**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 | The importance of validating input data is that it can open up a network for a slew of vulnerabilities. It is important to ensure that all inherently untrusted input is from trusted sources, and contains legitimate data. External sources have the ability to plant malicious code, program files and variables that could lead to data breaches, outages, and an overall unsecured network. It is important that all data coming into the system is valid and can be trusted to be within the network. |
| 1. Heed Compiler Warnings | Compiler warnings are issued when the compiler of the program pinpoints errors within the code. Though this won’t stop the process, it makes the developer aware of issues within the code. It essentially tells the developer that there is strange code syntax in the development or that it may crash. Using analysis tools, both static or dynamic, can ensure that there are no security vulnerabilities associated with these warnings. |
| 1. Architect and Design for Security Policies | Think of the architect and design of security policies as you would an actual structure. Consider the ease of access to the main floor, and then how the security could be bumped up from level to level based on need of access and authorization. It should be said that your security measures within code should be implemented in the same way. It is important to visualize the levels of security, and lay the frame work from the ground up broken into communication sets and priority for access. |
| 1. Keep It Simple | This is an age old tale in the world of developing. While it can be easy to get carried away in the lines of code, it is important to maintain the readability of the code and to lessen the potential for errors. If a code is kept simple, it is easily read, and therefore more easily debugged. The more complicated the code, the more room for error and potential vulnerabilities that can be exploited. |
| 1. Default Deny | The concept behind this is simple. It is better to deny everyone access and let in only the vetted users than in is to let everyone in and try to chase the bad ones out. In the structural development of security policies and access, be sure to delineate who is allowed access based on specific criterion, and to what level. |
| 1. Adhere to the Principle of Least Privilege | This, too, is a concept that can be carried across all realms of cyber security as well as into the real world. Access to files and programs should be based on the direct need of the user to complete their respective tasks. They should be given access to the files directly related to their current job. If in the off chance that elevated access be needed, this should be done in a monitored setting and only for the amount of time needed to access pertinent information. This will keep the opportunity for attackers to reach highly sensitive information to a minimum. |
| 1. Sanitize Data Sent to Other Systems | It is important to sanitize outgoing data before it leaves its origin. By ensuring that there is not an proprietary or potentially dangerous information in the data sent, it will ensure that it will not get into the wrong hands if intercepted. |
| 1. Practice Defense in Depth | Think of this as layers of protection through a fort. If the first layer of protection is breached, here is another line of defense waiting in wake for the next strike. This is the same concept as Defense in Depth. If an attacker manages to make it through the first line of defense, they will have more hurdles to jump over before they reach their payload. It is important however, to remember not to take it too far. While you want your code to be as secure as possible, you still want it to be simple, modular, and readable, as always. Don’t weigh down and convolute good code with unnecessary bloat. |
| 1. Use Effective Quality Assurance Techniques | While quality assurance is always the best practice in coding, after all it is best to present quality work, it can also act as another check and balance for the security of your code. By running constant tests and audits, not only will you ensue that you are producing quality work, but they can pinpoint vulnerabilities or room for improvement with security throughout the development process. |
| 1. Adopt a Secure Coding Standard | As many developers have found, coding can be very stylistic. However, it is important to find your forte language and program, and hold yourself to a standardized expectation of implementation. It is possible to work across languages and platforms, but ensure that you know what is expected of you, what you expect of the development process, and the checks and balances that need to be in place throughout. |

## 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** | INT30-C | Ensure that unsinged integer operations do not wrap |

| **Noncompliant Code** |
| --- |
| The code presented does not allow for preconditioned tests of the operands, and therefor can result in an unsigned interger wrap that can disallow for allocated memory needed to perform the operation. |
| void func(unsigned **int** ui\_a, unsigned **int** ui\_b) {    unsigned **int** usum = ui\_a + ui\_b;    /\* ... \*/  }  (Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682) |

| **Compliant Code** |
| --- |
| A compliant code in this instance allows for preconditioned test. This lessens the vulnerability of the code by ensuring that there is no unsigned wrap. |
| #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;    }    /\* ... \*/  }  (Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682) |

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

|  |
| --- |
| **Principles(s):** Validate Input data, Use effective quality assurance techniques: This principal is taken into account because it calls for the correct data to be entered and in the correct form. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description** |
| --- | --- | --- | --- |
| **Tool** | **Version** | **Checker** | **Description** |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 20.10 | **integer-overflow** | Fully checked |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 6.0p0 | **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.) |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **INTEGER\_OVERFLOW** | Implemented |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2021.1 | **C2910, C2911, C2912, C2913, C3383, C3384, C3385, C3386**  **C++2910, C++2911, C++2912, C++2913** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2021.1 | [**NUM.OVERFLOW**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**CWARN.NOEFFECT.OUTOFRANGE**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **493 S, 494 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2020.2 | **CERT\_C-INT30-a** **CERT\_C-INT30-b** **CERT\_C-INT30-c** | Avoid integer overflows Integer overflow or underflow in constant expression in '+', '-', '\*' operator Integer overflow or underflow in constant expression in '<<' operator |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2021a | [CERT C: Rule INT30-C](https://www.mathworks.com/help/bugfinder/ref/certcruleint30c.html) | Checks for:   * Unsigned integer overflow * Unsigned integer constant overflow   Rule partially covered. |
| [PRQA QA-C](https://wiki.sei.cmu.edu/confluence/display/c/PRQA+QA-C) | 9.7 | **2910 [C], 2911 [D], 2912 [A],**  **2913 [S], 3383, 3384, 3385, 3386** | Partially implemented |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | **2910, 2911, 2912, 2913** |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/c/PVS-Studio) | 7.07 | [**V658,**](https://www.viva64.com/en/w/v658/)[**V1028**](https://www.viva64.com/en/w/v1028/) |  |
| [TrustInSoft Analyzer](https://wiki.sei.cmu.edu/confluence/display/c/TrustInSoft+Analyzer) | 1.38 | **unsigned overflow** | Exhaustively verified. |

Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682

### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | INT32-C | Ensure that operations on signed integers do not result in overflow |

| **Noncompliant Code** |
| --- |
| The noncompliant code can result in overflow by not setting parameters around the value of the data being used. |
| void func(**signed** **int** si\_a, **signed** **int** si\_b) {  **signed** **int** sum = si\_a + si\_b;    /\* ... \*/  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

| **Compliant Code** |
| --- |
| By setting basic parameters for the integers used in the operation, overflow is disallowed thus making the code less vulnerable. |
| #include <limits.h>    void f(**signed** **int** si\_a, **signed** **int** si\_b) {  **signed** **int** sum;    if (((si\_b > 0) && (si\_a > (INT\_MAX - si\_b))) ||        ((si\_b < 0) && (si\_a < (INT\_MIN - si\_b)))) {      /\* Handle error \*/    } else {      sum = si\_a + si\_b;    }    /\* ... \*/  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

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

|  |
| --- |
| **Principles(s):** Validate input data, ensure quality: This standard allows for the smooth functionality of a program based on the input value given. This ensures that the code is well written and will function properly without the chance for overflow. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 20.10 | **integer-overflow** | Fully checked |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 6.0p0 | **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 |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **TAINTED\_SCALAR**  **BAD\_SHIFT** | Implemented |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2021.1 | **C2800, C2801, C2802, C2803, C2860, C2861, C2862, C2863**  **C++2800, C++2801, C++2802, C++2803, C++2860, C++2861, C++2862, C++2863** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **493 S, 494 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2020.2 | **CERT\_C-INT32-a** **CERT\_C-INT32-b** **CERT\_C-INT32-c** | Avoid integer overflows Integer overflow or underflow in constant expression in '+', '-', '\*' operator Integer overflow or underflow in constant expression in '<<' operator |
| [Parasoft Insure++](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) |  |  | Runtime analysis |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2021a | [CERT C: Rule INT32-C](https://www.mathworks.com/help/bugfinder/ref/certcruleint32c.html) | Checks for:   * Integer overflow * Tainted division operand * Tainted modulo operand   Rule partially covered. |
| [PRQA QA-C](https://wiki.sei.cmu.edu/confluence/display/c/PRQA+QA-C) | 9.7 | **2800, 2801, 2802, 2803,**  **2860, 2861, 2862, 2863** | Fully implemented |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | **2800, 2801, 2802, 2803,**  **2860, 2861, 2862, 2863** |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.07 | [**V1026**](https://www.viva64.com/en/w/v1026/) |  |
| [TrustInSoft Analyzer](https://wiki.sei.cmu.edu/confluence/display/c/TrustInSoft+Analyzer) | 1.38 | **signed\_overflow** | Exhaustively verified (see [one compliant and one non-compliant example](https://taas.trust-in-soft.com/tsnippet/t/06486475)). |

Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682

### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | STR30-C | Do not attempt to modify string literals |

| **Noncompliant Code** |
| --- |
| The noncompliant code initializes the character string to point to a string literal. Because of this, attempting to modify the string literal can result in defined behavior. |
| **char** \*str  = "string literal";  str[0] = 'S';  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

| **Compliant Code** |
| --- |
| In order to ensure compliance in this code, using an array to initialize the string allows for the modification of the string literal using size and values of the characters. This will result in safe modification and correctness of the string. |
| **char** str[] = "string literal";  str[0] = 'S';  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

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

|  |
| --- |
| **Principles(s):** Input data, quality assurance, coding standards: This standard comes back to the ability to code well, simply and with the ability to provide quality to the recipient. By ensuring that data is correct, errors can be avoided and the overall readability and quality of the code will be better. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 20.10 | **string-literal-modfication** **write-to-string-literal** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 6.9.0 | **CertC-STR30** | Fully implemented |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  | Can detect simple violations of this rule |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **PW** | Deprecates conversion from a string literal to "char \*" |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2021.1 | **C0556, C0752, C0753, C0754**  **C++3063, C++3064, C++3605, C++3606, C++3607** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2021.1 | [**C**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)[**XX.OVERWRITE\_CONST\_CHAR**](https://docs.roguewave.com/en/klocwork/current/) |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **157 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2020.2 | **CERT\_C-STR30-a** **CERT\_C-STR30-b** | A string literal shall not be modified Do not modify string literals |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **489, 1776** | Partially supported |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2021a | [CERT C: Rule STR30-C](https://www.mathworks.com/help/bugfinder/ref/certcrulestr30c.html) | Checks for writing to const qualified object (rule fully covered) |
| [PRQA QA-C](https://wiki.sei.cmu.edu/confluence/display/c/PRQA+QA-C) | 9.7 | **0556, 0752, 0753, 0754** | Partially implemented |
| [PRQA QA-C++](https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=142409849) | 4.4 | **3063, 3064, 3605, 3606, 3607, 3842** |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/c/PVS-Studio) | 7.07 | [**V675**](https://www.viva64.com/en/w/v675/) |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/c/RuleChecker) | 20.10 | **string-literal-modfication** | Partially checked |
| [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 | mem\_access | Exhaustively verified (see [one compliant and one non-compliant example](https://taas.trust-in-soft.com/tsnippet/t/952d807d)). |

Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682

### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | FIO30-C | Exclude user input from format strings |

| **Noncompliant Code** |
| --- |
| This section of code allows for unauthenticated users to change a string using an untrusted input value. It is accepting a line of user input that is not authenticated before returning an error message, and thus an attacker can input malicious code through this vulnerability. |
| #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);  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

| **Compliant Code** |
| --- |
| The solution is that in this line of code, instead of accepting part of the format string, it become a variable in the argument and doesn’t change the string format, thus reducing the chance of running malicious code. |
| #include <stdio.h>    void incorrect\_password(const **char** \*user) {    static const **char** msg\_format[] = "%s cannot be authenticated.\n";  **fprintf**(stderr, msg\_format, user);  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

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

|  |
| --- |
| **Principles(s):** Defense in depth, validate input, secure coding standard: With this code, there is the potential to open up the entire system for infiltration. By having the password in plain sight, it is possible for an attacker to exploit the obvious vulnerability. BY adding this layer of security, a better more secure code will be presented. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description** |
| --- | --- | --- | --- |
| [Splint](https://wiki.sei.cmu.edu/confluence/display/c/Splint) | 3.1.1 |  |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/c/PVS-Studio) | 7.07 | [**V618**](https://www.viva64.com/en/w/v618/) |  |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | **4916, 4917, 4918** |  |
| [PRQA QA-C](https://wiki.sei.cmu.edu/confluence/display/c/PRQA+QA-C) | 9.7 | **4916, 4917, 4918** |  |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2021a | [CERT C: Rule FIO30-C](https://www.mathworks.com/help/bugfinder/ref/certcrulefio30c.html) | Checks for tainted string format (rule partially covered) |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **592** | Partially supported: reports non-literal format strings |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2020.2 | **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 |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **86 D** | Partially Implemented |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2021.1 | [**SV.FMTSTR.GENERIC**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)[**SV.TAINTED.FMTSTR**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2021.1 | **C4916, C4917, C4918**  **C++4916, C++4917, C++4918** |  |
| [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 |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **TAINTED\_STRING** | Implemented |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 6.0p0 | **IO.INJ.FMT MISC.FMT** | Format string injection Format string |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 6.9.0 | **CertC-FIO30** | Partially implemented |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 20.10 |  | Supported via stubbing/taint analysis |

Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682

### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | MEM31-C | Free dynamically allocated memory when no longer needed |

| **Noncompliant Code** |
| --- |
| In this code, the pointer is never freed after the end of use of the pointer. As a result, this can lead to the over use of memory and system resources, and this can lead to a denial of service attack. |
| #include <stdlib.h>    enum { BUFFER\_SIZE = 32 };    **int** f(void) {  **char** \*text\_buffer = (**char** \*)**malloc**(BUFFER\_SIZE);    if (text\_buffer == NULL) {      return -1;    }    return 0;  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

| **Compliant Code** |
| --- |
| With this solution, and a call to free() the memory is deallocated thus ending the vulnerability on the system resources. |
| #include <stdlib.h>    enum { BUFFER\_SIZE = 32 };    **int** f(void) {  **char** \*text\_buffer = (**char** \*)**malloc**(BUFFER\_SIZE);    if (text\_buffer == NULL) {      return -1;    }    **free**(text\_buffer);    return 0;  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

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

|  |
| --- |
| **Principles(s):** Quality assurance, compiler warnings, secure coding: With the implementation of this rule, there is less of a chance for memory allocation errors, thus reducing the chance of a denial of service attack. Not only will it lessen the chance for errors, but it reduces a possible vulnerability. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 20.10 |  | Supported, but no explicit checker |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 6.9.0 | **CertC-MEM31** | Can detect dynamically allocated resources that are not freed |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 6.0p0 | **ALLOC.LEAK** | Leak |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  |  |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **RESOURCE\_LEAK**  **ALLOC\_FREE\_MISMATCH** | Finds resource leaks from variables that go out of scope while owning a resource |
| [Cppcheck](https://wiki.sei.cmu.edu/confluence/display/c/Cppcheck) | 1.66 | **leakReturnValNotUsed** | Doesn't use return value of memory allocation function |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2021.1 | **C2706, C2707, C2708**  **C++2706, C++2707, C++2708, C++3337, C++3338** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2021.1 | [**MLK.MIGHT**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**MLK.MUST**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**MLK.RET.MUST**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**MLK.RET.MIGHT**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **50 D** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2020.2 | **CERT\_C-MEM31-a** | Ensure resources are 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 | **429** | Fully supported |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2021a | [CERT C: Rule MEM31-C](https://www.mathworks.com/help/bugfinder/ref/certcrulemem31c.html) | Checks for memory leak (rule fully covered) |
| [PRQA QA-C](https://www.securecoding.cert.org/confluence/display/c/PRQA+QA-C) | 9.7 | **2706, 2707, 2708** |  |
| [PRQA QA-C++](https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=142409849) | 4.4 | **2706, 2707, 2708, 3337, 3338** |  |
| [SonarQube C/C++ Plugin](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151949) | 3.11 | [**S3584**](https://www.sonarsource.com/products/codeanalyzers/sonarcfamilyforcpp/rules-c.html#RSPEC-3584) |  |
| [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 | **malloc** | Exhaustively verified. |

Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682

### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | ERR53-CPP | Do not reference base classes or class data members in a constructor or destructor function-try-block handler |

| **Noncompliant Code** |
| --- |
| Because of checking the try block against str, it calls for undefined behavior. |
| #include <string>    class C {    std::string str;    public:    C(const std::string &s) try : str(s) {      // ...    } catch (...) {      if (!str.empty()) {        // ...      }    }  };  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

| **Compliant Code** |
| --- |
| By inspective the parameter instead of the class member, the undefined behavior is avoided. |
| #include <string>    class C {    std::string str;    public:    C(const std::string &s) try : str(s) {      // ...    } catch (...) {      if (!s.empty()) {        // ...      }    }  };  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

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

|  |
| --- |
| **Principles(s):** Quality assurance, secure coding standard: With this implementation, it will disallow the potential for compiler errors and undefined behavior. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **exception-handler-member-access** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 6.9.0 | **CertC++-ERR53** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | -Wexceptions |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.1 |  |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2021.1 | [**MISRA.CTOR.TRY.NON\_STATIC**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **549 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2020.2 | **CERT\_CPP-ERR53-a** | Handlers of a function-try-block implementation of a class constructor or destructor shall not reference nonstatic members from this class or its bases |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2020a | [CERT C++: ERR53-CPP](https://www.mathworks.com/help/bugfinder/ref/certcerr53cpp.html) | Checks for constructor or destructor function-try-block handler referencing base class or class data member (rule fully covered) |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | **3510** |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 20.10 | **exception-handler-member-access** | Fully checked |

Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages

### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | ERR51-CPP | Handle all exceptions thrown before main() begins executing |

| **Noncompliant Code** |
| --- |
| If exceptions are not caught before main() attempts to run, then it can cause the program to not run fully and/or correctly. These errors in programming can lead to denial of service attacks. |
| #include <string>    static const std::string global("...");    **int** main()  try {    // ...  } catch(...) {    // IMPORTANT: Will not catch exceptions thrown    // from the constructor of global  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

| **Compliant Code** |
| --- |
| This code was corrected by defining the string differently. Instead of defining it initially, it avoids the error by accepting a char. |
| static const **char** \*global = "...";    **int** main() {    // ...  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

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

|  |
| --- |
| **Principles(s):** Secure coding, quality assurance, defense in depth, compiler warnings: With the implementation of this rule, errors in running the code can be avoided. Additionally, this opens the vulnerability for a potential denial of service attack. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **main-function-catch-all early-catch-all** | Partially checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 6.9.0 | **CertC++-ERR51** |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.1 |  |  |
| [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) | 2020.2 | **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) | R2020a | [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) | 20.10 | **main-function-catch-all early-catch-all** | Partially checked |

Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages

### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Misc. | MSC41-C | Never hard code sensitive information |

| **Noncompliant Code** |
| --- |
| These lines of code represent the error in handling sensitive information. In hard coding the actual passphrase into the lines of code, it makes it very vulnerable to attack and required constant review, update and protection. |
| /\* 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;  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

| **Compliant Code** |
| --- |
| This error is corrected by not only checking the value of the password entered instead of the literal string entered, but it also has a predisposition to delete the entry when done. |
| /\* 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, 0, sizeof(code));    if (!flag) {  **printf**("Access denied\n");      return -1;    }  **printf**("Access granted\n");    // ...Work with system...    return 0;  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

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

|  |
| --- |
| **Principles(s):** Architect and Design, Defense in Depth, Secure Coding: The failure to adhere to this rule can leave a lot of room for vulnerabilities in a system. By having a password embedded in the code, doors are open for attack. This is a severe vulnerability and should be avoided. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 6.0p0 | **HARDCODED.AUTH HARDCODED.DNS HARDCODED.KEY HARDCODED.SALT** | Hardcoded Authentication Hardcoded DNS Name Hardcoded Crypto Key Hardcoded Crypto Salt |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2021.1 | **C3122**  **C++3842** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2020.2 | **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) | R2021a | [CERT C: Rule MSC41-C](https://www.mathworks.com/help/bugfinder/ref/certcrulemsc41c.html) | Checks for hard coded sensitive data (rule partially covered) |

Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages

### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Environment | ENV33-C | Do not call system() |

| **Noncompliant Code** |
| --- |
| This block of code allows for system() to be called to allow any command to be executed. This opens up the environment for a multitude of attacks. Malicious code can be sent directly to the system, and opens up the directory to infiltration. |
| #include <string.h>  #include <stdlib.h>  #include <stdio.h>    enum { BUFFERSIZE = 512 };    void func(const **char** \*input) {  **char** cmdbuf[BUFFERSIZE];  **int** len\_wanted = snprintf(cmdbuf, BUFFERSIZE,                              "any\_cmd '%s'", input);    if (len\_wanted >= BUFFERSIZE) {      /\* Handle error \*/    } else if (len\_wanted < 0) {      /\* Handle error \*/    } else if (**system**(cmdbuf) == -1) {      /\* Handle error \*/    }  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

| **Compliant Code** |
| --- |
| This code disallows full access to the system and directory by using CreateProcess(). Instead of allowing full access, this stops inheritance at the child process. |
| #include <Windows.h>    void func(**TCHAR** \*input) {    STARTUPINFO si = { 0 };    PROCESS\_INFORMATION pi;    si.cb = sizeof(si);    if (!CreateProcess(TEXT("any\_cmd.exe"), input, NULL, NULL, FALSE,                       0, 0, 0, &si, &pi)) {      /\* Handle error \*/    }    CloseHandle(pi.hThread);    CloseHandle(pi.hProcess);  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

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

|  |
| --- |
| **Principles(s):** Secure Coding, Defense in Depth, Architecture: This rule protects systems from being taken down from the directory. By calling the system, attackers could easily embed malicious code in the file system that could lead to a multitude of problems and vulnerabilities over all. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description** |
| --- | --- | --- | --- |
| **Tool** | **Version** | **Checker** | **Description** |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 20.10 | **stdlib-use-system** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 6.9.0 | **CertC-ENV33** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/c/Clang) | 3.9 | cert-env33-c | Checked by clang-tidy |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 6.0p0 | **BADFUNC.PATH.SYSTEM IO.INJ.COMMAND** | Use of system Command injection |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  |  |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **DONT\_CALL** | Implemented |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2021.1 | **C5018**  **C++5031** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2021.1 | [**MISRA.STDLIB.ABORT**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)[**SV.CODE\_INJECTION.SHELL\_EXEC**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**SV.TAINTED.INJECTION**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **588 S** | Fully implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2020.2 | **CERT\_C-ENV33-a** | Do not call the 'system()' function from the 'stdlib.h' or 'cstdlib' library with an argument other than '0' (null pointer) |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **586** | Fully supported |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2021a | [CERT C: Rule ENV33-C](https://www.mathworks.com/help/bugfinder/ref/certcruleenv33c.html) | Checks for unsafe call to a system function (rule fully covered) |
| [PRQA QA-C](https://wiki.sei.cmu.edu/confluence/display/c/PRQA+QA-C) | 9.7 | **5018** | Partially implemented |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/c/RuleChecker) | 20.10 | **stdlib-use-system** | Fully checked |
| [SonarQube C/C++ Plugin](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151949) | 3.11 | [**S990**](https://www.sonarsource.com/products/codeanalyzers/sonarcfamilyforcpp/rules-c.html#RSPEC-990) | Detects uses of "abort", "exit", "getenv" and "system" from <stdlib.h> |

Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages

### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Characters and Strings | STR50-CPP | Guarantee that storage for strings has sufficient space for character data and the null terminator |

| **Noncompliant Code** |
| --- |
| Being unbound, this code can lead to a buffer overflow. This is caused when the data is too larger for the buffer and can result to or from data manipulation. |
| #include <iostream>    void f() {  **char** buf[12];    std::cin >> buf;  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

| **Compliant Code** |
| --- |
| This potential error is remedied by using std::string instead of an array. |
| #include <iostream>  #include <string>    void f() {    std::string input;    std::string stringOne, stringTwo;    std::cin >> stringOne >> stringTwo;  }  Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046682 |

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

|  |
| --- |
| **Principles(s):** Quality assurance, Coding standards, compiler warnings: This is a rule that brings it back to basics. In order to ensure that there is a fully functional code, it is important that the memory is allocated and utilized as it should be. With this, you will ensure that the code is readable, well written and avoids data manipulation from buffer overflow, resulting in the execution of malicious code through the weaknesses presented. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 6.0p0 | **MISC.MEM.NTERM**  **LANG.MEM.BO LANG.MEM.TO** | No space for null terminator  Buffer overrun Type overrun |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.1 |  |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2021.1 | [**NNTS.MIGHT**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**NNTS.TAINTED**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [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) | 2020.2 | **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) | R2020a | [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   Rule partially covered. |
| [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) |  |

Confluence. (n.d.). Retrieved from https://wiki.sei.cmu.edu/confluence/pages

## Defense-in-Depth Illustration

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



# Project One

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

## Revise the C/C++ Standards

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

## Risk Assessment

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

## Automated Detection

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

## Automation

Provide a written explanation using the image provided.



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

[Insert your written explanations here.]

## Summary of Risk Assessments

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Rule** | **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| ENV33-C | High | Probable | Medium | P12 | L1 |
| ERR51-CPP | Low | Probable | Medium | P4 | L3 |
| ERR53-CPP | Low | Unlikely | Medium | P2 | L3 |
| FIO30-C | High | Likely | Medium | P18 | L1 |
| INT30-C | High | Likely | High | P9 | L2 |
| INT32-C | High | Likely | High | P9 | L2 |
| MEM31-C | Medium | Probable | Medium | P8 | L2 |
| MSC41-C | High | Probable | Medium | P12 | L1 |
| STR30-C | Low | Likely | Low | P9 | L2 |
| STR50-CPP | High | Likely | Medium | P18 | L1 |

## Create Policies for Encryption and Triple A

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

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

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

| 1. **Encryption** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Encryption in rest | Encryption at rest is the practice of encrypting the physical media, as in a hard drive on a computer. By requiring authentication on the physical device itself, it adds an additional level of security that would disallow malicious activity against the devices within a network. Instead of having an “open door” for the attacker, even if the device were to fall into their hands, and additional guard would be in place that would hinder their ability to use the information on the device. |
| Encryption at flight | Encryption at flight is the process that considers encryption while information is being transmitted. This may include hashing, or encrypting, data as it is passed through a network. By implementing this theory, information that is being passed would be useless without the proper decryption information. This protects the data from attack while it is being transmitted. |
| Encryption in use | Encryption in use is the act of needing to encrypt or decrypt data. This can be rationalized as a user needing to enter their password in order to verify their identity and authenticate the user. By implementing this practice, unauthorized users or attackers will be blocked access to the data. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | Authentication is a policy in which a user needs to prove who they are and their privileges before being granted access to data. Within this, the user needs to provide their personal username and password before being allowed to access data. This prevents unauthorized users from being able to access data or a network that they should not be allowed to. |
| Authorization | Once a user is authenticated, authorization must be granted. Authorization is the level that dictates what information may be accessed and when by an authenticated user. The principle of least privilege could come into play in this policy. A user should only be allowed to access the information that is required for their position and should not be allowed to access more secure files. |
| Accounting | Accounting is the policy that tracks the overall usage of a user on a network. This could include the system resources that are being allocated or the levels of access that are being used. With this, we can assure that network assets are not being abused, and can help to recognize when there is suspicious activity in play. |

**\***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 |  |
| 1.1 | 3/2021 | First Addition | Caroline Wilson |  |
| 1.2 | 4/2021 | Final Policy | Caroline Wilson |  |

# Appendix A Lookups

## Approved C/C++ Language Acronyms

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