**Green Pace Developer: Security Policy Guide – Kerrian Offermann**



# 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 | Create methods of verifying that what the user inputs into a field contains only the necessary characters, and no other code that might threaten the system. Common input validation techniques include restricting character type (ex: alphanumeric only) and restricting character length. |
| 1. Heed Compiler Warnings | Listen to warnings from compilers after compiling code. The compiler should be set with settings that will search for and signal against important errors in codes. It is also recommended to find other tools that could find give additional warnings based on additional issues that one’s default compiler cannot find. |
| 1. Architect and Design for Security Policies | Essentially, one must design their code with security policies in mind. If a system has certain requirements to keep itself secure then that must be considered when writing code. These security policies include its architecture and design, thus the principle’s name. |
| 1. Keep It Simple | Complex code leads to an increase in errors; therefore, it is recommended that codes be kept as simple as possible. Extraneous code should be avoided so more time and effort can be focused on keeping it secure. |
| 1. Default Deny | The default deny principle means that access to the system should be denied from the start. If a user wants permission to access the system or certain privileges, they must be made to prove they meet the requirements needed instead of it being given to them by default. |
| 1. Adhere to the Principle of Least Privilege | Rather than grant every user access to every feature in the system, they should be given the minimum number of privileges possible to perform necessary tasks. This means a basic user should not be granted permission to make changes to the system unless it is absolutely necessary for their job or purpose. |
| 1. Sanitize Data Sent to Other Systems | When sending data to other systems, make it habit to only send what is necessary by removing extra information. If unnecessary data is sent to another system and then that system falls under attack, the attackers will also have data about other systems and use that to perform another attack. Only send what is needed via sanitizing data. |
| 1. Practice Defense in Depth | Practicing defense in depth means relying on more than one layer of security. This method assures that if one line of defense falls then there is another waiting for an attacker—and another and another, etc. Defense in depth also protects against flaws in the code so if an error is made then there is still some protection available. |
| 1. Use Effective Quality Assurance Techniques | In order to ensure that the system and its security measures are functioning as it should, there needs to be a process in place to continuously test the system for errors. This can include standard testing for errors to penetration tests to see how simple it is to hack into a system. Quality assurance also includes a way to regularly update security according to new threats/vulnerabilities. |
| 1. Adopt a Secure Coding Standard | Adopting a secure coding standard means always having a process to both teach and learn from. Regardless of the size of a team, a secure coding standard establishes what is acceptable in the different codes being written. |

### 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** | **Detect Errors When Converting a String to a Number** |
| --- | --- | --- |
| **Data Type** | ERR-062-CPP | Attempting to convert parse an integer from a string can lead to errors such as creating a number that is out of range or extra information after the number. |

| **Noncompliant Code** |
| --- |
| The noncompliant code shows multiple numeric values being converted from an input stream, but there is no guarantee that the input will be an “int” which can lead to errors. |
| #include <iostream>    void f() {    int i, j;    std::cin >> i >> j;    // ...  } |

| **Compliant Code** |
| --- |
| The compliant code has exceptions in place to catch conversion errors. “Badbit” and “failbit” will also check for loss of integrity and treat them as exceptions. |
| #include <iostream>    void f() {    int i, j;      std::cin.exceptions(std::istream::failbit | std::istream::badbit);    try {      std::cin >> i >> j;      // ...    } catch (std::istream::failure &E) {      // Handle error    }  } |

| **Principles(s):** (1) *Validate Input Data* – By limiting the input to what is allowed and rejecting what is not, errors such as entering the wrong data type can be prevented. |
| --- |

**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++-ERR62** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | cert-err34-c | Checked by clang-tidy; only identifies use of unsafe C Standard Library functions corresponding to ERR34-C |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.3p0 | **BADFUNC.ATOF BADFUNC.ATOI BADFUNC.ATOL BADFUNC.ATOLL** | Use of atof Use of atoi Use of atol Use of atoll |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2023.1 | **C++3161** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2023.1 | **CERT.ERR.CONV.STR\_TO\_NUM** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.2 | **CERT\_CPP-ERR62-a** | The library functions atof, atoi and atol from library stdlib.h shall not be used |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Value-Returning Functions Must Return a Value from All Exit Paths** |
| --- | --- | --- |
| **Data Value** | MSC-052-CPP | When using a return statement in a function, ensure that the value being returned includes as possible outcomes to avoid undefined behavior. |

| **Noncompliant Code** |
| --- |
| It is possible for the return value in this function to be both negative and positive. Therefore, returning only a negative value for “a” is not returning all exit paths. |
| int absolute\_value(int a) {    if (a < 0) {      return -a;    }  } |

| **Compliant Code** |
| --- |
| In order to make this code compliant, the option for a positive return value is added based on whether or not the value of a is below or above 0. |
| int absolute\_value(int a) {    if (a < 0) {      return -a;    }    return a;  } |

| **Principles(s):** (2) *Heed Compiler Warnings* – A compiler will usually pick up issues like these if the correct settings are used. There are also additional tools to pick up on warnings like these, so it is recommended to use them to check for correct return values. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | **return-implicit** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-MSC52** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | **-Wreturn-type** | Does not catch all instances of this rule, such as function-try-blocks |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.3p0 | **LANG.STRUCT.MRS** | Missing return statement |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2023.1 | **DF2888** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2023.1 | **FUNCRET.GEN**  **FUNCRET.IMPLICIT** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **2 D, 36 S** | Fully implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.2 | **CERT\_CPP-MSC52-a** | All exit paths from a function, except main(), with non-void return type shall have an explicit return statement with an expression |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2023a | [CERT C++: MSC52-CPP](https://www.mathworks.com/help/bugfinder/ref/certcmsc52cpp.html) | Checks for missing return statements (rule partially covered) |
| [SonarQube C/C++ Plugin](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046388) | 4.10 | [**S935**](https://www.sonarsource.com/products/codeanalyzers/sonarcfamilyforcpp/rules-cpp.html#RSPEC-935) |  |
| [PRQA QA-C++](https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=142409849) | 4.4 | **1510** |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.24 | [**V591**](https://pvs-studio.com/en/docs/warnings/v591/) |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 22.10 | **return-implicit** | Fully checked |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Guarantee That Storage for Strings Has Sufficient Space for Character Data and The Null Terminator** |
| --- | --- | --- |
| **String Correctness** | STR-050-CPP | Avoid buffer overflows by ensuring that that the buffer is capable of handling the amount of data being send to it. |

| **Noncompliant Code** |
| --- |
| “buf[12]” is only capable of holding 12 characters of data, yet there is no limit in place for the input to guarantee that only 12 or less bytes of data will be sent each time. |
| #include <iostream>    void f() {    char buf[12];    std::cin >> buf;  } |

| **Compliant Code** |
| --- |
| To make the code compliant, std::string is used instead of a bounded array to make sure that data is not shortened or that a buffer overflow does not occur. |
| #include <iostream>  #include <string>    void f() {    std::string input;    std::string stringOne, stringTwo;    std::cin >> stringOne >> stringTwo;  } |

| **Principles(s):** (1) *Validate Input Data, (2) Heed Compiler Warnings* – buffer overflows can lead to many problems in a system. Therefore, having a method in place to prevent users from overflowing the buffer and then another to stop the programmer from programming it would apply to this issue. |
| --- |

**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 | **stream-input-char-array** | Partially checked + soundly supported |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.3p0 | **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) | 2023.1 | **C++5216**  **DF2835, DF2836, DF2839,** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2023.1 | **NNTS.MIGHT** **NNTS.TAINTED** **NNTS.MUST** **SV.UNBOUND\_STRING\_INPUT.CIN** |  |
| [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) | 2022.2 | **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 terminated 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) | R2023a | [CERT C++: STR50-CPP](https://www.mathworks.com/help/bugfinder/ref/certcstr50cpp.html) | Checks for:   * Use of dangerous standard function * Missing null in string array * Buffer overflow from incorrect string format specifier * Destination buffer overflow in string manipulation * Insufficient destination buffer size   Rule partially covered. |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 22.10 | **stream-input-char-array** | Partially checked |
| [SonarQube C/C++ Plugin](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046388) | 4.10 | [**S3519**](https://www.sonarsource.com/products/codeanalyzers/sonarcfamilyforcpp/rules-cpp.html#RSPEC-3519) |  |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Exclude User Input from Format Strings** |
| --- | --- | --- |
| **SQL Injection** | FIO-030-CPP | Prevent attacks like SQL injections by not allowing users to control contents of a format string and refrain from calling a formatted input/output function with a tainted value in its format string. |

| **Noncompliant Code** |
| --- |
| This function that is meant to check the user and password, and display an error message if they are not found, will accept the name of the user as a string and then output it when “fprintf()” is called. |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>    void incorrect\_password(const char \*user) {    int ret;    /\* User names are restricted to 256 or fewer characters \*/    static const char msg\_format[] = "%s cannot be authenticated.\n";    size\_t len = strlen(user) + sizeof(msg\_format);    char \*msg = (char \*)malloc(len);    if (msg == NULL) {      /\* Handle error \*/    }    ret = snprintf(msg, len, msg\_format, user);    if (ret < 0) {      /\* Handle error \*/    } else if (ret >= len) {      /\* Handle truncated output \*/    }    fprintf(stderr, msg);    free(msg);  } |

| **Compliant Code** |
| --- |
| By replacing “fprintf()” with “fputs()”, the output will be a standard error message created by the programmer rather than strings created by a possibly unauthorized user. |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>    void incorrect\_password(const char \*user) {    int ret;    /\* User names are restricted to 256 or fewer characters \*/    static const char msg\_format[] = "%s cannot be authenticated.\n";    size\_t len = strlen(user) + sizeof(msg\_format);    char \*msg = (char \*)malloc(len);    if (msg == NULL) {      /\* Handle error \*/    }    ret = snprintf(msg, len, msg\_format, user);    if (ret < 0) {      /\* Handle error \*/    } else if (ret >= len) {      /\* Handle truncated output \*/    }    fputs(msg, stderr);    free(msg);  } |

| **Principles(s):** *(1)* *Validate Input Data, (7) Sanitize Data Sent to Other Systems, (8) Practice Defense in Depth* – In order to stop SQL injections, one of the beginning steps is to prevent users from entering certain characters in the text fields provided. Once the login data is sent from one part of the system to another, it should be sanitized and stripped of any unnecessary data that does not apply to logging in. Finally, practicing defense in depth will establish a second, third, fourth, etc. barrier for the login information sent to ensure that it is a legitimate attempt to log into the system and not a malicious attack. |
| --- |

**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) | 22.04 |  | Supported via stubbing/taint analysis |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-FIO30** | Partially implemented |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.3p0 | **IO.INJ.FMT MISC.FMT** | Format string injection Format string |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  |  |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **TAINTED\_STRING** | Implemented |
| [GCC](https://wiki.sei.cmu.edu/confluence/display/c/GCC) | 4.3.5 |  | Can detect violations of this rule when the -Wformat-security flag is used |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2023.1 | **DF4916, DF4917, DF4918** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2023.1 | **SV.FMTSTR.GENERIC SV.TAINTED.FMTSTR** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **86 D** | Partially Implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2022.2 | **CERT\_C-FIO30-a** **CERT\_C-FIO30-b** **CERT\_C-FIO30-c** | Avoid calling functions printf/wprintf with only one argument other than string constant Avoid using functions fprintf/fwprintf with only two parameters, when second parameter is a variable Never use unfiltered data from an untrusted user as the format parameter |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **592** | Partially supported: reports non-literal format strings |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2023a | [CERT C: Rule FIO30-C](https://www.mathworks.com/help/bugfinder/ref/certcrulefio30c.html) | Checks for tainted string format (rule partially covered) |
| [PRQA QA-C](https://wiki.sei.cmu.edu/confluence/display/c/PRQA+QA-C) | 9.7 | **4916, 4917, 4918** |  |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | **4916, 4917, 4918** |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/c/PVS-Studio) | 7.24 | [**V618**](https://pvs-studio.com/en/docs/warnings/v618/) |  |
| [Splint](https://wiki.sei.cmu.edu/confluence/display/c/Splint) | 3.1.1 |  |  |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Do Not Access Freed Memory** |
| --- | --- | --- |
| **Memory Protection** | MEM-030-CPP | Pointers evaluated into memory that no longer exists or has been deallocated leads to issues such as undefined behavior and exploitable vulnerabilities. |

| **Noncompliant Code** |
| --- |
| In this noncompliant code, “s” has been deleted yet it is still being referenced post-deallocation. This can be exploited to run arbitrary code with permissions that once belonged to the valid “s” variable that was used. |
| #include <new>    struct S {    void f();  };    void g() noexcept(false) {    S \*s = new S;    // ...    delete s;    // ...    s->f();  } |

| **Compliant Code** |
| --- |
| To make this code compliant, “s” is not deleted from memory until it is no longer needed. Therefore, pointers will no longer evaluate a deallocated memory where “s” should be. |
| #include <new>    struct S {    void f();  };    void g() noexcept(false) {    S \*s = new S;    // ...    s->f();    delete s;  } |

| **Principles(s):** *(2)* *Heed Compiler Warnings, (9) Use Effective Quality Assurance Techniques* – When it comes to pointers and pointing to memory that no longer exists, compilers and other tools can pick up this error with the right settings. Even so, some quality assurance should be done through outside means (ex: reverse coding) to ensure that memory is not accessed after deallocation or that multiple pointers are not attempting to store memory in the same place. |
| --- |

**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) | 22.04 | **dangling\_pointer\_use** | Supported  Astrée reports all accesses to freed allocated memory. |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-MEM30** | Detects memory accesses after its deallocation and double memory deallocations |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.3p0 | **ALLOC.UAF** | Use after free |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  |  |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **USE\_AFTER\_FREE** | Can detect the specific instances where memory is deallocated more than once or read/written to the target of a freed pointer |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2023.1 | **DF4866, DF4867, DF4868, DF4871, DF4872, DF4873**  **C++3339, C++4303, C++4304** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2023.1 | **UFM.DEREF.MIGHT** **UFM.DEREF.MUST** **UFM.FFM.MIGHT** **UFM.FFM.MUST** **UFM.RETURN.MIGHT** **UFM.RETURN.MUST** **UFM.USE.MIGHT** **UFM.USE.MUST** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **51 D, 484 S, 112 D** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2022.2 | **CERT\_C-MEM30-a** | Do not use resources that have been freed |
| [Parasoft Insure++](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) |  |  | Runtime analysis |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **449, 2434** | Fully supported |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2023a | [CERT C: Rule MEM30-C](https://www.mathworks.com/help/bugfinder/ref/certcrulemem30c.html) | Checks for:   * Accessing previously freed pointer * Freeing previously freed pointer   Rule partially covered. |
| [PRQA QA-C](https://wiki.sei.cmu.edu/confluence/display/c/PRQA+QA-C) | 9.7 | **2731, 2732, 2733** |  |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | **3339, 4303, 4304** |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/c/PVS-Studio) | 7.24 | [**V586**](https://pvs-studio.com/en/docs/warnings/v586/), [**V774**](https://pvs-studio.com/en/docs/warnings/v774/) |  |
| [Splint](https://wiki.sei.cmu.edu/confluence/display/c/Splint) | 3.1.1 |  |  |
| [TrustInSoft Analyzer](https://wiki.sei.cmu.edu/confluence/display/c/TrustInSoft+Analyzer) | 1.38 | **dangling\_pointer** | Exhaustively verified (see [one compliant and one non-compliant example](https://taas.trust-in-soft.com/tsnippet/t/0d556bb8)). |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Avoid Side Effects in Arguments to Unsafe Macros** |
| --- | --- | --- |
| **Assertions** | PRE-031-C (Also Applies to C++) | Do not call an unsafe macro with arguments containing an assignment, increment, decrement, input/output, and other expressions with side effects. |

| **Noncompliant Code** |
| --- |
| The “assert()” macro can be used to test arguments and expressions in a convenient fashion. However, in this code it is used with an expression that increments (index++) what will lead to side effects. |
| #include <assert.h>  #include <stddef.h>    void process(size\_t index) {    assert(index++ > 0); /\* Side effect \*/    /\* ... \*/  } |

| **Compliant Code** |
| --- |
| In order to make this code compliant, the incremented “index++” was taken outside of the macro since it has side effects, and placed elsewhere. The “assert()” macro is now using an expression with no side effect. |
| #include <assert.h>  #include <stddef.h>    void process(size\_t index) {    assert(index > 0); /\* No side effect \*/    ++index;    /\* ... \*/  } |

| **Principles(s):** *(4)* *Keep It Simple, (2) Heed Compiler Warnings* – attempting to complicate codes by doing things such as adding multiple tasks such as incrementing and comparing inside a macro. This can be applied to this standard by encouraging users to consider more sensible ways to perform a task. Additionally, having compilers and other error-checking tools on hand to ensure the simplifying process works can add an extra layer of security. |
| --- |

**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) | 22.04 | **expanded-side-effect-multiplied** **expanded-side-effect-not-evaluated** **side-effect-not-expanded** | Partially checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-PRE31** | Fully implemented |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.3p0 | **LANG.PREPROC.FUNCMACRO** **LANG.STRUCT.SE.DEC** **LANG.STRUCT.SE.INC** | Function-Like Macro Side Effects in Expression with Decrement Side Effects in Expression with Increment |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **ASSERT\_SIDE\_EFFECTS** | Partially implemented  Can detect the specific instance where assertion contains an operation/function call that may have a side effect |
| [ECLAIR](https://wiki.sei.cmu.edu/confluence/display/c/ECLAIR) | 1.2 | **CC2.EXP31 CC2.PRE31** | Fully implemented |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2023.1 | **C3462, C3463, C3464,C3465,C3466,C3467**  **C++3225, C++3226, C++3227, C++3228, C++3229** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2023.1 | **PORTING.VAR.EFFECTS** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **9 S, 562 S, 572 S, 35 D, 1 Q** | Fully implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2022.2 | **CERT\_C-PRE31-b** **CERT\_C-PRE31-c** **CERT\_C-PRE31-d** | Assertions should not contain assignments, increment, or decrement operators Assertions should not contain function calls nor function-like macro calls Avoid side effects in arguments to unsafe macros |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **666, 2666** | Fully supported |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2023a | [CERT C: Rule PRE31-C](https://www.mathworks.com/help/bugfinder/ref/certcrulepre31c.html) | Checks for side effect in arguments to unsafe macro (rule partially covered) |
| [PRQA QA-C](https://wiki.sei.cmu.edu/confluence/display/c/PRQA+QA-C) | 9.7 | **3462, 3463, 3464, 3465, 3466, 3467** | Fully implemented |
| [PRQA QA-C++](https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=142409849) | 4.4 | **3225, 3226, 3227, 3228, 3229** |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/c/RuleChecker) | 22.04 | **expanded-side-effect-multiplied** **expanded-side-effect-not-evaluated** **side-effect-not-expanded** | Partia |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Handle All Exceptions** |
| --- | --- | --- |
| **Exceptions** | ERR-051-CPP | Ensure that there is a handler after an exception is thrown to prevent abnormal process termination and denial-of-service attacks. |

| **Noncompliant Code** |
| --- |
| This code is noncompliant because there are no handlers in either function to catch the exceptions that throwing\_func() will throw if an error occurs. This will lead to an unplanned termination of the process. |
| void throwing\_func() noexcept(false);    void f() {    throwing\_func();  }    int main() {    f();  } |

| **Compliant Code** |
| --- |
| Int main() is now programmed with a method of handling exceptions thrown. This prevents processes being ended by abnormal terminations and improves management of external resources. |
| void throwing\_func() noexcept(false);    void f() {    throwing\_func();  }    int main() {    try {      f();    } catch (...) {      // Handle error    }  } |

| **Principles(s):** *(9)* *Use Effective Quality Assurance Techniques* – the lack of a handler is typically a sign of misunderstanding or an oversight in the programmer. Therefore, having measures in place to assure code quality and scan codes for errors like these are important to this coding standard. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | **main-function-catch-all early-catch-all** | Partially checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-ERR51** |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.3p0 | **LANG.STRUCT.UCTCH** | Unreachable Catch |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2023.1 | **C++4035, C++4036, C++4037** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2023.1 | **MISRA.CATCH.ALL** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **527 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.2 | **CERT\_CPP-ERR51-a** **CERT\_CPP-ERR51-b** | Always catch exceptions Each exception explicitly thrown in the code shall have a handler of a compatible type in all call paths that could lead to that point |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2023a | [CERT C++: ERR51-CPP](https://www.mathworks.com/help/bugfinder/ref/certcerr51cpp.html) | Checks for unhandled exceptions (rule partially covered) |
| [PRQA QA-C++](https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=142409849) | 4.4 | **4035, 4036, 4037** |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 22.10 | **main-function-catch-all early-catch-all** | P |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Never Hard Code Sensitive Information** |
| --- | --- | --- |
| **Sensitive Information** | MSC-041-C (Also Applies to C++) | Never hard code sensitive information such as passwords or encryption keys. Anyone with access to executable or dynamic library files will be able to find them. In some cases, this can have legal consequences. |

| **Noncompliant Code** |
| --- |
| When this function authenticates a remote service for a code, it passes the code to the function as a string literal instead of protecting the information. Now it can be accessed by someone who does not need access to it. |
| /\* Returns nonzero if authenticated \*/  int authenticate(const char\* code);    int main() {    if (!authenticate("correct code")) {      printf("Authentication error\n");      return -1;    }      printf("Authentication successful\n");    // ...Work with system...    return 0;  } |

| **Compliant Code** |
| --- |
| The code is made compliant by requiring the user to supply the code and then erasing it once it is no longer needed. The code is not saved to any place where it can be accessed later by someone else. |
| /\* Returns nonzero if authenticated \*/  int authenticate(const char\* code);    int main() {  #define CODE\_LEN 50    char code[CODE\_LEN];    printf("Please enter your authentication code:\n");    fgets(code, sizeof(code), stdin);    int flag = authenticate(code);    memset\_s(code, sizeof(code), 0, sizeof(code));    if (!flag) {      printf("Access denied\n");      return -1;    }    printf("Access granted\n");    // ...Work with system...    return 0;  } |

| **Principles(s):** *(6)* *Adhere to the Principle of Least Privilege, (5) Default Deny, (9) Use Effective Quality Assurance Techniques, (10) Adopt a Secure Coding Standard* – When it comes to the coding standard of hardcoding passwords, training programmers not to add private information is the first step. However, in the event that this error is made, limiting who has access to the sensitive information in a code is one method of keeping the information from spreading. Whether it is tools or more experienced programmers in the system, QA should be done to sweep code for hardcoded passwords. Finally, rules should be established so the entire team understands that hardcoding passwords has dangerous consequences. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.3p0 | **HARDCODED.AUTH HARDCODED.DNS HARDCODED.KEY HARDCODED.SALT HARDCODED.SEED** | Hardcoded Authentication Hardcoded DNS Name Hardcoded Crypto Key Hardcoded Crypto Salt Hardcoded Seed in PRNG |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2023.1 | **C3122**  **C++3842** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2023.1 | **HCC** **HCC.PWD** **HCC.USER** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2022.2 | **CERT\_C-MSC41-a** | Do not hard code string literals |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **2460** | Assistance provided: reports when a literal is provided as an argument to a function parameter with the ‘noliteral’ argument Semantic; several Windows API functions are marked as such and the ‘-sem’ option can apply it to other functions as appropriate |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2023a | [CERT C: Rule MSC41-C](https://www.mathworks.com/help/bugfinder/ref/certcrulemsc41c.html) | Checks for hard coded sensitive data (rule partially co |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Do Not Abruptly Terminate the Program** |
| --- | --- | --- |
| **Proper Termination** | ERR-050-CPP | Before terminating a program, ensure that open streams with unwritten buffered data is flushed, open streams are closed, and temporary files are removed. Otherwise, it can lead to corrupted external files and other critical errors. |

| **Noncompliant Code** |
| --- |
| If “f()” is called in this noncompliant example, the exit handler (“std::at\_exit()”) can lead to std::terminate if an exception is thrown. Any initialized objects being now cause errors since the program was not closed correctly. |
| #include <cstdlib>    void throwing\_func() noexcept(false);    void f() { // Not invoked by the program except as an exit handler.    throwing\_func();  }    int main() {    if (0 != std::atexit(f)) {      // Handle error    }    // ...  } |

| **Compliant Code** |
| --- |
| To make the code compliant, it is rewritten so all exceptions are thrown in f() and rethrows do not lead to an improper termination of the program. |
| #include <cstdlib>    void throwing\_func() noexcept(false);    void f() { // Not invoked by the program except as an exit handler.    try {      throwing\_func();    } catch (...) {      // Handle error    }  }    int main() {    if (0 != std::atexit(f)) {      // Handle error    }    // ...  } |

| **Principles(s):** *(3)* *Architect and Design for Security Policies, (2) Heed Compiler Warnings* – Building a system with security policies in mind applies to this coding practice because prematurely terminating a program can lead to problems within the system that can hinder an entire team. Programmers should be encouraged to always check if the file is closed properly, and compilers should be set to remind them in the event that this is overlooked. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | **stdlib-use** | Partially checked |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.3p0 | **BADFUNC.ABORT BADFUNC.EXIT** | Use of abort Use of exit |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2023.1 | **C++5014** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2023.1 | **MISRA.TERMINATE** **CERT.ERR.ABRUPT\_TERM** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **122 S** | Enhanced Enforcement |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.2 | **CERT\_CPP-ERR50-a** **CERT\_CPP-ERR50-b** **CERT\_CPP-ERR50-c** **CERT\_CPP-ERR50-d** **CERT\_CPP-ERR50-e** **CERT\_CPP-ERR50-f** **CERT\_CPP-ERR50-g** **CERT\_CPP-ERR50-h** **CERT\_CPP-ERR50-i** **CERT\_CPP-ERR50-j** **CERT\_CPP-ERR50-k** **CERT\_CPP-ERR50-l** **CERT\_CPP-ERR50-m CERT\_CPP-ERR50-n** | The execution of a function registered with 'std::atexit()' or 'std::at\_quick\_exit()' should not exit via an exception Never allow an exception to be thrown from a destructor, deallocation, and swap Do not throw from within destructor There should be at least one exception handler to catch all otherwise unhandled exceptions An empty throw (throw;) shall only be used in the compound-statement of a catch handler Exceptions shall be raised only after start-up and before termination of the program Each exception explicitly thrown in the code shall have a handler of a compatible type in all call paths that could lead to that point Where a function's declaration includes an exception-specification, the function shall only be capable of throwing exceptions of the indicated type(s) Function called in global or namespace scope shall not throw unhandled exceptions Always catch exceptions Properly define exit handlers The 'abort()' function from the 'stdlib.h' or 'cstdlib' library shall not be used Avoid throwing exceptions from functions that are declared not to throw The 'quick\_exit()' and '\_Exit()' functions from the 'stdlib.h' or 'cstdlib' library shall not be used |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2023a | [CERT C++: ERR50-CPP](https://www.mathworks.com/help/bugfinder/ref/certcerr50cpp.html) | Checks for implicit call to terminate() function (rule partially covered) |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | **5014** |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.24 | [**V667**](https://pvs-studio.com/en/docs/warnings/v667/)**,** [**V2014**](https://pvs-studio.com/en/docs/warnings/v2014/) |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 22.10 | **stdlib-use** | Partially checked |
| [SonarQube C/C++ Plugin](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046388) | 4.10 | [**S990**](https://www.sonarsource.com/products/codeanalyzers/sonarcfamilyforcpp/rules-cpp.html#RSPEC-990) |  |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Close Files When They Are No Longer Needed** |
| --- | --- | --- |
| **File Handling** | FIO-051-CPP | Every call made to open a file must be matched with a call to close the file when no longer needed. Not closing a file before the program terminates can lead to unexpected errors with the external file and/or the program itself. |

| **Noncompliant Code** |
| --- |
| There is a call in this code to open a file that does not have a matching call to close it. Therefore, if an exception is thrown this program will terminate abruptly with the file still being open. |
| #include <exception>  #include <fstream>  #include <string>    void f(const std::string &fileName) {    std::fstream file(fileName);    if (!file.is\_open()) {      // Handle error      return;    }    // ...    std::terminate();  } |

| **Compliant Code** |
| --- |
| By adding code to close the file before terminating the program in the event of an exception, the program will always close the file first and exit second to prevent errors. |
| #include <exception>  #include <fstream>  #include <string>    void f(const std::string &fileName) {    std::fstream file(fileName);    if (!file.is\_open()) {      // Handle error      return;    }    // ...    file.close();    if (file.fail()) {      // Handle error    }    std::terminate();  } |

| **Principles(s):** *(2) Heed Compiler Warnings, (9) Use Effective Quality Assurance Techniques* – Compilers and other tools can be set to uphold this coding standard by notifying programmers when a file was opened but not closed. In the event that this somehow overlooked, though, QA measures should be put in place to ensure that all opened files are closed at some point within the code before running. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.3p0 | **ALLOC.LEAK** | Leak |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2023.1 | **DF4786, DF4787, DF4788** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2023.1 | **RH.LEAK** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.2 | **CERT\_CPP-FIO51-a** | Ensure resources are freed |
| [Parasoft Insure++](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) |  |  | Runtime detection |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2023a | [CERT C++: FIO51-CPP](https://www.mathworks.com/help/bugfinder/ref/certcfio51cpp.html) | Checks for resource leak (rule partially covered) |

### Defense-in-Depth Illustration

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



## Project One

There are seven steps outlined below that align with the elements you will be graded on in the accompanying rubric. When you complete these steps, you will have finished the security policy.

### Revise the C/C++ Standards

You completed one of these tables for each of your standards in the Module Three milestone. In Project One, add revisions to improve the explanation and examples as needed. Add rows to accommodate additional examples of compliant and noncompliant code. Coding standards begin on the security policy.

### Risk Assessment

Complete this section on the coding standards tables. Enter high, medium, or low for each of the headers, then rate it overall using a scale from 1 to 5, 5 being the greatest threat. You will address each of the seven policy standards. Fill in the columns of severity, likelihood, remediation cost, priority, and level using the values provided in the appendix.

### Automated Detection

Complete this section of each table on the coding standards to show the tools that may be used to detect issues. Provide the tool name, version, checker, and description. List one or more tools that can automatically detect this issue and its version number, name of the rule or check (preferably with link), and any relevant comments or description—if any. This table ties to a specific C++ coding standard.

### Automation

Provide a written explanation using the image provided.



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

Green Pace already has a solid Automation plan in mind in terms of assessing security measures, implementing them, and then keeping their system up to par. However, it would help their team to include regular training sessions for all employees as a part of the automation process. Every person on the team from the testers to the head of security to HR should be on the same page in terms of the security process, new threats, and how to escalate spotting these threats to the right party/parties. Some companies, for example, regularly send fake phishing emails to check which employees are opening them and which ones are reporting them. Measures like these help in addressing the unintentional security threats that occur within the company itself instead of only focusing on outside threats or only training members of the security team on learning about threats. If possible, this step can be added between “Assess and Plan” and “Design.” DevSecOps can assess what new threats are out there before providing company-wide measures of taking steps to protect against these threats. Once that information is shared, then they can begin to work on designing a system that will hopefully keep these threats away from Green Pace.

### Summary of Risk Assessments

| Rule | Severity | Likelihood | Remediation Cost | Priority | Level |
| --- | --- | --- | --- | --- | --- |
| ERR-062-CPP | Medium | Unlikely | Medium | P4 | L3 |
| MSC-052-CPP | Medium | Probable | Medium | P8 | L2 |
| STR-050-CPP | High | Likely | Medium | P18 | L1 |
| FIO-030-CPP | High | Likely | Medium | P18 | L1 |
| MEM-030-CPP | High | Likely | Medium | P18 | L1 |
| PRE-031-C | Low | Unlikely | Low | P3 | L3 |
| ERR-051-CPP | Low | Probable | Medium | P4 | L3 |
| MSC-041-C | High | Probable | Medium | P12 | L1 |
| ERR-050-CPP | Low | Probable | Medium | P4 | L3 |
| FIO-051-CPP | Medium | Unlikely | Medium | P4 | AuL3 |

### Create Policies for Encryption and Triple A

| 1. **Encryption** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Encryption at rest | Encryption at rest is encryption used to protect data that is stored/on a disc. When data at rest is encrypted, an algorithm is used with randomized code that requires a key to be solved. If the person attempting to access the disc does not have the correct key to match the encrypted code, then they cannot access the file. This policy should be used because whether it is a file, a part of the storage, or a specific user profile, there should be measures in place to ensure that only those who need access are given access. For example, folders with private information should not be accessed by anyone but the admin and those selected by the admin. Therefore, encryption at rest should be used to prevent anyone else from having access. |
| Encryption in flight | Encryption in flight is encryption used to protect data being sent over a network such as an email. Typically, the network where the data is being sent from is protected along with the network receiving the data. In order for the two systems to communicate, they must possess the required key to solve the encryption waiting on the other side. This can prevent information from being sent or received without authorization. An encryption in flight policy should be used because one of the most common ways that systems become infected is through a virus or malware being sent from the internet to a person’s system. Having a defense measure in place for received files, especially, is important. |
| Encryption in use | Encryption in use is encryption for files that are being accessed/read. When a file is opened, any temporary files can be encrypted while the file itself only allows authorized users to access it. The encryption in use policy is also useful when it comes to files that are left open with sensitive information. Files like these can be scrambled so even someone unauthorized accesses them, they will not be able to extract any private information. This policy should be used to protect opened files for those who are allowed to read them, and to make it useless to those who should not have access to them. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | Authentication is the process of ensuring that users attempting to access a system are who they say they are. This is typically done through user logins, but it is highly recommended that security measures such as text/email verifications be put in place as an added layer of defense against unauthorized users. This policy should be used because it essentially filters users who are “safe” and prevents those who are not from using another person’s data to access a system. |
| Authorization | Authorization is the process of granting privileges to users in a system. Typically, the principle of “least privilege” is applied where users are only granted privileges necessary to perform their jobs. Additionally, new users to the database are essentially expected to earn more privileges such as changing a database or creating more new users through an administrator rather than having these abilities by default. A policy like this should be used to prevent unauthorized users from making major changes to the database unless they are given the proper training and are proven to be trustworthy of having a higher level of access to the company’s database. |
| Accounting | Accounting is the process of checking activities, users, changes, etc. in the system by having a tracking system set in place. It is important to keep track of what happens inside of a system and make strange behavior known to the security team in the event that someone unauthorized is not accessing (or attempting to access) information they are not privy to. This policy should be used as a security measure because a reliable sign that a user is not to be trusted is their attempts—failed or otherwise—to access files that they are not required to access. For example, someone working in sales should not be attempting to access HR files containing the personal information of their coworkers. Accounting allows administrators to see behavior like this and react accordingly before real damage is done. |

### Map the Principles

**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 |  |
| 1.1 | 03/13/2023 | Milestone Three | Kerrian Offermann |  |
| 1.2 | 04/6/2023 | Project One | Kerrian Offermann |  |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

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