**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 | Any data that looks suspicious should be checked. Look at command line arguments, network interfaces, environmental variables and user controlled files |
| 1. Heed Compiler Warnings | Pay attention to warnings from the compiler. By fixing the code and getting red of the red will also stop security flaws. |
| 1. Architect and Design for Security Policies | Check that your design also takes into account visual and security safety. Be sure that it enforces security policies relevant to your design. |
| 1. Keep It Simple | Design should be simple and miniature. More complicated design would easily cause more complicated errors that need fixing. |
| 1. Default Deny | By default access is denied. It should only give permission to the individuals specified. |
| 1. Adhere to the Principle of Least Privilege | By making less access to all that leaves fewer accounts with higher access. These higher access accounts should be used quickly and rarely allowing less chances for hackers. |
| 1. Sanitize Data Sent to Other Systems | The calling process must be used to insure that all data is secure with no unused data they can use. |
| 1. Practice Defense in Depth | Use multiple layers of defense to ensure less chance of hacking or if nothing else making it harder for them to get with as little information gathered. |
| 1. Use Effective Quality Assurance Techniques | source code audits, fuzz testing and Penetration testing should all be used as an offensive way to stop hacking. |
| 1. Define Security Requirements | Having a list at the beginning will ensure that the system can efffectivelly be checked when needed. |

## 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** | [DCL50-CPP] | Don’t define an already named identifier. |

| **Noncompliant Code** |
| --- |
| X should be started with an underscore. |
| #include <cstddef>    unsigned int operator"" x(const char \*, std::size\_t); |

| **Compliant Code** |
| --- |
| The correct underscore is before x in the example below. |
| #include <cstddef>    unsigned int operator"" \_x(const char \*, std::size\_t); |

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

|  |
| --- |
| **Principles(s):** Using reserved identifiers can lead to incorrect program operation |

**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 | 6.9.0 | CertC++-DCL51 |  |
| RuleChecker | 20.10 | Reserved-identifier | Partially checked |
| SonarQube C/C++ Plugin | 4.10 | 978 |  |

### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | [FI008-J] | Distinguish between characters or bytes read from a stream and -1 |

| **Noncompliant Code** |
| --- |
| It will return -1 only when the input string is reached. It will then confuse a byte for a -1 and stop the loop short. |
| FileInputStream in;  // Initialize stream  byte data;  while ((data = (byte) in.read()) != -1) {  // ...  } |

| **Compliant Code** |
| --- |
| When the value returned is not -1 it will return false |
| FileInputStream in;  // Initialize stream  int inbuff;  byte data;  while ((inbuff = in.read()) != -1) {  data = (byte) inbuff;  // ...  } |

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

|  |
| --- |
| **Principles(s):** Ensure conversions of numeric types of narrower types do not result in lost or misinterpreted data. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Parasoft Jtest | 2020.2 | PB.LOGIC.CRRV | Check the return value of methods which read or skip input |

### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | [ERR34-C] | Find errors when converting a string to a number |

| **Noncompliant Code** |
| --- |
| Uses the sscanf() function to convert a string token to an integer. However it fails to report errors from strtol() |
| #include <stdio.h>    void func(const char \*buff) {  int matches;  int si;    if (buff) {  matches = sscanf(buff, "%d", &si);  if (matches != 1) {  /\* Handle error \*/  }  } else {  /\* Handle error \*/  }  } |

| **Compliant Code** |
| --- |
| Strtol() function converts a string token to an integer and ensures that the value is in range of int |
| #include <errno.h>  #include <limits.h>  #include <stdlib.h>  #include <stdio.h>    void func(const char \*buff) {  char \*end;  int si;    errno = 0;    const long sl = strtol(buff, &end, 10);    if (end == buff) {  fprintf(stderr, "%s: not a decimal number\n", buff);  } else if ('\0' != \*end) {  fprintf(stderr, "%s: extra characters at end of input: %s\n", buff, end);  } else if ((LONG\_MIN == sl || LONG\_MAX == sl) && ERANGE == errno) {  fprintf(stderr, "%s out of range of type long\n", buff);  } else if (sl > INT\_MAX) {  fprintf(stderr, "%ld greater than INT\_MAX\n", sl);  } else if (sl < INT\_MIN) {  fprintf(stderr, "%ld less than INT\_MIN\n", sl);  } else {  si = (int)sl;    /\* Process si \*/  }  } |

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

|  |
| --- |
| **Principles(s):** It is a rare violation of this rule to result in a security vulnerability unless it occurs in security-sensitive code. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Axivion Bauhaus Suite | 6.9.0 | CertC-ERR34 |  |
| Clang | 3.9 | cert-err34-c | Checked by clang-tidy |
| Klocwork | 2018 | MISRA.STDLIB.ATOI |  |
| PC-lint Plus | 1.4 | 586 | Assistance provided |

### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | [IDS00-J] | Prevent SQL injection. SGL query from an unreliable source. |

| **Noncompliant Code** |
| --- |
| A hacker can write validuser or 1=1 to easily attack |
| 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** |
| --- |
| Validates the length of the username stopping attacks and uses a ? character in place for the argument |
| 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):** Failure to sanitize user input before processing or storing it can result in injection attacks. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| The Checker Framework | 2.1.3 | Tainting Checker | Trust and security errors |
| Findbugs | 1.0 | SQL\_NONCONSTANT\_STRING\_PASSED\_TO\_EXECUTE | Implemented |
| Parasoft Jtest | 2020.2 | BD-SECURITY-TDSQL | Protect against SQL injection |
| SonarQube | 6.7 | S2077  S3649 | Executing SQL queries is security-sensitive  SQL queries should not be vulnerable to injection attacks |

### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [MSC04-J] | Do not leak memory. |

| **Noncompliant Code** |
| --- |
| n> 0 should be written as n>=0 this will cause it to miss one element and exhaust all available space |
| public class Leak {  static Vector vector = new Vector();    public void useVector(int count) {  for (int n = 0; n < count; n++) {  vector.add(Integer.toString(n));  }  // ...  for (int n = count - 1; n > 0; n--) { // Free the memory  vector.removeElementAt(n);  }  }    public static void main(String[] args) throws IOException {  Leak le = new Leak();  int i = 1;  while (true) {  System.out.println("Iteration: " + i);  le.useVector(1);  i++;  }  }  } |

| **Compliant Code** |
| --- |
| Corrects above mistake and writes it as n >=0 |
| public void useVector(int count) {  int n = 0;  try {  for (; n < count; n++) {  vector.add(Integer.toString(n));  }  // ...  } finally {  for (n = n - 1; n >= 0; n--) {  vector.removeElementAt(n);  }  }  } |

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

|  |
| --- |
| **Principles(s):** Memory leaks in Java applications may be exploited in a DoS attack. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Parasoft Jtest | 2020.2 | BD.RES.LEAKS | Ensure resources are deallocated |

### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | [MSC11-C] | Add diagnostic tests using assertions |

| **Noncompliant Code** |
| --- |
| Using the asser to verify memory allocation will cause termination opening the denial of service attack |
| char \*dupstring(const char \*c\_str) {  size\_t len;  char \*dup;    len = strlen(c\_str);  dup = (char \*)malloc(len + 1);  assert(NULL != dup);    memcpy(dup, c\_str, len + 1);  return dup;  } |

| **Compliant Code** |
| --- |
| This shows how to detect and handle memory exhaustion. |
| char \*dupstring(const char \*c\_str) {  size\_t len;  char \*dup;    len = strlen(c\_str);  dup = (char\*)malloc(len + 1);  /\* Detect and handle memory allocation error \*/  if (NULL == dup) {  return NULL;  }    memcpy(dup, c\_str, len + 1);  return dup;  } |

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

|  |
| --- |
| **Principles(s):** Assertions are a valuable diagnostic tool for finding and eliminating software defects that may result in vulnerabilities. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| CodeSonar | 6.0p0 | **LANG.FUNCS.ASSERTS** | Not enough assertions |
| Coverity | 2017.07 | **ASSERT\_SIDE\_EFFECT** | Can detect the specific instance where assertion contains an operation/function call that may have a side effect |
| Parasoft C/C++test | 2020.2 | CERT\_C-MSC11-a | Assert liberally to document internal assumptions and invariants |

### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [ERR55-CPP] | Honor exception specifications. |

| **Noncompliant Code** |
| --- |
| A function is declared as nonthrowing but it is possible for std::vector::resize() to throw an exception when the memory cannot be allocated. |
| #include <cstddef>  #include <vector>    void f(std::vector<int> &v, size\_t s) noexcept(true) {  v.resize(s); // May throw  } |

| **Compliant Code** |
| --- |
| The noexcept specification is removed allowing all exceptions |
| #include <cstddef>  #include <vector>    void f(std::vector<int> &v, size\_t s) {  v.resize(s); // May throw, but that is okay  } |

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

|  |
| --- |
| **Principles(s):** throwing unexpected exceptions disrupt control flow and can cause premature termination and denial of service. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astree | 20.10 | **unhandled-throw-noexcept** | Partially checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 6.9.0 | **CertC++-ERR55** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **56 D** | Partially implemented |
| [Parasoft C/C++Test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2020.2 | **CERT\_CPP-ERR55-a** | Where a function's declaration includes an exception-specification, the function shall only be capable of throwing exceptions of the indicated type(s) |

### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Data type | [INT00-C] | Understand the data model used by your implementations |

| **Noncompliant Code** |
| --- |
| It works for sizeof(int) == sizeof(long) others is causes an unexpected memory write to a buffer overflow |
| int f(void) {  FILE \*fp;  int x;  /\* ... \*/  if (fscanf(fp, "%ld", &x) < 1) {  return -1; /\* Indicate failure \*/  }    /\* ... \*/  return 0;  } |

| **Compliant Code** |
| --- |
| This uses the correct format for the type being used. |
| int f(void) {  FILE \*fp;  int x;  /\* Initialize fp \*/  if (fscanf(fp, "%d", &x) < 1) {  return -1; /\* Indicate failure \*/  }    /\* ... \*/  return 0;  } |

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

|  |
| --- |
| **Principles(s):** Making assumptions about the size of data types may lead to buffer-overflow-style attacks. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Axivion Bauhaus Suite | 6.9.0 | CertC-INT00 |  |
| PC-lint Plus | 1.4 | 559, 705, 706, 2403 | Assistance provided: Reports data type inconsistencies in format strings |
| Polyspace Bug Finder | R2020a | CERT C: Rec. INT00-C | Checks for:  Use of basic numerical types instead of typedef-s  Integer overflow or integer constant overflow  Format string specifiers and arguments mismatch  Rec. partially covered. |
| PVS-Studio | 7.07 | V629 |  |

### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| String Correctness | [FI047-C] | Use valid format strings |

| **Noncompliant Code** |
| --- |
| The error type argument to print is incorrectly matched with the s instead of d specifier |
| #include <stdio.h>    void func(void) {  const char \*error\_msg = "Resource not available to user.";  int error\_type = 3;  /\* ... \*/  printf("Error (type %s): %d\n", error\_type, error\_msg);  /\* ... \*/  } |

| **Compliant Code** |
| --- |
| Printf function matches their respective conversion specifications |
| #include <stdio.h>    void func(void) {  const char \*error\_msg = "Resource not available to user.";  int error\_type = 3;  /\* ... \*/  printf("Error (type %d): %s\n", error\_type, error\_msg);    /\* ... \*/  } |

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

|  |
| --- |
| **Principles(s):** Incorrectly specified format strings can result in memory corruption or abnormal program termination. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 6.9.0 | **CertC-FIO47** | Fully Implemented |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 6.0p0 | **IO.INJ.FMT MISC.FMT MISC.FMTTYPE** | Format string injection  Format string  Format string type error |
| Coverity | 2017.07 | PW | Reports when the number of arguments differs from the number of required arguments according to the format string |
| GCC | 4.3.5 |  | Can detect violations of this recommendation when the -Wformat flag is used |

### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Miscellaneous | [MSC01-C] | Strive for logical completeness |

| **Noncompliant Code** |
| --- |
| Fails to test if conditions where a is neither b or c. |
| if (a == b) {  /\* ... \*/  }  else if (a == c) {  /\* ... \*/  } |

| **Compliant Code** |
| --- |
| Correctly tests every situation |
| if (a == b) {  /\* ... \*/  }  else if (a == c) {  /\* ... \*/  }  else {  /\* Handle error condition \*/  } |

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

|  |
| --- |
| **Principles(s):** Failing to account for all possibilities within a logic statement can lead to a corrupted running state, potentially resulting in unintentional information disclosure or abnormal termination |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astrée | 20.10 | missing-else  switch-default | Partially checked |
| Compass/ROSE |  |  | Can detect some violations of this recommendation. In particular, it flags switch statements that do not have a default clause. ROSE should detect "fake switches" as well (that is, a chain of if statements each checking the value of the same variable). These if statements should always end in an else clause, or they should mathematically cover every possibility. For instance, consider the following:  if (x > 0) {  /\* ... \*/  } else if (x < 0) {  /\* ... \*/  } else if (x == 0) {  /\* ... \*/  } |
| GCC | 4.3.5 |  | Can detect some violations of this recommendation when the -Wswitch and -Wswitch-default flags are used |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **48 S, 59 S** | Fully implemented |

## Defense-in-Depth Illustration

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



# Project One

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

## Revise the C/C++ Standards

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

## Risk Assessment

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

## Automated Detection

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

## Automation

Provide a written explanation using the image provided.



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

* I think for the enforcing of standards should be in two locations. One in the Assess and plan section and also the transition and health check. That way it is thoroughly being used in the beginning and end of the process. This way no security is lost or forgotten.

## Summary of Risk Assessments

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

| Rule | Severity | Likelihood | Remediation Cost | Priority | Level |
| --- | --- | --- | --- | --- | --- |
| STD-001-CPP | High | Unlikely | Medium | High | 2 |
| DCL50-CPP | Low | Unlikely | Low | P3 | L3 |
| FI008-J | High | Probable | Medium | P12 | L1 |
| ERR34-C | Medium | Unlikely | Medium | P4 | L3 |
| IDS00-J | High | Probable | Medium | P12 | L1 |
| MSCO4-J | Low | Unlikely | High | P1 | L3 |
| MSC11-C | Low | Unlikely | High | P1 | L3 |
| ERR55-CPP | Low | Likely | Low | P9 | L2 |
| INT00-C | High | Unlikely | High | P3 | L3 |
| FI047-C | High | Unlikely | Medium | P6 | L2 |
| MSC01-C | Medium | Probable | Medium | P8 | L2 |

## 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 | This is data that is stored and protected requiring an encryption key to decode it that only authorized personnel possess. |
| Encryption at flight | This is data that is moving through the network but can still be accessed. |
| Encryption in use | This is data that is used on a daily basis. It is usually stored on a database that is accessed thru apps or programs. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | This is a term for a framework for intelligently controlling access to computer resources. Confirms Identity using a username and password. |
| Authorization | This term refers to the enforcing the policies. This gives permission to access certain parts of the site such as read and write access. |
| Accounting | This term refers to auditing usage and providing the information necessary to bill for services. |

**\***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 | 04/11/2021 | Security Template Revised | Danielle O’Bier |  |
| [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] | [Insert text.] |

# Appendix A Lookups

## Approved C/C++ Language Acronyms

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