**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 | Input validation is the process of ensuring inputs are suitable and in compliance with a predefined standard to protect the database from malformed data. This process eliminates a variety of software vulnerabilities since input is validated from all untrusted data sources. Validating input data should occur as soon as data is received from an external party. Syntactical validation should be used to ensure correct syntax of structured fields and semantic validation should be used to verify values and are correctly specified. |
| 1. Heed Compiler Warnings | Code should be compiled using the highest warning level available for the given compiler. This eliminates warnings by modifying the code. Static and dynamic analysis tools should be used to detect and eliminate additional security flaws. |
| 1. Architect and Design for Security Policies | The architecture and design should implement and enforce security policies. This protects the availability, confidentiality and integrity of the data being used within the system. |
| 1. Keep It Simple | The design should be kept as small and simple as possible. Complex security mechanisms increase the effort required to achieve an optimal level of assurance. |
| 1. Default Deny | By default, access should be denied, and the protection scheme should identify conditions under which access is permitted. Access decisions should be based on permission rather than exclusion. |
| 1. Adhere to the Principle of Least Privilege | Every process should execute using the least set of privileges necessary to successfully and securely run the program. Elevated permissions should only be accessible for the least amount of time required to complete its specified task. Adhering to this principle reduces the number of chances an attacker has to execute arbitrary code with elevated privileges. |
| 1. Sanitize Data Sent to Other Systems | All data passed to complex subsystems should first be sanitized. This includes command shells, commercial off-the-shelf (COTS) components and relational databases. The calling process is responsible for sanitizing the data since it understands its context. This process protects data from injection attacks from attackers trying to invoke unused functionality in these components. |
| 1. Practice Defense in Depth | Valuable data and information should be secured with overlapping and redundant layers of defense mechanisms. Each defense mechanism addresses a different attack vector but consequently, gives each layer its own vulnerabilities and holes. Using multiple different defenses together closes the gaps created by a singular defense mechanism in turn, creating a secured computer or system. |
| 1. Use Effective Quality Assurance Techniques | An effective quality assurance program should be implemented to effectively identify and eliminate vulnerabilities. Techniques such as independent security reviews, fuzz testing, penetration testing, and source code audits can be used to keep systems more secure. |
| 1. Adopt a Secure Coding Standard | A secure coding standard should be developed and applied for the target development language and platform. |

### C/C++ Ten Coding Standards

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

#### Coding Standard 1

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Type** | [STD-001-  -C] | FLP30-C. Do not use floating-point variables as loop counters |

| **Noncompliant Code** |
| --- |
| a floating-point variable of 0.1 is used as a loop counter |
| void func(void) {  for (float x = 0.1f; x <= 1.0f; x += 0.1f) {  /\* Loop may iterate 9 or 10 times \*/  }  } |

| **Compliant Code** |
| --- |
| The loop counter is an integer from which the floating-point value is derived |
| #include <stddef.h>    void func(void) {  for (size\_t count = 1; count <= 10; ++count) {  float x = count / 10.0f;  /\* Loop iterates exactly 10 times \*/  }  } |

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

| **Principles(s):** **Keep it simple –** do not use floating point numbers to represent fractions or as loop counters |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 23.04 | **for-loop-float** | Fully checked |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/c/Clang) | 3.9 | cert-flp30-c | Checked by clang-tidy |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **9009** | Fully supported |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | 39 S | Fully implemented |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | [STD-002-  -C] | EXP44-C. Do not rely on side effects in operands to sizeof, \_Alignof, or \_Generic |

| **Noncompliant Code** |
| --- |
| a++ is not evaluated |
| #include <stdio.h>    void func(void) {  int a = 14;  int b = sizeof(a++);  printf("%d, %d\n", a, b);  } |

| **Compliant Code** |
| --- |
| a is incremented outside of the sizeof operation |
| #include <stdio.h>    void func(void) {  int a = 14;  int b = sizeof(a);  ++a;  printf("%d, %d\n", a, b);  } |

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

| **Principles(s):** **Adopt a secure coding standard -** do not pass an operand that would otherwise yield a side effect since the side effect will not be generated |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.3p0 | LANG.STRUCT.SE.SIZEOf | Side effects in sizeof |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **MISRA C 2004 Rule 12.3** | Partially implemented |
| [ECLAIR](https://wiki.sei.cmu.edu/confluence/display/c/ECLAIR) | 1.2 | CC2.EXP06 | Fully implemented |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/c/RuleChecker) | 23.04 | alignof-side-effect generic-selection-side-effect sizeof | Fully checked |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | [STD-003-CPP] | Do not attempt to create a std::string from a null pointer |

| **Noncompliant Code** |
| --- |
| std::getenv() after a std::string object is created. Std::getev() returns a null pointer on failure leading to undefined behavior or other types of errors. |
| #include <cstdlib>  #include <string>    void f() {  std::string tmp(std::getenv("TMP"));  if (!tmp.empty()) {  // ...  }  } |

| **Compliant Code** |
| --- |
| Getenv() is called before the std::string object is constructed to check for null. |
| #include <cstdlib>  #include <string>    void f() {  const char \*tmpPtrVal = std::getenv("TMP");  std::string tmp(tmpPtrVal ? tmpPtrVal : "");  if (!tmp.empty()) {  // ...  }  } |

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

| **Principles(s):** **Keep it simple –** by using design patterns to handle implementation of various string types |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.3p0 | LANG.MEM.NPD | Null Pointer Dereference |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2022.2 | **CERT\_CPP-STR51-a** | Avoid null pointer dereferencing |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2023.1 | **DF4770, DF4771, DF4772, DF4773, DF4774** | Risk Prioritization |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2023a | [CERT C++: STR51-CPP](https://www.mathworks.com/help/bugfinder/ref/certcstr51cpp.html) | Checks for string operations on null pointer (rule partially covered). |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | [STD-004-  -C] | FIO30-C. Exclude user input from format strings |

| **Noncompliant Code** |
| --- |
| incorrect\_password() is called during identification and authentication to display an error message if the specified user is not found or the password is incorrect |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>    void incorrect\_password(const char \*user) {  int ret;  /\* User names are restricted to 256 or fewer characters \*/  static const char msg\_format[] = "%s cannot be authenticated.\n";  size\_t len = strlen(user) + sizeof(msg\_format);  char \*msg = (char \*)malloc(len);  if (msg == NULL) {  /\* Handle error \*/  }  ret = snprintf(msg, len, msg\_format, user);  if (ret < 0) {  /\* Handle error \*/  } else if (ret >= len) {  /\* Handle truncated output \*/  }  fprintf(stderr, msg);  free(msg);  } |

| **Compliant Code** |
| --- |
| Untrusted user input is passed through fprintf() instead of part of the format string |
| #include <stdio.h>    **void** incorrect\_password(**const** **char** \*user) {  **static** **const** **char** msg\_format[] = "%s cannot be authenticated.\n";  **fprintf**(stderr, msg\_format, user);  } |

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

| **Principles(s):** **Validate input –** ensures proper input validation so attackers cannot control the contents of a format string |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-FIO30** | Partially implemented |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **TAINTED\_STRING** | Implemented |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | 592 | Partially supported: reports non-literal format strings |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2023a | [CERT C: Rule FIO30-C](https://www.mathworks.com/help/bugfinder/ref/certcrulefio30c.html) | Checks for tainted string format (rule partially covered) |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | [STD-005-  -C] | MEM34-C. Only free memory allocated dynamically |

| **Noncompliant Code** |
| --- |
| realloc(), buf does not refer to dynamically allocated memory: |
| #include <stdlib.h>    enum { BUFSIZE = 256 };    void f(void) {  char buf[BUFSIZE];  char \*p = (char \*)realloc(buf, 2 \* BUFSIZE);  if (p == NULL) {  /\* Handle error \*/  }  } |

| **Compliant Code** |
| --- |
| buf refers to dynamically allocated memory |
| #include <stdlib.h>    enum { BUFSIZE = 256 };    void f(void) {  char \*buf = (char \*)malloc(BUFSIZE \* sizeof(char));  char \*p = (char \*)realloc(buf, 2 \* BUFSIZE);  if (p == NULL) {  /\* Handle error \*/  }  } |

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

| **Principles(s):** **Architect and design for security policies –** ensures that freeing the memory is included in the software architecture and design |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 23.04 | **invalid-free** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-MEM34** | Can detect memory deallocations for stack objects |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.3p0 | ALLOC.TM | Type Mismatch |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **424, 673** | Fully supported |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | [STD-006-  -C] | Use a static assertion to test the value of a constant expression |

| **Noncompliant Code** |
| --- |
| assert() macro is used to assert a property concerning a memory-mapped structure |
| #include <assert.h>    struct timer {  unsigned char MODE;  unsigned int DATA;  unsigned int COUNT;  };    int func(void) {  assert(sizeof(struct timer) == sizeof(unsigned char) + sizeof(unsigned int) + sizeof(unsigned int));  } |

| **Compliant Code** |
| --- |
| This portable compliant solution uses static\_assert |
| #include <assert.h>    struct timer {  unsigned char MODE;  unsigned int DATA;  unsigned int COUNT;  };    static\_assert(sizeof(struct timer) == sizeof(unsigned char) + sizeof(unsigned int) + sizeof(unsigned int),  "Structure must not have any padding"); |

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

| **Principles(s):** **Use effective quality assurance techniques –** to test values |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/c/Clang) | 3.9 | misc-static-assert | Checked by clang-tidy |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.3p0 | **(customization)** | Users can implement a custom check that reports uses of the assert() macro |
| [ECLAIR](https://wiki.sei.cmu.edu/confluence/display/c/ECLAIR) | 1.2 | **CC2.DCL03** | Fully implemented |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **44 S** | Fully implemented |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | [STD-007-CPP] | Handle all exceptions thrown before main() begins executing |

| **Noncompliant Code** |
| --- |
| The constructor for S may throw an exception that is not caught when globalS is constructed during program startup |
| struct S {  S() noexcept(false);  };    static S globalS; |

| **Compliant Code** |
| --- |
| uses a static const char \* instead of defining a std::string at global namespace scope |
| static const char \*global = "...";    int main() {  // ...  } |

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

| **Principles(s):** **Adopt a secure coding standard –** to catch exceptions |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 7.3p0 | **LANG.STRUCT.EXCP.THROW** | Use of throw |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | **potentially-throwing-static-initialization** | Partially checked |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | cert-err58-cpp | Checked by clang-tidy |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/cplusplus/RuleChecker) | 22.10 | **potentially-throwing-static-initialization** | Partially checked |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Concurrency | [STD-008-  -C] | Ensure that compound operations on shared variables are atomic |

| **Noncompliant Code** |
| --- |
| Declares a shared \_Bool flag variable and provides a toggle\_flag() method that negates the current value of flag |
| #include <stdbool.h>    static bool flag = false;    void toggle\_flag(void) {  flag = !flag;  }    bool get\_flag(void) {  return flag;  } |

| **Compliant Code** |
| --- |
| Atomic variables and a compare-and-exchange operation to guarantee that the correct value is stored in flag |
| #include <stdatomic.h>  #include <stdbool.h>    static atomic\_bool flag;    void init\_flag(void) {  atomic\_init(&flag, false);  }  void toggle\_flag(void) {  bool old\_flag = atomic\_load(&flag);  bool new\_flag;  do {  new\_flag = !old\_flag;  } while (!atomic\_compare\_exchange\_weak(&flag, &old\_flag, new\_flag));  }    bool get\_flag(void) {  return atomic\_load(&flag);} |

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

| **Principles(s):** **Adopt a secure coding standard –** to ensure compound operations are performed atomically to prevent data races and race conditions. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.3p0 | **CONCURRENCY.DATARACE** | Data Race |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/java/Coverity) | 7.5 | **GUARDED\_BY\_VIOLATION** **INDIRECT\_GUARDED\_BY\_VIOLATION** **NON\_STATIC\_GUARDING\_STATIC** **NON\_STATIC\_GUARDING\_STATIC** **SERVLET\_ATOMICITY** | Implemented |
| [ThreadSafe](https://wiki.sei.cmu.edu/confluence/display/java/ThreadSafe) | 1.3 | **CCE\_SL\_INCONSISTENT** **CCE\_CC\_CALLBACK\_ACCESS** **CCE\_SL\_MIXED** **CCE\_SL\_INCONSISTENT\_COL** **CCE\_SL\_MIXED\_COL** **CCE\_CC\_UNSAFE\_CONTENT** | Implemented |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Error Handling | [STD-009-  -C] | Use ferror() rather than errno to check for FILE stream errors |

| **Noncompliant Code** |
| --- |
| These implementations perform some operation that fails (with ENOTTY) if stdout is not a terminal. |
| errno = 0;  printf("This\n");  printf("is\n");  printf("a\n");  printf("test.\n");  if (errno != 0) {  fprintf(stderr, "printf failed: %s\n", strerror(errno));  } |

| **Compliant Code** |
| --- |
| ferror() is used to detect an error and if an early call to printf() fails, calls can modify errno so the program cannot rely on being able to detect the root cause of the original failure if it waits until after a sequence of library calls to check. |
| printf("This\n");  printf("is\n");  printf("a\n");  printf("test.\n");  if (ferror(stdout)) {  fprintf(stderr, "printf failed\n");  } |

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

| **Principles(s):** **Use effective quality assurance techniques -** to test the error indicator for a specified stream and return nonzero if the error indicator is set for the stream. |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [ECLAIR](https://wiki.sei.cmu.edu/confluence/display/c/ECLAIR) | 1.2 | **CC2.ERR01** | Fully implemented |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | 44 S | Enhanced Enforcement |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2022.2 | **CERT\_C-ERR01-a** | The error indicator 'errno' shall not be used |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| Input/Output  I/O | [STD-010-  -C] | Take care when using the rename() function |

| **Noncompliant Code** |
| --- |
| Any existing destination file is removed by rename() |
| const char \*src\_file = /\* ... \*/;  const char \*dest\_file = /\* ... \*/;  if (rename(src\_file, dest\_file) != 0) {  /\* Handle error \*/  } |

| **Compliant Code** |
| --- |
| If the destination file does not exist, the source file is renamed. The access() function is used to check for the existence of a file. |
| const char \*src\_file = /\* ... \*/;  const char \*dest\_file = /\* ... \*/;    if (access(dest\_file, F\_OK) != 0) {  if (rename(src\_file, dest\_file) != 0) {  /\* Handle error condition \*/  }  }  else {  /\* Handle file-exists condition \*/  } |

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

| **Principles(s):** **Keep it simple -** to either preserve or remove existing destination files |
| --- |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 7.3p0 | **(customization)** | Users can add a custom check for all uses of rename(). |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2023.1 | **C5015** | n/a |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **592 S** | Fully Implemented |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/c/PVS-Studio) | 7.25 | [**V576**](https://pvs-studio.com/en/docs/warnings/v576/) | n/a |

### 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.

Modifying Green Pace’s existing DevOps process to integrate security throughout each step will enforce DevSecOps. When assessing and planning, a security architecture should be created to specify where to apply security controls to reduce cyber breaches. During the design and build phases it is important to use secure coding practices such as input validation and session management. For more secure testing, application security testing should be performed such as white and black box testing and continuous testing to assist in catching security vulnerabilities before they make it into production. Throughout the production process, continuous monitoring should be used to detect security issues in real time. This falls hand in hand with continuous improvement as feedback can come from team members, external partners and end users in the real word environment. Enforcing automation into Green Pace’s current DevOps process will not only improve security, but also its ability to produce and deliver software more quicly and efficiently.

### 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-  -C] | Low | Probable | Low | **P6** | **L2** |
| [STD-002-  -C] | Low | Unlikely | Low | **P3** | **L3** |
| [STD-003-CPP] | High | Likely | Medium | **P18** | **L1** |
| [STD-004-  -C] | High | Likely | Medium | **P18** | **L1** |
| [STD-005-  -C] | High | Likely | Medium | **P18** | **L1** |
| [STD-006-  -C] | Low | Unlikely | High | **P1** | **L3** |
| [STD-007-CPP] | Low | Likely | Low | **P9** | **L2** |
| [STD-008-  -C] | Medium | Probable | Medium | **P8** | **L2** |
| [STD-009-  -C] | Low | Probable | Low | **P6** | **L2** |
| [STD-010-  -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 | Encryption at rest refers to encrypted data that is not actively traveling. This can mean traveling between devices or networks and this information can be saved on hard drives, databases, computers etc. By applying this policy, sensitive information remains confidential on the stored device by encrypting the data that is only readable with its corresponding encryption key. |
| Encryption at flight | Encryption at flight, or in transit, refers to encrypting information traveling between devices or networks. Creating strong foundations and implementing automated policies can make data at flight more secure. Encrypting data in transit should be used to defend data against unauthorized users. |
| Encryption in use | Encryption at rest refers to encrypting data that is actively being accessed by users. In this stage, data is most vulnerable due to the fact that it is immediately available. Authentication and implementing data controls before usage are best practices for encrypting data in use. Encrypting data in use enhances security between client apps and servers. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | Authentication refers to ensuring a system is being accessed by the registered user. This process checks that user’s credentials match those stored within a database. Methods of authentication include passwords, fingerprints or access cards. Authentication prevents unauthorized users from accessing confidential or sensitive information on systems, networks or databases. |
| Authorization | Authorization is a process that determines a user's level of access within a network. This gives users permission to only access resources and services needed to accomplish their job. Some users may only be allowed to read certain information while other users, such as admins, can use CRUD (create, read, update, delete) to alter information within a database. This policy limits access to important or sensitive information in turn, making content more secure. |
| Accounting | Accounting monitors and measures the resources a user is accessing. This uses features from both authentication and authorization to monitor the amount of data a user has sent/received, and the amount of time spent on the system. Accounting adds a final layer of security by logging user activity which can flag whether the person has been authorized and authenticated correctly. |

**\***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 | 05/21/2023 | 10 security principles and coding standards updated | Maya Ellison |  |
| 1.2 | 06/9/2023 | Template completed | Maya Ellison |  |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

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

**References**

Jatheon. (2022, May 18). *Encryption: Data at rest, data in motion and data in use*. Jatheon Technologies Inc. <https://jatheon.com/blog/data-at-rest-data-in-motion-data-in-use/>

Raza, M. (2023, March 3). *The 7 devsecops concepts & principles to ace for true devsecops*. Splunk. <https://www.splunk.com/en_us/blog/learn/devsecops-concepts-principles.html>

Seacord, R. (2018, May 2). *Top 10 secure coding practices*. Top 10 Secure Coding Practices - CERT Secure Coding - Confluence. https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices