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



# Green Pace Secure Development Policy

# Stephen Mathew

## 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 | Validate the data being inputted into the program. Always verify external data being brought into the program to make sure the external data is not corrupt or damaged to prevent damage to the program or the operating system. |
| 1. Heed Compiler Warnings | Follow the instructions of warning messages when compiling the code to prevent any open doors which can allow hackers to enter a program. Modify the program’s code to satisfy the warnings found during a debug of the program. |
| 1. Architect and Design for Security Policies | Create a software architecture and design your software to implement and enforce security policies. |
| 1. Keep It Simple | Keep the program code design short and simple to prevent any errors or warning, which are usually seen within complex design of coding for a program. |
| 1. Default Deny | Instead of removing data, make data only permissible to those who are allowed access to the data and deny access permission to everyone until they can prove they have the security keys to gain access to the data. |
| 1. Adhere to the Principle of Least Privilege | Every process should execute with the least amount of privileges necessary to complete the job. Any elevated permission should only be accessed for the least amount of time required to complete the task. |
| 1. Sanitize Data Sent to Other Systems | Sanitize all data passed to subsystems such as command shells, relational databases, and commercial off-the-shelf components |
| 1. Practice Defense in Depth | Use multi-layer defenses when managing safety of the computer. The multi-layer defense allows the system and/or device to stay protected even if one of the layer defenses gets compromised. |
| 1. Use Effective Quality Assurance Techniques | Good quality assurance techniques can be effective in identifying and eliminating vulnerabilities. Fuzz testing, penetration testing, and source code audits should all be used as part of an effective quality assurance program |
| 1. Adopt a Secure Coding Standard | Use a secured code standard whether you apply or develop the standard for your choice of language and platform. |

### 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-CCP] | Avoid casting an out-of-range enumeration value |

| **Noncompliant Code** |
| --- |
| [Attempts to check whether a given value is within the range of acceptable enumeration values. However, it  is doing this after casting to the enumeration type, which may not be able to represent the given integer  value. On a two's complement system, the valid range of values that can be represented by EnumType are  [0..3], so if a value outside of that range were passed to f(), the cast to EnumType would result in an  unspecified value, and using that value within the if statement results in unspecified behavior. |
| [enum EnumType {  First,  Second,  Third  };  void f(int intVar) {  EnumType enumVar = static\_cast<EnumType>(intVar);  If (enumVar < First || enumVar > Third) {  // Handle error  }  }  ] |

| **Compliant Code** |
| --- |
| Checks that the value can be represented by the enumeration type before performing the conversion to  ensure 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);  } |

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

| **Principles(s):** 1 – Validate Input Data: Casting to an unspecified result does not allow the input data to validated to give or revoke access to a system. |
| --- |

**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) | 7.6.0 | CertC++-INT50 |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.2 | C++3013 |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.2 | CERT\_CPP-INT50-a | An expression with enum underlying type shall only have values corresponding to the enumerators of the enumeration |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.22 | V1016 |  |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | [STD-002-CPP] | Use valid references, pointers, and iterators to reference elements of a container. |

| **Noncompliant Code** |
| --- |
| *pos* is invalidated after the first call to insert(), and loop iterations have undefined behavior. |
| #include <deque>  void f(const double \*items, std::size\_t count) {  std::deque<double> d;  auto pos = d.begin();  for (std::size\_t i = 0; I < count; ++i, ++pos) {  d.insert(pos, items[i] + 41.0);  }  } |

| **Compliant Code** |
| --- |
| pos is assigned a valid iterator on each insertion, preventing undefined behavior |
| #include <deque>  void f(const double \*items, std::size\_t count) {  std::deque<double> d;  auto pos = d.begin();  for (std::size\_t i = 0; i < count; ++i, ++pos) {  pos = d.insert(pos, items[i] + 41.0);  }  } |

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

| **Principles(s):** 5 – Default Deny: First, denies access to data inside a container unless valid references or pointers are related to the data inside the container. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 24.04 | overflow\_upon\_dereference |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.2 | C++4746, C++4747, C++4748, C++4749 |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.2 | CERT\_CPP-CTR51-a | Don’t modify the container while iterating over it |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.22 | V783 |  |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | [STD-003-CPP] | Ensure that storage for strings has enough space for character data and the null terminator |

| **Noncompliant Code** |
| --- |
| The input is unbounded, and the code can lead to a buffer overflow. |
| #include <iostream>  void f() {  char buf[12];  std::cin >> buf;  } |

| **Compliant Code** |
| --- |
| Ensuring that data is not cut and for protection against buffer overflows is to use std::string instead of  a bounded array. |
| #include <iostream>  #include <string>  void f() {  std::string input;  std::string stringOne, stringTwo;  std::cin >> stringOne >> stringTwo;  } |

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

| **Principles(s):** 1 – Architect and Design for Security Policies: Prevents any database architect issues by making sure sufficient storage is available |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.4 | MISC.MEM.NTERM  LANG.MEM.BO  LANG.MEM.TO | No space for null terminator  Buffer overrun  Type overrun |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.2 | C++2835, C++2836, C++2839, C++5216 |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2024.3 | NNTS.MIGHT  NNTS.TAINTED |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.9.3 | 489 S, 66 X, 70 X, 71 X | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.2 | CERT\_CPP-STR50-b  CERT\_CPP-STR50-c  CERT\_CPP-STR50-e  CERT\_CPP-STR50-f  CERT\_CPP-STR50-g | Avoid overflow due to reading a not zero terminate string  Avoid overflow when writing to a buffer  Prevent buffer overflows from tainted data  Avoid buffer write overflow from tainted data  Do not use the ‘char’ buffer to store input from ‘std::cin’ |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024a | CERT C++:STR50-CPP | Check for:   * Use of dangerous standard functions * Missing null in string array * Buffer overflow from incorrect string format specifier * Destination buffer overflow in string manipulation   Rule partially covered |
| [SonarQube C/C++ Plugin](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046388) | 4.11 | S3519 |  |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | [STD-004-CPP] | Do not store an already-owned pointer value in an unrelated smart pointer. |

| **Noncompliant Code** |
| --- |
| Two unrelated smart pointers are built from the same underlying pointer value. When the local,  variable p2 is destroyed, it deletes the pointer value it’s associated with. Then, when the local  variable p1 is destroyed, it deletes the same pointer value, which results in a double-free vulnerability. |
| #include <memory>  void f() {  int \*i = new int;  std::shared\_ptr<int> p1(i);  std::shared\_ptr<int> p2(i);  } |

| **Compliant Code** |
| --- |
| The std::shared\_ptr objects are related to one another through copy construction. When the local, automatic variable p2 is destroyed, the use count for the shared pointer value is decreased but still not zero. Then, when the local, automatic variable p1 is destroyed, the use count for the shared pointer value is decreased to zero, and the managed pointer is destroyed. This solution also calls std::make\_shared() instead of choosing a raw pointer and storing its value in a local variable. |
| #include <memory>  void f() {  std::shared\_ptr<int> p1 = std::make\_shared<int>();  std::shared\_ptr<int> p2(p1);  } |

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

| **Principles(s):** 3 – Architect and Design for Security Policies: Prevents issues from any stored variables being replaced and destroying later on in the code. This issue also relinquishes ownership of the managed pointer value. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 24.04 | dangling\_pointer\_use |  |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.6 | CertC++-MEM56 |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.2 | C++4721, C++4722, C++4723 |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.2 | CERT\_CPP-MEM56-a | Do not store an already-owned pointer value in an unrelated smart pointer |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024a | CERT C++:MEM56-CPP | Checks for use of already-owned pointers (rule fully covered) |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.22 | V1006 |  |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [STD-005-CPP] | Dynamically allocated resources are unallocated appropriately. |

| **Noncompliant Code** |
| --- |
| The local variable space is passed as the expression to the new operator. Then, resulting pointer of  that call is passed to ::operator delete(), which results in undefined behavior due to ::operator  delete() attempting to free memory that was not returned by ::operator new(). |
| #include <iostream>  struct S {  S() { std::cout << "S::S()" << std::endl; }  ~S() { std::cout << "S::~S()" << std::endl; }  };  void f() {  alignas(struct S) char space[sizeof(struct S)];  S \*s1 = new (&space) S;    //  delete s1;  } |

| **Compliant Code** |
| --- |
| Removes the call to ::operator delete(), instead calls s1's destructor. This is one of the few times  when using a destructor makes sense. |
| [#include <iostream>  struct S {  S() { std::cout << "S::S()" << std::endl; }  ~S() { std::cout << "S::~S()" << std::endl; }  };  void f() {  alignas(struct S) char space[sizeof(struct S)];  S \*s1 = new (&space) S;  //  s1->~S();  } |

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

| **Principles(s):** 8 – Practice Defense in Depth: Can cause a null-pointer or pointer behavior to be undefined and give results previous array new-expression to a command operator (new). |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 24.04 | **invalid\_dynamic\_memory\_allocation dangling\_pointer\_use** |  |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.6 | **CertC++-MEM51** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 16.0 | clang-analyzer-cplusplus.NewDeleteLeaks -Wmismatched-new-delete clang-analyzer-unix.MismatchedDeallocator | Checked by clang-tidy, but does not catch all violations of this rule |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.4 | **ALLOC.FNH ALLOC.DF ALLOC.TM** | Free non-heap variable Double free Type mismatch |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.2 | **C++2110, C++2111, C++2112, C++2113, C++2118, C++3337, C++3339, C++4262, C++4263, C++4264** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2024.3 | [CL.FFM.ASSIGN](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [CL.FFM.COPY](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [CL.FMM](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [FMM.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [FMM.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [FNH.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [FNH.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [FUM.GEN.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [FUM.GEN.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [UNINIT.CTOR.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [UNINIT.CTOR.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [UNINIT.HEAP.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [UNINIT.HEAP.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [UNINIT.STACK.ARRAY.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [UNINIT.STACK.ARRAY.PARTIAL.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [UNINIT.STACK.ARRAY.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [UNINIT.STACK.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [UNINIT.STACK.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.9.3 | **232 S, 236 S, 239 S, 407 S, 469 S, 470 S, 483 S, 484 S, 485 S, 64 D, 112 D** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.2 | **CERT\_CPP-MEM51-a CERT\_CPP-MEM51-b CERT\_CPP-MEM51-c CERT\_CPP-MEM51-d** | Use the same form in corresponding calls to new/malloc and delete/free Always provide empty brackets ([]) for delete when deallocating arrays Both copy constructor and copy assignment operator should be declared for classes with a nontrivial destructor Properly deallocate dynamically allocated resources |
| [Parasoft Insure++](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | R2023a |  | Runtime detection |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024a | [CERT C++: MEM51-CPP](https://www.mathworks.com/help/bugfinder/ref/certcmem51cpp.html) | Checks for:   * Invalid deletion of pointer * Invalid free of pointer * Deallocation of previously deallocated pointer   Rule partially covered. |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.22 | [**V515**](https://pvs-studio.com/en/docs/warnings/v515/)**,**[**V554**](https://pvs-studio.com/en/docs/warnings/v554/)**,**[**V611**](https://pvs-studio.com/en/docs/warnings/v611/)**,**[**V701**](https://pvs-studio.com/en/docs/warnings/v701/)**,**[**V748**](https://pvs-studio.com/en/docs/warnings/v748/)**,**[**V773**](https://pvs-studio.com/en/docs/warnings/v773/)**,**  [V1066](https://pvs-studio.com/en/docs/warnings/v1066/) |  |
| [SonarQube C/C++ Plugin](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046388) | 4.11 | S1232 |  |

#### Coding Standard 6

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

| **Noncompliant Code** |
| --- |
| [noncompliant code uses the assert() macro to assert a property concerning a memory-mapped structure that  is vital for the code to respond correctly. Using the runtime assertion is better than nothing, it needs to be  placed in a function and executed. |
| #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** |
| --- |
| Static assertions allow incorrect assumptions to be assessed at compiling instead of resulting in a  malfunction or runtime error. |
| #include <assert.h>  struct timer {  unsigned char MODE;  unsigned int DATA;  unsigned int COUNT;  };  static\_assert(sizeof(struct timer) == sizeof(unsigned char)  + sizeof(unsigned int) + sizeof(unsigned int),  "Structure must not have any padding"); |

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

| **Principles(s):** 9 – Use Effective Quality Assurance Techniques: Allows good programs and techniques to be used to test sections of code for any issues. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.6 | **CertC-DCL03** |  |
| Clang | 16.0 | misc-static-assert | Checked by clang-tidy |
| CodeSonar | 7.4 | **(customization)** | Users can implement a custom check that reports uses of the assert() macro |
| Compass/ROSE | 0.11.0 |  | Could detect violations of this rule merely by looking for calls to assert(), and if it can evaluate the assertion (due to all values being known at compile time), then the code should use static-assert instead; this assumes ROSE can recognize macro invocation |
| ECLAIR | 3.15 | **CC2.DCL03** | Fully implemented |
| LDRA tool suite | 9.9.3 | **44 S** | Fully implemented |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [STD-007-CPP] | Guarantee exception safety |

| **Noncompliant Code** |
| --- |
| The invariants of the class are that the array member is a valid pointer (could be NULL) and that  the nElems member stores the number of elements in the array pointed to by array. The function  removes the array and assigns the element counter, nElems, before allocating a new block of memory for the  copy. |
| #include <cstring>  class IntArray {  int \*array;  std::size\_t nElems;  public:  // .  ~IntArray() {  delete[] array;  }  IntArray(const IntArray& that); // nontrivial copy constructor  IntArray& operator=(const IntArray &rhs) {  if (this != &rhs) {  delete[] array;  array = nullptr;  nElems = rhs.nElems;  if (nElems) {  array = new int[nElems];  std::memcpy(array, rhs.array, nElems \* sizeof(\*array));  }  }  return \*this;  }  // .  }; |

| **Compliant Code** |
| --- |
| The copy assignment operator provides the strong exception safety guarantee. The function allocates new  storage for the copy before changing the state of the object. |
| #include <cstring>  class IntArray {  int \*array;  std::size\_t nElems;  public:  // .  ~IntArray() {  delete[] array;  }  IntArray(const IntArray& that); // nontrivial copy constructor  IntArray& operator=(const IntArray &rhs) {  int \*tmp = nullptr;  if (rhs.nElems) {  tmp = new int[rhs.nElems];  std::memcpy(tmp, rhs.array, rhs.nElems \* sizeof(\*array));  }  delete[] array;  array = tmp;  nElems = rhs.nElems;  return \*this;  }  // .  }; |

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

| **Principles(s):** 10 – Adopt a Secure Coding Standard: Allows developers to be held to a high standard when it comes to safety and security protocols during the project development in order to make sure any exception data is properly allowed the allocation of new data storage. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Helix QAC | 2024.2 | **C++4075, C++4076** |  |
| LDRA tool suite | 9.9.3 | 527 S, 56 D, 71 D | Partially implemented |
| Parasoft C/C++test | 2023.2 | [Insert text.] | Always catch exceptions Do not leave 'catch' blocks empty |
| PVS-Studio | 7.15 | V565, V1023, V5002 |  |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| [Student Choice] | [STD-008-CPP] | Do not define a C-style variadic function |

| **Noncompliant Code** |
| --- |
| Uses a C-style variadic function to add a bunch of integers together. The function reads arguments until the  value 0 is found. Calling this function without passing the value 0 as an argument (after the first two  arguments) results in undefined behavior. |
| #include <cstdarg>  int add(int first, int second, ...) {  int r = first + second;  va\_list va;  va\_start(va, second);  while (int v = va\_arg(va, int)) {  r += v;  }  va\_end(va);  return r;  } |

| **Compliant Code** |
| --- |
| A variadic function using a function parameter pack is used to implement the add() function, allowing  identical behavior for call sites |
| #include <type\_traits>  template <typename Arg, typename std::enable\_if<std::is\_integral<Arg>::va  lue>::type \* = nullptr>  int add(Arg f, Arg s) { return f + s; }  template <typename Arg, typename...  Ts, typename std::enable\_if<std::is\_integral<Arg>::value>::type \* =  nullptr>  int add(Arg f, Ts... rest) {  return f + add(rest...);  } |

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

| **Principles(s):** 2 – Heed Compiler Warnings: Rely in checker programs and IDE when debugging a program and fix issue due to issues with non-translatable libraries or include files from different languages such as C and C++. | **Principles(s):** |
| --- | --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 24.04 | **function-ellipsis** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.6 | **CertC++-DCL50** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 16.0 | cert-dcl50-cpp | Checked by clang-tidy. |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.4 | **LANG.STRUCT.ELLIPSIS** | Ellipsis |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.2 | **C++2012, C++2625** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2024.3 | [MISRA.FUNC.VARARG](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.9.3 | **41 S** | Fully Implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.2 | **CERT\_CPP-DCL50-a** | Functions shall not be defined with a variable number of arguments |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024a | [CERT C++: DCL50-CPP](https://www.mathworks.com/help/bugfinder/ref/certcdcl50cpp.html) | Checks for function definition with ellipsis notation (rule fully covered) |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 24.04 | **function-ellipsis** | Fully checked |
| [SonarQube C/C++ Plugin](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046388) | 4.11 | [FunctionEllipsis](https://www.sonarsource.com/products/codeanalyzers/sonarcfamilyforcpp/rules-cpp.html#RSPEC-923) |  |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| [Student Choice] | [STD-009-CPP] | Do not attempt to create a std::string from a null pointer |

| **Noncompliant Code** |
| --- |
| An std::string object is created from the results of a call to std::getenv(). However,  because std::getenv() returns a null pointer on failure, this code can lead to undefined when the  environment variable does not exist (or some other error occurs) |
| #include <cstdlib>  #include <string>  void f() {  std::string tmp(std::getenv("TMP"));  if (!tmp.empty()) {  // .  }  } |

| **Compliant Code** |
| --- |
| The results from the call to std::getenv() are checked for null before the std::string object is constructed |
| #include <cstdlib>  #include <string>  void f() {  const char \*tmpPtrVal = std::getenv("TMP");  std::string tmp(tmpPtrVal ? tmpPtrVal : "");  if (!tmp.empty()) {  // .  }  } |

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

| **Principles(s):** 4 – Keep It Simple: Making sure the program runs appropriately by keeping the code simple instead of complex which can cause failures and warnings. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 24.04 | **assert\_failure** |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.2 | **C++4770, C++4771, C++4772, C++4773, C++4774** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2024.3 | [NPD.CHECK.CALL.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.CHECK.CALL.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.CHECK.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [NPD.CHECK.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [NPD.CONST.CALL](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.CONST.DEREF](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.FUNC.CALL.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.FUNC.CALL.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.FUNC.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.FUNC.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [NPD.GEN.CALL.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.GEN.CALL.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.GEN.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.GEN.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [RNPD.CALL](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [RNPD.DEREF](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.2 | **CERT\_CPP-STR51-a** | Avoid null pointer dereferencing |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| [Student Choice] | [STD-010-CPP] | Range check element access |

| **Noncompliant Code** |
| --- |
| The value returned by the call to get\_index() might be greater than the number of elements stored in the  string, resulting in undefined behavior. |
| #include <string>  extern std::size\_t get\_index();  void f() {  std::string s("01234567");  s[get\_index()] = '1';  } |

| **Compliant Code** |
| --- |
| Solution uses the std::basic\_string::at() function, which behaves in a similar fashion to the  index operator[] but throws a std::out\_of\_range exception if pos >= size( |
| #include <stdexcept>  #include <string>  extern std::size\_t get\_index();  void f() {  std::string s("01234567");  try {  s.at(get\_index()) = '1';  } catch (std::out\_of\_range &) {  // Handle error  }  } |

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

| **Principles(s):** 4 – Keep It Simple: Simple code creation to prevent amount of elements vs string storage issues. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 24.04 | assert\_failure |  |
| CodeSonar | 7.4 | **LANG.MEM.BO LANG.MEM.BU LANG.MEM.TBA LANG.MEM.TO LANG.MEM.TU** | Buffer overrun Buffer underrun Tainted buffer access Type overrun Type underrun |
| Helix QAC | 2024.2 | **C++3162, C++3163, C++3164, C++3165** |  |
| Parasoft C/C++test | 2023.2 | **CERT\_CPP-STR53-a** | Guarantee that container indices are within the valid range |
| Polyspace Bug Finder | R2024a | [CERT C++: STR53-CPP](https://www.mathworks.com/help/bugfinder/ref/certcstr53cpp.html) | Checks for:   * Array access out of bounds * Array access with tainted index * Pointer dereference with tainted offset   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.

Automated testing is important because it helps check a programmer’s code using a system that can catch mistakes or warnings without needing someone to review it all the time. DevOps automation allows parts of the project to be tested with very little effort from a user. This testing makes sure that the main part of the project isn’t affected by unsafe code being added. It helps confirm that every piece of code, no matter who wrote it, meets the same quality standards, reducing the chances of security problems or issues in the project.

### 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 | Medium | Unlikely | Medium | Low | 2 |
| STD-002-CPP | High | Probable | High | Medium | 2 |
| STD-003-CPP | High | Likely | Medium | High | 5 |
| STD-004-CPP | High | Likely | Medium | High | 5 |
| STD-005-CPP | High | Likely | Medium | High | 5 |
| STD-006-CPP | Low | Unlikely | High | Low | 1 |
| STD-007-CPP | High | Likely | High | Medium | 3 |
| STD-008-CPP | High | Probable | Medium | High | 4 |
| STD-009-CPP | High | Likely | Medium | High | 5 |
| STD-010-CPP | High | Unlikely | Medium | Medium | 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 | An encryption in rest is when data is encrypted while it is being stored. The data is accessible, but the encryption prevents the data from being readable without a proper key. This encryption allows a company and/or government to stay protected and have additional defenses against any in-person crimes such as theft of device or thumb drives with sensitive data on it. |
| Encryption in flight | An encryption at flight is when data is encrypted while being transmitted. The data may not be encrypted while its being stored and/or being used, but the data will become encrypted as its being transferred to another storage location. This encryption protects sensitive data of an entity even if the data is intercepted by an outside source because the data will also be encrypted without the proper key. These encryptions are helpful when employees and/or users are allowed to telecommute or mobile work from the office. |
| Encryption in use | An encryption in use is when data is encrypted while being used and gives certain users certain access to the data depending on an employee’s security level. This encryption protects the business or government’s databases by creating layers of security to separate user activity from employees. These encryptions prevent new or lower security level employees from gaining total access to the data in the system. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | Authentication uses User Logins, passcodes, possible secure networks, and other security features such as fingerprint scanning and two-feature identification depending on the security level for certain users. |
| Authorization | Authorization uses the authentication features to identify the level of security access given to a certain user. The security level allows the user to have admin credentials in order to gain access to databases, files, and employee records. An admin user can make changes to databases and files. Also, admin users also are given the ability to add new users to have access to certain files in the system. |
| Accounting | Accounting uses the features from authentication and authorization to keep records on when data has been changed in a system and who made these changes. There are also certain systems that require a user to make a comment explaining why a user made certain changes to a system. These comments are created for both security reasons and future assistance. These defenses work together in order to give a system a multi-layer defense by the features relying on each other. |

**\***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:

* 3 – Architect and Design for Security Policies: Design a system and/or program to go hand in hand with security features during the beginning of the projects development and do not leave the security features to be at the end of the project.
* 4 – Keep It Simple: Make the OS development logs simple to track of any changes or discrepancies.
* 5 – Default Deny: User accounts are denied access until authorization and authentication is verified.
* 6 – Adhere to the least privilege: use the least privilege security access required for a retrieval of data instead of using the maximum-security privilege to prevent accidental data breaches.
* 7 – Sanitize Data Sent to Other Systems: Re-check files when suspicious activities are discovered for any security risks even after the data is already stored in the data
* 8 – Practice Defense in Depth: Allow multi-layer defense features to work together and notify each other of an attack or a collapse of a defense layer.
* 10 – Adopt a Secure Coding Standard: The security standard allows the development team to be on the same page and uphold certain standards when it comes to security of a system.
* Firewall logs
* 1 – ValidateInput Data: Validate any exterior data coming into the system to prevent security risks.
* 5 – Default Deny: User accounts are denied access until authorization and authentication is verified.
* 6 – Adhere to the least privilege: Adhere to the least privilege: use the least privilege security access required for a retrieval of data instead of using the maximum-security privilege to prevent accidental data breaches.
* 7 – Sanitize Data Sent to Other Systems: Re-check files when suspicious activities are discovered for any security risks even after the data is already stored in the data
* 8 – Practice Defense in Depth: Allow multi-layer defense features to work together and notify each other of an attack or a collapse of a defense layer.
* 10 – Adopt a Secure Coding Standard: The security standard allows the development team to be on the same page and uphold certain standards when it comes to security of a system.
* Anti-malware logs
* 1 – ValidateInput Data: Validate any exterior data coming into the system to prevent security risks.
* 5 – Default Deny: User accounts are denied access until authorization and authentication is verified.
* 6 – Adhere to the least privilege: Adhere to the least privilege: use the least privilege security access required for a retrieval of data instead of using the maximum-security privilege to prevent accidental data breaches.
* 7 – Sanitize Data Sent to Other Systems: Re-check files when suspicious activities are discovered for any security risks even after the data is already stored in the data
* 8 – Practice Defense in Depth: Allow multi-layer defense features to work together and notify each other of an attack or a collapse of a defense layer.
* 10 – Adopt a Secure Coding Standard: The security standard allows the development team to be on the same page and uphold certain standards when it comes to security of a system.

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 |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

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