**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 6](#_Toc52464060)

[Coding Standard 3 10](#_Toc52464061)

[Coding Standard 4 12](#_Toc52464062)

[Coding Standard 5 14](#_Toc52464063)

[Coding Standard 6 16](#_Toc52464064)

[Coding Standard 7 18](#_Toc52464065)

[Coding Standard 8 20](#_Toc52464066)

[Coding Standard 9 22](#_Toc52464067)

[Coding Standard 10 24](#_Toc52464068)

[Defense-in-Depth Illustration 26](#_Toc52464069)

[Project One 26](#_Toc52464070)

[1. Revise the C/C++ Standards 26](#_Toc52464071)

[2. Risk Assessment 26](#_Toc52464072)

[3. Automated Detection 26](#_Toc52464073)

[4. Automation 27](#_Toc52464074)

[5. Summary of Risk Assessments 27](#_Toc52464075)

[6. Create Policies for Encryption and Triple A 28](#_Toc52464076)

[7. Map the Principles 29](#_Toc52464077)

[Audit Controls and Management 30](#_Toc52464078)

[Enforcement 30](#_Toc52464079)

[Exceptions Process 30](#_Toc52464080)

[Distribution 31](#_Toc52464081)

[Policy Change Control 31](#_Toc52464082)

[Policy Version History 31](#_Toc52464083)

[Appendix A Lookups 31](#_Toc52464084)

[Approved C/C++ Language Acronyms 31](#_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 | Be cautious of all input from untrusted data sources. Precise validation against external data sources may reduce possible vulnerabilities. These sources may cause damage to the program and its surrounding applications. |
| 1. Heed Compiler Warnings | Pay closely attention to warnings after compiling the code. Apply high levels of security to eliminate warnings and secure data efficiently and effectively. Use testing tools to find vulnerabilities and find ways to strengthen the programs security. |
| 1. Architect and Design for Security Policies | Design and develop with the highest security protocols and coding standards. Implement and enforce security policies to reduce vulnerabilities. Divide programs as necessary into separate systems with safeguards in place. Add layers of security with specified user privileges assigned. |
| 1. Keep It Simple | Use best design practices. Keep the code simple as more complex systems are harder to catch errors and testing procedures may not catch all vulnerabilities. Maintaining and reading the code may make it more difficult to apply high security to complex code. |
| 1. Default Deny | Setting the default permissions to access denied is a layer of security and another step to keep the system stable, but still applying a process to allow authorized users access to the system. |
| 1. Adhere to the Principle of Least Privilege | Every process should limit the level of privileges and only apply the necessary amount to complete the task at hand. The higher level of permissions should be limited as this helps reduce an unauthorized attack. Adhering to this principle is another layer of security. |
| 1. Sanitize Data Sent to Other Systems | Make certain to sanitize data passed to any other subsystems, such as command shells, databases, etc. These can be vulnerable through command or injection attacks. The information can be susceptible to exploitation. |
| 1. Practice Defense in Depth | Implement multiple layers of defense. Using defense in depth strategies can adequately prevent unnecessary access and is another step of managing and safeguarding the system. Layers of security in most cases will limit a successful exploitation of sensitive data. |
| 1. Use Effective Quality Assurance Techniques | Quality assurance techniques may be an efficient way to reduce security vulnerabilities. QA is the process to test and analyze the code during and after the development stages and is an effective way to test and correcting errors prior to releasing the program. |
| 1. Adopt a Secure Coding Standard | Incorporating a secure coding standard is a method that can prevent issues during all stages of development and design. Abiding by these standards can eliminate potential harm to the system which benefits the company as well as its users. |

### 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** | [INT-050-CPP] | Do not cast to an out-of-range enumeration value |

| **Noncompliant Code** |
| --- |
| (Bounds Checking) This code attempts to determine whether the value is within the acceptable range |
| **enum** EnumType {    First,    Second,    Third  };    **void** f(**int** intVar) {    EnumType enumVar = **static\_cast**<EnumType>(intVar);    **if** (enumVar < First || enumVar > Third) {      // Handle error    }  } |

| **Compliant Code** |
| --- |
| (Bounds Checking) Checks the value to determine if it can be represented by the enumeration type prior to conversion and so results aren’t 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, 4: Keep it simple and as clean as possible, and 7: Sanitize Data sent to tother systems. Ensure there are no vulnerabilities prior to sending to other systems. |
| --- |

**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++-INT50 |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.1p0 | LANG.CAST.COERCE  LANG.CAST.VALUE | Coercion Alters Value  Cast Alters Value |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2022.3 | C++3013 |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.1 | CERT\_CPP-INT50-a | An expression with enum underlying type shall only have values corresponding to the enumerators of the enumeration |
| [PRQA QA-C++](https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=142409849) | 4.4 | 3013 |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.20 | V1016 |  |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | [INT-031-CPP] | Be certain that integer conversions do not result in misinterpreted or lost data. |

| **Noncompliant Code** |
| --- |
| This noncompliant code shows type range errors that can occur when converting a value of unsigned int type to signed int type. This results in a truncation error. The type range errors include loss of data (truncation and loss of sign (sign errors). |
| #include <limits.h>    **void** func(**void**) {    unsigned **long** **int** u\_a = ULONG\_MAX;  **signed** **char** sc;    sc = (**signed** **char**)u\_a; /\* Cast eliminates warning \*/    /\* ... \*/  } |

| **Compliant Code** |
| --- |
| Ensure to validate ranges when converting from an unsigned type to a signed type. This example shows how to convert a value of unsigned long int type to a value of signed char type. |
| #include <limits.h>    **void** func(**void**) {    unsigned **long** **int** u\_a = ULONG\_MAX;  **signed** **char** sc;  **if** (u\_a <= SCHAR\_MAX) {      sc = (**signed** **char**)u\_a;  /\* Cast eliminates warning \*/    } **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 and as clean as possible, and 9: Use effective quality assurance techniques. |
| --- |

**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=87152428) | 22.04 |  | Supported via MISRA C:2012 Rules 10.1, 10.3, 10.4, 10.6 and 10.7 |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.1p0 | **LANG.CAST.PC.AV** **LANG.CAST.PC.CONST2PTR** **LANG.CAST.PC.INT**  **LANG.CAST.COERCE** **LANG.CAST.VALUE**  **ALLOC.SIZE.TRUNC** **MISC.MEM.SIZE.TRUNC**  **LANG.MEM.TBA** | Cast: arithmetic type/void pointer Conversion: integer constant to pointer Conversion: pointer/integer  Coercion alters value Cast alters value  Truncation of allocation size Truncation of size  Tainted buffer access |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  | Can detect violations of this rule. However, false warnings may be raised if limits.h is included |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity)\* | 2017.07 | **NEGATIVE\_RETURNS**  **REVERSE\_NEGATIVE**  **MISRA\_CAST** | Can find array accesses, loop bounds, and other expressions that may contain dangerous implied integer conversions that would result in unexpected behavior  Can find instances where a negativity check occurs after the negative value has been used for something else  Can find instances where an integer expression is implicitly converted to a narrower integer type, where the signedness of an integer value is implicitly converted, or where the type of a complex expression is implicitly converted |
| [Cppcheck](https://wiki.sei.cmu.edu/confluence/display/c/Cppcheck) | 1.66 | **memsetValueOutOfRange** | The second argument to memset() cannot be represented as unsigned char |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2022.3 | C2850, C2851, C2852, C2853, C2855, C2856, C2857, C2858, C2890, C2891, C2892, C2893,  C2895, C2896, C2897, C2898, C2900, C2901, C2902, C2903, C2905, C2906, C2907, C2908  C++2850, C++2851, C++2852, C++2853, C++2855, C++2856, C++2857, C++2858, C++2890,  C++2891, C++2892, C++2893, C++2895, C++2896, C++2897, C++2898, C++2900, C++2901,  C++2902, C++2903, C++2905, C++2906, C++2907, C++2908, C++3000, C++3010 |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2022.3 | **PORTING.CAST.SIZE** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **93 S, 433 S, 434 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2022.1 | **CERT\_C-INT31-a CERT\_C-INT31-b CERT\_C-INT31-c CERT\_C-INT31-d CERT\_C-INT31-e CERT\_C-INT31-f CERT\_C-INT31-g CERT\_C-INT31-h CERT\_C-INT31-i CERT\_C-INT31-j CERT\_C-INT31-k CERT\_C-INT31-l CERT\_C-INT31-m CERT\_C-INT31-n CERT\_C-INT31-o** | An expression of essentially Boolean type should always be used where an operand is interpreted as a Boolean value An operand of essentially Boolean type should not be used where an operand is interpreted as a numeric value An operand of essentially character type should not be used where an operand is interpreted as a numeric value An operand of essentially enum type should not be used in an arithmetic operation Shift and bitwise operations should not be performed on operands of essentially signed or enum type An operand of essentially signed or enum type should not be used as the right hand operand to the bitwise shifting operator An operand of essentially unsigned type should not be used as the operand to the unary minus operator The value of an expression shall not be assigned to an object with a narrower essential type The value of an expression shall not be assigned to an object of a different essential type category Both operands of an operator in which the usual arithmetic conversions are performed shall have the same essential type category The second and third operands of the ternary operator shall have the same essential type category The value of a composite expression shall not be assigned to an object with wider essential type If a composite expression is used as one operand of an operator in which the usual arithmetic conversions are performed then the other operand shall not have wider essential type If a composite expression is used as one (second or third) operand of a conditional operator then the other operand shall not have wider essential type Avoid integer overflows |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2022b | CERT C: Rule INT31-C | * Integer conversion overflow * Call to memset with unintended value * Sign change integer conversion overflow * Tainted sign change conversion * Unsigned integer conversion overflow   Rule partially covered. |
| [PRQA QA-C](https://wiki.sei.cmu.edu/confluence/display/c/PRQA+QA-C) | 9.7 | **2850, 2851, 2852, 2853, 2855, 2856, 2857, 2858,**  **2890, 2891, 2892, 2893, 2895, 2896, 2897, 2898**  **2900, 2901, 2902, 2903, 2905, 2906, 2907, 2908** | Partially implemented |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | 2850, 2851, 2852, 2853, 2855, 2856, 2857, 2858,  2890, 2891, 2892, 2893, 2895, 2896, 2897, 2898,  2900, 2901, 2902, 2903, 2905, 2906, 2907, 2908,  3000, 3010 |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/c/PVS-Studio) | 7.21 | [**V562**](https://pvs-studio.com/en/docs/warnings/v562/)**,**[**V569**](https://pvs-studio.com/en/docs/warnings/v569/)**,**[**V642**](https://pvs-studio.com/en/docs/warnings/v642/), [V676](https://pvs-studio.com/en/docs/warnings/v676/), [V716](https://pvs-studio.com/en/docs/warnings/v716/), [V721](https://pvs-studio.com/en/docs/warnings/v721/), [**V724**](https://pvs-studio.com/en/docs/warnings/v724/), [V732](https://pvs-studio.com/en/docs/warnings/v732/), [V739](https://pvs-studio.com/en/docs/warnings/v739/), [V784](https://pvs-studio.com/en/docs/warnings/v784/), [V793](https://pvs-studio.com/en/docs/warnings/v793/), [V1019](https://pvs-studio.com/en/docs/warnings/v1019/),  [V1029](https://pvs-studio.com/en/docs/warnings/v1029/),[V1046](https://pvs-studio.com/en/docs/warnings/v1046/) |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/c/RuleChecker) | 22.04 |  | Supported via MISRA C:2012 Rules 10.1, 10.3, 10.4, 10.6 and 10.7 |
| [TrustInSoft Analyzer](https://wiki.sei.cmu.edu/confluence/display/c/TrustInSoft+Analyzer) | 1.38 | **signed\_downcast** | Exhaustively verified. |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | [STR-052-CPP] | Implement valid references, pointers, and iterators to reference elements of a basic\_string. |

| **Noncompliant Code** |
| --- |
| The iterator “loc” is invalidated after the first call to the method insert () and is undefined. |
| #include <string>    **void** f(**const** std::string &input) {    std::string email;      // Copy input into email converting ";" to " "    std::string::iterator loc = email.begin();  **for** (auto i = input.begin(), e = input.end(); i != e; ++i, ++loc) {      email.insert(loc, \*i != ';' ? \*i : ' ');    }  } |

| **Compliant Code** |
| --- |
| The value of the iterator “loc” is updated then is incremented at the end of the loop. |
| #include <string>    **void** f(**const** std::string &input) {    std::string email;      // Copy input into email converting ";" to " "    std::string::iterator loc = email.begin();  **for** (auto i = input.begin(), e = input.end(); i != e; ++i, ++loc) {      loc = email.insert(loc, \*i != ';' ? \*i : ' ');    }  } |

**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 |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.1p0 | **ALLOC.UAF** | Use After Free |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2022.3 | **C++4746, C++4747, C++4748, C++4749** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.1 | **CERT\_CPP-STR52-a** | Use valid references, pointers, and iterators to reference elements of a basic\_string |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2022b | [CERT C++: STR52-CPP](https://www.mathworks.com/help/bugfinder/ref/certcstr52cpp.html) | Checks for use of invalid string iterator (rule partially covered). |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | [FIO-051-CPP] | The importance of file closure when no longer needed. |

| **Noncompliant Code** |
| --- |
| The object file is constructed. The terminate () method not properly used, which causes the object to not close properly after use. |
| #include <exception>  #include <fstream>  #include <string>    **void** f(**const** std::string &fileName) {    std::fstream file(fileName);  **if** (!file.is\_open()) {      // Handle error  **return**;    }    // ...    std::terminate();  } |

| **Compliant Code** |
| --- |
| Using the method Std::fstream::close() prior to calling std::terminate() will ensure the file resources are closed appropriately. |
| #include <exception>  #include <fstream>  #include <string>    **void** f(**const** std::string &fileName) {    std::fstream file(fileName);  **if** (!file.is\_open()) {      // Handle error  **return**;    }    // ...    file.close();  **if** (file.fail()) {      // Handle error    }    std::terminate();  } |

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

| **Principles(s):** 3: Architect and design for security policies, 6: Adhere to the principle of least privileage, 8: Practice defense in depth, and 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** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.1p0 | **ALLOC.LEAK** | Leak |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2022.3 | **C++4786, C++4787, C++4788** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2022.3 | **RH.LEAK** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.1 | **CERT\_CPP-FIO51-a** | Ensure resources are freed |
| [Parasoft Insure++](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) |  |  | Runtime detection |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2022b | [CERT C++: FIO51-CPP](https://www.mathworks.com/help/bugfinder/ref/certcfio51cpp.html) | Checks for resource leak (rule partially covered) |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [MEM-055-CPP] | Abide by the replacement dynamic storage management policies and procedures. |

| **Noncompliant Code** |
| --- |
| Here the global operator new(std::size\_t) function is replaced by a custom implementation and it fails to honor the behavior that is required by the function that it replaces. |
| #include <new>    **void** \*operator **new**(std::**size\_t** size) {  **extern** **void** \*alloc\_mem(std::**size\_t**); // Implemented elsewhere; may return nullptr  **return** alloc\_mem(size);  }    **void** operator **delete**(**void** \*ptr) noexcept; // Defined elsewhere  **void** operator **delete**(**void** \*ptr, std::**size\_t**) noexcept; // Defined elsewhere |

| **Compliant Code** |
| --- |
| To be compliant, this implements the required behavior for the global allocator function that is replaced. It adequately throws a std::bad\_alloc exception when the allocation fails. |
| #include <new>    **void** \*operator **new**(std::**size\_t** size) {  **extern** **void** \*alloc\_mem(std::**size\_t**); // Implemented elsewhere; may return nullptr  **if** (**void** \*ret = alloc\_mem(size)) {  **return** ret;    }  **throw** std::bad\_alloc();  }    **void** operator **delete**(**void** \*ptr) noexcept; // Defined elsewhere  **void** operator **delete**(**void** \*ptr, std::**size\_t**) noexcept; // Defined elsewhere |

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

| **Principles(s):** 3: Architect and design for security policies, and 10: Adopt a secure coding standard. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2022.3 | **C++4736, C++4737, C++4738, C++4739** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2022.3 | **CERT.MEM.OVERRIDE.DELETE** **CERT.MEM.OVERRIDE.NEW** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.1 | **CERT\_CPP-MEM55-a** | The user defined 'new' operator should throw the 'std::bad\_alloc' exception when the allocation fails |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2022b | [CERT C++: MEM55-CPP](https://www.mathworks.com/help/bugfinder/ref/certcmem55cpp.html) | Checks for replacement allocation/deallocation functions that do not meet requirements of the Standard (rule fully covered) |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | [MSC-038-CPP] | Ensure not to treat a predefined identifier as an object. |

| **Noncompliant Code** |
| --- |
| The standard assert()macro is suppressed. This is due to it attempting to pass it as a function pointer to the execute\_handler() function and is an undefined behavior when attempting to suppress the assert() macro |
| #include <assert.h>    **typedef** **void** (\*handler\_type)(**int**);    **void** execute\_handler(handler\_type handler, **int** value) {    handler(value);  }    **void** func(**int** e) {    execute\_handler(&(**assert**), e < 0);  } |

| **Compliant Code** |
| --- |
| The assert() macro becomes contained in a helper function, which removes the undefined behavior. |
| #include <assert.h>    **typedef** **void** (\*handler\_type)(**int**);    **void** execute\_handler(handler\_type handler, **int** value) {    handler(value);  }    **static** **void** assert\_handler(**int** value) {  **assert**(value);  }    **void** func(**int** e) {    execute\_handler(&assert\_handler, e < 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, 3: Architect and design for security policies, and 9: Use effective quality assurance techniques. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 22.04 |  | Supported, but no explicit checker |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.1p0 | **BADMACRO.STDARG\_H** | Use of <stdarg.h> Feature |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2022.3 | **C3437, C3475**  **C++3127, C++5039** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2022.1 | **CERT\_C-MSC38-a** | A function-like macro shall not be invoked without all of its arguments |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2022b | [CERT C: Rule MSC38-C](https://www.mathworks.com/help/bugfinder/ref/certcrulemsc38c.html) | Checks for predefined macro used as an object (rule fully covered) |
| [PRQA QA-C](https://wiki.sei.cmu.edu/confluence/display/c/PRQA+QA-C) | 9.7 | **3437, 3475** |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/c/RuleChecker) | 22.04 |  | Supported, but no explicit checker |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [ERR-58-CPP] | Handle all exceptions thrown before main() begins executing. |

| **Noncompliant Code** |
| --- |
| During startup, an exception may be thrown by the constructor S, if it is not captured when the globalS is first running. |
| **struct** S {    S() noexcept(**false**);  };    **static** S globalS; |

| **Compliant Code** |
| --- |
| By making the globalS into a local variable using static storage duration. This allows exceptions thrown during object construction. |
| **struct** S {    S() noexcept(**false**);  };    S &globalS() {  **try** {  **static** S s;  **return** s;    } **catch** (...) {      // Handle error, perhaps by logging it and gracefully terminating the application.    }    // Unreachable.  } |

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

| **Principles(s):** 3: Architect and design for security and 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** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **potentially-throwing-static-initialization** | Partially checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-ERR58** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | cert-err58-cpp | Checked by clang-tidy |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.1p0 | **LANG.STRUCT.EXCP.THROW** | Use of throw |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2022.3 | **C++4634, C++4636, C++4637, C++4639** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.1 | **CERT\_CPP-ERR58-a** | Exceptions shall be raised only after start-up and before termination of the program |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2022b | [CERT C++: ERR58-CPP](https://www.mathworks.com/help/bugfinder/ref/certcerr58cpp.html) | Checks for exceptions raised during program startup (rule fully covered) |
| [PRQA QA-C++](https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=142409849) | 4.4 | **4634, 4636, 4637, 4639** |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 20.10 | **potentially-throwing-static-initialization** | Partially checked |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Exceptions (Student Choice) | [DCL-057-CPP] | Ensure not to let exceptions escape from functions such as destructors or deallocations. |

| **Noncompliant Code** |
| --- |
| A global deallocation is declared noexcept or (false) and throws and exception. This becomes an undefined behavior when the conditions aren’t properly met and throws the deallocation function. |
| #include <stdexcept>    **bool** perform\_dealloc(**void** \*);    **void** operator **delete**(**void** \*ptr) noexcept(**false**) {  **if** (perform\_dealloc(ptr)) {  **throw** std::logic\_error("Something bad");    }  } |

| **Compliant Code** |
| --- |
| The compliant code doesn’t throw exceptions if the deallocation fails. Instead it fails as gracefully as possible. |
| #include <cstdlib>  #include <stdexcept>    **bool** perform\_dealloc(**void** \*);  **void** log\_failure(**const** **char** \*);    **void** operator **delete**(**void** \*ptr) noexcept(**true**) {  **if** (perform\_dealloc(ptr)) {      log\_failure("Deallocation of pointer failed");      std::**exit**(1); // Fail, but still call destructors    }  } |

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

| **Principles(s):** 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** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **destructor-without-noexcept** **delete-without-noexcept** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-DCL57** |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.1p0 | **LANG.STRUCT.EXCP.CATCH**  **LANG.STRUCT.EXCP.THROW** | Use of catch  Use of throw |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2022.3 | **C++2045, C++2047, C++4032, C++4631** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2022.3 | **MISRA.DTOR.THROW** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **453 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.1 | **CERT\_CPP-DCL57-a** **CERT\_CPP-DCL57-b** | Never allow an exception to be thrown from a destructor, deallocation, and swap Always catch exceptions |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2022b | [CERT C++: DCL57-CPP](https://www.mathworks.com/help/bugfinder/ref/certcdcl57cpp.html) | Checks for class destructors exiting with an exception (rule partially covered) |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.21 | [V509](https://pvs-studio.com/en/docs/warnings/v509/)**,**[V1045](https://pvs-studio.com/en/docs/warnings/v1045/) |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 20.10 | **destructor-without-noexcept** **delete-without-noexcept** | Fully checked |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Data Type (Student Choice) | [DCL-053-CPP] | Ensure not to write syntactically ambiguous declarations. |

| **Noncompliant Code** |
| --- |
| The noncompliant code attempts to declare a local variable while executing the default constructor. |
| #include <iostream>    **struct** Widget {    Widget() { std::cout << "Constructed" << std::endl; }  };    **void** f() {    Widget w();  } |

| **Compliant Code** |
| --- |
| This code represents two equally compliant ways to write the declaration. First elide the () after the variable declaration. Second, use a braced-init-list to direct-initialize the local variable. |
| #include <iostream>    **struct** Widget {    Widget() { std::cout << "Constructed" << std::endl; }  };    **void** f() {    Widget w1; // Elide the parentheses    Widget w2{}; // Use direct initialization  } |

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

| **Principles(s):** 3: Architect and Design for Security polices, 4: keep it simple. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Tool | Version | Checker | Description |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.1p0 | **LANG.STRUCT.DECL.FNEST** | Nested Function Declaration |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2022.3 | **C++1109, C++2510** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2022.3 | **CERT.DCL.AMBIGUOUS\_DECL** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **296 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.1 | **CERT\_CPP-DCL53-a** **CERT\_CPP-DCL53-b** **CERT\_CPP-DCL53-c** | Parameter names in function declarations should not be enclosed in parentheses Local variable names in variable declarations should not be enclosed in parentheses Avoid function declarations that are syntactically ambiguous |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2022b | [CERT C++: DCL53-CPP](https://www.mathworks.com/help/bugfinder/ref/certcdcl53cpp.html) | Checks for declarations that can be confused between:   * Function and object declaration * Unnamed object or function parameter declaration   Rule fully covered. |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | **2502, 2510** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | -Wvexing-parse |  |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Strings  (Student Choice) | [STR-051-CPP] | Ensure not to attempt to create a std::string from a null pointer. |

| **Noncompliant Code** |
| --- |
| This code shows a std::string object being created from the results of a call to std::getenv(). The called function returns a null pointer on failure which leads to an undefined behavior if the event of the environment variable doesn’t exist. |
| #include <cstdlib>  #include <string>    **void** f() {    std::string tmp(std::**getenv**("TMP"));  **if** (!tmp.empty()) {      // ...    }  } |

| **Compliant Code** |
| --- |
| The compliant codes show the results from the call to std::getenv() being checked first for null prior to the std::string object being 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, and 9: Use effective quality assurance technniques |
| --- |

**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) | 20.10 | **assert\_failure** |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.1p0 | **LANG.MEM.NPD** | Null Pointer Dereference |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2022.3 | **C++4770, C++4771, C++4772, C++4773, C++4774** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2022.3 | **NPD.CHECK.CALL.MIGHT** **NPD.CHECK.CALL.MUST** **NPD.CHECK.MIGHT** **NPD.CHECK.MUST** **NPD.CONST.CALL** **NPD.CONST.DEREF** **NPD.FUNC.CALL.MIGHT** **NPD.FUNC.CALL.MUST** **NPD.FUNC.MIGHT** **NPD.FUNC.MUST** **NPD.GEN.CALL.MIGHT** **NPD.GEN.CALL.MUST** **NPD.GEN.MIGHT** **NPD.GEN.MUST** **RNPD.CALL** **RNPD.DEREF** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.1 | **CERT\_CPP-STR51-a** | Avoid null pointer dereferencing |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2022b | [CERT C++: STR51-CPP](https://www.mathworks.com/help/bugfinder/ref/certcstr51cpp.html) | Checks for string operations on null pointer (rule fully 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.

**DevSecOps Diagram Explanation:**

First and foremost, DevSecOps, also known as “Development, Security, and Operations,” is an integration of security throughout the entire software development lifecycle. This includes the initial design to the testing and deployment of the software, and the integration and software delivery. The major advantages of DevSecOps are its security and speed. The team is more susceptible of producing better and more secure code with this integration and can be cost effective for the organization. Using the previous DevOps process and infrastructure from Green Pace, we can ensure security is incorporated throughout the entire process.

In the planning stages, security should be assessed, and best choices will be chosen, and a plan is created to implement that secure coding. In the Design and building stages, security protocols are implemented within the code. Once enough code is written, test for vulnerabilities. When new code is modified, test for vulnerabilities again. Using stronger testing methods and tools, the environment will be analyzed using dynamic testing to detect any flaws. The Transition and health check is the final stages of testing, and the vulnerabilities should be rectified. Once the environment is active, there should be continuous monitoring and logging events of any unauthorized detection. The environment and team should respond appropriately with any intrusion detected, such as rolling back or turning off the system to stop the threat. The final stages of this diagram the environment must be maintained and stabilized after any security breaches and the compromised data and access points must be repaired and strengthened.

### 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 |
| --- | --- | --- | --- | --- | --- |
| [INT-050-CPP] | Medium | Unlikely | Medium | P4 | L3 |
| [INT-031-CPP] | High | Probable | High | **P6** | **L2** |
| [STR-052-CPP] | High | Probable | High | P6 | **L2** |
| [FIO-051-CPP] | Medium | Unlikely | Medium | **P4** | **L3** |
| [MEM-055-CPP] | High | Likely | Medium | **P18** | **L1** |
| [MSC-038-CPP] | Low | Unlikely | Medium | **P2** | **L3** |
| [ERR-58-CPP] | Low | Likely | Low | **P9** | **L2** |
| [DCL-057-CPP] | Low | Likely | Medium | **P6** | **L2** |
| [DCL-053-CPP] | Low | Unlikely | Medium | **P2** | **L3** |
| [STR-051-CPP] | High | Likely | Medium | **P18** | **L1** |

### Create Policies for Encryption and Triple A

Include all three types of encryptions (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 in rest | This type of encryption is essential for protecting data from a breach. Encryption at rest is the process of converting any sensitive data into another form using an algorithm. The key is needed to decrypt the sensitive data, and only authorized personnel will have the access to decode the data. |
| Encryption at flight | This type of encryption is essential for protecting data transmitted over aa network, that being internal or over the internet. Data transmitted over the internet is potentially at a high risk. When you Encrypt data in flight, that refers to encrypting the data being transmitted over the network. Some examples may be using VPN’s, avoid using self-signed certificates, and using transport layer security (TLS) for your https connections. These are a few examples that can reduce risk. |
| Encryption in use | Data in use is one of the higher forms of vulnerability. Since this data is stored in clear text during its use, it is the most susceptible to exploitation. Encryption in use will always protect the data and is encrypted throughout the entire data life cycle. Any anomalies that are detected are analyzed and blocks any suspicious requests. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | This process verifies the identity and credentials of the client/user wishing to gain access and connect to the network. The network in turn verifies the credentials to ensure the identity belongs to that client/user. Authentication is a layer of defense and determines if the request is approved. |
| Authorization | This process determines which privilege is granted to the client/user of a particular credential and given a specified level of access. Clients/users must obtain an authorization before tasks requested are granted. Authorization is needed to grant access to the system and should be limited to the necessary level only for a limited time to prevent unauthorized access. |
| Accounting | This process is a form of logging or collecting information on resource utilization on a specific section or on all network access. Accounting is vital to security on any environment. Having a record of what has transpired is needed to ensure the correct behavior/work is conducted on the system at all times. |

**\***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 | 09/18/2022 | Initial Template | Kenneth Gollaher |  |
| 1.1 | 10/09/2022 | Final template | Kenneth Gollaher |  |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

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