cppcheck | Static analysis tool for C/C++ to detect bugs and issues. |
| Security Scanner | 5.6.7 | SecureScan | Scans applications for vulnerabilities, including SQL injections. |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [STD-005-CPP] | Set access permissions. |

| **Noncompliant Code** |
| --- |
| In this code, there is no permission check for reading the data, which can lead to unauthorized access |
| int\* ptr = new int[10];  for (int y = 0; y < 10; ++y) {  ptr[y] = y;  }  delete[] ptr; |

| **Compliant Code** |
| --- |
| Solution to implement permissions that prevent unauthorized access and ensure memory protection |
| int\* ptr = new int[10];  for (int y = 0; y < 10; ++y) {  if (hasPermission(ptr, y)) {  ptr[y] = y;  }  }  delete[] ptr; |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** 3.Architect and Design for Security Policies; 5.Default Deny; 8.Practice Defense in Depth |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Medium | Probable | High | **P4** | **L3** |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.1p0 | **(customization)** | CodeSonar's custom checking infrastructure allows users to implement checks such as the following.   * A check for all uses of fopen(). * A check for calls to open() with only two arguments.   A check for calls to open() where the third argument does not satisfy some specified requirement. |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2024.2 | **C5013** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **44 S** | Enhanced Enforcement |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | [STD-006-CPP] | Declare identifiers before using them |

| **Noncompliant Code** |
| --- |
| In this code, the identifier is used without a declaration, leading to an implicit declaration by the compiler |
| x = 1 |

| **Compliant Code** |
| --- |
| This code includes a type specifier to declare the identifier properly, to prevent the errors if the condition is not met. |
| Int x= 1 |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** 2.Head Compiler Warnings; 4.Keep it Simple; 9.Use Effective Quality Assurance Techniques |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Unlikely | Low | **P3** | **L3** |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Cppcheck Premium](https://wiki.sei.cmu.edu/confluence/display/c/Cppcheck+Premium) | 24.9.0 | **premium-cert-dcl31-c** | Partially Implemented Can detect implicit int |
| [ECLAIR](https://wiki.sei.cmu.edu/confluence/display/c/ECLAIR) | 1.2 | **CC2.DCL31** | Fully implemente |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2024.2 | **C0434, C2050, C2051, C3335** | Fully implemented |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2024.2 | **CWARN.IMPLICITINT** **MISRA.DECL.NO\_TYPE** **MISRA.FUNC.NOPROT.CALL** **RETVOID.IMPLICIT** | Fully implemented |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [STD-007-CPP] | Detect errors when converting a string to a number |

| **Noncompliant Code** |
| --- |
| In this code converts the string token stored in the buff to a signed integer value using the atoi() function. |
| #include <stdlib.h>    void func(const char \*buff) {  int si;    if (buff) {  si = atoi(buff);  } else {  /\* Handle error \*/  }  } |

| **Compliant Code** |
| --- |
| This noncompliant example uses the sscanf() function to convert a string token to an integer. The sscanf() function has the same limitations as atoi(). |
| #include <stdio.h>    void func(const char \*buff) {  int matches;  int si;    if (buff) {  matches = sscanf(buff, "%d", &si);  if (matches != 1) {  /\* Handle error \*/  }  } else {  /\* Handle error \*/  }  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** 1.Validate Input Data; 4.Keep It Simple; 9.Use effective Quality Assurance Techniques; 10.Adopt a Secure Coding Standard |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Medium | Unlikely | Medium | **P4** | **L3** |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-ERR62** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | cert-err34-c | Checked by clang-tidy; only identifies use of unsafe C Standard Library functions corresponding to ERR34-C |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 8.1p0 | **BADFUNC.ATOF** **BADFUNC.ATOI** **BADFUNC.ATOL** **BADFUNC.ATOLL** | Use of atof Use of atoi Use of atol Use of atoll |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Expressions** | [STD-008-CPP] | Do not access a variable through a pointer of an incompatible type |

| **Noncompliant Code** |
| --- |
| An object of type float is incremented through an int \*. The programmer can use the unit in the last place to get the next representable value for a floating-point type.  However, accessing an object through a pointer of an incompatible type is undefined behavior. |
| #include <stdio.h>    void f(void) {    if (sizeof(int) == sizeof(float)) {      float f = 0.0f;      int \*ip = (int \*)&f;      (\*ip)++;      printf("float is %f\n", f);    }  } |

| **Compliant Code** |
| --- |
| In this compliant solution, the standard C function nextafterf() is used to round toward the highest representable floating-point value. |
| #include <float.h>  #include <math.h>  #include <stdio.h>    void f(void) {    float f = 0.0f;    f = nextafterf(f, FLT\_MAX);    printf("float is %f\n", f);  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** 1. Validate Input Data; 2.Heed Compiler Warnings; 10.Adopt a Secure Coding Standard |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Medium | Unlikely | High | **P2** | **L3** |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2024.2 | **MISRA.CAST.FUNC\_PTR.2012** **MISRA.CAST.INCOMPLETE\_PTR\_TO\_ANY.2012** **MISRA.CAST.OBJ\_PTR\_TO\_NON\_INT.2012** **MISRA.CAST.OBJ\_PTR\_TO\_OBJ\_PTR.2012** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **94 S, 554 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2023.1 | **CERT\_C-EXP39-a** **CERT\_C-EXP39-b** **CERT\_C-EXP39-c** **CERT\_C-EXP39-d** **CERT\_C-EXP39-e** **CERT\_C-EXP39-f** | There shall be no implicit conversions from integral to floating type A cast should not be performed between a pointer to object type and a different pointer to object type Avoid accessing arrays and pointers out of bounds Avoid buffer overflow from tainted data due to defining incorrect format limits Avoid buffer read overflow from tainted data Avoid buffer write overflow from tainted data |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2024a | [CERT C: Rule EXP39-C](https://www.mathworks.com/help/bugfinder/ref/certcruleexp39c.html) | Checks for cast to pointer pointing to object of different type (rule partially covered) |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Preprocessor** | [STD-009-CPP] | Do Not Modify the Standard Namespaces |

| **Noncompliant Code** |
| --- |
| Using namespace std while declaring variables or functions directly in the std namespace is bad because it can cause problems in the code. |
| Namespace std {  Int x;  } |

| **Compliant Code** |
| --- |
| Instead, create your own namespace or use a non-standard namespace for your functions to avoid conflicts. |
| Namespace mystd {  Int x;  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** 2.Heed Compiler Warnings; 9.Use Effective Quality Assurance Techniques; 10. Adopt a Secure Coding Standard |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | Unlikely | Medium | **P6** | **L2** |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.2 | **C++3180, C++3181, C++3182** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2024.2 | **CERT.DCL.STD\_NS\_MODIFIED** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.1 | **CERT\_CPP-DCL58-a** | Do not modify the standard namespaces 'std' and 'posix' |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024a | [CERT C++: DCL58-CPP](https://www.mathworks.com/help/bugfinder/ref/certcdcl58cpp.html) | Checks for modification of standard namespaces (rule fully covered) |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Integers** | [STD-010-CPP] | Ensure that division and remainder operations do not result in divide-by-zero errors |

| **Noncompliant Code** |
| --- |
| In this code, division by zero results in an error, but there is no error handling to manage it. This can cause undefined behavior |
| #include <iostream>  int main() {  int x = 10;  int y = 0;  int z = x / y;  std::cout << z << std::endl;  return 0;  } |

| **Compliant Code** |
| --- |
| In this compliant code, a solution is provided to handle division by zero, ensuring the error is managed so the program runs smoothly. |
| #include <iostream>  int main() {  int x = 10;  int y = 0;  int z;  // Prevent divide-by-zero error  if (y != 0) {  z = x / y;  std::cout << z << std::endl;  } else {  std::cout << "Error: Division by zero is not allowed." << std::endl;  }  return 0;  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):**1. Validate Input Data; 2.Heed Compiler Warnings; 9.Use Effective Quality Assurance Techniques |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Likely | Medium | **P6** | **L2** |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [SonarQube](https://wiki.sei.cmu.edu/confluence/display/java/SonarQube) | 9.9 | [S3518](https://rules.sonarsource.com/java/RSPEC-3518) | [Zero should not be a possible denominator](https://rules.sonarsource.com/java/RSPEC-3518) |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/java/Coverity) | 7.5 | **DIVIDE\_BY\_ZERO** | Implemented |
| [Parasoft Jtest](https://wiki.sei.cmu.edu/confluence/display/java/Parasoft) | 2024.1 | **CERT.NUM02.ZERO** | Avoid division by zero |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/java/PVS-Studio) | 7.33 | [V6020](https://pvs-studio.com/en/docs/warnings/v6020/) |  |

### Defense-in-Depth Illustration

This illustration provides a visual representation of the defense-in-depth best practice of layered security.



## Project One

There are seven steps outlined below that align with the elements you will be graded on in the accompanying rubric. When you complete these steps, you will have finished the security policy.

### Revise the C/C++ Standards

You completed one of these tables for each of your standards in the Module Three milestone. In Project One, add revisions to improve the explanation and examples as needed. Add rows to accommodate additional examples of compliant and noncompliant code. Coding standards begin on the security policy.

### Risk Assessment

Complete this section on the coding standards tables. Enter high, medium, or low for each of the headers, then rate it overall using a scale from 1 to 5, 5 being the greatest threat. You will address each of the seven policy standards. Fill in the columns of severity, likelihood, remediation cost, priority, and level using the values provided in the appendix.

### Automated Detection

Complete this section of each table on the coding standards to show the tools that may be used to detect issues. Provide the tool name, version, checker, and description. List one or more tools that can automatically detect this issue and its version number, name of the rule or check (preferably with link), and any relevant comments or description—if any. This table ties to a specific C++ coding standard.

### Automation

Provide a written explanation using the image provided.



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.

### DevSecOps is the transition from DevOps, focusing on security from the beginning of the Software Development Life Cycle (SDLC), starting with the design phase and continuing through to the end. The diagram here explains how security is applied pre-production, including the design, assessment, planning, building, verification, and testing phases. During production, it encompasses transition and health checks, monitoring and detection, response, maintenance, and stabilization. Addressing security concerns from the beginning of the project is much easier and more cost-effective than trying to patch issues at the end of development or after the software goes live.

### Summary of Risk Assessments

Consolidate all risk assessments into one table including both coding and systems standards, ordered by standard number.

| Rule | Severity | Likelihood | Remediation Cost | Priority | Level |
| --- | --- | --- | --- | --- | --- |
| STD-004-CPP | High | Medium | Medium | **High (12)** | **1** |
| STD-001-CPP | Low | Probable | Low | **Medium (6)** | **2** |
| STD-002-CPP | High | Probable | High | **Medium (6)** | **2** |
| STD-009-CPP | High | Unlikely | Medium | **Medium (6)** | **2** |
| STD-010-CPP | Low | Likely | Medium | **Medium (6)** | **2** |
| STD-003-CPP | Low | Likely | Low | **Medium (9)** | **2** |
| STD-008-CPP | Medium | Unlikely | High | **Low (2)** | **3** |
| STD-006-CPP | Low | Unlikely | Low | **Low (3)** | **3** |
| STD-005-CPP | Medium | Probable | High | **Low (4)** | **3** |
| STD-007-CPP | Medium | Unlikely | Medium | **Low (4)** | **3** |

### Create Policies for Encryption and Triple A

Include all three types of encryption (in flight, at rest, and in use) and each of the three elements of the Triple-A framework using the tables provided***.***

* 1. Explain each type of encryption, how it is used, and why and when the policy applies.
  2. Explain each type of Triple-A framework strategy, how it is used, and why and when the policy applies.

Write policies for each and explain what it is, how it should be applied in practice, and why it should be used.

| 1. **Encryption** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Encryption at rest | Refers to the method of protecting data or backing it up to safeguard against negative outcomes. This method uses encryption tools such as AES (Advanced Encryption Standard), which ensures that stored data remains secure and unreadable without the right decryption keys. |
| Encryption in flight | Refers to the method of protecting data while it is being sent or transmitted over a network to safeguard against negative outcomes. In this method, two users, such as the sender and receiver, share the same encryption key to encrypt and decrypt the data. This can be protected through examples such as email encryption, DLP solutions, and solid network security features, such as firewalls and authentication. |
| Encryption in use | This refers to the method of protecting data while it is being used, such as when we input or update data, to safeguard sensitive information and minimize the risk of data theft. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | A process of ensuring that the right person is accessing a system or data. It is important because it protects sensitive information from unauthorized access and helps maintain the integrity and security of systems by verifying users' identities. |
| Authorization | A process that follows authentication, where a user is granted permissions to access specific resources or perform certain actions. This process helps prevent negative outcomes such as data theft and SQL injection attacks, while also determining the limitations of a user's access rights. |
| Accounting | A process of tracking and recording user interactions within a system, including details such as the time, user identity, and actions performed. This helps maintain a history of activities, allowing for the identification of past events and verification of legitimate user access. By implementing auditing, organizations can quickly detect and prevent unauthorized access. |

**\***Use this checklist for the Triple A to be sure you include these elements in your policy:

* User logins
* Changes to the database
* Addition of new users
* User level of access
* Files accessed by users

### Map the Principles

Map the principles to each of the standards, and provide a justification for the connection between the two. In the Module Three milestone, you added definitions for each of the 10 principles provided. Now it’s time to connect the standards to principles to show how they are supported by principles. You may have more than one principle for each standard, and the principles may be used more than once. Principles are numbered 1 through 10. You will list the number or numbers that apply to each standard, then explain how each of these principles supports the standard. This exercise demonstrates that you have based your security policy on widely accepted principles. Linking principles to standards is a best practice.

**NOTE:** Green Pace has already successfully implemented the following:

* Operating system logs
* Firewall logs
* Anti-malware logs

The only item you must complete beyond this point is the Policy Version History table.

## Audit Controls and Management

Every software development effort must be able to provide evidence of compliance for each software deployed into any Green Pace managed environment.

Evidence will include the following:

* Code compliance to standards
* Well-documented access-control strategies, with sampled evidence of compliance
* Well-documented data-control standards defining the expected security posture of data at rest, in flight, and in use
* Historical evidence of sustained practice (emails, logs, audits, meeting notes)

## Enforcement

The office of the chief information security officer (OCISO) will enforce awareness and compliance of this policy, producing reports for the risk management committee (RMC) to review monthly. Every system deployed in any environment operated by Green Pace is expected to be in compliance with this policy at all times.

Staff members, consultants, or employees found in violation of this policy will be subject to disciplinary action, up to and including termination.

## Exceptions Process

Any exception to the standards in this policy must be requested in writing with the following information:

* Business or technical rationale
* Risk impact analysis
* Risk mitigation analysis
* Plan to come into compliance
* Date for when the plan to come into compliance will be completed

Approval for any exception must be granted by chief information officer (CIO) and the chief information security officer (CISO) or their appointed delegates of officer level.

Exceptions will remain on file with the office of the CISO, which will administer and govern compliance.

## Distribution

This policy is to be distributed to all Green Pace IT staff annually. All IT staff will need to certify acceptance and awareness of this policy annually.

## Policy Change Control

This policy will be automatically reviewed annually, no later than 365 days from the last revision date. Further, it will be reviewed in response to regulatory or compliance changes, and on demand as determined by the OCISO.

## Policy Version History

| Version | Date | Description | Edited By | Approved By |
| --- | --- | --- | --- | --- |
| 1.0 | 08/05/2020 | Initial Template | David Buksbaum |  |
| 2.0 | 09/21/2024 | Filling Answers | Astrid French | Kaan Esendemir |
| 3.0 | 10/11/2024 | Final Answers | Astrid French | Kaan Esendemir |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

| Language | Acronym |
| --- | --- |
| C++ | CPP |
| C | CLG |
| Java | JAV |