**Adam Conger**

**CS 405**

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



# Green Pace Secure Development Policy

## Contents

[Overview 2](#_heading=h.gjdgxs)

[Purpose 2](#_heading=h.30j0zll)

[Scope 2](#_heading=h.1fob9te)

[Module Three Milestone 2](#_heading=h.3znysh7)

[Ten Core Security Principles 2](#_heading=h.4d34og8)

[C/C++ Ten Coding Standards 3](#_heading=h.2s8eyo1)

[Coding Standard 1 4](#_heading=h.3rdcrjn)

[Coding Standard 2 5](#_heading=h.26in1rg)

[Coding Standard 3 6](#_heading=h.lnxbz9)

[Coding Standard 4 7](#_heading=h.35nkun2)

[Coding Standard 5 8](#_heading=h.1ksv4uv)

[Coding Standard 6 9](#_heading=h.2jxsxqh)

[Coding Standard 7 10](#_heading=h.z337ya)

[Coding Standard 8 11](#_heading=h.28h4qwu)

[Coding Standard 9 13](#_heading=h.nmf14n)

[Coding Standard 10 14](#_heading=h.37m2jsg)

[Defense-in-Depth Illustration 15](#_heading=h.1mrcu09)

[Project One 15](#_heading=h.46r0co2)

[1.](#_heading=h.2lwamvv) Revise the C/C++ Standards 15

[2.](#_heading=h.111kx3o) Risk Assessment 15

[3.](#_heading=h.3l18frh) Automated Detection 15

[4.](#_heading=h.206ipza) Automation 15

[5.](#_heading=h.4k668n3) Summary of Risk Assessments 16

[6.](#_heading=h.2zbgiuw) Create Policies for Encryption and Triple A 16

[7.](#_heading=h.1egqt2p) Map the Principles 17

[Audit Controls and Management 18](#_heading=h.3ygebqi)

[Enforcement 18](#_heading=h.2dlolyb)

[Exceptions Process 18](#_heading=h.sqyw64)

[Distribution 19](#_heading=h.3cqmetx)

[Policy Change Control 19](#_heading=h.1rvwp1q)

[Policy Version History 19](#_heading=h.4bvk7pj)

[Appendix A Lookups 19](#_heading=h.2r0uhxc)

[Approved C/C++ Language Acronyms 19](#_heading=h.1664s55)

## 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 | It is essential to pass only validated input to protect from untrusted data sources. Failure to validate risks serious software vulnerabilities. |
| 1. Heed Compiler Warnings | Use both static and dynamic testing tools to find security flaws. |
| 1. Architect and Design for Security Policies | Implement security policies through the software design. |
| 1. Keep It Simple | A complicated design increases chances for errors. Complex mechanisms require much more security. |
| 1. Default Deny | Access should be based on permission. Identify permission conditions. |
| 1. Adhere to the Principle of Least Privilege | Every task should be executed with the least set of privileges necessary. Elevated privileges should only be given when necessary to decrease the window of opportunity for hackers. |
| 1. Sanitize Data Sent to Other Systems | Attackers can invoke unused functionality through attacks like SQL injection. Data should be sanitized before invoking a subsystem. |
| 1. Practice Defense in Depth | It is essential to manage risk with multiple defense strategies. This reduces the likelihood of a vulnerability remaining in the codebase at the time of deployment. |
| 1. Use Effective Quality Assurance Techniques | An effective QA Program includes fuzz testing, penetration testing and source code audits. Independent reviews lead to more security. |
| 1. Adopt a Secure Coding Standard | A secure coding standard is required for target development language and platform. |

### 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** | **Do not declare or define a reserved identifier** |
| --- | --- | --- |
| **Data Type** | [STD-DCL51-CPP] | Defining an identifier in a reserved context yields undefined behavior. |

| **Noncompliant Code** |
| --- |
| Many programs use reserved names such as header guards, and may clash with a preestablished name. |
| #indef \_MY\_HEADER\_H\_  #define \_MY\_HEADER\_H\_  // contents of <my\_header.h>  #endif \_MY\_HEADER\_H\_ |

| **Compliant Code** |
| --- |
| Avoids using leading underscores |
| #indef 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):** 3; 9; It is important to understand what type of undef behavior we have, also we need to be vigilant in cases where QA may only come from a single dev. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | Reserved-identifier | Partially checked |
| Axivion-Bauhaus Suite | 7.2.0 | CertC++-DCL51 | [Insert text.] |
| Clang | 3.9 | Wreserved-id-macro | Not enabled by default but enable with Weverything, does not catch all instances of the rule. |
| CodeSonar | 7.1p0 | LANG.UD.NU.MK | Macro name is C keyword |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Do not depend on the order of evaluation for side effects** |
| --- | --- | --- |
| **Data Value** | [STD-EXP50-CPP] | There are numerous unintended restrictions and sequencing restrictions that can occur from expressions that have unsequenced ordering. |

| **Noncompliant Code** |
| --- |
| Evaluated more than once and creates undefined expression |
| void f(int i, const int \*b) {  int a = i + b[++i];  // ...  } |

| **Compliant Code** |
| --- |
| Evaluation can be interpreted in only one way. |
| **void** f(**int** i, **const** **int** \*b) {    ++i;  **int** a = i + b[i];    // ...  } |

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

| **Principles(s):** 1;4. Keep the application easy and the code base understandable and explicit. Ensure input data is valid. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | Reserved-identifier | Partially checked |
| Axivion-Bauhaus Suite | 7.2.0 | CertC++-DCL51 | [Insert text.] |
| Clang | 3.9 | Wreserved-id-macro | Not enabled by default but enable with Weverything, does not catch all instances of the rule. |
| CodeSonar | 7.1p0 | LANG.UD.NU.MK | Macro name is C keyword |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Use valid references, pointers, and iterators to reference elements of a basic\_string** |
| --- | --- | --- |
| **String Correctness** | [STD-STR52-CPP] | Invalidated references result in undefined behaviors. |

| **Noncompliant Code** |
| --- |
| Iterator loc is invalidated, behavior undefined |
| #include <string>    void f(const std::string &input) {  std::string email;    // Copy input into email converting ";" to " "  std::string::iterator loc = email.begin();  for (auto i = input.begin(), e = input.end(); i != e; ++i, ++loc) {  email.insert(loc, \*i != ';' ? \*i : ' ');  }  } |

| **Compliant Code** |
| --- |
| Iterator is incremented at end of loop |
| #include <string>    void f(const std::string &input) {  std::string email;    // Copy input into email converting ";" to " "  std::string::iterator loc = email.begin();  for (auto i = input.begin(), e = input.end(); i != e; ++i, ++loc) {  loc = email.insert(loc, \*i != ';' ? \*i : ' ');  }  } |

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

| **Principles(s):** 1;2; Ensure input data is valid and make sure there are no invalidations. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | Reserved-identifier | Partially checked |
| Axivion-Bauhaus Suite | 7.2.0 | CertC++-DCL51 | [Insert text.] |
| Clang | 3.9 | Wreserved-id-macro | Not enabled by default but enable with Weverything, does not catch all instances of the rule. |
| CodeSonar | 7.1p0 | LANG.UD.NU.MK | Macro name is C keyword |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Prevent SQL injection** |
| --- | --- | --- |
| **SQL Injection** | [STD-ISD00-J | SQL queries originating from untrusted sources can create security vulnerabilities. |

| **Noncompliant Code** |
| --- |
| Permits unsanitized input |
| 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 + "'";  Statement stmt = connection.createStatement();  ResultSet rs = stmt.executeQuery(sqlString);    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** |
| --- |
| Sanitized input |
| 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  }  }  } |

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

| **Principles(s):** 7, ensure that originating data is sent from a secure server or a known source. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | Reserved-identifier | Partially checked |
| Axivion-Bauhaus Suite | 7.2.0 | CertC++-DCL51 | [Insert text.] |
| Clang | 3.9 | Wreserved-id-macro | Not enabled by default but enable with Weverything, does not catch all instances of the rule. |
| CodeSonar | 7.1p0 | LANG.UD.NU.MK | Macro name is C keyword |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Properly deallocate dynamically allocated resources** |
| --- | --- | --- |
| **Memory Protection** | [STD-MEM51-CPP] | Do not call a deallocation function on anything other than nullptr. |

| **Noncompliant Code** |
| --- |
| End functionality of this results in undefined behavior. |
| #include <iostream>    struct S {  S() { std::cout << "S::S()" << std::endl; }  ~S() { std::cout << "S::~S()" << std::endl; }  };    void f() {  alignas(struct S) char space[sizeof(struct S)];  S \*s1 = new (&space) S;    // ...    delete s1;  } |

| **Compliant Code** |
| --- |
| Removes call to delete and calls s1’s destructor, one of a few times where this is permitted. |
| #include <iostream>    struct S {  S() { std::cout << "S::S()" << std::endl; }  ~S() { std::cout << "S::~S()" << std::endl; }  };    void f() {  alignas(struct S) char space[sizeof(struct S)];  S \*s1 = new (&space) S;    // ...    s1->~S();  } |

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

| **Principles(s):** 4. Complicated design can increase the chance of errors during deployment. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | Invalid dynamic memory allocation | Partially checked |
| Axivion-Bauhaus Suite | 7.2.0 | CertC++-MEM51 | [Insert text.] |
| Clang | 3.9 | Wreserved-id-macro | Checked by clang-tidy but doesn’t catch all |
| CodeSonar | 7.1p0 | Use ALLOC |  |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Use a static assertion to test the value of a constant expression** |
| --- | --- | --- |
| **Assertions** | [STD-DCL03-C | Find and eliminate software defects by using assertions |

| **Noncompliant Code** |
| --- |
| Diagnostic is only occurring at runtime. |
| #include <assert.h>    struct timer {  unsigned char MODE;  unsigned int DATA;  unsigned int COUNT;  };    int func(void) {  assert(sizeof(struct timer) == sizeof(unsigned char) + sizeof(unsigned int) + sizeof(unsigned int));  } |

| **Compliant Code** |
| --- |
| Preprocessor conditional statement is used and results in no runtime penalty. |
| struct timer {  unsigned char MODE;  unsigned int DATA;  unsigned int COUNT;  };    #if (sizeof(struct timer) != (sizeof(unsigned char) + sizeof(unsigned int) + sizeof(unsigned int)))  #error "Structure must not have any padding"  #endif |

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

| **Principles(s):** 3, 9 Ensure QA is done vigilant and deliberately during code reviews. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | Reserved-identifier | Partially checked |
| Axivion-Bauhaus Suite | 7.2.0 | CertC++-DCL51 | [Insert text.] |
| Clang | 3.9 | Wreserved-id-macro | Not enabled by default but enable with Weverything, does not catch all instances of the rule. |
| CodeSonar | 7.1p0 | LANG.UD.NU.MK | Macro name is C keyword |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Handle all exceptions** |
| --- | --- | --- |
| **Exceptions** | [STD-ERR51-CPP] | Programs can encounter an unrecoverable exception and terminate but not allow it to remain uncaught. |

| **Noncompliant Code** |
| --- |
| f() nor main() catch exceptions. |
| void throwing\_func() noexcept(false);    void f() {  throwing\_func();  }    int main() {  f();  } |

| **Compliant Code** |
| --- |
| The main entry point handles all exceptions |
| void throwing\_func() noexcept(false);    void f() {  throwing\_func();  }    int main() {  try {  f();  } catch (...) {  // Handle error  }  } |

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

| **Principles(s):** 4,5,6,8 Make sure that there is a single entry point for data and it only needs to be init once. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | Main function catch all | Partially checked |
| Axivion-Bauhaus Suite | 7.2.0 | CertC++-ERR51 | [Insert text.] |
| Helix | 2022.3 | CPP 4035, 6, 7 |  |
| CodeSonar | 7.1p0 | LANG.STRUCT.UCTCH | Unreachable catch |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Do not invoke virtual functions from constructors or destructors** |
| --- | --- | --- |
| Object Oriented Programming | [STD-OOP50-CPP] | Calling a derived class function from a base class under construction is dangerous and may access resources that have already been released. |

| **Noncompliant Code** |
| --- |
| Attempts to seize and release and object’s resources through calls to virtual function. |
| struct B {  B() { seize(); }  virtual ~B() { release(); }    protected:  virtual void seize();  virtual void release();  };    struct D : B {  virtual ~D() = default;    protected:  void seize() override {  B::seize();  // Get derived resources...  }    void release() override {  // Release derived resources...  B::release();  }  }; |

| **Compliant Code** |
| --- |
| Call a nonvirtual, private member function instead |
| class B {  void seize\_mine();  void release\_mine();    public:  B() { seize\_mine(); }  virtual ~B() { release\_mine(); }    protected:  virtual void seize() { seize\_mine(); }  virtual void release() { release\_mine(); }  };    class D : public B {  void seize\_mine();  void release\_mine();    public:  D() { seize\_mine(); }  virtual ~D() { release\_mine(); }    protected:  void seize() override {  B::seize();  seize\_mine();  }    void release() override {  release\_mine();  B::release();  }  }; |

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

| **Principles(s):** 5 and 6. Make sure that everything in the application is only allowed at certain authentication or authorization levels. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | Reserved-identifier | Partially checked |
| Axivion-Bauhaus Suite | 7.2.0 | CertC++-DCL51 | [Insert text.] |
| Clang | 3.9 | Wreserved-id-macro | Not enabled by default but enable with Weverything, does not catch all instances of the rule. |
| CodeSonar | 7.1p0 | LANG.UD.NU.MK | Macro name is C keyword |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Close files when they are no longer needed** |
| --- | --- | --- |
| Input Output | [STD-FIO51-CPP | Use stream object by value semantics, not dynamic memory allocation |

| **Noncompliant Code** |
| --- |
| Underlying object not properly closed |
| #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** |
| --- |
| File resources properly closed |
| #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();  } |

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

| **Principles(s):** 4, 8 Make sure the application is simple and concise so other developers can acknowledge and understand what is happening overall. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| CodeSonar | 7.1p0 | Leak | Leak |
| HELIXQAC | 2022.3 | CPP 4786 4787 4788 | [Insert text.] |
| Klockwork | 2022.3 | RH leak | [Insert text.] |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Do not allow data races in multithreaded code** |
| --- | --- | --- |
| Concurrency | [STD-CON43-C] | Synchronization techniques will help avoid software flaws. |

| **Noncompliant Code** |
| --- |
| Operations are not atomic and operations can occur concurrently, resulting in saving only one operation. |
| static volatile int account\_balance;    void debit(int amount) {  account\_balance -= amount;  }    void credit(int amount) {  account\_balance += amount;  } |

| **Compliant Code** |
| --- |
| Attackers cannot hack the race condition to steal money from the bank because of the mutex making operations atomic. |
| #include <threads.h>    static int account\_balance;  static mtx\_t account\_lock;    int debit(int amount) {  if (mtx\_lock(&account\_lock) == thrd\_error) {  return -1; /\* Indicate error to caller \*/  }  account\_balance -= amount;  if (mtx\_unlock(&account\_lock) == thrd\_error) {  return -1; /\* Indicate error to caller \*/  }  return 0; /\* Indicate success \*/  }    int credit(int amount) {  if (mtx\_lock(&account\_lock) == thrd\_error) {  return -1; /\* Indicate error to caller \*/  }  account\_balance += amount;  if (mtx\_unlock(&account\_lock) == thrd\_error) {  return -1; /\* Indicate error to caller \*/  }  return 0; /\* Indicate success \*/  }    int main(void) {  if(mtx\_init(&account\_lock, mtx\_plain) == thrd\_error) {  /\* Handle error \*/  }  /\* ... \*/  } |

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

| **Principles(s):** 8,9 – It is important that the application has the correct iterative functionality. Additionally we will also want the application to be run on runtime and secure through deployment. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 22 | Reserved-identifier | Partially checked |
| Coverity | 2017.7 | Missing lock (partial) | [Insert text.] |
| Helix | 2022.3 | C1765, 1766, 1770, 1771 |  |
| CodeSonar | 7.1p0 | CONCURRENCY.DATARACE |  |

### 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.

We will want to ensure that all authentication methods return the correct access and privileges to certain actions and resources within our application. Overall we will also want to allow for authorization to be correct and true. We can apply our Triple A policies to make sure the overall application is secure enough for deployment and automated security checks during integration testing before it’s deployed for public use.

### 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-DCL51-CPP | Low | Unlikely | Low | P3 | L3 |
| STD-EXP50-CPP | Medium | Probable | Medium | P8 | L2 |
| STD-STR52-CPP | High | Probable | High | P6 | L2 |
| STD-ISD00-J | High | Probable | Medium | P12 | L1 |
| STD-MEM51-CPP | High | Likely | Medium | P18 | L1 |
| STD-DCL03-C | High | Probable | High | P6 | L2 |
| STD-ERR51-CPP | Low | Probable | Medium | P4 | L3 |
| STD-OOP50-CPP | Low | Unlikely | Medium | P2 | L3 |
| STD-FIO51-CPP | Medium | Unlikely | Medium | P4 | L3 |
| STD-CON43-C | Medium | Probable | High | 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.] | [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 | Inactive data is still encrypted and kept secure from hackers when the company is not developing. |
| Encryption at flight | Active data sent from the backend to the front end in application deployment is encrypted. |
| Encryption in use | Data transferred in real time is also encrypted to ensure security and prevent meltdowns. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | User Logins all have roles: Enables organizations to keep networks secure, only permitting authenticated users to access protected resources. |
| Authorization | User level of access is defined by role: Allows permissions to enact a given action on specific resources. |
| Accounting | Files accessed by users are logged by their credentials: this ensures a valid history is kept in records and data structures for auditing. |

**\***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 |  |
| 2.0 | 10/09/2022 | Update Template | Adam Conger | Prof. Gappy |
| 3.0 | 10/09/2023 | Update Template | TBD | TBD |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

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