**Green Pace Developer: Security Policy Guide**



# 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** |  |
| --- | --- |
| 1. ValidateInput Data | Any data entered or sent across servers should be checked for proper structure based on the use and application. If invalid data is found, it should be discarded and re-entered. To prevent buffer overflows, length should also be checked, and extra data should be discarded. |
| 1. Heed Compiler Warnings | All compiler warnings should be considered necessary to be fixed before deployment, even if the application is seemingly functional. Outdated or superseded legacy functions should always be replaced by new alternatives, with updated in-line comments and documentation. |
| 1. Architect and Design for Security Policies | All developed systems should adhere to the ten principles in this table. Controls should be set up in proportion to the potential threat level depending on data or how it will be used. Regular risk assessment should also be conducted by those with senior experience in their respective department. |
| 1. Keep It Simple | System design should be kept consistent, so that it is easy to understand for not only current employees but also future ones. It should be easy to understand how a component of a system works by looking through its documentation. Analysis of a system should be done to ensure effectiveness is maintained even with lowered complexity. |
| 1. Default Deny | Authorization should be given on a case-by-case basis. A new employee or user should only be given permission based on necessary usage of a system. A standard should be set to avoid taking risks that imply trust, by starting from a completely fresh slate when it comes to permissions. |
| 1. Adhere to the Principle of Least Privilege | Frequent analysis should be done to ensure that privileges aren’t over-extended, especially for employees moving from one department to another. Permissions should only be granted based on the work that is absolutely necessary. If an employee’s required tasks change, their permissions should also reflect this. |
| 1. Sanitize Data Sent to Other Systems | Extra care should be taken to validate input and securely encrypt data sent remotely. The same standards applied to the “Validate Input Data” section should be used here, and personal information that is not required to be sent should be discarded. |
| 1. Practice Defense in Depth | Layers of security should be implemented while avoiding complexity; guidelines mentioned in the “Keep It Simple” section. Redundancy should be encouraged rather than avoided. The type of data should be considered when deciding on the number of defense layers to protect it with. |
| 1. Use Effective Quality Assurance Techniques | All systems should be thoroughly examined and tested by multiple engineers, who collectively can decide on the criteria used. Testing should continue even after a system is deployed and operational. Any and all changes should be followed by a re-evaluation and new testing period. |
| 1. Adopt a Secure Coding Standard | Every function should be thoroughly documented, with a description of the function itself, the return value (if applicable), and any parameters used. All external input must be checked and validated before it is used, even if a built-in variable check (such as the case for Strings in C++) is in place. Exceptions errors should clearly explain the cause, and potential effect on the program and its data going forward. |

### C/C++ Ten Coding Standards

#### Coding Standard 1

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Type** | [STD-001-CPP] | [Never qualify a reference type with const or volatile](https://wiki.sei.cmu.edu/confluence/display/cplusplus/DCL52-CPP.+Never+qualify+a+reference+type+with+const+or+volatile) |

| **Noncompliant Code** |
| --- |
| A constant char reference is declared, being assigned a value. Afterwards, there is an attempt to modify this read-only reference. |
| #include <iostream>    void f(char c) {  const char &p = c;  p = 'p'; // Error: read-only variable is not assignable  std::cout << c << std::endl;  } |

| **Compliant Code** |
| --- |
| The const qualifier is removed. |
| #include <iostream>    void f(char c) {  char &p = c;  p = 'p';  std::cout << c << std::endl;  } |

| **Principles(s):**  2. Heed compiler warnings – Depending on the compiler, a program may still compile with this warning. It’s important to fix it for readability before release builds and deployment. |
| --- |

**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) | 2024.4 | C++0014 |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2024.4 | CERT.DCL.REF\_TYPE.CONST\_OR\_VOLATILE |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2024.2 | CERT\_CPP-DCL52-a | Never qualify a reference type with 'const' or 'volatile' |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024b | [CERT C++: DCL52-CPP](https://www.mathworks.com/help/bugfinder/ref/certcdcl52cpp.html) | Checks for:   * const-qualified reference types * Modification of const-qualified reference types   Rule fully covered. |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 |  | Clang checks for violations of this rule and produces an error without the need to specify any special flags or options. |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | [STD-002-CPP] | [Do not access an object outside of its lifetime](https://wiki.sei.cmu.edu/confluence/display/cplusplus/EXP54-CPP.+Do+not+access+an+object+outside+of+its+lifetime) |

| **Noncompliant Code** |
| --- |
| An object pointer is used to call a non-static member function before it is initialized. |
| struct S {  void mem\_fn();  };    void f() {  S \*s;  s->mem\_fn();  } |

| **Compliant Code** |
| --- |
| The pointer is initialized before calling the member function. Afterwards, it is removed from memory. |
| struct S {  void mem\_fn();  };    void f() {  S \*s = new S;  s->mem\_fn();  delete s;  } |

| **Principles(s):**  2. Heed Compiler Warnings – A warning about uninitialized memory should always be addressed, even if it’s inconsequential to the program’s operation.  9. Use Effective Quality Assurance Techniques – Catching this issue early can be done by adopting a standard of always checking that variables are initialized before they are used during quality assurance testing. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | return-reference-local dangling\_pointer\_use | Partially checked |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | -Wdangling-initializer-list | Catches some lifetime issues related to incorrect use of std::initializer\_list<> |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 9.0p0 | IO.UAC ALLOC.UAF | Use after close Use after free |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.4 | C++4003, C++4026  DF2812, DF2813, DF2814, DF2930, DF2931, DF2932, DF2933, DF2934, |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2024.4 | CL.FFM.ASSIGN CL.FFM.COPY LOCRET.ARG LOCRET.GLOB LOCRET.RET UFM.DEREF.MIGHT UFM.DEREF.MUST UFM.FFM.MIGHT UFM.FFM.MUST UFM.RETURN.MIGHT UFM.RETURN.MUST UFM.USE.MIGHT UFM.USE.MUST UNINIT.HEAP.MIGHT UNINIT.HEAP.MUST UNINIT.STACK.ARRAY.MIGHT UNINIT.STACK.ARRAY.MUST UNINIT.STACK.ARRAY.PARTIAL.MUST UNINIT.STACK.MIGHT UNINIT.STACK.MUST |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | 42 D, 53 D, 77 D, 1 J, 71 S, 565 S | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2024.2 | CERT\_CPP-EXP54-a CERT\_CPP-EXP54-b CERT\_CPP-EXP54-c | Do not use resources that have been freed The address of an object with automatic storage shall not be returned from a function The address of an object with automatic storage shall not be assigned to another object that may persist after the first object has ceased to exist |
| [Parasoft Insure++](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) |  |  | Runtime detection |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024b | [CERT C++: EXP54-CPP](https://www.mathworks.com/help/bugfinder/ref/certcexp54cpp.html) | Checks for:   * Non-initialized variable or pointer * Use of previously freed pointer * Pointer or reference to stack variable leaving scope * Accessing object with temporary lifetime   Rule partially covered. |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.36 | [V758](https://pvs-studio.com/en/docs/warnings/v758/), [V1041](https://pvs-studio.com/en/docs/warnings/v1041/), [V1099](https://pvs-studio.com/en/docs/warnings/v1099/) |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 22.10 | return-reference-local | Partially checked |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | [STD-003-CPP] | [Do not attempt to create a std::string from a null pointer](https://wiki.sei.cmu.edu/confluence/display/cplusplus/STR51-CPP.+Do+not+attempt+to+create+a+std%3A%3Astring+from+a+null+pointer) |

| **Noncompliant Code** |
| --- |
| A function call that has the possibility to return a null string is used to create a std::string variable. |
| #include <cstdlib>  #include <string>    void f() {  std::string tmp(std::getenv("TMP"));  if (!tmp.empty()) {  // ...  }  } |

| **Compliant Code** |
| --- |
| An additional null check is added. |
| #include <cstdlib>  #include <string>    void f() {  const char \*tmpPtrVal = std::getenv("TMP");  std::string tmp(tmpPtrVal ? tmpPtrVal : "");  if (!tmp.empty()) {  // ...  }  } |

| **Principles(s):**  4. Keep It Simple – Avoiding creation of a string by using a function’s return value can reduce complexity and increase code readability by keeping the declaration and function call check on a separate line. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | assert\_failure |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 9.0p0 | LANG.MEM.NPD | Null Pointer Dereference |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.4 | DF4770, DF4771, DF4772, DF4773, DF4774 |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2024.4 | NPD.CHECK.CALL.MIGHT NPD.CHECK.CALL.MUST NPD.CHECK.MIGHT NPD.CHECK.MUST NPD.CONST.CALL NPD.CONST.DEREF NPD.FUNC.CALL.MIGHT NPD.FUNC.CALL.MUST NPD.FUNC.MIGHT NPD.FUNC.MUST NPD.GEN.CALL.MIGHT NPD.GEN.CALL.MUST NPD.GEN.MIGHT NPD.GEN.MUST RNPD.CALL RNPD.DEREF |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2024.2 | CERT\_CPP-STR51-a | Avoid null pointer dereferencing |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2024b | [CERT C++: STR51-CPP](https://www.mathworks.com/help/bugfinder/ref/certcstr51cpp.html) | Checks for string operations on null pointer (rule partially covered). |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | [STD-004-JAV] | [Prevent SQL injection](https://wiki.sei.cmu.edu/confluence/display/java/IDS00-J.+Prevent+SQL+injection) |

| **Noncompliant Code** |
| --- |
| A prepared statement is used to sanitize any untrusted data. However, arguments such as username in this case are allowed regardless. |
| import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.sql.Statement;    class Login {  public Connection getConnection() throws SQLException {  DriverManager.registerDriver(new  com.microsoft.sqlserver.jdbc.SQLServerDriver());  String dbConnection =  PropertyManager.getProperty("db.connection");  // Can hold some value like  // "jdbc:microsoft:sqlserver://<HOST>:1433,<UID>,<PWD>"  return DriverManager.getConnection(dbConnection);  }    String hashPassword(char[] password) {  // Create hash of password  }    public void doPrivilegedAction(  String username, char[] password  ) throws SQLException {  Connection connection = getConnection();  if (connection == null) {  // Handle error  }  try {  String pwd = hashPassword(password);  String sqlString = "select \* from db\_user where username=" +  username + " and password =" + pwd;  PreparedStatement stmt = connection.prepareStatement(sqlString);    ResultSet rs = stmt.executeQuery();  if (!rs.next()) {  throw new SecurityException("User name or password incorrect");  }    // Authenticated; proceed  } finally {  try {  connection.close();  } catch (SQLException x) {  // Forward to handler  }  }  }  } |

| **Compliant Code** |
| --- |
| Placeholder ‘?’ characters are used for each argument, and username length is properly validated. |
| public void doPrivilegedAction(  String username, char[] password  ) throws SQLException {  Connection connection = getConnection();  if (connection == null) {  // Handle error  }  try {  String pwd = hashPassword(password);    // Validate username length  if (username.length() > 8) {  // Handle error  }    String sqlString =  "select \* from db\_user where username=? and password=?";  PreparedStatement stmt = connection.prepareStatement(sqlString);  stmt.setString(1, username);  stmt.setString(2, pwd);  ResultSet rs = stmt.executeQuery();  if (!rs.next()) {  throw new SecurityException("User name or password incorrect");  }    // Authenticated; proceed  } finally {  try {  connection.close();  } catch (SQLException x) {  // Forward to handler  }  }  } |

| **Principles(s):**  1. ValidateInput Data; 7. Sanitize Data Sent to Other Systems – Data sent should only include the parameters necessary for function, without potentially harmful extra fields or information. This should be checked any time a request is sent to retrieve or modify data in a database. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [The Checker Framework](https://wiki.sei.cmu.edu/confluence/display/java/The+Checker+Framework) | 2.1.3 | Tainting Checker | Trust and security errors (see Chapter 8) |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/java/CodeSonar) | 9.0p0 | JAVA.IO.INJ.SQL | SQL injection |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/java/Coverity) | 7.5 | SQLI FB.SQL\_PREPARED\_STATEMENT\_GENERATED\_ FB.SQL\_NONCONSTANT\_STRING\_PASSED\_TO\_EXECUTE | Implemented |
| [Findbugs](https://wiki.sei.cmu.edu/confluence/display/java/Findbugs) | 1.0 | SQL\_NONCONSTANT\_STRING\_PASSED\_TO\_EXECUTE | Implemented |
| [Fortify](https://wiki.sei.cmu.edu/confluence/display/java/Fortify) | 1.0 | HTTP\_Response\_Splitting SQL\_Injection\_\_Persistence SQL\_Injection | Implemented |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/java/Klocwork) | 2024.4 | SV.DATA.DB SV.SQL SV.SQL.DBSOURCE | Implemented |
| [Parasoft Jtest](https://wiki.sei.cmu.edu/confluence/display/java/Parasoft) | 2024.2 | CERT.IDS00.TDSQL | Protect against SQL injection |
| [SonarQube](https://wiki.sei.cmu.edu/confluence/display/java/SonarQube) | 9.9 | [S2077](https://rules.sonarsource.com/java/RSPEC-2077)  [S3649](https://rules.sonarsource.com/java/RSPEC-3649) | [Executing SQL queries is security-sensitive](https://rules.sonarsource.com/java/RSPEC-2077)  [SQL queries should not be vulnerable to injection attacks](https://rules.sonarsource.com/java/RSPEC-3649) |
| [SpotBugs](https://wiki.sei.cmu.edu/confluence/display/java/SpotBugs) | 4.6.0 | SQL\_NONCONSTANT\_STRING\_PASSED\_TO\_EXECUTE SQL\_PREPARED\_STATEMENT\_GENERATED\_FROM\_NONCONSTANT\_STRING | Implemented |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [STD-005-CPP] | [Do not access freed memory](https://wiki.sei.cmu.edu/confluence/display/cplusplus/MEM50-CPP.+Do+not+access+freed+memory) |

| **Noncompliant Code** |
| --- |
| Zero bytes of memory are allocated using operator new(), resulting in undefined behavior when it is dereferenced. |
| #include <new>    void f() noexcept(false) {  unsigned char \*ptr = static\_cast<unsigned char \*>(::operator new(0));  \*ptr = 0;  // ...  ::operator delete(ptr);  } |

| **Compliant Code** |
| --- |
| new is used instead of operator new(). |
| void f() noexcept(false) {  unsigned char \*ptr = new unsigned char;  \*ptr = 0;  // ...  delete ptr;  } |

| **Principles(s):**  4. Keep It Simple – Using the built-in *new* function saves keystrokes/unnecessary complexity and ensures safety for the variable type used. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | dangling\_pointer\_use |  |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | CertC++-MEM50 |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | clang-analyzer-cplusplus.NewDelete clang-analyzer-alpha.security.ArrayBoundV2 | Checked by clang-tidy, but does not catch all violations of this rule. |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 9.0p0 | ALLOC.UAF | Use after free |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Rose) |  |  |  |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | v7.5.0 | USE\_AFTER\_FREE | Can detect the specific instances where memory is deallocated more than once or read/written to the target of a freed pointer |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.4 | C++4303, C++4304 |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2024.4 | UFM.DEREF.MIGHT UFM.DEREF.MUST UFM.FFM.MIGHT UFM.FFM.MUST UFM.RETURN.MIGHT UFM.RETURN.MUST UFM.USE.MIGHT UFM.USE.MUST |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | 483 S, 484 S | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2024.2 | CERT\_CPP-MEM50-a | Do not use resources that have been freed |
| [Parasoft Insure++](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) |  |  | Runtime detection |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024b | [CERT C++: MEM50-CPP](https://www.mathworks.com/help/bugfinder/ref/certcmem50cpp.html) | Checks for:   * Pointer access out of bounds * Deallocation of previously deallocated pointer * Use of previously freed pointer   Rule partially covered. |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.36 | [V586](https://pvs-studio.com/en/docs/warnings/v586/), [V774](https://pvs-studio.com/en/docs/warnings/v774/) |  |
| [Splint](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Splint) | 5.0 |  |  |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | [STD-006-JAV] | [Never use assertions to validate method arguments](https://wiki.sei.cmu.edu/confluence/display/java/MET01-J.+Never+use+assertions+to+validate+method+arguments) |

| **Noncompliant Code** |
| --- |
| Assertions are used to validate arguments of a public method, leading to issues if assertions are disabled. |
| public static int getAbsAdd(int x, int y) {  assert x != Integer.MIN\_VALUE;  assert y != Integer.MIN\_VALUE;  int absX = Math.abs(x);  int absY = Math.abs(y);  assert (absX <= Integer.MAX\_VALUE - absY);  return absX + absY;  } |

| **Compliant Code** |
| --- |
| The method arguments are validated manually through if statements. |
| public static int getAbsAdd(int x, int y) {  if (x == Integer.MIN\_VALUE || y == Integer.MIN\_VALUE) {  throw new IllegalArgumentException();  }  int absX = Math.abs(x);  int absY = Math.abs(y);  if (absX > Integer.MAX\_VALUE - absY) {  throw new IllegalArgumentException();  }  return absX + absY;  } |

| **Principles(s):**  9. Use Effective Quality Assurance Techniques – Any code used for debugging purposes should be removed before release builds are deployed. This ensures that unneeded functions which could be potentially executed and exploited are not accidentally left in. |
| --- |

**Threat Level**

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

**Automation**

No known automated detection tools.

#### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [STD-007-CPP] | [Guarantee exception safety](https://wiki.sei.cmu.edu/confluence/display/cplusplus/ERR56-CPP.+Guarantee+exception+safety) |

| **Noncompliant Code** |
| --- |
| An array object is deallocated, and memory is re-allocated using new. However, it’s possible for the new operator to throw an exception, which would lead to undefined behavior when using the array in the future. |
| #include <cstring>    class IntArray {  int \*array;  std::size\_t nElems;  public:  // ...    ~IntArray() {  delete[] array;  }      IntArray(const IntArray& that); // nontrivial copy constructor  IntArray& operator=(const IntArray &rhs) {  if (this != &rhs) {  delete[] array;  array = nullptr;  nElems = rhs.nElems;  if (nElems) {  array = new int[nElems];  std::memcpy(array, rhs.array, nElems \* sizeof(\*array));  }  }  return \*this;  }    // ...  }; |

| **Compliant Code** |
| --- |
| new is used *before* changing the state of the array object, allowing for exceptions to be checked first. |
| #include <cstring>    class IntArray {  int \*array;  std::size\_t nElems;  public:  // ...    ~IntArray() {  delete[] array;  }    IntArray(const IntArray& that); // nontrivial copy constructor    IntArray& operator=(const IntArray &rhs) {  int \*tmp = nullptr;  if (rhs.nElems) {  tmp = new int[rhs.nElems];  std::memcpy(tmp, rhs.array, rhs.nElems \* sizeof(\*array));  }  delete[] array;  array = tmp;  nElems = rhs.nElems;  return \*this;  }    // ...  }; |

| **Principles(s):**  1. ValidateInput Data – Always check that memory is properly initialized before it’s used for a variable that will store user-inputted data, in order to avoid buffer overflows. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 9.0p0 | ALLOC.LEAK | Leak |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.4 | C++4075, C++4076 |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | 527 S, 56 D, 71 D | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2024.2 | CERT\_CPP-ERR56-a CERT\_CPP-ERR56-b | Always catch exceptions Empty 'catch' blocks should not be used |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024b | [CERT C++: ERR56-CPP](https://www.mathworks.com/help/bugfinder/ref/certcerr56cpp.html) | Checks for exceptions violating class invariant (rule fully covered). |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.36 | [V565](https://pvs-studio.com/en/docs/warnings/v565/), [V1023](https://pvs-studio.com/en/docs/warnings/v1023/), [V5002](https://pvs-studio.com/en/docs/warnings/v5002/) |  |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Input Output** | [STD-008-CPP] | [Close files when they are no longer needed](https://wiki.sei.cmu.edu/confluence/display/cplusplus/FIO51-CPP.+Close+files+when+they+are+no+longer+needed) |

| **Noncompliant Code** |
| --- |
| A file stream is not properly closed, because std::terminate() does not call its destructor. |
| #include <exception>  #include <fstream>  #include <string>    void f(const std::string &fileName) {  std::fstream file(fileName);  if (!file.is\_open()) {  // Handle error  return;  }  // ...  std::terminate();  } |

| **Compliant Code** |
| --- |
| std::fstream.close() is called and checked for errors before proceeding to call std::terminate(). |
| #include <exception>  #include <fstream>  #include <string>    void f(const std::string &fileName) {  std::fstream file(fileName);  if (!file.is\_open()) {  // Handle error  return;  }  // ...  file.close();  if (file.fail()) {  // Handle error  }  std::terminate();  } |

| **Principles(s):**  5. Default Deny; 6. Adhere to the Principle of Least Privilege – Files should not be accessible during any other time than when access is absolutely needed. Closing the file immediately after operations are done ensures that the file stream cannot be utilized by attackers during any other point during the program’s execution. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 9.0p0 | ALLOC.LEAK | Leak |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.4 | DF4786, DF4787, DF4788 |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2024.4 | RH.LEAK |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2024.2 | CERT\_CPP-FIO51-a | Ensure resources are freed |
| [Parasoft Insure++](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) |  |  | Runtime detection |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024b | [CERT C++: FIO51-CPP](https://www.mathworks.com/help/bugfinder/ref/certcfio51cpp.html) | Checks for resource leak (rule partially covered) |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Algorithms** | [STD-009-CPP] | [Do not use std::rand() for generating pseudorandom numbers](https://wiki.sei.cmu.edu/confluence/display/cplusplus/MSC50-CPP.+Do+not+use+std%3A%3Arand%28%29+for+generating+pseudorandom+numbers) |

| **Noncompliant Code** |
| --- |
| The std::rand() function has been found to be predictable in the way that it generates numbers, which could potentially cause security issues for encryption algorithms that rely on it. |
| #include <cstdlib>  #include <string>    void f() {  std::string id("ID"); // Holds the ID, starting with the characters "ID" followed  // by a random integer in the range [0-10000].  id += std::to\_string(std::rand() % 10000);  // ...  } |

| **Compliant Code** |
| --- |
| A distribution object and density function are used alongside the pseudorandom generation engine, ensuring values are properly distributed within a specified range. |
| #include <random>  #include <string>    void f() {  std::string id("ID"); // Holds the ID, starting with the characters "ID" followed  // by a random integer in the range [0-10000].  std::uniform\_int\_distribution<int> distribution(0, 10000);  std::random\_device rd;  std::mt19937 engine(rd());  id += std::to\_string(distribution(engine));  // ...  } |

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

| **Principles(s):**  3. Architect and Design for Security Policies – Although functionality wouldn’t vary much between solutions, using *rand()* makes it easier for a program’s execution to more easily be “predicted”. For security purposes, it’s best to use the solution that will result in maximized randomness for functions that require it. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | bad-function (AUTOSAR.26.5.1A) | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | CertC++-MSC50 |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 4.0 (prerelease) | cert-msc50-cpp | Checked by clang-tidy |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 9.0p0 | BADFUNC.RANDOM.RAND | Use of rand |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Rose) |  |  |  |
| [ECLAIR](https://wiki.sei.cmu.edu/confluence/display/c/ECLAIR) | 1.2 | CC2.MSC30 | Fully implemented |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.4 | C++5028 |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2024.4 | CERT.MSC.STD\_RAND\_CALL |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | 44 S | Enhanced Enforcement |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2024.2 | CERT\_CPP-MSC50-a | Do not use the rand() function for generating pseudorandom numbers |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024b | [CERT C++: MSC50-CPP](https://www.mathworks.com/help/bugfinder/ref/certcmsc50cpp.html) | Checks for use of vulnerable pseudo-random number generator (rule partially covered) |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 22.10 | bad-function (AUTOSAR.26.5.1A) | Fully checked |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Functions** | [STD-010-CPP] | [Value-returning functions must return a value from all exit paths](https://wiki.sei.cmu.edu/confluence/display/cplusplus/MSC52-CPP.+Value-returning+functions+must+return+a+value+from+all+exit+paths) |

| **Noncompliant Code** |
| --- |
| A value is only returned for negative results in this case, but not for 0 or positive results. |
| int absolute\_value(int a) {  if (a < 0) {  return -a;  }  } |

| **Compliant Code** |
| --- |
| Values greater than or equal to 0 now properly return a value as well. |
| int absolute\_value(int a) {  if (a < 0) {  return -a;  }  return a;  } |

| **Principles(s):**  2. Heed Compiler Warnings - This standard is generally identified by compilers and should not be ignored.  3. Architect and Design for Security Policies – Even if a function should never reach a particular exit path, these cases should still be considered to prevent against reverse engineering of assembly *jmp* statements which could cause the program to reach an unintended path. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | return-implicit | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | CertC++-MSC52 |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | -Wreturn-type | Does not catch all instances of this rule, such as *function-try-blocks* |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 9.0p0 | LANG.STRUCT.MRS LANG.STRUCT.NVNR | Missing return statement Non-void noreturn, |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.4 | DF2888 |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2024.4 | FUNCRET.GEN  FUNCRET.IMPLICIT |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | 2 D, 36 S | Fully implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2024.2 | CERT\_CPP-MSC52-a | All exit paths from a function, except main(), with non-void return type shall have an explicit return statement with an expression |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024b | [CERT C++: MSC52-CPP](https://www.mathworks.com/help/bugfinder/ref/certcmsc52cpp.html) | Checks for missing return statements (rule partially covered) |
| [SonarQube C/C++ Plugin](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046388) | 4.10 | [S935](https://www.sonarsource.com/products/codeanalyzers/sonarcfamilyforcpp/rules-cpp.html#RSPEC-935) |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.36 | [V591](https://pvs-studio.com/en/docs/warnings/v591/) |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 22.10 | return-implicit | Fully checked |

### 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



Currently, vulnerability testing is started in the “Verify and test” phase of pre-production. By enforcing a post-build requirement of running a project through all tools cumulatively used in the “Automation” sections of the coding standards, this could be possibly moved up to the “Build” phase. Doing so would give more flexibility in changing the design if necessary and allowing for close monitoring before moving onto production.

Within production, log collection could also be automated as part of the health check, immediately after deployment. For very high-traffic applications, it may be beneficial to incorporate tools such as LogicMonitor (<https://www.logicmonitor.com/>) or ELK Stack (<https://www.elastic.co/elastic-stack>) which can parse large log files and allow for quicker threat response than would happen through manually monitored log analysis. In the case that a threat does occur, automatic rollbacks could also be performed after the vulnerability has been detected.

### Summary of Risk Assessments

| Rule | Severity | Likelihood | Remediation Cost | Priority | Level |
| --- | --- | --- | --- | --- | --- |
| STD-003-CPP | High | Likely | Medium | P18 | L1 |
| STD-004-JAV | High | Likely | Medium | P18 | L1 |
| STD-005-CPP | High | Likely | Medium | P18 | L1 |
| STD-007-CPP | High | Likely | High | P9 | L2 |
| STD-006-JAV | Medium | Probable | Medium | P8 | L2 |
| STD-010-CPP | Medium | Probable | Medium | P8 | L2 |
| STD-002-CPP | High | Probable | High | P6 | L2 |
| STD-009-CPP | Medium | Unlikely | Low | P6 | L2 |
| STD-008-CPP | Medium | Unlikely | Medium | P4 | L3 |
| STD-001-CPP | Low | Unlikely | Low | P3 | L3 |

### Create Policies for Encryption and Triple A

| 1. **Encryption** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Encryption at rest | Applies to all data being stored, whether on a local hard drive or within a remote database. All data without exception should be encrypted before and after its access. This works as a safeguard against data breaches, which could potentially expose sensitive user data or propriety company information. |
| Encryption in flight | Applies to all data being transferred or sent. Data should always be encrypted during transit by both the sender and receiver – never sent as plaintext. This helps to protect against Man-in-the-Middle (MITM attacks, or other interception techniques. |
| Encryption in use | Applies to all data actively being utilized or modified. Both of the above policies should be used here as well, and files being currently read or written to should have the data encrypted and decrypted in real time. This policy helps to specifically prevent active threats such as SQL injection, or above-mentioned Man-in-the-Middle (MITM) attacks. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | Verification of users within a system. This is the main way employees are identified, and their permissions and level of access are determined by their unique identifier(s) such as a username and password. Their username identifier is also used for system logging purposes. |
| Authorization | After the user has been identified, authorization is given for company functions or resources based on their permission level. By default, new users have the lowest level of access which is expanded on solely based on absolute necessity, and this access is re-evaluated when their job status changes. |
| Accounting | Any changes to either local files or databases are automatically logged with the time, date, and username of the employee who made the change. This is vital to ensure company systems are being maintained and used based on our security standards and best practices, as well as allowing for inquiry into specific reasoning for the changes. |

### 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

## Audit Controls and Management

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

Evidence will include the following:

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

## Enforcement

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

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

## Exceptions Process

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

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

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

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

## Distribution

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

## Policy Change Control

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

## Policy Version History

| Version | Date | Description | Edited By | Approved By |
| --- | --- | --- | --- | --- |
| 1.0 | 08/05/2020 | Initial Template | David Buksbaum |  |
| 1.1 | 03/23/2025 | Core security principles and coding standards added. | Michelle Powers |  |
| 1.2 | 04/10/2025 | Automation, risk assessment, and encryption/Triple A policies filled out. Security policies mapped to coding standards, with explanations. | Michelle Powers |  |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

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