**Green Pace Developer: Security Policy Guide**



Green Pace Secure Development Policy

Contents

[Overview 2](#_Toc52464053)

[Purpose 2](#_Toc52464054)

[Scope 2](#_Toc52464055)

[Module Three Milestone 2](#_Toc52464056)

[Ten Core Security Principles 2](#_Toc52464057)

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

[Coding Standard 1 4](#_Toc52464059)

[Coding Standard 2 5](#_Toc52464060)

[Coding Standard 3 6](#_Toc52464061)

[Coding Standard 4 7](#_Toc52464062)

[Coding Standard 5 8](#_Toc52464063)

[Coding Standard 6 9](#_Toc52464064)

[Coding Standard 7 10](#_Toc52464065)

[Coding Standard 8 11](#_Toc52464066)

[Coding Standard 9 13](#_Toc52464067)

[Coding Standard 10 14](#_Toc52464068)

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

[Project One 15](#_Toc52464070)

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

[2. Risk Assessment 15](#_Toc52464072)

[3. Automated Detection 15](#_Toc52464073)

[4. Automation 15](#_Toc52464074)

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

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

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

[Audit Controls and Management 18](#_Toc52464078)

[Enforcement 18](#_Toc52464079)

[Exceptions Process 18](#_Toc52464080)

[Distribution 19](#_Toc52464081)

[Policy Change Control 19](#_Toc52464082)

[Policy Version History 19](#_Toc52464083)

[Appendix A Lookups 19](#_Toc52464084)

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

# Overview

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

# Purpose

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

# Scope

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

# Module Three Milestone

## Ten Core Security Principles

| **Principles** | Write a short paragraph explaining each of the 10 principles of security. |
| --- | --- |
| 1. ValidateInput Data | Being one of the main sources of vulnerability, one should always validate input including external sources such as the command line, network interfaces, user-controlled files, etc. |
| 1. Heed Compiler Warnings | Utilize warning capabilities of the compiler and modify code to eliminate potential security flaws. |
| 1. Architect and Design for Security Policies | Design software to implement and enforce security policies such as established user privileges. |
| 1. Keep It Simple | Keep project design as small and simple as possible due to the increased risk for security flaws as the project grows in complexity as well as the additional work required to mitigate vulnerabilities. |
| 1. Default Deny | Standard permission for the system should be to deny access and to solely execute functionality based on the assigned levels of privileges. |
| 1. Adhere to the Principle of Least Privilege | Each process should execute with the minimal number of privileges required. If higher privileges are needed, the permission should only be accessed for the minimal amount of time required to complete the privileged task. This reduces windows of opportunity for potential hackers. |
| 1. Sanitize Data Sent to Other Systems | Sanitize data passed to subsystems such as the command shell or databases. Attackers can exploit functionality of a subsystem by issuing the command elsewhere (ex: a SQL command as a username). Since the calling process understands the context, it should sanitize the data prior to being sent to the subsystem. |
| 1. Practice Defense in Depth | Utilize multiple defensive strategies to strengthen the security with different layers of defense. Where one security may be bypassed, another one may catch or mitigate the vulnerability. |
| 1. Use Effective Quality Assurance Techniques | Implement tests, techniques, and procedures to ensure quality. Examples can include code reviews, fuzz testing, penetration testing, etc. |
| 1. Adopt a Secure Coding Standard | Develop/Apply a secure coding standard for the language and platform used. |

## C/C++ Ten Coding Standards

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

### Coding Standard 1

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Type** | [STD-001-CPP] | When creating variables, keep in mind the min/max range of the data type and code around these restrictions. |

| **Noncompliant Code** |
| --- |
| Choosing char for a data type limited by the bounds set could cause issues for a variable that can/will be changed later. |
| char password[5];  password = “pass”;  password = “newpass”; // BufferOverflow |

| **Compliant Code** |
| --- |
| Utilizing strings for a variable that can frequently change would be a better option (validation should still be implemented). |
| std::string password = “pass”;  password = “newpass”; |

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

|  |
| --- |
| **Principles(s):** This standard relates to encryption in use because it involves the protection of data being used. A buffer overflow can lead to this data being leaked or unauthorized functionality/scripting being used. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **overflow\_upon\_dereference** |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.1 | **C++3802** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.1 | **CERT\_CPP-CTR53-a** **CERT\_CPP-CTR53-b** | Do not use an iterator range that isn't really a range Do not compare iterators from different containers |
| [PRQA QA-C++](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=88046345) | 4.4 | **3802** |  |

### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | [STD-002-CPP] | Since certain data values can only be precise with certain data types, one should handle data conversions accordingly to maintain precise values. |

| **Noncompliant Code** |
| --- |
| Can result in loss of precision and cause unexpected results. |
| void func(void) {  signed long int s\_a = LONG\_MAX;  signed char sc = (signed char)s\_a;  } |

| **Compliant Code** |
| --- |
| Considers loss of data value precision to enable the ability to handle the situation accordingly. |
| void func(void) {  signed long int s\_a = LONG\_MAX;  signed char sc;  if ((s\_a < SCHAR\_MIN) || (s\_a > SCHAR\_MAX)) {  /\* Handle error \*/  } else {  sc = (signed char)s\_a;  }  } |

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

|  |
| --- |
| **Principles(s):** This standard relates to encryption in use because it involves the protection of data being used. Imprecise data values when handling transactional values such as the purchase/sale of items involving credit/debit cards could lead to unexpected results and could be potentially abused/exploited. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Parasoft Jtest](https://wiki.sei.cmu.edu/confluence/display/java/Parasoft) | 2021.1 | **CERT.NUM13.AIC** | Avoid implicit casts from integer data types to floating point data types |

### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | [STD-003-CPP] | Ensure strings have sufficient space for character data and null terminator |

| **Noncompliant Code** |
| --- |
| When handling user input, it is simple to cause a buffer overflow (std::string would be better option). |
| char user\_input[20];  std::cin >> user\_input; |

| **Compliant Code** |
| --- |
| While std::astring would be better overall, if a char array is mandatory, a check should be implemented to ensure appropriate space. |
| char user\_input[20];  std::string bufferString;  while(bufferString.length() <= 0 || bufferString.length() > 20) {  std::cout << "Enter a value: ";  std::cin >> bufferString;  }  strcpy(user\_input, bufferString.data()); |

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

|  |
| --- |
| **Principles(s):** This standard relates to encryption in use because it involves the protection of data being used. Appropriate spacing not being controlled can lead to a buffer overflow which would open a window of vulnerability where data or unintended functionality/scripting can be exploited. |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | Unlikely | High | High | 1 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 6.0p0 | **MISC.MEM.NTERM**  **LANG.MEM.BO LANG.MEM.TO** | No space for null terminator  Buffer overrun Type overrun |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.1 | **C++2835, C++2836, C++2839, C++5216** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2021.1 | [**NNTS.MIGHT**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**NNTS.TAINTED**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **489 S, 66 X, 70 X, 71 X** | Partially implemented |

### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | [STD-004-CPP] | Ensure proper safety procedures to bolster security when engaging with SQL statements and user input. |

| **Noncompliant Code** |
| --- |
| String concatenation allows hackers to include additional functionality that is not intended. |
| std::string sql = "SELECT \* FROM users WHERE email = '" + email + "'"; |

| **Compliant Code** |
| --- |
| Prepared statements restricts functionality regardless of user input (input should still be validated/sanitized). |
| sql::connection \*con;  sql::preparedStatement \*prep\_stmt  prep\_stmt = con->prepareStatement("INSERT INTO test(id, label) VALUES (?, ?)"); |

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

|  |
| --- |
| **Principles(s):** This standard relates to encryption in flight because it involves the communication between two tools being used. Having prepared statements helps secure against sql injection which could be used to gain access to unauthorized data. |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Very High | Very Likely | High | High | 1 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [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** | Protect Against SQL Injection |
| [Findbugs](https://wiki.sei.cmu.edu/confluence/display/java/Findbugs) | 1.0 | **SQL\_NONCONSTANT\_STRING\_PASSED\_TO\_EXECUTE** | Protect Against SQL Injection |
| [Fortify](https://wiki.sei.cmu.edu/confluence/display/java/Fortify) | 1.0 | **HTTP\_Response\_Splitting** **SQL\_Injection\_\_Persistence** **SQL\_Injection** | Protect Against SQL Injection |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/java/Klocwork) |  | **SV.DATA.BOUND** **SV.DATA.DB** **SV.HTTP\_SPLIT** **SV.PATH** **SV.PATH.INJ** **SV.SQL** | Protect Against SQL Injection |

### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [STD-005-CPP] | Consider the memory size for different data types and determine if additional code or change in type is required to handle memory concerns. |

| **Noncompliant Code** |
| --- |
| Susceptible to buffer overflow that could go unnoticed for awhile |
| T add\_numbers(T const& start, T const& increment, unsigned long int const& steps)  {  T result = start;  for (unsigned long int i = 0; i < steps; ++i)  {  result += increment;  }  return result;  } |

| **Compliant Code** |
| --- |
| Checks for buffer overflow and reacts accordingly. |
| T add\_numbers(T const& start, T const& increment, unsigned long int const& steps)  {  T result = start;  T temp = start;  for (unsigned long int i = 0; i < steps; ++i)  {  temp = result;  result += increment;  if (result <= temp || isinf(result)) {  return 0;  };  }  return result;  } |

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

|  |
| --- |
| **Principles(s):** This standard relates to encryption in use because it involves the protection of data being used. A buffer overflow can lead to this data being leaked or unauthorized functionality/scripting being used. |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Medium | Likely | Low | Medium | 4 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Rose) |  |  |  |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Coverity) | 7.5 | **CHECKED\_RETURN** | Finds inconsistencies in how function call return values are handled |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.1 | **C++3225, C++3226, C++3227, C++3228, C++3229, C++4632** |  |

### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | [STD-006-CPP] | Utilize assertions to test assumptions, clarify code to people who may not understand functionality, or to refamiliarize oneself in the future instead of stating the obvious. |

| **Noncompliant Code** |
| --- |
| Does not test assumptions for potential errors and is only states the obvious. |
| Int i = 0;  ASSERT (i == 0); |

| **Compliant Code** |
| --- |
| Tests against assumptions to detect potential errors and allows one to quickly understand how the functionality should play out. |
| int i = 0;  int j = 9;    for( ; (i < 10) ; (++i, --j) ) {  array1[i] = array2[j];  }    ASSERT( j == -1 ); |

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

|  |
| --- |
| **Principles(s):** This standard relates to encryption in use because it involves the protection of data being used. By creating assertion one can increase security for any data being handled by handling edge cases and checking for potential exploits with the program. |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Likely | Low | Low | 5 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-ERR58** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | **Cert-err58-cpp** |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.1 | **C++4634, C++4636, C++4637, C++4639** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.1 | **CERT\_CPP-ERR58-a** |  |

### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [STD-007-CPP] | Exceptions should be used for exceptional situations and not as a means to simply keep a program running. |

| **Noncompliant Code** |
| --- |
| Serves little purpose and doesn’t handle exception well. If program doesn’t crash, program may incur unexpected consequences as a result. |
| try  {  basicFunc();  }  catch (…)  {  cout << “error” << endl;  } |

| **Compliant Code** |
| --- |
| Deliberate usage of exception handling. Utilizes a likely fail case and reacts accordingly. |
| try  {  file.open ("file.txt");  }  catch (std::ifstream::failure e)  {  /\* handle exception accordingly \*/  } |

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

|  |
| --- |
| **Principles(s):** This standard relates to encryption in use because it involves the protection of data being used. Handling exceptions accordingly protects data being used and performs functionality to act as a failsafe. |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Unlikely | Low | Low | 5 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **main-function-catch-all early-catch-all** | Partially checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-ERR51** |  |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.1 | **C++4035, C++4036, C++4037** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **527 S** | Partially implemented |

### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| File Handling | [STD-008-CPP] | When handling files, ensure proper handling procedures to reduce risk. |

| **Noncompliant Code** |
| --- |
| Does not call upon destructors. Attackers can exhaust system resources leading to further vulnerabilities. |
| std::fstream file(fileName);  if (!file.is\_open()) {  // Handle error  return;  }  // ...  std::terminate(); |

| **Compliant Code** |
| --- |
| Ensures system resources are properly closed. |
| 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):** This standard relates to encryption in flight because it involves the communication between two tools being used. File handling done incorrectly can lead to vulnerabilities where system resources can be exhausted and further exploits can be made. |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | Likely | High | High | 1 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 6.0p0 | **ALLOC.LEAK** | Leak |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.1 | **C++4786, C++4787, C++4788** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2021.1 | [**RH.LEAK**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.1 | **CERT\_CPP-FIO51-a** | Ensure resources are freed |

### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Function Return | [STD-009-CPP] | Ensure a value is returned from all potential paths of a function to avoid integrity vulnerabilities and exploitations. |

| **Noncompliant Code** |
| --- |
| Does not return anything if value fails condition check. |
| int absolute\_value(int a) {  if (a < 0) {  return -a;  }  } |

| **Compliant Code** |
| --- |
| Each code path returns a value. |
| int absolute\_value(int a) {  if (a < 0) {  return -a;  }  return a;  } |

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

|  |
| --- |
| **Principles(s):** This standard relates to encryption in use because it involves the protection of data being used. By handling all potential paths from a function the application is protected against certain vulnerabilities and exploits. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **return-implicit** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-MSC52** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | **-Wreturn-type** | Does not catch all instances of this rule, such as function-try-blocks |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 6.0p0 | **LANG.STRUCT.MRS** | Missing return statement |

### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Containers | [STD-010-CPP] | When handling containers, ensure that the indices and/or iterators are of a valid range. |

| **Noncompliant Code** |
| --- |
| Data from untrusted sources can pass the ‘Int pos’ parameter as a negative value resulting in an invalid range. |
| void insert\_in\_table(int \*table, std::size\_t tableSize, int pos, int value) {  if (pos >= tableSize) {  // Handle error  return;  }  table[pos] = value;  } |

| **Compliant Code** |
| --- |
| Since pos is passed as size\_t, it cannot be passed in as a negative number. |
| void insert\_in\_table(int \*table, std::size\_t tableSize, std::size\_t pos, int value) {  if (pos >= tableSize) {  // Handle error  return;  }  table[pos] = value;  } |

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

|  |
| --- |
| **Principles(s):** This standard relates to encryption in use because it involves the protection of data being used. A buffer overflow can lead to this data being leaked or unauthorized functionality/scripting being used. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 20.10 | **overflow\_upon\_dereference** |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 6.0p0 | **LANG.MEM.BO LANG.MEM.BU LANG.MEM.TO LANG.MEM.TU LANG.MEM.TBA LANG.STRUCT.PBB LANG.STRUCT.PPE** | Buffer overrun Buffer underrun Type overrun Type underrun Tainted buffer access Pointer before beginning of object Pointer past end of object |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2021.1 | **C++2891, C++3139, C++3140** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2021.1 | [**ABV.ANY\_SIZE\_ARRAY**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**ABV.GENERAL**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**ABV.STACK**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**ABV.TAINTED**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**SV.TAINTED.ALLOC\_SIZE**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**SV.TAINTED.CALL.INDEX\_ACCESS**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**SV.TAINTED.CALL.LOOP\_BOUND**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [**SV.TAINTED.INDEX\_ACCESS**](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |

## 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.

In order to automate enforcements of the standards of this policy, it would be beneficial to start with security in mind. How code is implemented and unit tests are made start with this policy in mind. By doing so, time and effort would not get wasted into adding security features that could even change the design of the application after much of the work has been done. No code should be committed to the main branch of the application until formally reviewed following this policy as a guideline. In addition, it would be important to review policy standards before any work reaches production as well as before pre-production begins to uphold security standards and catch any mistakes that may have slipped through.

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

## Create Policies for Encryption and Triple A

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

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

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

| 1. **Encryption** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Encryption in rest | The protection of data on a disk including disk, files, and database encryption. Applies when data is stored and protects data if it gets exposed/stolen. |
| Encryption at flight | The protection of data in transit. Applies when data is moved from one location to another and helps secure the channel from outside influence with tools such as VPNs, HTTPS, SSL, etc. |
| Encryption in use | The protection of data being used. Applies to data such as credit card information, passwords, and personal information. Secures data used throughout its lifecycle. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | Used to identify and confirm who/what is accessing the application in order to restrict access or grant privileges as needed. |
| Authorization | The method of granting specific users privileges/restrictions based on their level of authorized access to ensure functionality/data is only used by specific individuals as needed. |
| Accounting | The ability to measure a user’s amount of time accessing the application, when/where it is accessed, how much/often data is accessed, etc. Usage information is applied in conjunction with authentication to hold users accountable by knowing who has done what. |

**\***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 | 05/22/2021 | Coding Standards | Heriberto Torres |  |
| 1.2 | 06/10/2021 | Project One | Heriberto Torres |  |

# Appendix A Lookups

## Approved C/C++ Language Acronyms

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