**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 | This is where most security vulnerabilities are found. Making sure to always validate any data that doesn’t come from a trusted internal source can lead to major breaches of security. Breaking apart incoming data and then parsing it before re-combining it is a common practice to help prevent security issues. |
| 1. Heed Compiler Warnings | For code to actually function it must be compiled correctly. SO having checks and safety checks that are examining code as it is being compiled it’s another way to catch issues before the damage is done. |
| 1. Architect and Design for Security Policies | Having layers of privileges for different levels of data is a important way to minimize any possible breaches. If somehow a hack does get through but that information or privileges, they gain access to are minimal then so will be the damage. |
| 1. Keep It Simple | Having complicated or ineffective code can lead to vulnerabilities being lost in the lattice work of something that isn’t simple. Having simple generalized functions that have built in data security within them that can be used in many aspects of a coding project is a great example of having simple code that performs a task well and securely. |
| 1. Default Deny | Having the default of a system to be no power to modify the system or access it is an industry standard. Making those privileges be locked behind several sets of “keys” in most cases being a series of checks, makes it hard for attacks to pass all these checks without being caught. |
| 1. Adhere to the Principle of Least Privilege | Each time a process is executed, or something is modified, it should be done with the minimal number of privileges required. For example, a getter function for bringing data shouldn’t have the privileges to change a user’s login info. |
| 1. Sanitize Data Sent to Other Systems | This entails the permanent removal of data that is sent to other systems that isn’t required. If you only need to send a small portion of data to another system you need to sanitize the folder it is being sent to that permanently deletes the data to the point even advanced forensic tools can’t retrain it. Compared to standard deviation that just removes it from the data branches. |
| 1. Practice Defense in Depth | There is no catch all security procedures that will always be airtight. So, it is important to practice DID which is the concept of layering security protocols to minimize vulnerabilities. By having multiple layers of security that specialize in certain aspects and then have other protocols specialized in the previous protocol’s holes crash a powerful wall of security. |
| 1. Use Effective Quality Assurance Techniques | Using several different methodologies and people when reviewing code for security concerns results in more secure code. Like having DID be implemented in the coding practice this is having several layers of checking the actual security so that vulnerabilities that one product or person missed can be caught by another source. |
| 1. Adopt a Secure Coding Standard | Having a plan for how to keep your code secure is a important step before development. Knowing what language(s) you are using and how to keep them secure having precedent that will be followed through the development phase is important for security. |

### 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-DCL50-CPP | Do not define a C-style variadic function |

| **Noncompliant Code** |
| --- |
| Never use variadic functions because they have no way of checking how many incoming arguments or their data types, are being passed to the function resulting in a security risk. |
| #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** |
| --- |
| This code allows for the flexibility of a variadic function while also having the data be checked as its parsed and only accepting one data type. |
| #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):** ValidateInput Data |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 22.10 | Function-ellipsis | Fully Checked |
| Clang | 7.2.0 | Cert-dc150-cpp | Checked by clang-tidy |
| CodeSonar | 8.1p0 | LANG.STRUCT.ELLIPSIS | Ellipsis |
| Helix QAC | 2024.1 | C++2012,C++2625 |  |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | [STD-EXP50-CPP | ValidateInput Data |

| **Noncompliant Code** |
| --- |
| Using some shortcuts for modifying objects cause them to be modified twice in a single line rustling in confusion of sequencing order resulting in a security vulnerability. |
| // i is modified twice in the same full expression  i = ++i + 1;    // i is read other than to determine the value to be stored  a[i++] = i; |

| **Compliant Code** |
| --- |
| The order of evaluation is sperate from operand meaning this code can only be interpreted on one way when executing. |
| **void** f(**int** i, **const** **int** \*b) {    ++i;  **int** a = i + b[i];    // ...  } |

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

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

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Medium | Probable | Medium | P8 | L2 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Axicion Bauhayus Suite | 7.2.0 | CertC++-EXP50 |  |
| Clang | 3.9 | -Wunsequenced | Can detect simple violations of this rule where path-sensitive analysis is not required |
| CodeSonar | 8.1p0 | **LANG.STRUCT.SE.DEC** **LANG.STRUCT.SE.INC** | Side Effects in Expression with Decrement Side Effects in Expression with Increment |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Rose) |  |  | Can detect simple violations of this rule. It needs to examine each expression and make sure that no variable is modified twice in the expression. It also must check that no variable is modified once, then read elsewhere, with the single exception that a variable may appear on both the left and right of an assignment operator |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | [STD-STR51-CPP | Do not attempt to create a std::string from a null pointer |

| **Noncompliant Code** |
| --- |
| Trying to create a string from a function call that isn’t always populated with data can result in a undefined behavior because of the null pointer. |
| #include <cstdlib>  #include <string>    **void** f() {    std::string tmp(std::**getenv**("TMP"));  **if** (!tmp.empty()) {      // ...    }  } |

| **Compliant Code** |
| --- |
| This example checks that the called data exists in memory before calling it. It then parses the memory to populate the string tmp only if it exits. |
| #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):** ValidateInput Data |
| --- |

**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) | 22.10 | **assert\_failure** |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.1p0 | |  |  | | --- | --- | | **LANG.MEM.NPD** |  | | Null Pointer Dereference |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.1 | **DF4770, DF4771, DF4772, DF4773, DF4774** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2024.1 | **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** |  |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | [STD-FIO50-CPP | Do not alternately input and output from a file stream without an intervening position call |

| **Noncompliant Code** |
| --- |
| This data pushes data to the file and then reads data from the file before intervening allowing for SQL injections. |
| #include <fstream>  #include <string>    **void** f(**const** std::string &fileName) {    std::fstream file(fileName);  **if** (!file.is\_open()) {      // Handle error  **return**;    }      file << "Output some data";    std::string str;    file >> str;  } |

| **Compliant Code** |
| --- |
| This code pushes data to the file and then before reading from it then begins a new input stream from the file to allow for more secure data checking as its reading. |
| #include <fstream>  #include <string>    **void** f(**const** std::string &fileName) {    std::fstream file(fileName);  **if** (!file.is\_open()) {      // Handle error  **return**;    }      file << "Output some data";      std::string str;    file.seekg(0, std::ios::beg);    file >> str;  } |

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

| **Principles(s):** ValidateInput Data |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-FIO50** |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 8.1p0 | **IO.IOWOP**  **IO.OIWOP** | Input After Output Without Positioning  Output After Input Without Positioning |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.1 | **DF4711, DF4712, DF4713** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.1 | **CERT\_CPP-FIO50-a** | Do not alternately input and output from a stream without an intervening flush or positioning call |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [STD-MEM50-CPP | Do not access freed memory |

| **Noncompliant Code** |
| --- |
| This code reference freed memory which runs the code from the deallocated memory that could have been tampered with between the function call and the deallocation. |
| #include <new>    **struct** S {  **void** f();  };    **void** g() noexcept(**false**) {    S \*s = **new** S;    // ...  **delete** s;    // ...    s->f();  } |

| **Compliant Code** |
| --- |
| This memory is only deallocated after it has been used and is no longer needed. |
| #include <new>    **struct** S {  **void** f();  };    **void** g() noexcept(**false**) {    S \*s = **new** S;    // ...    s->f();  **delete** s;  } |

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

| **Principles(s):** Adhere to the Principle of Least Privilege |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | **dangling\_pointer\_use** |  |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-MEM50** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | clang-analyzer-cplusplus.NewDelete clang-analyzer-alpha.security.ArrayBoundV | Checked by clang-tidy, but does not catch all violations of this rule. |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 8.1p0 | **ALLOC.UAF** | Use after free |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | [STD-DCL03-C | Use a static assertion to test the value of constant expression |

| **Noncompliant Code** |
| --- |
| This assertion on runs when called in another function compared to at time of assertion or runtime until the function that is called is run. |
| #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** |
| --- |
| This solution allows for the assertion to be processed at runtime resulting in a check of the data while also not adding runtime space or cost. |
| #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):** Use Effective Quality Assurance Techniques |
| --- |

**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](https://wiki.sei.cmu.edu/confluence/display/c/Clang) | 3.9 | misc-static-assert | Checked by clang-tidy |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.1p0 | **(customization)** | Users can implement a custom check that reports uses of the assert() macro |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  | Could detect violations of this rule merely by looking for calls to assert(), and if it can evaluate the assertion (due to all values being known at compile time), then the code should use static-assert instead; this assumes ROSE can recognize macro invocation |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [STD-ERR60-CPP | Exception objects mist ne no throw copy constructible |

| **Noncompliant Code** |
| --- |
| This code doesn’t state and nonexempt meaning the exception is going to throw a copy of the exemption which can result in security and runtime risks. |
| #include <exception>  #include <string>    **class** S : **public** std::exception {    std::string m;  **public**:    S(**const** **char** \*msg) : m(msg) {}    **const** **char** \*what() **const** noexcept override {  **return** m.c\_str();    }  };    **void** g() {    // If some condition doesn't hold...  **throw** S("Condition did not hold");  }    **void** f() {  **try** {      g();    } **catch** (S &s) {      // Handle error    }  } |

| **Compliant Code** |
| --- |
| This code asserts that it cant throw a copy of the exception and asserts that it isn’t a copy. |
| #include <stdexcept>  #include <type\_traits>    **struct** S : std::runtime\_error {    S(**const** **char** \*msg) : std::runtime\_error(msg) {}  };    static\_assert(std::is\_nothrow\_copy\_constructible<S>::value,                "S must be nothrow copy constructible");    **void** g() {    // If some condition doesn't hold...  **throw** S("Condition did not hold");  }    **void** f() {  **try** {      g();    } **catch** (S &s) {      // Handle error    }  } |

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

| **Principles(s):** Heed Compiler Warnings |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | cert-err60-cpp | |  | | --- | | Checked by clang-tidy | |  | |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.1 | **C++3508** | [Insert text.] |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.1 | **CERT\_CPP-ERR60-a** **CERT\_CPP-ERR60-b** | Exception objects must be nothrow copy constructible An explicitly declared copy constructor for a class that inherits from 'std::exception' should have a non-throwing exception specification |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2023b | [CERT C++: ERR60-CPP](https://www.mathworks.com/help/bugfinder/ref/certcerr60cpp.html) | Checks for throwing exception object in copy constructor (rule fully covered) |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Data Type | [STD-INT30-C | Ensure that unsigned integer operators do not wrap |

| **Noncompliant Code** |
| --- |
| This code also for objects un\_a and Un\_b to have the possibility to wrap because their behavior is unexpected it can lead to a memory exploitable vulnerability. |
| **void** func(unsigned **int** ui\_a, unsigned **int** ui\_b) {    unsigned **int** usum = ui\_a + ui\_b;    /\* ... \*/  } |

| **Compliant Code** |
| --- |
| This code doesn’t allow for an overflow and checks that the unknown variables when populated behave within the allowed parameters to remove the memory vulnerability. |
| #include <limits.h>    **void** func(unsigned **int** ui\_a, unsigned **int** ui\_b) {    unsigned **int** usum;  **if** (UINT\_MAX - ui\_a < ui\_b) {      /\* Handle error \*/    } **else** {      usum = ui\_a + ui\_b;    }    /\* ... \*/  } |

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

| **Principles(s):** Input validation |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 23.04 | **integer-overflow** | |  | | --- | | Fully checked | |  | |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=125337650) | 7.2.0 | **CertC-INT30** | Implemented |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.1p0 | **ALLOC.SIZE.ADDOFLOW ALLOC.SIZE.IOFLOW ALLOC.SIZE.MULOFLOW ALLOC.SIZE.SUBUFLOW MISC.MEM.SIZE.ADDOFLOW MISC.MEM.SIZE.BAD MISC.MEM.SIZE.MULOFLOW MISC.MEM.SIZE.SUBUFLOW** | Addition overflow of allocation size Integer overflow of allocation size Multiplication overflow of allocation size Subtraction underflow of allocation size Addition overflow of size Unreasonable size argument Multiplication overflow of size Subtraction underflow of size |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  | Can detect violations of this rule by ensuring that operations are checked for overflow before being performed (Be mindful of exception INT30-EX2 because it excuses many operations from requiring [validation](https://wiki.sei.cmu.edu/confluence/display/c/BB.+Definitions#BB.Definitions-validation), including all the operations that would validate a potentially dangerous operation. For instance, adding two unsigned ints together requires validation involving subtracting one of the numbers from UINT\_MAX, which itself requires no validation because it cannot wrap.) |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| String Correctness | [STD-STR37-C | Arguments to character-handline functions must be representable as an unsigned char |

| **Noncompliant Code** |
| --- |
| This example doesn’t assign the memory location of t as a unsigned char allowing for a memory vulnerability because of string manipulation. |
| #include <ctype.h>  #include <string.h>    **size\_t** count\_preceding\_whitespace(**const** **char** \*s) {  **const** **char** \*t = s;  **size\_t** length = **strlen**(s) + 1;  **while** (**isspace**(\*t) && (t - s < length)) {      ++t;    }  **return** t - s;  } |

| **Compliant Code** |
| --- |
| This code casts the variable of t to be a unsigned char memory location meaning that it won’t possible pull from deallocated memory that can be abused and will allocate new memory for the function |
| #include <ctype.h>  #include <string.h>    **size\_t** count\_preceding\_whitespace(**const** **char** \*s) {  **const** **char** \*t = s;  **size\_t** length = **strlen**(s) + 1;  **while** (**isspace**((unsigned **char**)\*t) && (t - s < length)) {      ++t;    }  **return** t - s;  } |

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

| **Principles(s):** Input Validation |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 23.04 | **ctype-limits** | |  | | --- | | Partially checked | |  | |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-STR37** | Fully implemented |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.1p0 | **MISC.NEGCHAR** | Negative character value |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  | Could detect violations of this rule by seeing if the argument to a character handling function (listed above) is not an unsigned char |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Memory Manipulation | [STD-MEM34-C | Only free memory allocated dynamically |

| **Noncompliant Code** |
| --- |
| This code has c\_str be called by free() regardless of how the memory is allocated meaning that if it wasn’t dynamically allocate it could cause a heap corruption. |
| #include <stdlib.h>  #include <string.h>  #include <stdio.h>    **enum** { MAX\_ALLOCATION = 1000 };    **int** main(**int** argc, **const** **char** \*argv[]) {  **char** \*c\_str = NULL;  **size\_t** len;    **if** (argc == 2) {      len = **strlen**(argv[1]) + 1;  **if** (len > MAX\_ALLOCATION) {        /\* Handle error \*/      }      c\_str = (**char** \*)**malloc**(len);  **if** (c\_str == NULL) {        /\* Handle error \*/      }  **strcpy**(c\_str, argv[1]);    } **else** {      c\_str = "usage: $>a.exe [string]";  **printf**("%s\n", c\_str);    }  **free**(c\_str);  **return** 0;  } |

| **Compliant Code** |
| --- |
| This code checks that the memory is allocated correctly and if not it exits the statement before it has a chance to error out the heap. |
| #include <stdlib.h>  #include <string.h>  #include <stdio.h>    **enum** { MAX\_ALLOCATION = 1000 };    **int** main(**int** argc, **const** **char** \*argv[]) {  **char** \*c\_str = NULL;  **size\_t** len;    **if** (argc == 2) {      len = **strlen**(argv[1]) + 1;  **if** (len > MAX\_ALLOCATION) {        /\* Handle error \*/      }      c\_str = (**char** \*)**malloc**(len);  **if** (c\_str == NULL) {        /\* Handle error \*/      }  **strcpy**(c\_str, argv[1]);    } **else** {  **printf**("%s\n", "usage: $>a.exe [string]");  **return** EXIT\_FAILURE;    }  **free**(c\_str);  **return** 0;  } |

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

| **Principles(s):** Architect and Design for Security Policies |
| --- |

**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=87152428) | 23.04 | **invalid-free** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-MEM34** | Can detect memory deallocations for stack objects |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/c/Clang) | 3.9 | **clang-analyzer-unix.Malloc** | Checked by clang-tidy; can detect some instances of this rule, but does not detect all |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.1p0 | **ALLOC.TM** | Type Mismatch |

### Defense-in-Depth Illustration

This illustration provides a visual representation of the defense-in-depth best practice of layered security.



## Project One

### Automation

Provide a written explanation using the image provided.



Automation will be used for the enforcement of and compliance to the standards defined in this policy. Green Pace already has a well-established DevOps process and infrastructure. Define guidance on where and how to modify the existing DevOps process to automate enforcement of the standards in this policy. Use the DevSecOps diagram and provide an explanation using that diagram as context.

The risk assessment and automation should take place at the planning phase, end of sprints, development, and before launch. Risk management will be prevalent throughout the whole development process and you should have your case diagrams in place before developing code. This means you should know what the class shall be doing, what risks are implied with such a class, and the correct ways to prevent those vulnerabilities. Then upon completion of that piece of code, you should use any of the suggested third-party tools and methods to check for security vulnerabilities.

### 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 |
| [STD-DCL03-C | Low | Unlikely | High | P1 | L3 |
| [STD-INT30-C | High | Likely | High | **P9** | **L2** |
| [STD-MEM34-C | High | Likely | Medium | **P18** | **L1** |
| [STD-STR37-C | Low | Unlikely | Low | **P3** | **L3** |
| STD-DCL50-CPP | High | Probable | Medium | P12 | L1 |
| STD-MEM50-CPP | High | High | Medium | P18 | L1 |
| STD-FIO50-CPP | Low | Likely | Medium | P6 | L2 |
| STD-EXP50-CPP | Medium | Probable | Medium | P8 | L2 |
| STD-STR51-CPP | High | Likely | Medium | P18 | L1 |
| STD-ERR60-CPP | Low | Probable | Medium | **P4** | **L3** |

### Create Policies for Encryption and Triple A

| 1. **Encryption** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Encryption at rest | This means when the program is idle it isn’t taking up resources or still has access to outside data that could lead to vulnerabilities. If it isn’t being used it should have close access to anything outside itself. |
| Encryption in flight | Data is encrypted with a cipher or some other form of data manipulation that prevents catching the data. There should also be measures for the system that is expecting the data that they are checking that the data they received was actually the correct data and not a Trojan horse |
| Encryption in use | In use the program needs to make sure that it is consistently checking the data it is taking, and pushing out, and what parts of memory it is accessing aren’t vital that then are passed as commands to prevent SQL injections. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | This is the process of confirming that the actions being taken have the authority to do that action. This applies because it uses the principle of layered access so that not everything has access to each portion of a system. |
| Authorization | This allows for confirmation that the thing trying to ace information has clearance to get that information. This is important because it adds a layer of security to the data a system has. |
| Accounting | This is keeping count of the data and how it is manipulated. It is important so that you have an accurate count and tracking of data changes. |

**\***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 |