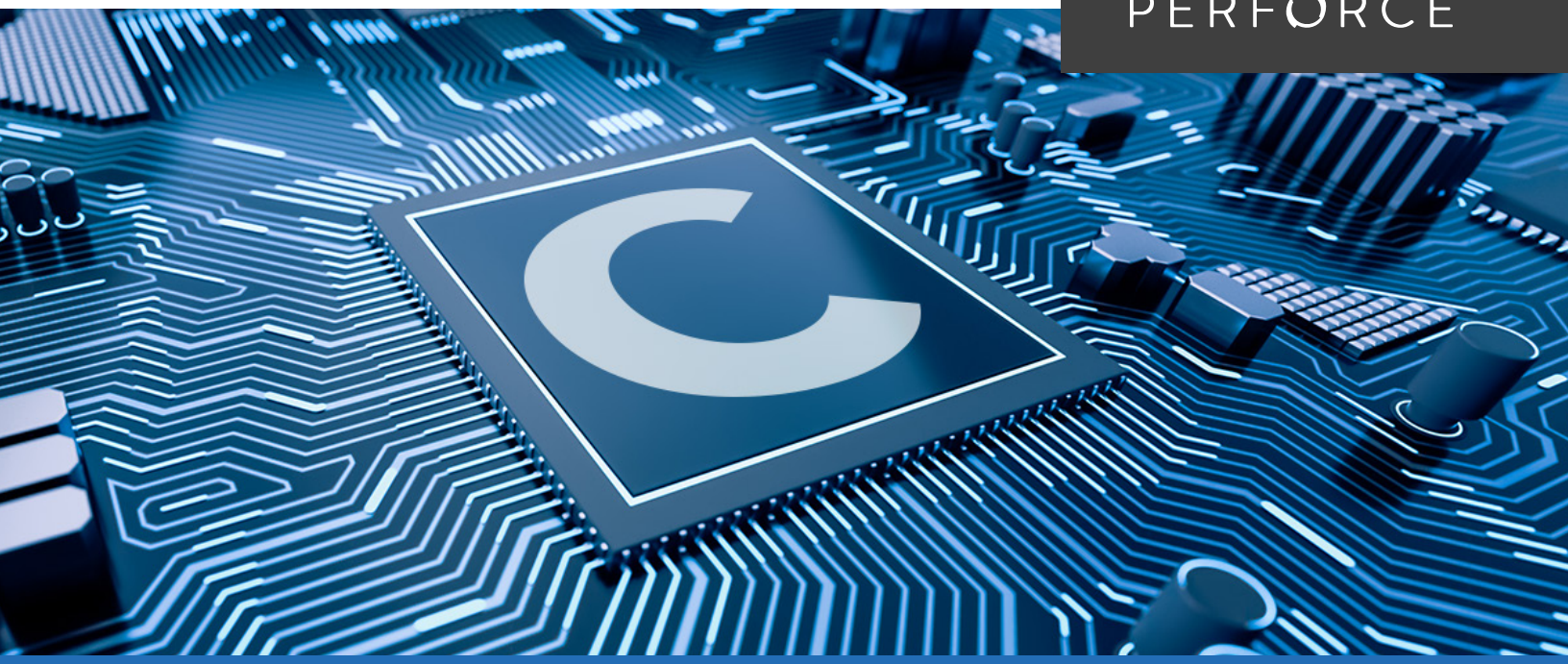
PERFORCE



WHITE PAPER

How to Write Secure Code in C

Introduction

Software security is a top concern today. You can't risk any security vulnerabilities – particularly if you're developing software for embedded systems. And that means your code needs to be secure and free of coding errors.

When you think about software security, you probably think about passwords and access control. Or viruses, spoofing, and phishing attacks.

These are common security concerns. And security features, such data encryption and authentication protocols, mitigate these vulnerabilities.

But even if you've implemented these security features, software can remain vulnerable.

To ensure secure software, you need to start at the source — the code level. Otherwise, coding errors will compromise your program.

Coding Errors Compromise Security

The Software Engineering Institute (SEI) estimates that up to 90% of reported security incidents result from exploiting vulnerabilities in software code or design. And these vulnerabilities allow hackers to access private data or take unauthorized control of a system.

So, a simple coding error can lead to a hacking threat. A hacker could take control of your computer, your home automation device, your home entertainment device, or your car. Worse still, a hacker could even take control of a nuclear power plant.

EXAMPLE OF A SECURITY VULNERABILITY: BUFFER OVERFLOW IN C

To illustrate how this might happen, let's look at just one example. Buffer overflow is a common security vulnerability in C programming.

What Is Buffer Overflow?

Buffer overflow occurs when data is written outside the boundary of the allocated memory.

For example:

```
char buff[10];
buff[10] = 'a';
```

Here, an array of 10 bytes (index 0 to 9) is declared. But the program then attempts to write a character one byte beyond the array's boundary. If the memory neighboring the array is used later in the program, then it will lead to unexpected behavior.

This is bad enough. And it can get worse. A buffer overflow can allow a hacker to take control of a system.

How Buffer Overflow Invites Hackers

Hackers can use buffer overflow errors to cause a program to crash, corrupt the data, or simply steal information.

When a program runs, it uses an area of memory referred to as the 'stack'. Variables within the scope of the currently executing function will be stored on the stack. The address of the function call will also be stored to allow return statements to return to the correct location.

When the function returns to the calling function, the program execution continues from where it left off. So, if the return address on the stack is modified to point to some alternative malicious instructions, then those instructions will be executed when the function returns.

If the program is receiving data — and there is no check in place to ensure that the input buffer cannot overflow — then it will be possible to design an input, or 'payload', that contains mali-

cious code. This overflows the input buffer and overwrites the return address on the stack with the address of the malicious code.

PREVENTING SECURITY VULNERABILITIES IS CRITICAL

Preventing security vulnerabilities — such as buffer overflow — is critical. And this can be done by making sure the code itself is written without exploitable gaps.

After all, putting stronger locks on your front door is no use if the windows are left open. So, to improve security, you'll need to ensure secure code.

4 Ways to Ensure Secure Code in C

Writing secure code is important. And when it comes to C programming, there are four key sources of information to help you ensure secure code.

1. CWE

You can identify security weaknesses from the Common Weakness Enumeration (CWE).

What Is CWE?

CWE is a community-developed list of common software security weaknesses in C. It's maintained by the MITRE Corporation. This list can be used as a baseline for weakness identification, mitigation, and prevention.

CWE's List of Software Security Weaknesses

The CWE list prioritizes weaknesses. The top 25 entries are prioritized using input from more than two dozen different organizations. They evaluate each weakness based on frequency and importance. Many of the weaknesses (in C programs) listed in CWE relate to buffer overflow.

The top 25 list also adds a small set of the most effective 'Monster Mitigations'. This helps developers reduce or eliminate entire groups of the top 25 weaknesses. It also helps with many of the other 800 weaknesses that are documented in the CWE list.

CWE focuses on stopping vulnerabilities at the source. This is done by educating designers, programmers, and testers on how to eliminate common mistakes — before software is even shipped.

2. CERT C

You can apply the CERT C coding standard to your code.

What Is CERT C?

The CERT C coding standard is published by the CERT Division at the Software Engineering Institute (SEI). SEI is a research and development center operated by Carnegie Mellon University. It's primarily funded by the U.S. Department of Defense and the Department of Homeland Security.

CERT C Security Rules

Secure coding experts continually develop the CERT C guidelines on a wiki.

Each guideline consists of:

- A title
- A description
- An example of non-compliant code
- Examples of compliant solutions

The guidelines cover coding and implementation errors, as well as low-level design errors. The aim

is to eliminate in secure coding practices and undefined behaviors that can lead to vulnerabilities.

CERT C defines a vulnerability as:

A set of conditions that allows an attacker to violate an explicit or implicit security policy.

The defect may be minor. This means it doesn't affect the performance or results produced by the software. But it nevertheless may be exploited by an attack. And that results in a significant security breach.

RECOMMENDED READING

Secure Coding in C and C++

by Robert Seacord

An essential resource for all C developers.

3. ISO/IEC TS 17961:2013 "C SECURE"

You can apply the ISO/IEC TS 17961:2013 "C Secure" coding rules.

What Is ISO/IEC TS 17961:2013?

ISO/IEC TS 17961:2013 establishes a set of coding rules. These rules enable static code analyzers to diagnose insecure code beyond the requirements of the language standard.

C Secure Coding Rules

ISO/IEC TS 179671:2013 includes rules for secure coding in C. It also includes examples for each rule.

The purpose of C Secure is to specify secure coding rules that can be automatically enforced. These can be used to detect security flaws in C programming. To be considered a security flaw, a software bug must be triggerable by the actions of a malicious user or attacker.

Analyzers that implement these rules must be able to effectively discover secure coding errors — without generating excessive false positives.

4. MISRA C

You can also use MISRA to ensure secure coding in C.

What Is MISRA?

MISRA provides best practice guidelines for the development of safety-related systems. Its C coding standards have been widely adopted across many industries.

MISRA C Security Rules

MISRA C:2012 Amendment 1 was published in 2016. It provides additional security guidelines for C programming, including new rules and directives. It also includes examples of compliant and non-compliant code.

These guidelines can be used to prevent coding errors that lead to safety issues and security vulnerabilities.

Why MISRA C Security Rules Are Ideal for Embedded Systems

MISRA C security rules are ideal for embedded systems. That's because MISRA C security is on par with that of other secure coding standards for C. Plus, MISRA C is trusted across embedded systems industries. And it's a go-to coding standard in the automotive industry.

EXAMPLE OF A MISRA C SECURITY RULE

MISRA C security rules can prevent coding errors and security weaknesses, such as buffer overflow.

Here's an example of a MISRA C security rule:

MISRA C Rule 18.1

"A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand."

This rule does the same thing as the following CERT C rule.

ARR30-C

"Do not form or use out-of-bounds pointers or array subscripts."

And both relate to multiple CWE weaknesses in C, one of which is:

CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer

"The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer."

Following either the MISRA C rule or the CERT rule will ensure secure code — and avoid common weaknesses in CWE. This is because writing to an out-of-range pointer (or pointer operand) could result in a buffer overflow — and vulnerable code. Reading from an out-of-range pointer (or pointer operand) could accidentally reveal information to hackers.

So, by ensuring these rules are followed, you'll avoid serious coding errors. You can enforce MISRA and CERT rules by using a static code analyzer, such as Helix QAC.

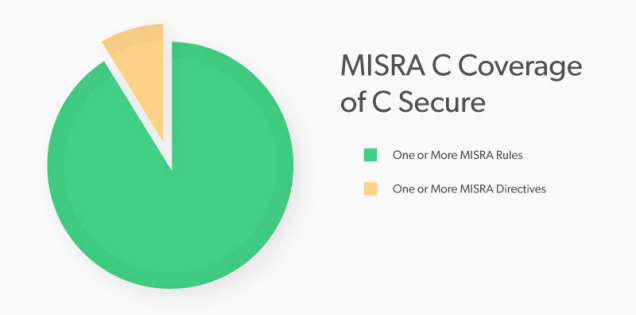
COMPARING MISRA C AND OTHER STANDARDS

This is why the MISRA C coding standard is also ideal for environments where software security has more emphasis than safety.

In fact, MISRA has published two addenda to the MISRA C:2012 standard to help developers map MISRA rules to the C Secure and CERT C standards.

Comparing MISRA C and C Secure

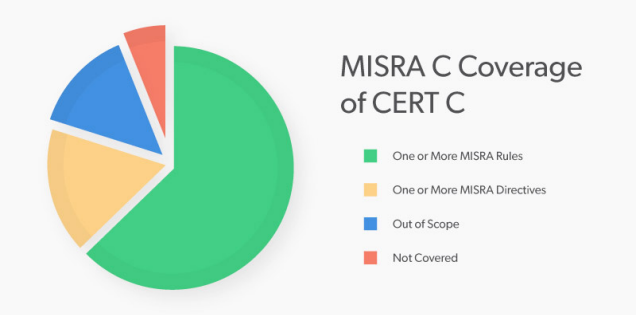
MISRA C:2012 – Addendum 2 shows how each MISRA rule maps to the C Secure rules in ISO/IEC TS 17961:2013.



Every rule in C Secure is covered by a rule or directive in MISRA C. And any static code analyzer (such as Helix QAC) that fully supports MISRA C will also comply with the C Secure standard. So, you can use the standards interchangeably for security.

Comparing MISRA C and CERT C

MISRA C:2012 – Addendum 3 shows how each rule maps to the CERT C rules.



CERT C is designed for C11. MISRA C:2012 was designed for C99.

There are 15 C11-specific rules in CERT C that are out of scope for MISRA C:2012. Of the CERT C rules (within the scope of MISRA C:2012), there are only four that aren't covered. So, MISRA C covers a large share of security rules from CERT C.

Note: Violations of all four of these rules can be detected automatically using Helix QAC.

Write Secure Code With Helix QAC

You can enforce MISRA rules (in C or C++) automatically with Helix QAC. This significantly reduces the amount of time you need to spend performing manual code inspections. So, you'll free up development resources and deliver your program on time — while improving the quality of your software.

See how Helix QAC applies MISRA rules by visiting perforce.com/helix-qac-demo.

About Perforce

Perforce is a leading provider of enterprise scale software solutions to technology developers and development operations ("DevOps") teams requiring productivity, visibility, and scale during all phases of the development lifecycle. Enterprises across the globe rely on its agile planning and ALM tools, developer collaboration, static code analysis, version control and repository management solutions as the foundation for successful DevOps at scale. Perforce is trusted by the world's most innovative brands, including NVIDIA, Pixar, Scania, Ubisoft, and VMware. Perforce has offices in Minneapolis, MN, Alameda, CA, Mason, OH, Boston, MA, the United Kingdom, Finland, Sweden, Germany, India, and Australia, and sales partners around the globe. For more information, please visit www.perforce.com