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



# Green Pace Secure Development Policy

# Merlin Martinez

## 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 | By validating the input data, we ensure only the right information enters in the system. This means that any bad and malicious input data can be prevented. In addition, this means that we can verify the accuracy and the integrity of the data that’s being input in the system. So, then we can avoid any data corruption, unauthorized access and any system crash that can occur. In addition, by implementing the right validation for the user input like data length, validation range etc. in addition the book that we been reading in the lecture talks about SEI CERT C++ Rule 07. Input Output (FIO) which includes compliant solutions with fputs() or fprints(). |
| 1. Heed Compiler Warnings | The Heed warnings allow the developer to know that there is an error or issue within the code. This means as well that there might be a buffer overflow within the project. There are many ways to reduce risks, overflow etc. and some of them are using the right IDE and enabling the basic runtime checks (/RTCs) within the C/C++ options will allow compiler generated runtime checks for buffer overflows. |
| 1. Architect and Design for Security Policies | Ensuring that the right security is implemented in the software is necessary using the best practices for security policies, such as separating a system into sub systems with different authorization or privilege levels. This ensures that the right people have the right security clearances this will prevent any unauthorized people to access the system. |
| 1. Keep It Simple | Having a simple design allows you to reduce any risks in the codes and whenever the users are using the system, this as well means that it helps with the minimalization in the complexity of security required. This means proper spacing, proper variable names and useful comments always make sure to follow the KISS rule. Make it short and simple. |
| 1. Default Deny | Using the default deny this allows that when the access and access is permitted through the conditions of the protection scheme deems that permission is authorized. Either by firewall policy as well blocking all inbound and outbound traffic that has not been permitted expressly by the policy. |
| 1. Adhere to the Principle of Least Privilege | This will help to reduce a big risk, and this is because allowing the user to have the necessary levels to complete their task without giving them too much power, this means that. MYSQL is able to employ the principle of least privilege when several user accounts are each designated toward their own unique DB task and privileges. This reduces any risk of unauthorized users to have unnecessary privileges. |
| 1. Sanitize Data Sent to Other Systems | By sanitizing the data that is being sent to other systems this can help avoid any injection attacks or threats to the systems. This allows to check for potential issues prior reaching the system by using the right data sanitation that includes a parameterizing an SQL query to avoid SQL injection. Two main approach that can be taken into consideration would be black-listing and white-listing these two can help with the vulnerability of the system for example by using a deny by default approach would be to white-listing, this is a list that only valid inputs are accepted, allowing fail-safe behavior on unexpected inputs. This is a good, preferred method as we can’t always predict every invalid value to exclude via a black-list. |
| 1. Practice Defense in Depth | Ensuring using multiple layers would help with avoiding any risks. This means that if one layer was breached it has plenty of layers left that would secure the system. These layers of protection could be firewalls, intrusion detection, antivirus/antimalware, VPN or even Virtual Machine. We could add as well maintaining the servers continuously, using harden code and crypto library as well can be of big help to prevent any SQL injection. |
| 1. Use Effective Quality Assurance Techniques | There are many ways to ensure the code is reliable, secure and that can avoid unnecessary bugs and is by doing the necessary quality assurances techniques like penetration testing, vulnerability scanning, code reviews, security audits, and vigorous testing etc. can help with quality assurance techniques. |
| 1. Adopt a Secure Coding Standard | Ensuring you are applying the right methos to the code can help a lot as well as a secure coding standard is a must in any project this helps secure any standard that Is being adhered to SEI CERT C++ Coding Standard. This not only applies to C++ or SQL but as well any other coding languages. |

### 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 cast to an out-of-range enumeration value |

[**https://wiki.sei.cmu.edu/confluence/display/cplusplus/INT50-CPP.+Do+not+cast+to+an+out-of-range+enumeration+value**](https://wiki.sei.cmu.edu/confluence/display/cplusplus/INT50-CPP.+Do+not+cast+to+an+out-of-range+enumeration+value)

| **Noncompliant Code** |
| --- |
| The noncompliant code example attempts to check whether a given value is within the range of acceptable enumeration values |
| **enum** EnumType {    First,    Second,    Third  };    **void** f(**int** intVar) {    EnumType enumVar = **static\_cast**<EnumType>(intVar);    **if** (enumVar < First || enumVar > Third) {      // Handle error    }  } |

| **Compliant Code** |
| --- |
| The solution checks that the value is able to be represented by the enumeration type before performing the conversion to guarantee the conversion does not result in an unspecified value. |
| **enum EnumType {**  **First,**  **Second,**  **Third**  **};**    **void f(int intVar) {**  **if (intVar < First || intVar > Third) {**  **// Handle error**  **}**  **EnumType enumVar = static\_cast<EnumType>(intVar);**  **}**    **static** **void** dealloc(**void** \*ptr) noexcept {  **if** (h) {        (**void**)::HeapFree(h, 0, ptr);      }    }  };    **HANDLE** HeapAllocator::h = nullptr;  **bool** HeapAllocator::init = **false**;    **void** \*operator **new**(std::**size\_t** size) noexcept(**false**) {  **return** HeapAllocator::alloc(size);  }    **void** operator **delete**(**void** \*ptr) noexcept {  **return** HeapAllocator::dealloc(ptr); |

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

| **Principles(s):** [Name the principle and explain how it maps to this standard.] |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Polyspace Bug Finder | R2024a | CERT C++: INT50-CPP | Checks for casting to out-of-range enumeration value (rule fully covered) |
| RuleChecker | 22.10 | cast-integer-to-enum | Partially checked |
| PVS-Studio | 7.31 | V1016 | [Insert text.] |
| Helix QAC | 2024.1 | C++3013 | [Insert text.] |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | [STD-002-CPP] | Do not read uninitialized memory |

[**https://wiki.sei.cmu.edu/confluence/display/cplusplus/EXP53-CPP.+Do+not+read+uninitialized+memory**](https://wiki.sei.cmu.edu/confluence/display/cplusplus/EXP53-CPP.+Do+not+read+uninitialized+memory)

| **Noncompliant Code** |
| --- |
| an uninitialized local variable is evaluated as part of an expression to print its value, resulting in undefined behavior. |
| #include <iostream>    **void** f() {  **int** i;    std::cout << i;  } |

| **Compliant Code** |
| --- |
| You need to initialize object prior to printing its value |
| #include <iostream>    **void** f() {  **int** i = 0;    std::cout << i;  } |

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

| **Principles(s):** [Name the principle and explain how it maps to this standard.] |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astrée | 22.10 | uninitialized-read | Partially checked |
| Clang | 3.9 | -Wuninitialized clang-analyzer-core.UndefinedBinaryOperatorResult | Does not catch all instances of this rule, such as uninitialized values read from heap-allocated memory. |
| PVS-Studio | 7.31 | V546, V573, V614, V670, V679, V730, V788, V1007, V1050 | [Insert text.] |
| RuleChecker | 22.10 | uninitialized-read | Partially checked |

#### Coding Standard 3

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

[**https://wiki.sei.cmu.edu/confluence/display/c/STR30-C.+Do+not+attempt+to+modify+string+literals**](https://wiki.sei.cmu.edu/confluence/display/c/STR30-C.+Do+not+attempt+to+modify+string+literals)

| **Noncompliant Code** |
| --- |
| This noncompliant code example shows how the char pointer str is initialized to the address of a string literal. Attempting to modify the string literal is undefined behavior: |
| **char** \*str  = "string literal";  str[0] = 'S'; |

| **Compliant Code** |
| --- |
| This code is able to create a copy of the string literal in the space allocated to the character array str. The string stored in str can be modified safely. |
| **char** str[] = "string literal";  str[0] = 'S'; |

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

| **Principles(s):** [Name the principle and explain how it maps to this standard.] |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Axivion Bauhaus Suite | 7.2.0 | CertC-STR30 | Fully implemented |
| Coverity | 2017.07 | PW | Deprecates conversion from a string literal to "char \*" |
| PVS-Studio | 7.31 | V675 | [Insert text.] |
| Splint | 3.1.1 | [Insert text.] | [Insert text.] |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | [STD-004-CPP] | Prevent SQL injection |

[**https://wiki.sei.cmu.edu/confluence/display/java/IDS00-J.+Prevent+SQL+injection**](https://wiki.sei.cmu.edu/confluence/display/java/IDS00-J.+Prevent+SQL+injection)

| **Noncompliant Code** |
| --- |
| This noncompliant code example shows how the JDBC code is used to authenticate a user to a system. The password is passed as a char array, the database connection is created, and then the passwords are hashed. |
| **import java.sql.Connection;**  **import java.sql.DriverManager;**  **import java.sql.ResultSet;**  **import java.sql.SQLException;**  **import java.sql.Statement;**    **class Login {**  **public Connection getConnection() throws SQLException {**  **DriverManager.registerDriver(new**  **com.microsoft.sqlserver.jdbc.SQLServerDriver());**  **String dbConnection =**  **PropertyManager.getProperty("db.connection");**  **// Can hold some value like**  **// "jdbc:microsoft:sqlserver://<HOST>:1433,<UID>,<PWD>"**  **return DriverManager.getConnection(dbConnection);**  **}**    **String hashPassword(char[] password) {**  **// Create hash of password**  **}**    **public void doPrivilegedAction(**  **String username, char[] password**  **) throws SQLException {**  **Connection connection = getConnection();**  **if (connection == null) {**  **// Handle error**  **}**  **try {**  **String pwd = hashPassword(password);**  **String sqlString = "select \* from db\_user where username=" +**  **username + " and password =" + pwd;**  **PreparedStatement stmt = connection.prepareStatement(sqlString);**    **ResultSet rs = stmt.executeQuery();**  **if (!rs.next()) {**  **throw new SecurityException("User name or password incorrect");**  **}**    **// Authenticated; proceed**  **} finally {**  **try {**  **connection.close();**  **} catch (SQLException x) {**  **// Forward to handler**  **}**  **}**  **}**  **}** |

| **Compliant Code** |
| --- |
| This compliant solution uses a parametric query with a ? character as a placeholder for the argument. This code also validates the length of the username argument, preventing an attacker from submitting an arbitrarily long user name. |
| **public** **void** doPrivilegedAction(    String username, **char**[] password  ) **throws** SQLException {    Connection connection = getConnection();  **if** (connection == **null**) {      // Handle error    }  **try** {      String pwd = hashPassword(password);        // Validate username length  **if** (username.length() > 8) {        // Handle error      }        String sqlString =        "select \* from db\_user where username=? and password=?";      PreparedStatement stmt = connection.prepareStatement(sqlString);      stmt.setString(1, username);      stmt.setString(2, pwd);      ResultSet rs = stmt.executeQuery();  **if** (!rs.next()) {  **throw** **new** SecurityException("User name or password incorrect");      }        // Authenticated; proceed    } **finally** {  **try** {        connection.close();      } **catch** (SQLException x) {        // Forward to handler      }    }  } |

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

| **Principles(s):** [Name the principle and explain how it maps to this standard.] |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | Likely | Medium | P18 | L1 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| CodeSonar | 8.1p0 | JAVA.IO.INJ.SQL | SQL Injection (Java) |
| Coverity | 7.5 | SQLI  FB.SQL\_PREPARED\_STATEMENT\_GENERATED\_  FB.SQL\_NONCONSTANT\_STRING\_PASSED\_TO\_EXECUTE | Implemented |
| SpotBugs | 4.6.0 | SQL\_NONCONSTANT\_STRING\_PASSED\_TO\_EXECUTE  SQL\_PREPARED\_STATEMENT\_GENERATED\_FROM\_NONCONSTANT\_STRING | Implemented |
| Klocwork | 2024.1 | **SV.DATA.DB** **SV.SQL** **SV.SQL.DBSOURCE** | Implemented |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [STD-005-CPP] | Do not access freed memory |

[**https://wiki.sei.cmu.edu/confluence/display/cplusplus/MEM50-CPP.+Do+not+access+freed+memory**](https://wiki.sei.cmu.edu/confluence/display/cplusplus/MEM50-CPP.+Do+not+access+freed+memory)

| **Noncompliant Code** |
| --- |
| In this noncompliant code example, s is dereferenced after it has been deallocated. |
| #include <new>    **struct** S {  **void** f();  };    **void** g() noexcept(**false**) {    S \*s = **new** S;    // ...  **delete** s;    // ...    s->f();  } |

| **Compliant Code** |
| --- |
| In this compliant solution, the dynamically allocated memory is not deallocated until it is no longer required. |
| #include <new>    **struct** S {  **void** f();  };    **void** g() noexcept(**false**) {    S \*s = **new** S;    // ...    s->f();  **delete** s;  } |

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

| **Principles(s):** [Name the principle and explain how it maps to this standard.] |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | Likely | Medium | P18 | L1 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| CodeSonar | 8.1p0 | ALLOC.UAF | Use after free |
| Astrée | 22.10 | dangling\_pointer\_use | [Insert text.] |
| Helix QAC | 2024.1 | C++4303, C++4304 | [Insert text.] |
| Parasoft Insure++ | [Insert text.] | [Insert text.] | Runtime detection |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | [STD-006-CLG] | Use a static assertion to test the value of a constant expression |

[**https://wiki.sei.cmu.edu/confluence/display/c/DCL03-C.+Use+a+static+assertion+to+test+the+value+of+a+constant+expression**](https://wiki.sei.cmu.edu/confluence/display/c/DCL03-C.+Use+a+static+assertion+to+test+the+value+of+a+constant+expression)

| **Noncompliant Code** |
| --- |
| This noncompliant code uses the assert() macro to assert a property concerning a memory-mapped structure that is essential for the code to behave correctly: |
| #include <assert.h>    **struct** timer {    unsigned **char** MODE;    unsigned **int** DATA;    unsigned **int** COUNT;  };    **int** func(**void**) {  **assert**(**sizeof**(**struct** timer) == **sizeof**(unsigned **char**) + **sizeof**(unsigned **int**) + **sizeof**(unsigned **int**));  } |

| **Compliant Code** |
| --- |
| For assertions involving only constant expressions, a preprocessor conditional statement may be used, as in this compliant solution: |
| **struct** timer {    unsigned **char** MODE;    unsigned **int** DATA;    unsigned **int** COUNT;  };    #if (sizeof(struct timer) != (sizeof(unsigned char) + sizeof(unsigned int) + sizeof(unsigned int)))    #error "Structure must not have any padding"  #endif |

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

| **Principles(s):** [Name the principle and explain how it maps to this standard.] |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Unlikely | High | P1 | L3 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| LDRA tool suite | 9.7.1 | 44 S | Fully implemented |
| ECLAIR | 1.2 | **CC2.DCL03** | Fully implemented |
| CodeSonar | 8.1p0 | **(customization)** | Users can implement a custom check that reports uses of the assert() macro |
| Clang | 3.9 | misc-static-assert | Checked by clang-tidy |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [STD-007-CPP] | Handle all exceptions. |

[**https://wiki.sei.cmu.edu/confluence/display/cplusplus/ERR51-CPP.+Handle+all+exceptions**](https://wiki.sei.cmu.edu/confluence/display/cplusplus/ERR51-CPP.+Handle+all+exceptions)

| **Noncompliant Code** |
| --- |
| In this noncompliant code example, neither f() nor main() catch exceptions thrown by throwing\_func(). Because no matching handler can be found for the exception thrown, std::terminate() is called. |
| **void** throwing\_func() noexcept(**false**);    **void** f() {    throwing\_func();  }    **int** main() {    f();  } |

| **Compliant Code** |
| --- |
| In this compliant solution, the main entry point handles all exceptions, which ensures that the stack is unwound up to the main() function and allows for graceful management of external resources. |
| **void** throwing\_func() noexcept(**false**);    **void** f() {    throwing\_func();  }    **int** main() {  **try** {      f();    } **catch** (...) {      // Handle error    }  } |

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

| **Principles(s):** [Name the principle and explain how it maps to this standard.] |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| RuleChecker | 22.10 | **main-function-catch-all early-catch-all** | Partially checked |
| Helix QAC | 2024.1 | C++4035, C++4036, C++4037 | [Insert text.] |
| LDRA tool suite | 9.7.1 | 527 S | Partially implemented |
| RuleChecker | 22.10 | **main-function-catch-all early-catch-all** | Partially checked |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Exceptions | [STD-008-CPP] | Detect errors when converting a string to a number. |

[**https://wiki.sei.cmu.edu/confluence/display/cplusplus/ERR62-CPP.+Detect+errors+when+converting+a+string+to+a+number**](https://wiki.sei.cmu.edu/confluence/display/cplusplus/ERR62-CPP.+Detect+errors+when+converting+a+string+to+a+number)

| **Noncompliant Code** |
| --- |
| In this noncompliant code example, multiple numeric values are converted from the standard input stream. However, if the text received from the standard input stream cannot be converted into a numeric value that can be represented by an int, the resulting value stored into the variables i and j may be unexpected. |
| #include <iostream>    **void** f() {  **int** i, j;    std::cin >> i >> j;    // ...  } |

| **Compliant Code** |
| --- |
| In this compliant solution, exceptions are enabled so that any conversion failure results in an exception being thrown. |
| #include <iostream>    **void** f() {  **int** i, j;      std::cin.exceptions(std::istream::failbit | std::istream::badbit);  **try** {      std::cin >> i >> j;      // ...    } **catch** (std::istream::failure &E) {      // Handle error    }  } |

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

| **Principles(s):** [Name the principle and explain how it maps to this standard.] |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Clang | 3.9 | cert-err34-c | Checked by clang-tidy; only identifies use of unsafe C Standard Library functions corresponding to ERR34-C |
| Parasoft C/C++test | 2023.1 | **CERT\_CPP-ERR62-a** | the library functions atof, atoi and atol from library stdlib.h shall not be used |
| Polyspace Bug Finder | R2024a | CERT C++: ERR62-CPP | Checks for unvalidated string-to-number conversion (rule fully covered) |
| 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 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [STD-009-CPP] | Do not abruptly terminate the program |

[**https://wiki.sei.cmu.edu/confluence/display/cplusplus/ERR50-CPP.+Do+not+abruptly+terminate+the+program**](https://wiki.sei.cmu.edu/confluence/display/cplusplus/ERR50-CPP.+Do+not+abruptly+terminate+the+program)

| **Noncompliant Code** |
| --- |
| the call for f(), which was registered as an exit handler with std::at\_exit(), may result in a call to std::terminate() because throwing\_func() may throw an exception. |
| #include <cstdlib>    **void** throwing\_func() noexcept(**false**);    **void** f() { // Not invoked by the program except as an exit handler.    throwing\_func();  }    **int** main() {  **if** (0 != std::**atexit**(f)) {      // Handle error    }    // ...  } |

| **Compliant Code** |
| --- |
| The solution would be, f() handles all exceptions thrown by throwing\_func() and does not rethrow. |
| #include <cstdlib>    **void** throwing\_func() noexcept(**false**);    **void** f() { // Not invoked by the program except as an exit handler.  **try** {      throwing\_func();    } **catch** (...) {      // Handle error    }  }    **int** main() {  **if** (0 != std::**atexit**(f)) {      // Handle error    }    // ...  } |

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

| **Principles(s):** [Name the principle and explain how it maps to this standard.] |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Probable | MEDIUM | P4 | L3 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| LDRA tool suite | 9.7.1 | **122 S** | Enhanced Enforcement |
| CodeSonar | 8.1p0 | B**ADFUNC.ABORT BADFUNC.EXIT** | Use of abort Use of exit |
| Polyspace Bug Finder | R2024a | CERT C++: ERR50-CPP | Checks for implicit call to terminate() function (rule partially covered) |
| RuleChecker | 22.10 | stdlib-use | Partially checked |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Expressions | [STD-010-CPP] | Do not access an object outside of its lifetime |

[**https://wiki.sei.cmu.edu/confluence/display/cplusplus/EXP54-CPP.+Do+not+access+an+object+outside+of+its+lifetime**](https://wiki.sei.cmu.edu/confluence/display/cplusplus/EXP54-CPP.+Do+not+access+an+object+outside+of+its+lifetime)

| **Noncompliant Code** |
| --- |
| on this this noncompliant code example we see how a pointer to an object is used to call a non-static member function of the object prior to the beginning of the pointer's lifetime, resulting in undefined behavior. |
| **struct** S {  **void** mem\_fn();  };    **void** f() {    S \*s;    s->mem\_fn();  } |

| **Compliant Code** |
| --- |
| The storage is obtained for the pointer prior to calling S::mem\_fn(). |
| **struct** S {  **void** mem\_fn();  };    **void** f() {    S \*s = **new** S;    s->mem\_fn();  **delete** s;  } |

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

| **Principles(s):** [Name the principle and explain how it maps to this standard.] |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| HIGH | PROBABLE | HIGH | P6 | L2 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astrée | 22.10 | return-reference-local dangling\_pointer\_use | Partially checked |
| Clang | 3.9 | -Wdangling-initializer-list | Catches some lifetime issues related to incorrect use of std::initializer\_list<> |
| CodeSonar | 8.1P0 | IO.UAC  ALLOC.UAF | Use after close  Use after free |
| Polyspace Bug Finder | R2024a | CERT C++: EXP54-CPP | Checks for:   * Non-initialized variable or pointer * Use of previously freed pointer * Pointer or reference to stack variable leaving scope * Accessing object with temporary lifetime   Rule partially covered. |

### 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.

The way I would modify the existing DevOps process infrastructure is by integrating the necessary security measures into each step of the DevOps toolchain this means that I would incorporate the necessary security measures in the DevOps pipeline so I can ensure the necessary effective execution of security standards throughout the software development lifecycle.

I addition I would add that when the program gets to be in production continuous testing should be priority as well integrity checks and defense in depths to prevent any type of hacks. Momitor and detection, network monitoring, penetration testing, network monitoring and performance logs are some of the methods of continuous threat detection, these methods should be done early and often to prevent any issues.

### 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-001-CPP | High | likely | High | P9 | 2 |
| STD-003-CPP | High | Likely | Medium | P18 | 1 |
| STD-005-CPP | High | Likely | Medium | P18 | 1 |
| STD-008-CPP | Medium | Unlikely | Medium | P4 | 3 |
| STD-007-CPP | Low | Probable | Medium | P4 | 3 |
| STD-006-CLG | Low | Unlikely | High | P1 | 3 |
| STD-002-CPP | High | Likely | High | P9 | 2 |
| STD-010-CPP | High | Probable | High | P6 | 2 |
| STD-004-CPP | High | Probable | Medium | P12 | 1 |
| STD-009-CPP | High | Probable | High | P6 | 2 |

### 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 | Store data should be encrypted so it can be protected. As well database encryption by using MySQL and having backup data. |
| Encryption in flight | This means that any data that is moving should be protected. Like for example if data is being share between networks, cloud, hard drive and etc. utilizing secure methods like data leak prevention mechanisms built into cloud services, SSH with expiration date on the link and etc. |
| Encryption in use | This means that the data that is being use at the time is being protected, in order to protect this type of data you need to ensure that the data control and protection exists prior to use. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | By authenticating the user, using the necessary valid credentials, there are many ways to authenticate the users like for example using static passwords, one-time passwords, certifications, biometric, tokens etc., are some ways to use authentication and to ensure the system is not getting hacked into. |
| Authorization | The authorization shows who has the necessary credentials and privileges. This is really important because it shows who and what kind of access the user has to information this help to lower vulnerabilities this help protect sensible data. |
| Accounting | This monitor logs users activities like, for example, timestamps, , accessed resources, and data transfer information. This is a good resource because helps in creating a trail in the user activity, and also for the purposes of forensic analysis and investigation. |

**\***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 | 07/19/2024 | Module3 | Merlin Martinez |  |
| 1.2 | 07/19/2024 | Module6 | Merlin Martinez |  |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

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