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

Justin Fields 6-2



# 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 | Enter and delete a lot of vulnerabilities as able to be done validating incoming data  from an untrusted source. |
| 1. Heed Compiler Warnings | Compiling and testing the code as much as possible to try and keep the code as  tight as possible. A warning might be just an annoyance, but it also could be  something serious, that means check them all as much as possible just in case. |
| 1. Architect and Design for Security Policies | When design the code with security you have to meet the standard for the code that work right, your able to save time by building safeguards in or you can fix vulnerabilities when they are found. |
| 1. Keep It Simple | When keeping your code clean and simple it makes it easier, also the security of your code. This principle applies for security as the more understandable something is the easier to protect. |
| 1. Default Deny | The Standardize my access decisions on permission than exclusion. The  default access will denied the specific conditions to permit access. |
| 1. Adhere to the Principle of Least Privilege | The principle of less important goes with default denial, there a standard of  having users that need better access they go through the right channels to  gain that access. When going through the processes of gaining access it should  have implemented that only the access that is most important is given and  limiting possible risk by making less information available. |
| 1. Sanitize Data Sent to Other Systems | Making sure that your data sent to another systems sanitized all the vulnerabilities or sensitive information is important. Also, when sending code with this kind of  information could move to exploits that could have any number of bad  outcomes depending on some situation. |
| 1. Practice Defense in Depth | Some systems will need more than one line of protection, combining lines of  redundant and many defense will help make a better safeguard a system if it did not it would cause a breach and chaos. |
| 1. Use Effective Quality Assurance Techniques | The Quality assurance techniques will grow chances of finding and deleting  vulnerabilities. Creating multiple testing phases, independent security reviews and  external security reviews can add more secure systems. |
| 1. Adopt a Secure Coding Standard | Making sure to have a secure coding standard in development in any language and  platform you are good to have. |

### C/C++ Ten Coding Standards

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

#### Coding Standard 1

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Type** | [STD-001-  CPP] | When defining C-style variadic functions can create vulnerabilities  when they forget to check arguments being passed. |

| **Noncompliant Code** |
| --- |
| The functions are designed to show values until 0 value is found or not found after two arguments it  will cause issues. |
| #include <cstdarg>    int add(int first, int second, ...) {  int r = first + second;  va\_list va;  va\_start(va, second);  while (int v = va\_arg(va, int)) {  r += v;  }  va\_end(va);  return r;  } |

| **Compliant Code** |
| --- |
| The function does have a built in add statement that helps stop the top issues. |
| #include <type\_traits>    template <typename Arg, typename std::enable\_if<std::is\_integral<Arg>::value>::type \* = nullptr>  int add(Arg f, Arg s) { return f + s; }    template <typename Arg, typename... Ts, typename std::enable\_if<std::is\_integral<Arg>::value>::type \* = nullptr>  int add(Arg f, Ts... rest) {  return f + add(rest...);  } |

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

| **Principles(s):** The Validating Input Data is creating a proper input.  The Architect and Design for Security Policies is writing code to stop vulnerabilities.  The Keep it simple is an applies as making sure your code is small as possible is also a best practice.  The Use Effective Quality Assurance Techniques can test that they are effective as possible.  The Adopt a secure coding standard is having security a priority helps to stop vulnerabilities. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **function-ellipsis** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-DCL50** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | cert-dcl50-cpp | Checked by clang-tidy. |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 6.1p0 | **LANG.STRUCT.ELLIPSIS** | Ellipsis |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | [STD-002-  CPP] | Talking about a reserved identifier wrong creates an issue it will not really  be reserved. |

| **Noncompliant Code** |
| --- |
| To have a name standards that are not met then create undefined behavior. |
| #ifndef \_MY\_HEADER\_H\_  #define \_MY\_HEADER\_H\_    // Contents of <my\_header.h>    #endif // \_MY\_HEADER\_H\_ |

| **Compliant Code** |
| --- |
| When taking out the trailing and leading underscores it will stop the issue. |
| #ifndef MY\_HEADER\_H  #define MY\_HEADER\_H    // Contents of <my\_header.h>    #endif // MY\_HEADER\_H |

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

| **Principles(s):** When a compiler warnings are always pay attention for warning sign as there the reason  Architect and Design for Security Policies is writing code to stop vulnerabilities.  Keep it easy to understand and applies in keeping the code as small as possible is best practice.  Effective Quality Assurance Techniques show tests when effective as possible  Adopt a secure coding standard creates security that helps stop vulnerabilities |
| --- |

**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=222953724) | 20.10 | **reserved-identifier** | Partially checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-DCL51** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | -Wreserved-id-macro -Wuser-defined-literals | The -Wreserved-id-macro flag is not enabled by default or with -Wall, but is enabled with -Weverything. This flag does not catch all instances of this rule, such as redefining reserved names. |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 6.1p0 | **LANG.ID.NU.MK**  **LANG.STRUCT.DECL.RESERVED** | Macro name is C keyword  Declaration of reserved name |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | [STD-003-  CPP] | Don’t qualify a reference with a const or volatile. Also, cv-qualifying a  reference type Can create an undefined behavior. Also, compiler will deliver fatal diagnostic and when it can’t and might create different results. |

| **Noncompliant Code** |
| --- |
| Most const-qualified reference type to the char is create instead on a reference to const-qualified char. |
| #include <iostream>    void f(char c) {  char &const p = c;  p = 'p';  std::cout << c << std::endl;  } |

| **Compliant Code** |
| --- |
| When removing a const qualifier it will stop an issue from occurring. |
| #include <iostream>    void f(char c) {  char &p = c;  p = 'p';  std::cout << c << std::endl;  } |

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

| **Principles(s):** A heed compiler warnings will look at warnings and the reason.  The Architect and Design in Security Policies is writing code to stop vulnerabilities.  The Keep it simple can applies as keeping code as little as you can is a best practice.  The Use Effective Quality Assurance Techniques can create tests that are effective as it can.  The Adopt a secure coding standard I will have security as a priority that helps prevent vulnerabilities. |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Unlikely | Low | **P3** | **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++-DCL52** |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.2 | **C++0014** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2021.1 | [**CERT.DCL.REF\_TYPE.CONST\_OR\_VOLATILE**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.1 | **CERT\_CPP-DCL52-a** | Never qualify a reference type with 'const' or 'volatile' |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | [STD-004-  CPP] | It not a good idea to write syntactically ambiguous declarations. When write code that can only be understood one way is not a good idea. |

| **Noncompliant Code** |
| --- |
| An argument you can declare an anonymous object is also called a single-argument converting  constructor or interpreted when declaring an object named m and can also default constructing it. |
| #include <mutex>    static std::mutex m;  static int shared\_resource;    void increment\_by\_42() {  std::unique\_lock<std::mutex>(m);  shared\_resource += 42;  } |

| **Compliant Code** |
| --- |
| Lock will have an identifier and proper converting constructor when called. |
| #include <mutex>    static std::mutex m;  static int shared\_resource;    void increment\_by\_42() {  std::unique\_lock<std::mutex> lock(m);  shared\_resource += 42;  } |

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

| **Principles(s):** The Validating Input Data is creating a proper input.  The Architect and Design for Security Policies is writing code to stop vulnerabilities.  The Keep it simple is an applies as making sure your code is small as possible is also a best practice.  The Use Effective Quality Assurance Techniques can test that they are effective as possible.  The Adopt a secure coding standard is having security a priority helps to stop vulnerabilities. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **296 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.1 | **CERT\_CPP-DCL53-a** **CERT\_CPP-DCL53-b** | Always declare functions at file scope Identifier declared in a local or function prototype scope shall not hide an identifier declared in a global or namespace scope |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2021a | [CERT C++: DCL53-CPP](https://www.mathworks.com/help/bugfinder/ref/certcdcl53cpp.html) | Checks for declarations that can be confused between:   * Function and object declaration * Unnamed object or function parameter declaration   Rule fully covered. |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | **2502, 2510** |  |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [STD-005-  CPP] | The Overload allocation and the deallocation functions are together in the scope.  The failure could create undefined behavior. |

| **Noncompliant Code** |
| --- |
| Allocation is overloaded on a huge scale there no deallocation function declared. |
| #include <Windows.h>  #include <new>    void \*operator new(std::size\_t size) noexcept(false) {  static HANDLE h = ::HeapCreate(0, 0, 0); // Private, expandable heap.  if (h) {  return ::HeapAlloc(h, 0, size);  }  throw std::bad\_alloc();  }    // No corresponding global delete operator defined. |

| **Compliant Code** |
| --- |
| Deallocation becomes declared and should stop the overload condition. |
| #include <Windows.h>  #include <new>    class HeapAllocator {  static HANDLE h;  static bool init;    public:  static void \*alloc(std::size\_t size) noexcept(false) {  if (!init) {  h = ::HeapCreate(0, 0, 0); // Private, expandable heap.  init = true;  }    if (h) {  return ::HeapAlloc(h, 0, size);  }  throw std::bad\_alloc();  }    static void dealloc(void \*ptr) noexcept {  if (h) {  (void)::HeapFree(h, 0, ptr);  }  }  };    HANDLE HeapAllocator::h = nullptr;  bool HeapAllocator::init = false;    void \*operator new(std::size\_t size) noexcept(false) {  return HeapAllocator::alloc(size);  }    void operator delete(void \*ptr) noexcept {  return HeapAllocator::dealloc(ptr);  } |

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

| **Principles(s):** The Validating Input Data is creating a proper input.  The Architect and Design for Security Policies is writing code to stop vulnerabilities.  The Keep it simple is an applies as making sure your code is small as possible is also a best practice.  The Use Effective Quality Assurance Techniques can test that they are effective as possible.  The Adopt a secure coding standard is having security a priority helps to stop vulnerabilities. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **new-delete-pairwise** | Partially checked |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | misc-new-delete-overloads | Checked with clang-tidy. |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.1 | **CERT\_CPP-DCL54-a** | Always provide new and delete together |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2021a | [CERT C++: DCL54-CPP](https://www.mathworks.com/help/bugfinder/ref/certcdcl54cpp.html) | Checks for mismatch between overloaded operator new and operator delete (rule fully covered) |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | [STD-006-  CPP] | The avoid information leakage is when passing a class object to a trust  boundary. Data passing should become verified before it creates an issues. |

| **Noncompliant Code** |
| --- |
| Data can transfer can contain sensitive information. |
| #include <cstddef>    struct test {  int a;  char b;  int c;  };    // Safely copy bytes to user space  extern int copy\_to\_user(void \*dest, void \*src, std::size\_t size);    void do\_stuff(void \*usr\_buf) {  test arg{1, 2, 3};  copy\_to\_user(usr\_buf, &arg, sizeof(arg));  } |

| **Compliant Code** |
| --- |
| The structure data before copying should stop these problems. |
| #include <cstddef>  #include <cstring>    struct test {  int a;  char b;  int c;  };    // Safely copy bytes to user space.  extern int copy\_to\_user(void \*dest, void \*src, std::size\_t size);    void do\_stuff(void \*usr\_buf) {  test arg{1, 2, 3};  // May be larger than strictly needed.  unsigned char buf[sizeof(arg)];  std::size\_t offset = 0;    std::memcpy(buf + offset, &arg.a, sizeof(arg.a));  offset += sizeof(arg.a);  std::memcpy(buf + offset, &arg.b, sizeof(arg.b));  offset += sizeof(arg.b);  std::memcpy(buf + offset, &arg.c, sizeof(arg.c));  offset += sizeof(arg.c);    copy\_to\_user(usr\_buf, buf, offset /\* size of info copied \*/);  } |

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

| **Principles(s):** The Validating Input Data is creating a proper input.  The Architect and Design for Security Policies is writing code to stop vulnerabilities.  The Keep it simple is an applies as making sure your code is small as possible is also a best practice.  The Use Effective Quality Assurance Techniques can test that they are effective as possible.  The Adopt a secure coding standard is having security a priority helps to stop vulnerabilities. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-DCL55** |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.2 | **C++4941, C++4942, C++4943** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.1 | **CERT\_CPP-DCL55-a** | A pointer to a structure should not be passed to a function that can copy data to the user space |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [STD-007-  CPP] | The avoid cycles in the initialization of static objects. When an function is reentered in the initialization of a static object, the behavior can become undefined. |

| **Noncompliant Code** |
| --- |
| The process to add factorial function utilizing caching the adding of the static array cache  is recursion creating undefined behavior. |
| #include <stdexcept>  int fact(int i) noexcept(false) {  if (i < 0) {  // Negative factorials are undefined.  throw std::domain\_error("i must be >= 0");  }  static const int cache[] = {  fact(0), fact(1), fact(2), fact(3), fact(4), fact(5),  fact(6), fact(7), fact(8), fact(9), fact(10), fact(11),  fact(12), fact(13), fact(14), fact(15), fact(16)  };  if (i < (sizeof(cache) / sizeof(int))) {  return cache[i];  }  return i > 0 ? i \* fact(i - 1) : 1;  } |

| **Compliant Code** |
| --- |
| This cannot show the static cache is the part that causing the issue. |
| #include <stdexcept>    int fact(int i) noexcept(false) {  if (i < 0) {  // Negative factorials are undefined.  throw std::domain\_error("i must be >= 0");  }    // Use the lazy-initialized cache.  static int cache[17];  if (i < (sizeof(cache) / sizeof(int))) {  if (0 == cache[i]) {  cache[i] = i > 0 ? i \* fact(i - 1) : 1;  }  return cache[i];  }    return i > 0 ? i \* fact(i - 1) : 1;  } |

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

| **Principles(s):** The Architect and Design in Security Policies is writing code to stop vulnerabilities.  The Keep it simple can applies as keeping code as little as you can is a best practice.  The Use Effective Quality Assurance Techniques can create tests that are effective as it can.  The Adopt a secure coding standard I will have security as a priority that helps prevent vulnerabilities. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **6 D** | Enhanced Enforcement |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.1 | **CERT\_CPP-DCL56-a** | Avoid initialization order problems across translation units by replacing non-local static objects with local static objects |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |

#### Coding Standard 8

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

| **Noncompliant Code** |
| --- |
| This class destructor have an exception and create an undefined behavior. |
| #include <stdexcept>    class S {  bool has\_error() const;    public:  ~S() noexcept(false) {  // Normal processing  if (has\_error()) {  throw std::logic\_error("Something bad");  }  }  }; |

| **Compliant Code** |
| --- |
| It should catch most exceptions and delete it. |
| class SomeClass {  Bad bad\_member;  public:  ~SomeClass()  try {  // ...  } catch(...) {  // Catch exceptions thrown from noncompliant destructors of  // member objects or base class subobjects.    // NOTE: Flowing off the end of a destructor function-try-block causes  // the caught exception to be implicitly rethrown, but an explicit  // return statement will prevent that from happening.  return;  }  }; |

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

| **Principles(s):** The Architect and Design in Security Policies is writing code to stop vulnerabilities.  The Keep it simple can applies as keeping code as little as you can is a best practice.  The Use Effective Quality Assurance Techniques can create tests that are effective as it can.  The Adopt a secure coding standard I will have security as a priority that helps prevent vulnerabilities. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **destructor-without-noexcept delete-without-noexcept** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-DCL57** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **453 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.1 | **CERT\_CPP-DCL57-a** **CERT\_CPP-DCL57-b** | Never allow an exception to be thrown from a destructor, deallocation, and swap Always catch exceptions |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| [Student Choice] | [STD-009-  CPP] | Make sure to not change the standard namespaces. Also show a new declaration in the namespace can create undefined behavior that not utilized correctly. |

| **Noncompliant Code** |
| --- |
| You add an x is to the namespace creating an undefined behavior. |
| namespace std {  int x;  } |

| **Compliant Code** |
| --- |
| When adding without a reserved name this does not create an undefined behavior. |
| namespace nonstd {  int x;  } |

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

| **Principles(s):** The Architect and Design in Security Policies is writing code to stop vulnerabilities.  The Keep it simple can applies as keeping code as little as you can is a best practice.  The Use Effective Quality Assurance Techniques can create tests that are effective as it can.  The Adopt a secure coding standard I will have security as a priority that helps prevent vulnerabilities. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-DCL58** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.1 | **CERT\_CPP-DCL58-a** | Do not modify the standard namespaces 'std' and 'posix' |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2021a | [CERT C++: DCL58-CPP](https://www.mathworks.com/help/bugfinder/ref/certcdcl58cpp.html) | Checks for modification of standard namespaces (rule fully covered) |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | **4032, 4035, 4631** | [Insert text.] |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| [Student Choice] | [STD-010-  CPP] | Should not define the unnamed namespace in a header file. Adding to an  unnamed namespace in a header file will create issues. |

| **Noncompliant Code** |
| --- |
| When a variable is defined in the unnamed namespace and a result is each translation unit works on its own instance. |
| // a.h  #ifndef A\_HEADER\_FILE  #define A\_HEADER\_FILE    namespace {  int v;  }    #endif // A\_HEADER\_FILE    // a.cpp  #include "a.h"  #include <iostream>    void f() {  std::cout << "f(): " << v << std::endl;  v = 42;  // ...  }    // b.cpp  #include "a.h"  #include <iostream>    void g() {  std::cout << "g(): " << v << std::endl;  v = 100;  }    int main() {  extern void f();  f(); // Prints v, sets it to 42  g(); // Prints v, sets it to 100  f();  g();  } |

| **Compliant Code** |
| --- |
| Variable is defined in one translation unit. Also visible to and ends in the expected output. |
| // a.h  #ifndef A\_HEADER\_FILE  #define A\_HEADER\_FILE    extern int v;    #endif // A\_HEADER\_FILE    // a.cpp  #include "a.h"  #include <iostream>    int v; // Definition of global variable v    void f() {  std::cout << "f(): " << v << std::endl;  v = 42;  // ...  }    // b.cpp  #include "a.h"  #include <iostream>    void g() {  std::cout << "g(): " << v << std::endl;  v = 100;  }    int main() {  extern void f();  f(); // Prints v, sets it to 42  g(); // Prints v, sets it to 100  f(); // Prints v, sets it back to 42  g(); // Prints v, sets it back to 100  } |

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

| **Principles(s):** The Architect and Design in Security Policies is writing code to stop vulnerabilities.  The Keep it simple can applies as keeping code as little as you can is a best practice.  The Use Effective Quality Assurance Techniques can create tests that are effective as it can.  The Adopt a secure coding standard I will have security as a priority that helps prevent vulnerabilities. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **unnamed-namespace-header** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-DCL59** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | cert-dcl59-cpp | Checked by clang-tidy |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **286 S, 512 S** | Fully implemented |

### Defense-in-Depth Illustration

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



## Project One

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

### Revise the C/C++ Standards

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

### Risk Assessment

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

### Automated Detection

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

### Automation

Provide a written explanation using the image provided.



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

[Insert your written explanations here.]

### Summary of Risk Assessments

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

| Rule | Severity | Likelihood | Remediation Cost | Priority | Level |
| --- | --- | --- | --- | --- | --- |
| STD-001-CPP | High | Probable | Medium | P12 | L1 |
| [STD-002-  CPP] | Low | Unlikely | Low | **P3** | **L3** |
| [STD-003-  CPP] | Low | Unlikely | Low | **P3** | **L3** |
| [STD-004-  CPP] | Low | Unlikely | Medium | **P2** | **L3** |
| [STD-005-  CPP] | Low | Probable | Low | **P6** | **L2** |
| [STD-006-  CPP] | Low | Unlikely | High | **P1** | **L3** |
| [STD-007-  CPP] | Low | Unlikely | Medium | **P2** | **L3** |
| [STD-008-  CPP] | Low | Likely | Medium | **P6** | **L2** |
| [STD-009-  CPP] | High | Unlikely | Medium | **P6** | **L2** |
| [STD-010-  CPP] | Medium | Unlikely | Medium | **P4** | **L3** |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
|  | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
|  | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |
|  |  |  |  |  |  |

### Create Policies for Encryption and Triple A

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

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

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

| 1. **Encryption** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Encryption in rest | The Encryption at rest is created to stop an attacker from accessing the unencrypted  data and making sure the data are encrypted on disk. If an attacker gets the hard drive  with encrypted data and not the encryption keys, an attacker will protect the  encryption to read the data. |
| Encryption at flight | A process of encrypting data when the data is being transmitted. In applications,  like remote replication, data might become unencrypted and it at rest on drive arrays,  and encrypted when it is being transmitted to provide protection. |
| Encryption in use | The Encryption of data in-use is the process of protecting data it works in memory, the way it works is by the password protected profiles it protects the  memory of many users the data stored in memory for that profile can be used to  compromise their data in rest/flight. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | The Authentication is the process when a user is picked to has  access to the system. When a user login and password information for the user  to access parts of the system. Few new methods are 2 steps  authentication or multi-tier authentication. |
| Authorization | The Authorization is the access that a user has in the system. Also, a user could read, create, delete and modify files in the database. Are able to access if a user can add or delete files and users in the system. |
| Accounting | Make sure to watch and record activity of the users in the system. It is called accounting making sure this process is working. Then you will have a good understanding of who is trying to access the system and what they are doing with that access when they have  authorization to the data on the system. |

**\***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 | 10/10/2021 | Initial Template | Ahlam Alhweiti |  |
| 1.11 | 10/14/2021 | Project One | Ahlam Alhweiti | Ahlam Alhweiti |
| 1.12 | [Insert text.] | The Final Revision | Ahlam Alhweiti | [Insert text.] |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

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