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



# Green Pace Secure Development Policy

## Contents

[Overview 2](#_Toc52464053)

[Purpose 2](#_Toc52464054)

[Scope 2](#_Toc52464055)

[Module Three Milestone 2](#_Toc52464056)

[Ten Core Security Principles 2](#_Toc52464057)

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

[Coding Standard 1 4](#_Toc52464059)

[Coding Standard 2 5](#_Toc52464060)

[Coding Standard 3 6](#_Toc52464061)

[Coding Standard 4 7](#_Toc52464062)

[Coding Standard 5 8](#_Toc52464063)

[Coding Standard 6 9](#_Toc52464064)

[Coding Standard 7 10](#_Toc52464065)

[Coding Standard 8 11](#_Toc52464066)

[Coding Standard 9 13](#_Toc52464067)

[Coding Standard 10 14](#_Toc52464068)

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

[Project One 15](#_Toc52464070)

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

[2. Risk Assessment 15](#_Toc52464072)

[3. Automated Detection 15](#_Toc52464073)

[4. Automation 15](#_Toc52464074)

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

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

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

[Audit Controls and Management 18](#_Toc52464078)

[Enforcement 18](#_Toc52464079)

[Exceptions Process 18](#_Toc52464080)

[Distribution 19](#_Toc52464081)

[Policy Change Control 19](#_Toc52464082)

[Policy Version History 19](#_Toc52464083)

[Appendix A Lookups 19](#_Toc52464084)

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

## Overview

Software development at Green Pace requires consistent implementation of secure principles to all developed applications. Consistent approaches and methodologies must be maintained through all policies that are uniformly defined, implemented, governed, and maintained over time.

## Purpose

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

## Scope

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

## Module Three Milestone

### Ten Core Security Principles

| **Principles** | Write a short paragraph explaining each of the 10 principles of security. |
| --- | --- |
| 1. ValidateInput Data | Validating input data can help prevent software vulnerabilities. External data sources should always be verified and validated. Sources such as command line arguments, environmental variables and network interfaces should all be double checked before use. |
| 1. Heed Compiler Warnings | Always read and understand warnings issued by the compiler. Always use the highest warning levels that are available so that you are notified of everything that might cause an issue. Static and dynamic analysis tools can also be of help when trying to detect and eliminate security risks. |
| 1. Architect and Design for Security Policies | When designing software, it is very important that you protect the architecture by following any security policies that have been issued to you. Assume that you will need to make it as secure as reasonably achievable. |
| 1. Keep It Simple | Keep code design simple and easy to read. Doing this will allow you to quickly detect and rectify any vulnerabilities that come up in your code. Simple code will be much easier to secure than complex code. |
| 1. Default Deny | Access decisions should be standardized based on permissions. This is much more secure than standardizing based on exclusions. Access should be denied by default and permissions should be given situationally. |
| 1. Adhere to the Principle of Least Privilege | Process executions should have the least amount of privileges that is required to complete a task. Elevated privileges may pose a security risk and should only be used for the amount of time that it takes to complete the task. |
| 1. Sanitize Data Sent to Other Systems | Data should be secure and clean before it is sent to other systems. Since injection attacks can manipulate the components of subsystems, these subsystems should have sanitized data. |
| 1. Practice Defense in Depth | There should always be multiple layers of security. If one layer of security measures were to fail, there would be a safety net of other layers to prevent further attacks. Defense in depth will also allow for fail safes to be implemented in the form of deeper layers of security. |
| 1. Use Effective Quality Assurance Techniques | Quality assurance techniques are very important to securing code because it can include multiple testing phases. In addition to testing, there can be regular security reviews that can ensure that the code is secure. |
| 1. Adopt a Secure Coding Standard | Adopting a secure coding standard is very important to developing secure code. Care should be taken to learn the vulnerabilities that are present with whatever programming language is used at the time. |

### C/C++ Ten Coding Standards

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

#### Coding Standard 1

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Type** | [STD-**001-CPP**] | Do not cast to an out of range enumeration value |

| **Noncompliant Code** |
| --- |
| Noncompliant code will check whether a given value is within range of acceptable enumeration values. After casting the type, it may not be able to represent the given integer value. |
| Enum EnumType {  First,  Second,  Third  };  Void f(int intVar) {  EnumType enumVar = static\_cast<EnumType>(intVar);  If (enumVar < First || enumVar > Third) {  // Handle error  }  } |

| **Compliant Code** |
| --- |
| This compliant code solution will check the value represented by the enumeration type. IT will then perform the conversion to guarantee that the conversion does not result in an unspecified value. This will restrict the converted value to one of the specific enumerator type. |
| 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):** Unspecified values may result in a buffer overflow which can lead to an attack. Since enumerators are rarely used for indexing into arrays, it is likely that this can result in data integrity violations as opposed to arbitrary code executions. |
| --- |

**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 |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.2 | C++3013 |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.2 | 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) | 7.15 | V1016 |  |

#### Coding Standard 2

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

| **Noncompliant Code** |
| --- |
| Pos is invalidated after the first call to insert(). Loop iterators have undefined behavior. |
| #inclue <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. |
| #inclue <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):** Deny 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) | 20.10 | overflow\_upon\_dereference |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.2 | C++4746, C++4747, C++4748, C++4749 |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.2 | CERT\_CPP-CTR51-a | Do not modify container while iterating over it |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.15 | V783 |  |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | [STD-**003-CPP**] | Guarantee string storage has required space for character data and the null terminator |

| **Noncompliant Code** |
| --- |
| The input is not bound. Buffer overflow may occur. |
| #include <iostream>  Void f() {  Char buf[12];  Std::cin >> buf;  } |

| **Compliant Code** |
| --- |
| Ensure that data is not truncated and protect against buffer overflow by using 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):** Prevent any database architect issues by ensuring 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) | 6.1p0 | 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) | 2021.2 | C++2835, C++2836, C++2839, C++5216 |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2021.3 | NNTS.MIGHT  NNTS.TAINTED |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | 489 S, 66 X, 70 X, 71 X | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.2 | 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) | R2021b | 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) | 4.10 | 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** |
| --- |
| Different smart pointers are constructed from the same pointer value. The other pointer is affected by altering the first one. This 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** |
| --- |
| When the local automatic variable (p2) is destroyed, the count for the shared pointer value is decremented. This solution calls std::make\_shared() instead of allocating 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):** Prevent any stored variable from being replaced and destroyed. 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) | 20.10 | dangling\_pointer\_use |  |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | CertC++-MEM56 |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.2 | C++4721, C++4722, C++4723 |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.2 | 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) | R2021b | 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) | 7.15 | V1006 |  |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [STD-**005-CPP**] | Properly deallocate dynamically allocated resources |

| **Noncompliant Code** |
| --- |
| Local variable is being passed resulting in the pointer being passed to ::powerator delete() which results in an 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() as opposed to calling s1’s destructor. |
| #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):** Passing a pointer value to a deallocation function that was not previously obtained will result in an undefined behavior. This will lead to exploitable vulnerabilities. |
| --- |

**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 | **invalid\_dynamic\_memory\_allocation dangling\_pointer\_use** |  |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-MEM51** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | 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) | 6.1p0 | **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) | 2021.2 | **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) | 2021.3 | [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) | 9.7.1 | 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) | 2021.2 | CERT\_CPP-MEM51-aCERT\_CPP-MEM51-bCERT\_CPP-MEM51-cCERT\_CPP-MEM51-d | Guarantee that container indices are within the valid range |
| [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) | R2021b | [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. |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | 2110, 2111, 2112, 2113, 2118,  3337, 3339, 4262, 4263, 4264 |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.15 | [**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) | 4.10 | 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** |
| --- |
| Uses the assert() macro to assert a memory mapped structure that is essential to the code. |
| #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** |
| --- |
| A preprocessor conditional statement cant be used for assertions that have only constant expressions. |
| #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):** Static assertion is a valuable diagnostic tool for finding and eliminating software defects that can result in vulnerabilities. |
| --- |

**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) | 7.2.0 | **CertC-DCL03** |  |
| Clang | 3.9 | misc-static-assert | Checked by clang-tidy |
| CodeSonar | 6.1p0 |  | Users can implement a custom check that reports uses of the assert() macro |
| ECLAIR | 1.2 | **CC2.DCL03** | Fully implemented |
| LDRA tool suite | 9.7.1 | **44 S** | Fully implemented |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [STD-**007-CPP**] | Handle all exceptions thrown before main() begins executing |

| **Noncompliant Code** |
| --- |
| The constructor for S may throw an exception that is not caught when globalS is constructed. |
| Struct S {  S() noexcept(false);  };  Static S globalS; |

| **Compliant Code** |
| --- |
| This solution will make globalS into a local variable with static storage duration. This will allow exceptions thrown during object construction to be identified. |
| Struct S {  S() noexcept(false);  };  S &globalS() {  Try {  Static S s;  Return s;  }catch (…) {  }  } |

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

| **Principles(s):** Throwing an exception that cannot be caught results in program termination. This sort of situation will lead to a denial of service attack. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Helix QAC | 2021.2 | **C++4075, C++4076** |  |
| LDRA tool suite | 9.7.1 | 527 S, 56 D, 71 D | Partially implemented |
| Parasoft C/C++test | 2021.2 |  | 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** |
| --- | --- | --- |
| Misc | [STD-**008-CPP**] | Range check element access |

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

| **Compliant Code** |
| --- |
| When pos >= size(), an exception will be thrown. |
| #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 &) {    }  } |

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

| **Principles(s):** Prevent unnecessary storage issues by keeping the code simple and easy to maintain. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
|  | 20.10 | assert\_failure |  |
| CodeSonar | 6.1p0 | **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 | 2021.2 | **C++3162, C++3163, C++3164, C++3165** |  |
| Parasoft C/C++test | 2021.2 | **CERT\_CPP-STR53-a** | Guarantee that container indices are within the valid range |
| Polyspace Bug Finder | R2021b | [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. |

#### Coding Standard 9

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

| **Noncompliant Code** |
| --- |
| std::getenv() will return a null pointer on failure which can lead to undefined behavior if the environment variable does not exist. |
| #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 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):** Prevent unnecessary issues by keeping the code simple and easy to maintain. |
| --- |

**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** |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.2 | **C++4770, C++4771, C++4772, C++4773, C++4774** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.2 | **CERT\_CPP-STR51-a** | Avoid null pointer dereferencing |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Misc | [STD-**010-CPP**] | Do not define a C-style variadic function |

| **Noncompliant Code** |
| --- |
| A C-style variadic function is used to add series of integers together. This function will lead to an undefined behavior error if 0 is not passed as an argument. |
| #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** |
| --- |
| The proper parameters are used to implement the add() function. |
| #include <type\_traits>    template <typename Arg, typename std::enable\_if<std::is\_integral<Arg>::value>::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):** It is important to heed compiler warnings. Using the functions built into IDE’s can make programming a lot easier and safer. |
| --- |

**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) | 20.10 | **function-ellipsis** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-DCL50** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | cert-dcl50-cpp | Checked by clang-tidy. |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 6.1p0 | **LANG.STRUCT.ELLIPSIS** | Ellipsis |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **41 S** | Fully Implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.2 | **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) | R2021b | [CERT C++: DCL50-CPP](https://www.mathworks.com/help/bugfinder/ref/certcdcl50cpp.html) | Checks for function definition with ellipsis notation (rule fully covered) |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 1.1 | **2012, 2625** |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 20.10 | **function-ellipsis** | Fully checked |

### 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.

Automation should be checked to ensure that its code is up to the same standards as any developer. Automation testing should be used whenever automation is allowed to create code so that it maintains the quality of the project and is free from errors. The DevOps process allows for entire sections of code to be tested before it is seen by a human developer. Code is continuously reviewed and tested for its integrity and security. Automation testing and review will work to protect code from any flaws that can set the project back as well as handle any security risks as they come up. DevOps currently does not address the security concerns of a modern company. DevSecOps improves upon the DevOps model by causing an engineering shift and focuses on security. In the diagram, I think that the main points that I would like to highlights would be the monitor and detect sections since this is a very important part of DevSecOps. In this section we are looking for errors and bugs in the code that could cause vulnerabilities. The entire DevSecOps is a cycle and everything sort of works together to create secure code.

### 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 | Unlikely | Medium | Medium | 2 |
| STD-009-CPP | High | Likely | Medium | High | 5 |
| STD-010-CPP | High | Probable | Medium | High | 4 |

### 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 | Encryption at rest occurs when we encrypt data that is not currently being utilized. This policy protects data files from misuse and attack. Breaching this data would require brute force or the decryption keys in order to access. |
| Encryption at flight | Encryption in flight refers to protecting data as it is transitioning or moving through a network. This policy will protect data while it is at its most vulnerable point. This type of protection can be achieved through the use of SSL/TLS connections between a web server and a database. This type of connection will secure the data as it moves between these two systems. |
| Encryption in use | Encryption in use occurs when we protect data that is currently being used. This can take place through the use of protected memory and Homomorphic encryption. These safeguards would require brute force, which would take a lot of time. Encryption in use can occur when we have a web server that has pulled data from a database and delivering data to a customer. Since an end user is involved, this type of encryption is very important. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | Authentication is a process in which an application can verify your identity. This can be done by the use of login information, passcodes or any other means of identification. Two factor authentication is also a great example. These verification methods allow only authorized entry into a system and minimize risk of attack. |
| Authorization | Authentication is a process that grants authenticated users access to a system. Authorization can also allow what each user can access in a system using a security level. For example, an administrator would have full authorization to access the entire system. Authorization is also where the principle of default deny takes place. Most users by default will not have much authorization in a system. |
| Accounting | Accounting keeps records of all changes made to the system. This is why when changes are made to a system, users will often have to comment on what they did. This is for the accounting policy. This is also important because through this policy you can keep track of what user is accessing what information. This is important because if a system were to have sensitive information, you would know exactly who was accessing what information or data. |

**\***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
* 3 – Architect and Design for Security Policies: Design a project that will have set security features and clear development plans. Make sure that security functions are implemented early.
* 4 – Keep It Simple: Make the OS development logs simple and easy to understand.
* 5 – Default Deny: User accounts are denied access until authorization and authentication is verified.
* 6 – Adhere to the least privilege: By default, every user gets the least amount of privileges possible and then privileges are granted based on circumstances.
* 7 – Sanitize Data Sent to Other Systems: Files should be scrubbed for sensitive data when they are transitioning to other systems. This will prevent data leaks from occurring.
* 8 – Practice Defense in Depth: Allow multilayer defense features to work together and notify each other of an attack. This type of policy will allow you to catch errors that other layers can miss.
* 10 – Adopt a Secure Coding Standard: This policy ensures that the entire development team is up to the same standard.
* 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: By default, every user gets the least amount of privileges possible and then privileges are granted based on circumstances.
* 7 – Sanitize Data Sent to Other Systems: Files should be scrubbed for sensitive data when they are transitioning to other systems. This will prevent data leaks from occurring.
* 8 – Practice Defense in Depth: Allow multilayer defense features to work together and notify each other of an attack. This type of policy will allow you to catch errors that other layers can miss.
* 10 – Adopt a Secure Coding Standard: This policy ensures that the entire development team is up to the same standard.
* 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: By default, every user gets the least amount of privileges possible and then privileges are granted based on circumstances.
* 7 – Sanitize Data Sent to Other Systems: Files should be scrubbed for sensitive data when they are transitioning to other systems. This will prevent data leaks from occurring.
* 8 – Practice Defense in Depth: Allow multilayer defense features to work together and notify each other of an attack. This type of policy will allow you to catch errors that other layers can miss.
* 10 – This policy ensures that the entire development team is up to the same standard.

**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 |  |
| 1.5 | 10/02/2022 | Standard and Principles | Nicholas Boodoo | Nicholas Boodoo |
| 2.0 | 10/07/2022 | Complete | Nicholas Boodoo | Nicholas Boodoo |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

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