**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 | By validating input data, software vulnerabilities can be prevented and eliminated. This can also stop buffer overflows within the software. If overflow occurs, the program should be prepared to deal with the problem or to be stopped in its tracks by displaying error messages. Input sources must always be identified even though all input data may not be harmful or entered purposely. Command line arguments, network interfaces, environmental variables, and user-controlled files are input sources that may pose to be harmful. |
| 1. Heed Compiler Warnings | Compiler warnings detect errors in the code that should be addressed before preceding to run a program. This can be eliminated by correcting the errors in the code or by writing errorless(which is almost impossible) code. Compiler flags can also be used at runtime to check for overflows, uninitialized variables, and stack pointer corruption. |
| 1. Architect and Design for Security Policies | Architect and design for security policies include how to manage access to the system, account creations, tools to use to prevent threats and detect them regularly(in real time), roles as to who will be responsible for day-to-day operations and control of end users, and auditing purposes to ensure the system is protected. Basically, this structure and design ensures security throughout the system and determines who will be in charge based on each role. |
| 1. Keep It Simple | Designs can become complex fast. It is easier to maintain code this way. It also eliminates the room for errors. The more complex the design, the longer it takes to debug the code and the longer it will take to find errors. Keeping it simples makes the code more manageable and easier to read especially with in line comments. |
| 1. Default Deny | Default deny is a security principle that basically states that by default, all access if denied unless under certain conditions then access will be granted. This is a base access decision. |
| 1. Adhere to the Principle of Least Privilege | The least privilege principle only executes with the least number of privileges to completely run the program. This privilege gives the attacker no higher privileges when trying to run the injected code and can prevent the attacker from obtaining access to important data/files and systems. |
| 1. Sanitize Data Sent to Other Systems | This principle prevents SQL injection from occurring. Information is being sent from one system to another (subsystem) such as command shells and databases. For attackers to not gain access, it is important to do validation checks for input and excluding “or” and equals(because it will always return true, and the attacker will be able to gain access to sensitive information. |
| 1. Practice Defense in Depth | By practicing defense in depth, this principle allows for multiple security measures to be in place to protect the system from hackers trying to gain access to it, vulnerabilities, and system failures. If one security measure fails, another will attempt to stop the attack and so on. Security elements include firewalls and anti-virus software, |
| 1. Use Effective Quality Assurance Techniques | Quality assurance techniques include code reviews, testing, input validation, and audits to prevent and discover errors sooner rather than later. It’s better to test now than later and as often as possible to prevent hours of backtracking and finding/fixing vulnerabilities early. |
| 1. Adopt a Secure Coding Standard | This principle allows one to adopt a secure coding standard by using one that is already in place using the specific programming language or creating a new one based on the programming language being used to develop the software such as Java, C++, and/or Python. |

## 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** | INT32-C | Ensure that operations on signed integers do not result in overflow  This is overflow that is undefined. Integer operations will overflow if the resulting value cannot be represented by the underlying representation of the integer. |

| **Noncompliant Code** |
| --- |
| This example can result in a signed integer overflow while adding signed operands a and b. |
| void func(signed int a, signed int b) {  signed int sum = a + b;  } |

| **Compliant Code** |
| --- |
| This solution ensures that overflow will not happen due to the addition operators. |
| #include <limits.h>    void f(signed int a, signed int b) {  signed int sum;  if (((b > 0) && (a > (INT\_MAX - b))) ||  ((b < 0) && (a < (INT\_MIN - b)))) {  /\* Handle error \*/  } else {  sum = a + b;  }  } |

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

|  |
| --- |
| **Principles(s):** Principle 1 is to validate user input. This principle can alert the user if errors have occurred due to overflow. Principle 2 is heed compiler warnings. This can be used to inform the user that there is an error. Compiler flags can be thrown to detect buffer overflows. Principle 9(Use Effective Quality Assurance Techniques) can be used to test input validation before implementing the program to avoid error messages and/or handle errors due to overflow. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Astrée | 20.10 | integer-overflow | Fully checked |
| CodeSonar | 6.0p0 | **ALLOC.SIZE.ADDOFLOW** **ALLOC.SIZE.IOFLOW** **ALLOC.SIZE.MULOFLOW** **ALLOC.SIZE.SUBUFLOW** **MISC.MEM.SIZE.ADDOFLOW** **MISC.MEM.SIZE.BAD** **MISC.MEM.SIZE.MULOFLOW** **MISC.MEM.SIZE.SUBUFLOW** | Addition overflow of allocation size Integer overflow of allocation size Multiplication overflow of allocation size Subtraction underflow of allocation size Addition overflow of size Unreasonable size argument Multiplication overflow of size Subtraction underflow of size |
| Coverity | 2017.07 | **TAINTED\_SCALAR**  **BAD\_SHIFT** | Implemented |
| [LDRA](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) tool suite | 9.7.1 | **493 S, 494 S** | Partially implemented |

### Coding Standard 2

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Data Value** | STR50-CPP | Guarantee that storage for strings has sufficient space for character data and the null terminator  Buffer overflow occurs when there is not enough space to hold the data that is being copied to the buffer. To eliminate buffer overflow, make sure there is equivalent space to store the data and use truncation. |

| **Noncompliant Code** |
| --- |
| Input data is boundless. |
| #include <iostream>    void s() {  char buf[16];  std::cin >> buf;  } |

| **Compliant Code** |
| --- |
| This solution does not use an unbounded array and uses std::string to eliminate buffer overflows. |
| #include <iostream>  #include <string>    void s() {  std::string input;  std::string one, two;  std::cin >> one >> two;  } |

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

|  |
| --- |
| **Principles(s):** Principle 1 (Validate user input) can validate the data by eliminating buffer overflows before they occur by creating unbound less arrays. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Code](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar)Sonar | 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 3

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **String Correctness** | STR03-C | Do not inadvertently truncate a string  Functions used to truncate strings that exceed the buffer limts. |

| **Noncompliant Code** |
| --- |
| The remaining characters will be truncated because there are no null characters and it will only copy a specific amount of characters. |
| char \*data;  char s[16];  strncpy(s, data, sizeof(s)); |

| **Compliant Code** |
| --- |
| This example has the correct amount of space and the buffer is large enough to hold the string that is being copied over. |
| char \*data = NULL;  char s[16];    if (data == NULL) {  /\* Handle null pointer error \*/  }  else if (strlen(data) >= sizeof(s)) {  /\* Handle overlong string error \*/  }  else {  strcpy(s, data);  } |

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

|  |
| --- |
| **Principles(s):** Principle 1(Validate user input) can be applied to standard to ensure the buffer is large enough to accommodate the string by defining the data and allowing for errors to be handled. Principle 9(Use Effective Quality Assurance Techniques) ensures that all data will be tested for errors early in the development lifecycle. |

**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) | 6.0p0 | **MISC.MEM.NTERM** | No Space For Null Terminator |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  | Could detect violations in the following manner: all calls to strncpy() and the other functions should be followed by an assignment of a terminating character to null-terminate the string |
| [GCC](http://gcc.gnu.org/) | 8.1 | [-Wstringop-truncation](https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wstringop-truncation) | Detects string truncation by strncat and strncpy. |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2021.1 | [NNTS.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NNTS.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |

### Coding Standard 4

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **SQL Injection** | IDS00-J | Prevent SQL injection  Vulnerabilities occur from untrusted sources and can result in altering query statements that can retrieved personal information and other sensitive data. |

| **Noncompliant Code** |
| --- |
| The data is not sanitized using input arguments allowing an attacker to enter “1=1.” |
| 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 statement = connection.createStatement();  ResultSet results = stmt.executeQuery(sqlString);    if (!results.next()) {  throw new SecurityException(  "User name or password incorrect"  );  }    // Authenticated; proceed  } finally {  try {  connection.close();  } catch (SQLException x) {  // Forward to handler  }  }  }  } |

| **Compliant Code** |
| --- |
| The data is sanitized using prepared statements. |
| public void doPrivilegedAction(  String username, char[] password  ) throws SQLException {  Connection connection = getConnection();  if (connection == null) {  // Handle error  }  try {  String hashpwd = hashPassword(password);    // Validate username length  if (user\_name.length() > 6) {  // Handle error  }    String sqlString =  "select \* from db\_user where user\_name=? and password=?";  PreparedStatement stmt = connection.prepareStatement(sqlString);  stmt.setString(1, username);  stmt.setString(2, hashpwd);  ResultSet rs = stmt.executeQuery();  if (!result.next()) {  throw new SecurityException("Username 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):** Principle 5(Default Deny) is mapped to this standard to deny access to prevent the sql injection from happening. Principle 6(Adhere to the principle of least privilege**)** can deny the attacker high level privileges when running the injected code **.** Principle 7(Sanitize Data Sent to Other Systems) can be used to sanitize data being sent/received by using prepared statements. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [The Checker Framework](https://wiki.sei.cmu.edu/confluence/display/java/The+Checker+Framework) | 2.1.3 | **Tainting Checker** | Trust and security errors (see Chapter 8) |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/java/Coverity) | 7.5 | **SQLI** **FB.SQL\_PREPARED\_STATEMENT\_GENERATED\_** **FB.SQL\_NONCONSTANT\_STRING\_PASSED\_TO\_EXECUTE** | Implemented |
| [Findbugs](https://wiki.sei.cmu.edu/confluence/display/java/Findbugs) | 1.0 | **SQL\_NONCONSTANT\_STRING\_PASSED\_TO\_EXECUTE** | Implemented |
| [Fortify](https://wiki.sei.cmu.edu/confluence/display/java/Fortify) | 1.0 | **HTTP\_Response\_Splitting** **SQL\_Injection\_\_Persistence** **SQL\_Injection** | Implemented |

### Coding Standard 5

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Memory Protection** | MEM52-CPP | Detect and handle memory allocation errors |

| **Noncompliant Code** |
| --- |
| Memory allocation is not checked and does not throw expections. |
| #include <cstring>    void s(const int \*array, std::size\_s size) noexcept {  int \*scopy = new int[size];  std::memcpy(scopy, array, size \* sizeof(\*copy));  // ...  delete [] scopy;  } |

| **Compliant Code** |
| --- |
| The no throw command tests and returns allocated space(pointer) or a null pointer |
| #include <cstring>  #include <new>    void s(const int \*array, std::size\_s size) noexcept {  int \*scopy = new (std::nothrow) int[size];  if (!scopy) {  // Handle error  return;  }  std::memcpy(scopy, array, size \* sizeof(\*scopy));  // ...  delete [] scopy;  } |

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

|  |
| --- |
| **Principles(s):** Principle 9(Use Effective Quality Assurance Techniques) tests the no throw commands to verify if there is enough space or a null pointer. |

**Threat Level**

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

**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** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2021.1 | [NPD.CHECK.CALL.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.CHECK.CALL.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.CHECK.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [NPD.CHECK.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [NPD.CONST.CALL](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.CONST.DEREF](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.FUNC.CALL.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.FUNC.CALL.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.FUNC.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.FUNC.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [NPD.GEN.CALL.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.GEN.CALL.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.GEN.MIGHT](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [NPD.GEN.MUST](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/)  [RNPD.CALL](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) [RNPD.DEREF](https://support.roguewave.com/documentation/klocwork/en/current/certcandcsecurecodingstandardidsmappedtoklocworkcandccheckers/) |  |

### Coding Standard 6

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Assertions** | DCL03-C | Use a static assertion to test the value of a constant expression  This standard finds and eliminates defects and bugs that will cause vulnerabilities in the software. |

| **Noncompliant Code** |
| --- |
| It needs to be placed in the structure and can only be ran at runtime. |
| #include <assert.h>    struct timer {  unsigned char MODE;  unsigned int DATA;  unsigned int COUNT;  };    int time(void) {  assert(sizeof(struct timer) == sizeof(unsigned char) + sizeof(unsigned int) + sizeof(unsigned int));  } |

| **Compliant Code** |
| --- |
| This code can be run at compile time and will no display a runtime error or crash. |
| #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 insulation"); |

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

|  |
| --- |
| **Principles(s):** Principle 2(Heed Compiler Warnings) can be used to compile with the highest level warning during compiling**.**  Principle 9(Use Effective Quality Assurance Techniques) maps to this standard because it tests and prevents errors and vulnerabilities early. |

**Threat Level**

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

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 6.9.0 | **CertC-DCL03** |  |
| [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) | 6.0p0 | **(customization)** | Users can implement a custom check that reports uses of the assert() macro |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  | Could detect violations of this rule merely by looking for calls to assert(), and if it can evaluate the assertion (due to all values being known at compile time), then the code should use static-assert instead; this assumes ROSE can recognize macro invocation |

### Coding Standard 7

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **Exceptions** | ERR51-CPP | Handle all exceptions  This allows all exceptions to be handled properly by transferring control to the nearest handler to prevent the program from closing abnormally. |

| **Noncompliant Code** |
| --- |
| No matching handler is found and terminate is called automatically as default. |
| void throwing\_except() noexcept(false);    void f() {  throwing\_except ();  }    int main() {  s();  } |

| **Compliant Code** |
| --- |
| This code handles all exceptions and manages external resources. |
| void throwing\_except() noexcept(false);    void f() {  throwing\_except();  }    int main() {  try {  s();  } catch (...) {  // Handle error  }  } |

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

|  |
| --- |
| **Principles(s):** Principle 2(Heed Compiler Warnings) catches and applies highest level of compiler warnings. Principle 3(Architect and Design for Security Policies) to manage external resources. |

**Threat Level**

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

**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) | 6.9.0 | **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** |
| --- | --- | --- |
| Input Output | FIO51-CPP | Close files when they are no longer needed  A call to open and close must match in order for the program to terminate or the last pointer of the return value has ended. |

| **Noncompliant Code** |
| --- |
| Object is not properly closed because the constructor called to open and no destructor is called. |
| #include <exception>  #include <tstream>  #include <string>    void t(const std::string &fileName) {  std::tstream file(fileName);  if (!file.is\_open()) {  // Handle error  return;  }  // ...  std::terminate();  } |

| **Compliant Code** |
| --- |
| The resource is closed properly. |
| #include <exception>  #include <tstream>  #include <string>    void t(const std::string &fileName) {  {  std::tstream file(fileName);  if (!tfile.is\_open()) {  // Handle error  return;  }  } // tfile is closed properly because it is destroyed  std::terminate();  } |

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

|  |
| --- |
| **Principles(s):** Principle 3(Architect and Design for Security Policies) to manage resources. Principle 10 (Adopt a Secure Coding Standard) can adapt to a specific type of problem such as closing resources properly by using a policy already in place or by creating a new one to handle closing of resources. |

**Threat Level**

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

**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** |
| --- | --- | --- |
| Memory Management | MEM50-CPP. | Do not access freed memory  Dereferencing the pointer, operation, type casting, and memory that has been deallocated by a memory management functions are all examples of undefined behavior that will result in dangling pointers that can expose vulnerabilities. |

| **Noncompliant Code** |
| --- |
| This code can be run using arbitrary code because p is deallocated. |
| #include <new>    struct P {  void t();  };    void g() noexcept(false) {  P \*p = new P;  // ...  delete p;  // ...  p->t();  } |

| **Compliant Code** |
| --- |
| The memory can be used until it is deallocated. |
| #include <new>    struct P {  void t();  };    void h() noexcept(false) {  P \*p = new P;  // ...  p->h();  delete p;  } |

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

|  |
| --- |
| **Principles(s):** Principle 6(Adhere to the Principle of Least Privilege) to prevent arbitrary code from being run by allocating the memory. |

**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=222953724) | 20.10 | **dangling\_pointer\_use** |  |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 6.9.0 | **CertC++-MEM50** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | clang-analyzer-cplusplus.NewDelete clang-analyzer-alpha.security.ArrayBoundV2 | Checked by clang-tidy, but does not catch all violations of this rule. |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 6.0p0 | **ALLOC.UAF** | Use after free |

### Coding Standard 10

| **Coding Standard** | **Label** | **Name of Standard** |
| --- | --- | --- |
| **EXCEPTIONS** | ERR57-CPP | Do not leak resources when handling exceptions  This standard eliminates the need to write complex cleanup code. This ensures that objects will not be left in uninitialized states by reclaiming resources. |

| **Noncompliant Code** |
| --- |
| This code is not properly released which causes a resource leak. |
| #include <new>    struct SomeType {  SomeType() noexcept; // Performs nontrivial initialization.  ~SomeType(); // Performs nontrivial finalization.  void process\_item() noexcept(false);  };    void f() {  SomeType \*slm = new (std::nothrow) SomeType();  if (!slm) {  // Handle error  return;  }    try {  slm->process\_item();  } catch (...) {  // Process error, but do not recover from it; rethrow.  throw;  }  delete slm;  } |

| **Compliant Code** |
| --- |
| Exception handler calls delete which makes the slm free. |
| #include <new>    struct SomeType {  SomeType() noexcept; // Performs nontrivial initialization.  ~SomeType(); // Performs nontrivial finalization.    void process\_item() noexcept(false);  };    void f() {  SomeType \*slm = new (std::nothrow) SomeType();  if (!slm) {  // Handle error  return;  }  try {  slm->process\_item();  } catch (...) {  // Process error, but do not recover from it; rethrow.  delete slm;  throw;  }  delete slm;  } |

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

|  |
| --- |
| **Principles(s):** Principle 4(Keep It Simple) applies to this standard by writing smaller condensed code with in-line comments. Principle 8(Practice Defense in Depth) to prevent leaked resources. |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Probable | High | **P2** | **L3** |

**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++4756, C++4757, C++4758** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **50 D** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2021.1 | **CERT\_CPP-ERR57-a** | Ensure resources are freed |

## Defense-in-Depth Illustration

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



# Project One

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

## Revise the C/C++ Standards

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

## Risk Assessment

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

## Automated Detection

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

## Automation

Provide a written explanation using the image provided.



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

Automation enforcement of the standards should begin in the Design sector and Verify and Test sector of the DevSecOps. This will ensure that all written code can be designed to be kept simple, easy to navigate, and easy to understand. Verify and Test of automation will ensure vulnerabilities and errors are caught early in the development cycle and will aid in preventing sql injections by denying access to attackers. Monitor and detect sector will comply with the default deny and defense in depth principles to prevent attackers from accessing the system in any way possible. Automation tools will make it easier to identify and prevent attacks, errors, and vulnerabilities.

## 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 |
| --- | --- | --- | --- | --- | --- |
| INT32-C | High | Likely | High | **P9** | **L2** |
| STR50-CPP | High | Likely | Medium | **P18** | **L1** |
| STR03-C | Medium | Probable | Medium | **P8** | **L2** |
| IDS00-J | High | Probable | Medium | P12 | L1 |
| MEM52-CPP | High | Likely | Medium | **P18** | **L1** |
| DCL03-C | Low | Unlikely | High | **P1** | **L3** |
| ERR51-CPP | Low | Probable | Medium | **P4** | **L3** |
| FIO51-CPP | Medium | Unlikely | Medium | **P4** | **L3** |
| MEM50-CPP | High | Likely | Medium | **P18** | **L1** |
| ERR57-CPP | Low | Probable | High | **P2** | **L3** |

## 

## 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 rests protects the unencrypted data by encryption(other form of data) that cannot be read unless there is an encryption key to decode it. This data is usually on a disk, and it prevents hackers/unauthorized users from accessing it |
| Encryption at flight | Encryption at flight is data that is encrypted while it is being sent or received. This prevents the data from being read while in transmission. |
| Encryption in use | Encryption in use is used in all stages of encryption rather it is in rest or in flight or in use. Regardless, if the data is at rest, in flight or in use, it is protected at every stage whether it is from the source or the destination such as storage locations and memory. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | Authentication is the very first step in the Triple A Framework. This is used by applications or systems to verify/identify users and making sure the user is who the user says they are. Users have usernames and passwords to verify identity. Other forms of identification are single sign-on (SSO) systems, biometrics, digital certificates, and public key infrastructure. This prevents unauthorized users from accessing an account or specific data in the systems. |
| Authorization | Authorization is a set of rules or policies that define who can access specific information. It is also used to enforce restrictions to deny access to specific users. Other types of authorization include route assignments, IP address filtering, bandwidth traffic management, and encryption. |
| Accounting | Accounting basically is logs that are created to monitor the user’s activity in the program or sessions. It records the time, activities, login credentials, and sent/received data. |

**\***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 | Milestone One/Standards and Principals | Shakira Medlock | Instructor |
| 1.2 | 06/07/2021 | Project One/Risks and Automation | Shakira Medlock |  |

# Appendix A Lookups

## Approved C/C++ Language Acronyms

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