Applications

Safe-Secure C has numerous business applications that can enhance developer's and user's perceptions of C language programs.

Embedded Systems

Embedded systems have high-reliability requirements where the cost of failure can be catastrophic. Safe-Secure provides an extra layer of protection from liability due to critical software failures.

Compiler Software

Compiler vendors can significantly improve the security of applications generated by their compiler providing a competitive advantage over other compilers and programming languages.

Tool Vendors

Source-analysis tool vendors can add a new dimension of safety and security verification.

Operating Systems

Operating system vendors can improve system security and reliability by identifying potential buffer overflow issues in drivers.

Manufacturing

Large manufacturers can identify issues in embedded software.

Consulting

Consulting firms can provide specialized remediation consulting for eliminating vulnerabilities in client code.

Product Features

Function Interface Checking

Automatically infers the requirements on the interface of each callable function.

Buffer Bounds Checking

Uses top-down interpretation combined with static analysis to identify potential array and pointer buffer bounds violations.

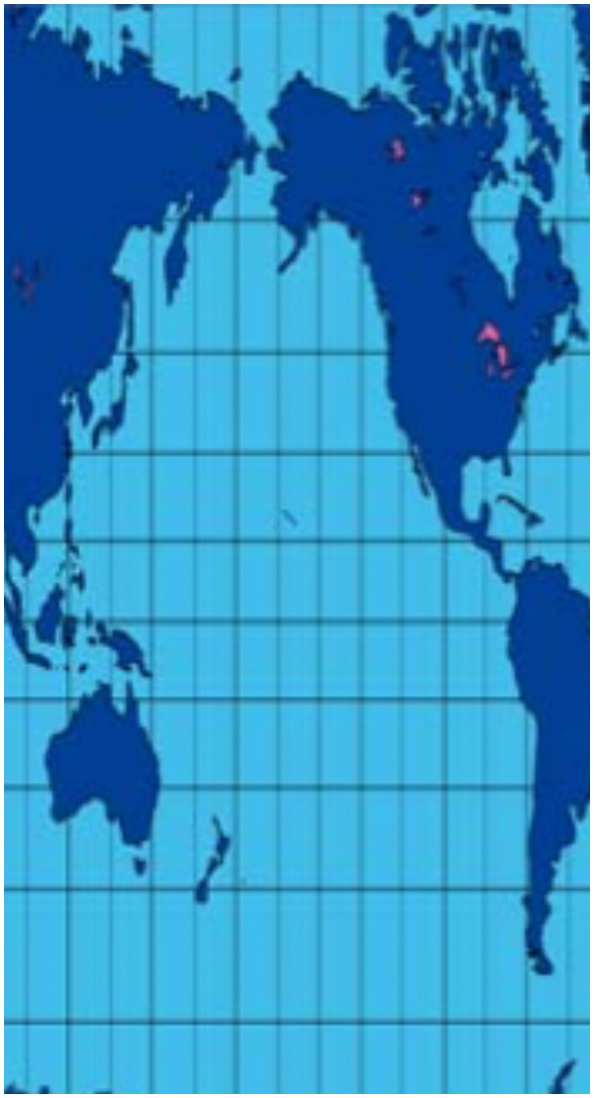
Post Link Checking

Supplements the compilation and linking mechanism by producing and using bounds-data files which record requirements and guarantees for the defined and undefined symbols in one or more corresponding object files.

Plum Hall, Inc.
3 Waihona Box 44610
Kamuela HI 96743
Phone 808-882-1255
Fax 808-882-1556
http://www.plumhall.com/
sscc@plumhall.com

Plum Hall, Inc. Safe-Secure™ C

Safe and Secure Bounds Checking for C Software



Software Security

According to CERT Coordination Center statistics, more than 90% of software security incidents are caused by attackers exploiting known software defect types. The most common form of software vulnerability, particularly in C and C++ programs are buffer overflows.

The preponderance of buffer overflows in C and C+has made these languages the bane of the software security community and has caused the redirection of many software projects to other languages that are perceived to be more secure.

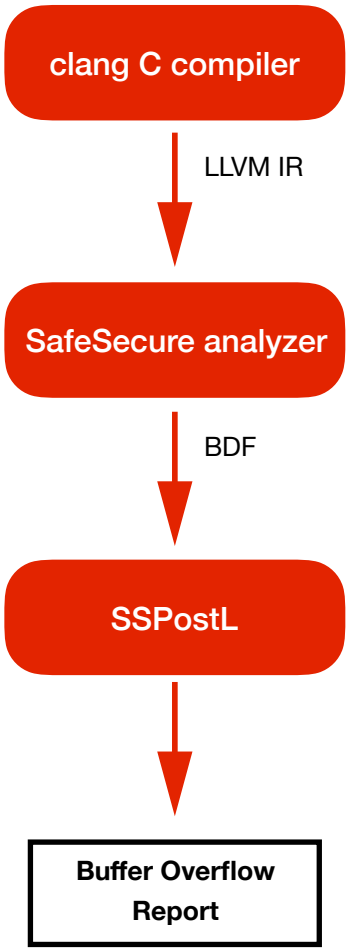
By using Plum Hall sscc™ (Safe-Secure™ C), programs written in C can be just as safe and secure as programs written in Java or C#, while preserving C efficiency. This combination of security and performance preserves the viability of C in a more security-conscious age.

Leveraging the LLVM Compiler Infrastructure

Safe-Secure C leverages the popular LLVM compiler tool chain infrastructure. SafeSecure reads bitcodes produced by the clang compiler available on all Linux systems. The bitcodes are analyzed in a top-down fashion by the call tree to statically analyze the C program. The static analysis is augmented in loops by an interpreter which dynamically computes program variable values. Safe-Secure checks and reports bounds violations and produces Plum Hall patented Bounds Data Table data, which is then combined in a post link step with BDT from other linked C programs to check global bounds violations.

How Safe-Secure Works

Safe-Secure C consists of two major components: bounds analyzer/bounds data generator, and post linker bounds checker. These components can be integrated into compilers and software analysis tools to detect buffer overflows and other common security vulnerabilities in C programs.



Benefits

sscc combines static and dynamic analysis methods to create a hybrid solution for detecting buffer overflows and other common vulnerabilities.

Compile-time analysis identifies possible buffer overflows. Patent-pending static analysis techniques are used in combination with top-down interpretation to ensure security across compiled units.

Safe-Secure offers new techniques which can detect and prevent buffer overflow issues in C software programs thus improving security, reliability and safety while at the same time reducing liability arising from undetected vulnerabilities.