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



# Green Pace Secure Development Policy

## Contents

[Overview 2](#_Toc52464053)

[Purpose 2](#_Toc52464054)

[Scope 2](#_Toc52464055)

[Module Three Milestone 2](#_Toc52464056)

[Ten Core Security Principles 2](#_Toc52464057)

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

[Coding Standard 1 4](#_Toc52464059)

[Coding Standard 2 5](#_Toc52464060)

[Coding Standard 3 6](#_Toc52464061)

[Coding Standard 4 7](#_Toc52464062)

[Coding Standard 5 8](#_Toc52464063)

[Coding Standard 6 9](#_Toc52464064)

[Coding Standard 7 10](#_Toc52464065)

[Coding Standard 8 11](#_Toc52464066)

[Coding Standard 9 13](#_Toc52464067)

[Coding Standard 10 14](#_Toc52464068)

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

[Project One 15](#_Toc52464070)

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

[2. Risk Assessment 15](#_Toc52464072)

[3. Automated Detection 15](#_Toc52464073)

[4. Automation 15](#_Toc52464074)

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

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

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

[Audit Controls and Management 18](#_Toc52464078)

[Enforcement 18](#_Toc52464079)

[Exceptions Process 18](#_Toc52464080)

[Distribution 19](#_Toc52464081)

[Policy Change Control 19](#_Toc52464082)

[Policy Version History 19](#_Toc52464083)

[Appendix A Lookups 19](#_Toc52464084)

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

## Overview

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

## Purpose

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

## Scope

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

## Module Three Milestone

### Ten Core Security Principles

| **Principles** | Write a short paragraph explaining each of the 10 principles of security. |
| --- | --- |
| 1. ValidateInput Data | To try to eliminate as many vulnerabilities as it can while validating incoming data from untrusted sources and even some trusted sources just to make sure that no malicious data is coming through. |
| 1. Heed Compiler Warnings | Compiles and tests the cost as much as possible to try and keep the code tight. Warnings could be annoying but checking all of them is the proper choice because one of them could become serious. |
| 1. Architect and Design for Security Policies | If you design code with thinking of putting in as much security as possible but still meet the standard for how the code is supposed to function. |
| 1. Keep It Simple | Keeping the code as clean as possible makes it easier to read and to perform edits if needed. Keeping the code clean makes it easier for those who are not familiar with your code, it helps them to be able to understand why it written the way it was. |
| 1. Default Deny | Denial is a good standard to have in the coding. You can give access to certain people in certain departments if needed, but those who aren’t a developer, or a programmer should not really have access to the code. |
| 1. Adhere to the Principle of Least Privilege | Principle of least privilege goes hand in hand with default denial. This sets the standard of who gets what kind of higher up access and those that will have to go through the proper chain of command to gain any higher access. Less people with higher up access for confidential records the safer those records are. |
| 1. Sanitize Data Sent to Other Systems | While developing your code be sure that any data being sent to other systems is safe and doesn’t contain any sensitive information. Sending any of that sensitive information could lead to breaches in the system. |
| 1. Practice Defense in Depth | Its always safe to use more than one layer of protection. Having multiple combined layers and having them be somewhat redundant could help defend against any potential breach. |
| 1. Use Effective Quality Assurance Techniques | Quality Assurance can be a pain for developers because they are supposed to find any bugs or issues within the code and exploit it. That way the developer can go back and make the necessary changes to make sure that the code is secure and functions. |
| 1. Adopt a Secure Coding Standard | Secure coding is a great principle to have in the front of your mind while developing code. Even having quality assurance is great because it helps make your code more secure and it should help prevent any possible bugs later down the codes life. |

### 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] | Defining C – The style variadic functions can possibly have some cracks in the armor since it doesn’t check arguments being passed. |

| **Noncompliant Code** |
| --- |
| This function is designed to read the values until 0 is found but if 0 isn’t found in two arguments there could be an error message that displays. |
| #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 function below has an added statement to help prevent issues in the above coding. |
| #include <tail\_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) {  Returnf + add(rest…);  } |

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

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

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Clang | 3.9 | Cert-dcl50-cpp | Checked by clang-tidy |
| Axivion Bauhaus suite | 6.9.0 | Certc++dcl50 |  |
| Astree | 20.10 | Function-ellipsi | Fully checked |
| CodeSonar | 5.4p0 | Lang.struct.ellipsis | Ellipsis |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | [STD-002-CPP] | Defining a reserved identifier incorrectly can cause issues. |

| **Noncompliant Code** |
| --- |
| Naming standards not met and causes undefined behavior. |
| #ifndef \_MY\_HEADER\_H\_  #define \_MY\_HEADER\_H\_  // Contents of <my\_header.h>  #endif // \_MY\_HEADER\_H\_ |

| **Compliant Code** |
| --- |
| By removing the trailing and leading underscores it prevents the issue. |
| #ifndef MY\_HEADER\_H  #define MY\_HEADER\_H  // Contents of <my\_header.h>  #endif // MY\_HEADER\_H |

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

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

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | Reserved-identifier | Partially checked |
| Axivion Bauhaus suite | 6.9.0 | Certc++-dl51 |  |
| Clang | 3.9 | -wreserved-id-macro  -wuser-defined-literals | Wreserved-id-macro flag is not enabled by default |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | [STD-003-CPP] | Cv-qualifying a reference type could result in an undefined behavior. A compiler should send a fatal error and it may not display any results. |

| **Noncompliant Code** |
| --- |
| Const-qualified reference to a char is formed instead of a reference to a const-qualified char |
| #include <iostream>  Void f(char c) {  Char & const p = c;  P = ‘p’;  Std::cout << c << std::end1;  } |

| **Compliant Code** |
| --- |
| Remove the const qualifier to prevent the issue from occurring. |
| #include <iostream>  Void f(char c) {  Char &p = c;  P = ‘p’;  Std::cout << c << std::end1;  } |

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

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

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Axivion Bauhaus suite | 6.9.0 | Certc++-dcl52 |  |
| Polyspace bug finder | R2020a | Certc++:dcl52-cpp | Fully covered |
| Parasoft c/c++test | 2020.2 | Cert\_cpp-dcl52-a | Never qualify a reference type with “const” or “ volatile” |
| Prqa qa-c++ | 4.4 | 0014 |  |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | [STD-004-CPP] | You should try to write code the same way every time. That way co-workers or even you can go back and review what was done and make sure that it is secure, makes sense and functions. |

| **Noncompliant Code** |
| --- |
| The argument can be taken to declare an anonymous object and call it a single-argument converting constructor, or it could interpret it as declaring an object named m and default the constructing of it. |
| #include <mutex>  Static std::mutex::m;  Static int shared\_resource;  Void increment\_by\_28() {  Std::unique\_lock<std::mutex>(m);  Shared\_resources += 28;  } |

| **Compliant Code** |
| --- |
| The lock is given an identifier and the proper converting constructor will be called. |
| #include <mutex>  Static std::mutex m;  Static int shared\_resource;  Void increment\_by\_28() {  Std::unique\_lock<std::mutex> lock(m);  Shared\_resource += 28;  } |

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

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

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Ldra tool suite | 9.7.1 | 296 s | partially |
| Parasoft c/c++test | 2020.2 | Cert-cpp-dcl53-a  Cert-cpp-dcl53-b | Declare functions at file scope |
| Polyspace bug finder | R2020a | Cert c++:dcl53-cpp | Fully covered |
| Prqa qa-c++ | 4.4 | 2502, 2510 |  |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [STD-005-CPP] | Overload allocation and deallocation functions as a pair. Doing so could lead to undefined behavior. |

| **Noncompliant Code** |
| --- |
| The allocation is overloaded and there is no deallocation function declared. |
| #include <Windows.h>  #include <new>  Void \*operator new(std::size\_t size) noexcept(false) {  Static HANDLE h = ::HeapCreate(0, 0, 0);  If (h) {  Return::HeapAlloc(h, o, size);  }  Throw std::bad\_alloc();  } |

| **Compliant Code** |
| --- |
| Deallocation is declared and it should prevent overload condition. |
| #include <Windows.h>  #include <new>  Class HeapAllocator {  Static HANDLE h;  Static bool init;  Public:  Static void \*alloc(std::size\_t size) nonexcept(false) {  If(!init) {  H = ::HeapCreate(0, 0, 0);  Init = true;  }  If(h) {  Return::HeapAlloc(h, 0, size);  }  Throw std::bad\_alloc();  }  Static void dealloc(void\*ptr) noexcept {  If(h) {  (void)::HeapFree(h, 0, ptr);  }  }  };  HANDLE HeapAllocator::h = nullptr;  Bool HeapAllocator::init = false;  Void\*operator new(std::size\_t size) noexcept(false) {  Return HeapAllocator::alloc(size);  }  Void operator delete(void\*ptr) noexcept {  Return HeapAllocator::dealloc(ptr);  } |

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

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

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | New-delete-pairwise | Partially checked |
| Clang | 3.9 | Misc-new-delete-overloads | Checked |
| Parasoft c/c++test | 2020.2 | Cert-cpp-dcl54-a | Provide new and delete |
| Polyspace bug finder | R2020a | Cert c++:dcl54-cpp | Partially checked |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | [STD-006-CPP] | Try to prevent information leaks when passing a class object across a trust boundary. The passing data needs to be verified or else it could cause issues. |

| **Noncompliant Code** |
| --- |
| Data beings transferred by any means could contain sensitive information. |
| #include <cstddef>  Struct test {  Int a;  Char b;  Int c;  };  Extern int copy\_to\_user(void\*dest, void\*src, std::size\_t size);  Void do\_stuff(void\*usr\_buf) {  Test arg{1, 2, 3};  Copy\_to\_user(usr\_buf, &arg, sieof(arg));  } |

| **Compliant Code** |
| --- |
| Tried serializing the structure data before coping it which should prevent any issues. |
| #include <cstddef>  #include <cstring>  Struct test {  Int a;  Char b;  Int c;  };  Extern int copy\_to\_user(void\*dest, void\*src, std::size\_t size);  Void do\_stuff(void\*usr\_buf) {  Test arg{1, 2, 3};  Unsigned char buf[sizeof(arg)];  Std::size\_t offset = 0;  Std::memcpy(buf + offset, &arg.a, sizeof(arg.a));  Offset += sieof(arg.a);  Std::memcpy(buf + offset, &arg.b, sizeof(arg.b));  Offset += sizeof(arg.b);  Std::memcpy(buf + offset, &arg.c, sizeof(arg.c));  Offset += sizeof(arg.c);  Copy\_to\_user(usr\_buf, buf, offset /\* size of info copied \*/);  } |

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

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

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Axivion Bauhaus suite | 6.9.0 | Certc++-dcl55 |  |
| Parasoft c/c++test | 2020.2 | Cert-cpp-dcl55-a | Partially |
|  |  |  |  |
|  |  |  |  |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [STD-007-CPP] | Try to not use circles during the initialization of static objects. If a function is reentered during the initialization phase it could cause undefined behavior. |

| **Noncompliant Code** |
| --- |
| Attempting to implement factional function utilizing caching so that the initialization of the static array circle to see if this creates an undefined behavior. |
| #include <stdexcept>  Int fact(int i) noexcept(false) {  If (I < 0) {  Throw std::domain\_error(“I must be >==0”);  }  Static const int cache[] = {  Fact(0), fact(1), fact(2), fact(3), fact(4), fact(5),  Fact(6), fact(7), fact(8), fact(9), fact(10), fact(11),  Fact(12), fact(13), fact(14), fact(15), fact(16)  };  If(I < (sizeof(cache) / sizeof(int))) {  Return cache[i];  }  Return I > 0 ? I \* fact(I – 1) : 1;  } |

| **Compliant Code** |
| --- |
| Does not utilize the static cache which is the reason for the issue. |
| #include <stdexcept>  Int fact(int i) noexcept(false) {  If (I < 0) {  Throw std::domain\_error(“I must be >=0”);  }  Static int cache[18];  If (I < (sizeof(cache) / sizeof(int))) {  If (0 == cache[i] {  Cache[i] = I > 0 ? I \* fact(I – 1) : 1;  }  Return cache[i];  }  Return I > 0 ? I \* fact(I – 1) : 1;  } |

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

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

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Ldra tool suite | 9.7.1 | 6 d | Enhanced enforcement |
| Parasoft c/c++test | 2020.2 | Cert-cpp-dcl56-a | Partially hecked |
|  |  |  |  |
|  |  |  |  |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| [Student Choice] | [STD-008-CPP] | Cannot let exceptions escape from destructors or deallocation functions. |

| **Noncompliant Code** |
| --- |
| Class destructor might cause an exception and may cause undefined behavior. |
| #inlcude <stdexcept>  Class S {  Bool has\_error() const;  Public:  ~s() noexcept(false) {  If (has\_error()) {  Throw std::logic\_error(“something bad”);  }  }  }; |

| **Compliant Code** |
| --- |
| This should catch any exceptions and destroy them too. |
| Class someClass {  Bad bad\_member;    Public:  ~SomeClass()  Try {  } catch(…) {  Return;  }  }; |

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

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

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | Destructor-without-noexcept  Delete-without-noexcept | Fully checked |
| Axivion Bauhaus suite | 6.9.0 | Certc++-dcl57 |  |
| LDRA tool suite | 9.7.1 | 453 s | partially |
| Parasoft c/c++test | 2020.2 | Cert-cpp-dcl57-a  Cert-cpp-dcl57-b | Catch exceptions |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| [Student Choice] | [STD-009-CPP] | You should not touch the standard namespaces. Trying to input any new names into the namespaces might cause undefined behavior. |

| **Noncompliant Code** |
| --- |
| G is added to namespace to try and cause undefined behavior. |
| Namespace std {  Int G;  } |

| **Compliant Code** |
| --- |
| Putting a without reserved name this should not cause any undefined behavior. |
| Namespace nonstd {  Int G;  } |

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

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

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Axivion Bauhaus suite | 6.9.0 | Certc++dcl58 |  |
| Parasoft c/c++test | 2020.2 | Cert-cpp-dcl58-a | Do not modify the standard namespaces |
| Polyspace bug finder | R2020a | Cert c++: dcl58-cpp | Fully covered |
| PRQA QA-C++ | 4.4 | 4032, 4035, 4631 |  |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| [Student Choice] | [STD-010-CPP] | To cause an undefined behavior, trying to put an unnamed namespace in the header file. |

| **Noncompliant Code** |
| --- |
| The variable is defined in an unnamed namespace. This should cause undefined behavior. |
| #ifndef A\_HEADER\_FILE  #define A\_HEADER\_FILE  Namespace {  Int v;  }  #include “a.h”  #include <iostream>  Void f() {  Std::cout << “f()”: “<< v << std::end1;  V = 42;  }  #include “a.h”  #include <iostream>  Void g() {  Std::cout << “g(): “<< v << std::end1;  V = 100;  }  Int main() {  Extern void f();  F();  G();  F();  G();  } |

| **Compliant Code** |
| --- |
| Variable is defined by one translation unit that is visible to all and the results should be an expected output. |
| #ifndef A\_HEADER\_FILE  #define A\_HEADER\_FILE  Extern int v;  #include “a.h”  #include <iostream>  Int v;  Void f() {  Std::cout << “f(): “ << v << std::end1;  V = 42;  }  #include “a.h”  #include <iostream>  Void g() {  Std::cout << “g(): “<< v << std::end1;  V = 100;  }  Int main() {  Extern void f();  F();  G();  F();  G();  } |

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

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

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | Unnamed-namespace-header | Fully checked |
| Axivion Bauhaus suite | 6.9.0 | Certc++-dcl59 |  |
| Clang | 3.9 | Cert-dcl59-cpp | Checked |
| LDRA tool suite | 9.7.1 | 286 s, 512 s | Fully |

### 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.

[Insert your written explanations here.]

### Summary of Risk Assessments

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

| Rule | Severity | Likelihood | Remediation Cost | Priority | Level |
| --- | --- | --- | --- | --- | --- |
| STD-001-CPP | High | Unlikely | Medium | High | 2 |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |

### Create Policies for Encryption and Triple A

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

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

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

| 1. **Encryption** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Encryption in rest | [Insert text.] |
| Encryption at flight | [Insert text.] |
| Encryption in use | [Insert text.] |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | [Insert text.] |
| Authorization | [Insert text.] |
| Accounting | [Insert text.] |

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

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

### Map the Principles

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

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

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

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

## Audit Controls and Management

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

Evidence will include the following:

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

## Enforcement

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

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

## Exceptions Process

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

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

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

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

## Distribution

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

## Policy Change Control

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

## Policy Version History

| Version | Date | Description | Edited By | Approved By |
| --- | --- | --- | --- | --- |
| 1.0 | 08/05/2020 | Initial Template | David Buksbaum |  |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

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