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



# Green Pace Secure Development Policy

## Contents

[Overview 2](#_Toc52464053)

[Purpose 2](#_Toc52464054)

[Scope 2](#_Toc52464055)

[Module Three Milestone 2](#_Toc52464056)

[Ten Core Security Principles 2](#_Toc52464057)

[C/C++ Ten Coding Standards 3](#_Toc52464058)

[Coding Standard 1 4](#_Toc52464059)

[Coding Standard 2 5](#_Toc52464060)

[Coding Standard 3 6](#_Toc52464061)

[Coding Standard 4 7](#_Toc52464062)

[Coding Standard 5 8](#_Toc52464063)

[Coding Standard 6 9](#_Toc52464064)

[Coding Standard 7 10](#_Toc52464065)

[Coding Standard 8 11](#_Toc52464066)

[Coding Standard 9 13](#_Toc52464067)

[Coding Standard 10 14](#_Toc52464068)

[Defense-in-Depth Illustration 15](#_Toc52464069)

[Project One 15](#_Toc52464070)

[1. Revise the C/C++ Standards 15](#_Toc52464071)

[2. Risk Assessment 15](#_Toc52464072)

[3. Automated Detection 15](#_Toc52464073)

[4. Automation 15](#_Toc52464074)

[5. Summary of Risk Assessments 16](#_Toc52464075)

[6. Create Policies for Encryption and Triple A 16](#_Toc52464076)

[7. Map the Principles 17](#_Toc52464077)

[Audit Controls and Management 18](#_Toc52464078)

[Enforcement 18](#_Toc52464079)

[Exceptions Process 18](#_Toc52464080)

[Distribution 19](#_Toc52464081)

[Policy Change Control 19](#_Toc52464082)

[Policy Version History 19](#_Toc52464083)

[Appendix A Lookups 19](#_Toc52464084)

[Approved C/C++ Language Acronyms 19](#_Toc52464085)

## Overview

Software development at Green Pace requires consistent implementation of secure principles to all developed applications. Consistent approaches and methodologies must be maintained through all policies that are uniformly defined, implemented, governed, and maintained over time.

## Purpose

This policy defines the core security principles; C/C++ coding standards; authorization, authentication, and auditing standards; and data encryption standards. This article explains the differences between policy, standards, principles, and practices (guidelines and procedure): [Understanding the Hierarchy of Principles, Policies, Standards, Procedures, and Guidelines](https://www.linkedin.com/pulse/understanding-hierarchy-principles-policies-standards-wally-beddoe/).

## Scope

This document applies to all staff that create, deploy, or support custom software at Green Pace.

## Module Three Milestone

### Ten Core Security Principles

| **Principles** | Write a short paragraph explaining each of the 10 principles of security. |
| --- | --- |
| 1. ValidateInput Data | Prior to ingesting data from an outside source, the data needs to be checked for erroneous types. This is used to prevent attacks through this domain such as SQL injection or buffer overflows. |
| 1. Heed Compiler Warnings | When compiling code, you need to ensure you are being mindful to what your compiler is notifying you of. Compilers are typically updated with the latest security standards, so they will notify you potential areas within your code that need adjusting, such as unused variables or ones that are the wrong scope. |
| 1. Architect and Design for Security Policies | Security should not be something to consider after the infrastructure is deployed. During the initial phases of planning for a deployment, security should be built into it. This could address areas of layer 1 or physical security, such as implementing firewalls or cloud security groups that handle access control lists (ACLs). |
| 1. Keep It Simple | When creating systems or applications, its useful to keep their design simple in nature. By doing so, you can eliminate the complexity from certain domains which allows administrators and engineers to work with the technology better. |
| 1. Default Deny | When working with access control lists or processes that handle control, it is always better to deny all as default. Then, you can allow only what you specifically detail to the system. This ensures that nothing goes under your radar, and you can be confident that any connections or programs that are used are allowed. |
| 1. Adhere to the Principle of Least Privilege | With systems, privileges are ones key to having control over the environment. One should only ever be assigned permissions or privileges to only accomplish what is outlined for their specific role. The highest level of privileges should never be given unless only absolutely necessary. |
| 1. Sanitize Data Sent to Other Systems | When data is outgoing, it could potentially contain information that is critically sensitive to an organization. This could be passwords, networking or device information or even PII. It is crucial to scrub this type of data with either blanks or placeholders to allow the original data to still be sent. |
| 1. Practice Defense in Depth | You should never rely on a single point to secure your systems. In order words, do not only rely on a host level firewall to block any malicious actors from accessing your infrastructure. Instead, implement ACLs, firewall rules, correctly assigned user permissions and anti-virus. |
| 1. Use Effective Quality Assurance Techniques | Quality assurance is the key to ensuring to the best of an organization’s ability, that what is reaching the end customer is bug free and secure. Without these techniques in place, the customer may be accessing a product that allows for a security violations or has open doors for bad actors, since these were not previously detected. |
| 1. Adopt a Secure Coding Standard | To adopt a secure coding standard is to implement security within the applications code, and not to only rely on the surrounding systems. This could be accounting for buffer under or overflows, input validation or even properly handling of sensitive information. |

### 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 | Use the correct data type for the correct purpose |

| **Noncompliant Code** |
| --- |
| String used in place of a char |
| std::string ch = "A";  character std::cout << ch << std::endl; |

| **Compliant Code** |
| --- |
| Character used for char data type |
| char ch = 'A';  std::cout << ch << std::endl; |

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

| **Principles(s):** 3: Architect and Design for Security Policies. A secure design can enforce strong and defined typing rules |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Medium | Medium | Low | Medium | 2 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| SonarQube | 25.2 | Format strings should comply with ISO standards | Incorrec type conversions |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | STD-002-CPP | Data values should match the requirements of the data type |

| **Noncompliant Code** |
| --- |
| Assigning a string data value to an integer data type |
| int age = "25"; |

| **Compliant Code** |
| --- |
| [Compliant description] |
| Int age = 25; |

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

| **Principles(s):** 1: Validate Input Data. Data values need to be within the allowed range and format |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Medium | High | Low | High | 3 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Clang-Tidy | 21.0.0 | cppcoreguidelines-init-variables | Checks proper integer initialization |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | STD-003-CPP | Strings should be set to a series of characters |

| **Noncompliant Code** |
| --- |
| A string being set to an integer value |
| Std:: string num = 25 |

| **Compliant Code** |
| --- |
| A string being set to a string value |
| Std:string num = “twenty five” |

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

| **Principles(s):** 1: Validate Input Data. Data values need to be within the allowed range and format |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | High | Medium | High | 4 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-ERR34** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/c/Clang) | 3.9 | cert-err34-c | Checked by clang-tidy |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.3p0 | **BADFUNC.ATOF** **BADFUNC.ATOI** **BADFUNC.ATOL** **BADFUNC.ATOLL**  **(customization)** | Use of atof Use of atoi Use of atol Use of atoll  Users can add custom checks for uses of other undesirable conversion functions. |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | STD-004-STD | Inputs must be handled to avoid SQL injections |

| **Noncompliant Code** |
| --- |
| Typical user input that is open to any string |
| std::string userInput = "John' OR '1'='1"; |

| **Compliant Code** |
| --- |
| Leveraging the sqlite3\_prepare\_v2, we can treat the input as separate from SQL logic |
| std::string userInput = "John' OR '1'='1";  sqlite3\_prepare\_v2(db, userInput.c\_str(), -1, &stmt, nullptr); |

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

| **Principles(s):** 1: Validate Input Data. Data values need to be within the allowed range and format  8: Practice Defense in Depth. SQL injection attacks must be properly handled |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | High | Medium | High | 4 |

**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) | 8.1p0 | **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 |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | STD-005-CPP | Pointers shall be properly disposed of |

| **Noncompliant Code** |
| --- |
| Contains a dangling pointer |
| int\* ptr = new int(42);  delete ptr; |

| **Compliant Code** |
| --- |
| Leverage smart pointers |
| std::unique\_ptr<int> ptr = std::make\_unique<int>(42); |

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

| **Principles(s):** 5: Default Deny & 6: Adhere to the Principle of Least Privilege. Pointers need to be properly managed to avoid memory leaks |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | High | Medium | High | 4 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-INT36** | Fully implemented |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/c/Clang) | 3.9 | -Wint-to-pointer-cast, -Wint-conversion | Can detect some instances of this rule, but does not detect all |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.3p0 | **LANG.CAST.PC.CONST2PTR** **LANG.CAST.PC.INT** | Conversion: integer constant to pointer Conversion: pointer/integer |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | STD-006-CP | Leverage assertions when possible |

| **Noncompliant Code** |
| --- |
| Missing assertion |
| if (b == 0) {  std::cout << "Error: Division by zero!" << std::endl;  return -1;  }  return a / b; |

| **Compliant Code** |
| --- |
| Clear assertion is visible to halt the program |
| assert(b != 0 && "Division by zero detected!");  return a / b; |

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

| **Principles(s):** 10: Adopt a Secure coding standard. Assertions can eliminate incorrect assumptions |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Medium | Low | Low | 1 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/java/CodeSonar) | 8.1p0 | **JAVA.STRUCT.SE.ASSERT** | Assertion contains side effects |
| [Parasoft Jtest](https://wiki.sei.cmu.edu/confluence/display/java/Parasoft) | 2024.2 | **CERT.EXP06.EASE** | Expressions used in assertions must not produce side effects |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | STD-007-CPP | Use exceptions when fatal errors can be present |

| **Noncompliant Code** |
| --- |
| No way to catch for failing values |
| Return a/b; |

| **Compliant Code** |
| --- |
| Proper try catch to identify the exception |
| try {  Return a/b;  } catch (const std::runtime\_error& e) {  std::cerr << "Exception caught: " << e.what() << std::endl;  } |

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

| **Principles(s):** 9: Use effective quality assurance techniques. Exceptions can improve the reliability with a smaller number of crashes |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Medium | Medium | Medium | Medium | 3 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 8.3p0 | **LANG.STRUCT.UCTCH** | Unreachable Catch |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.4 | **C++4035, C++4036, C++4037** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2024.4 | **MISRA.CATCH.ALL** |  |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| File Handling | STD-008-CPP | Files must be properly closed |

| **Noncompliant Code** |
| --- |
| File is not closed after completed operations |
| std::ofstream file("output.txt");  file << "Writing to file..." << std::endl; |

| **Compliant Code** |
| --- |
| File is properly closed after operations |
| std::ofstream file("output.txt");  file << "Writing to file..." << std::endl;  file.close(); |

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

| **Principles(s):** 5: Default Deny & 8: Practice Defense in Depth. Proper file handling is needed to avoid resource exhaustion |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Medium | Medium | Low | Medium | 2 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| CodeSonar | 8.3p0 | ALLOC.LEAK | Leak |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.4 | **DF4786, DF4787, DF4788** |  |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Buffer overflow | STD-009-CPP | Avoid buffer overflows |

| **Noncompliant Code** |
| --- |
| Buffer overflow created |
| char buffer[10]; // Fixed-size buffer  std::strcpy(buffer, "This is a very very long string"); |

| **Compliant Code** |
| --- |
| Perform bounds checking to avoid buffer overflow |
| char buffer[10];  std::strncpy(buffer, "This is a long string", sizeof(buffer) - 1);  buffer[sizeof(buffer) - 1] = '\0'; |

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

| **Principles(s):** 10: Adopt a secure coding standard. Overflows can cause crashes and other various security exploits |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | High | High | High | 5 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 24.04 | **malloc-size-insufficient** | Partially checked  Besides direct rule violations, all undefined behaviour resulting from invalid memory accesses is reported by Astrée. |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=125337650) | 7.2.0 | **CertC-MEM35** |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.3p0 | **ALLOC.SIZE.ADDOFLOW** **ALLOC.SIZE.IOFLOW** **ALLOC.SIZE.MULOFLOW** **ALLOC.SIZE.SUBUFLOW** **ALLOC.SIZE.TRUNC** **IO.TAINT.SIZE** **MISC.MEM.SIZE.BAD** **LANG.MEM.BO** **LANG.MEM.BU** **LANG.STRUCT.PARITH** **LANG.STRUCT.PBB** **LANG.STRUCT.PPE** **LANG.MEM.TBA** **LANG.MEM.TO** **LANG.MEM.TU** | Addition overflow of allocation size Addition overflow of allocation size Multiplication overflow of allocation size Subtraction underflow of allocation size Truncation of allocation size Tainted allocation size Unreasonable size argument Buffer Overrun Buffer Underrun Pointer Arithmetic Pointer Before Beginning of Object Pointer Past End of Object Tainted Buffer Access Type Overrun Type Underrun |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Null Pointer | STD-010-CPP | Avoid dereference null pointer |

| **Noncompliant Code** |
| --- |
| Incorrectly dereference null pointer |
| int\* ptr = nullptr;  pointer \*ptr = 42; |

| **Compliant Code** |
| --- |
| Ensures the pointer is not null |
| int\* ptr = nullptr;  if (ptr) {  \*ptr = 42;  std::cout << "Value: " << \*ptr << std::endl;  } else {  std::cerr << "Error: Null pointer dereference prevented" << std::endl; } |

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

| **Principles(s):** 2: Heed compiler warnings. Segmentation faults and crashes can appear from not dereferencing null pointers |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | High | Medium | High | 4 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 24.04 | **null-dereferencing** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-EXP34** |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.3p0 | **LANG.MEM.NPD** **LANG.STRUCT.NTAD** **LANG.STRUCT.UPD** | Null pointer dereference Null test after dereference Unchecked parameter dereference |

### 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.

Automation should be enforced during the Design and Build stages, or development stages of the DevOps process. The security policies outline in this document should not be seen as an afterthought, but rather be built and architected in. By using tools such as SonarQube, automated checks should be built into the CI process to only allow merges of code after they have been return successful and clean code scans. Once merged, the code can undergo functional tests and later on user tests.

### 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 | Medium | Medium | Low | Medium | 2 |
| STD-002-CPP | Medium | High | Low | High | 3 |
| STD-003-CPP | High | High | Medium | High | 4 |
| STD-004-CPP | High | High | Medium | High | 4 |
| STD-005-CPP | High | High | Medium | High | 4 |
| STD-006-CPP | Low | Medium | Low | Low | 1 |
| STD-007-CPP | Medium | Medium | Medium | Medium | 3 |
| STD-008-CPP | Medium | Medium | Low | Medium | 2 |
| STD-009-CPP | High | High | High | High | 5 |
| STD-010-CPP | High | High | Medium | High | 4 |

### 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 at rest | This encryption method involves encrypting data that is stored. The encrypted data can be accessed, however without a proper decryption key it is usable. |
| Encryption in flight | Data that is in transmission from one system to another can be encrypted. This protects from attacks such as man in the middle. |
| Encryption in use | Encrypting data that is currently in use allows for protection against any hijacking parties to read sensitive data. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | Ensuring a user or person is who they claim to be. |
| Authorization | Cross referencing a user to any roles they are assigned, to ensure they are accessing the correct information in the correct manner. |
| Accounting | Auditing any an all actions performed by users that perform authorized tasks. |

**\***Use this checklist for the Triple A to be sure you include these elements in your policy:

* User logins
* Changes to the database
* Addition of new users
* User level of access
* Files accessed by users

### Map the Principles

Map the principles to each of the standards, and provide a justification for the connection between the two. In the Module Three milestone, you added definitions for each of the 10 principles provided. Now it’s time to connect the standards to principles to show how they are supported by principles. You may have more than one principle for each standard, and the principles may be used more than once. Principles are numbered 1 through 10. You will list the number or numbers that apply to each standard, then explain how each of these principles supports the standard. This exercise demonstrates that you have based your security policy on widely accepted principles. Linking principles to standards is a best practice.

**NOTE:** Green Pace has already successfully implemented the following:

* Operating system logs
* Firewall logs
* Anti-malware logs

The only item you must complete beyond this point is the Policy Version History table.

## Audit Controls and Management

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

Evidence will include the following:

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

## Enforcement

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

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

## Exceptions Process

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

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

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

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

## Distribution

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

## Policy Change Control

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

## Policy Version History

| Version | Date | Description | Edited By | Approved By |
| --- | --- | --- | --- | --- |
| 1.0 | 08/05/2020 | Initial Template | David Buksbaum |  |
| 1.1 | 02/10/2025 | Milestone Update | Tyler Pimental |  |
| 1.2 | 02/22/2025 | Project 1 | Tyler Pimental |  |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

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