

Static analysis tools for detecting buffer overflow vulnerabilities

Sreedevi Jagavarapu

Introduction

- Introduction
- Splint
- Uno
- Orion
- Conclusion

The Problem

- Programs are buggy
- Manual inspection, “print” statements, debuggers are not always effective and are time consuming
- There is lack of information on which software analyzer tools are most helpful

Static Analysis Tools

- Static Analysis :
- analyze programs without running them
 - Analysis is possible before a program is compliable
 - No test suite is necessary
 - May not find every memory error

Some Static Analysis Tools

C/C++

FlexeLint	Coverity	Flawfinder
Discover	PolySpace	CodeAssure
Klocwork	Prexis	CodeSurfer
PREfast	Fortify	Orion
ITS4	DMS	

Java

Klocwork	Orion
Fortify	FindBugs

C (only)

Splint	Smatch
Blast	Uno
CQual	MOPS

Easy to Use

- Splint: very easy to use
 - Knowledge of annotations is not compulsory
 - Knowledge of annotations allows full functionality
 - Good documentation
 - Easy to run e.g. `host% splint filename.c`

Splint

- Annotations
 - Denoted using C comments identified by an @ character following the /* comment marker
- Running Splint is an iterative process
- Splint checks approximately 1,000 lines per second, so it is fairly easy to run Splint
- The goal is to eliminate all the warnings by modifying code or adjusting annotations

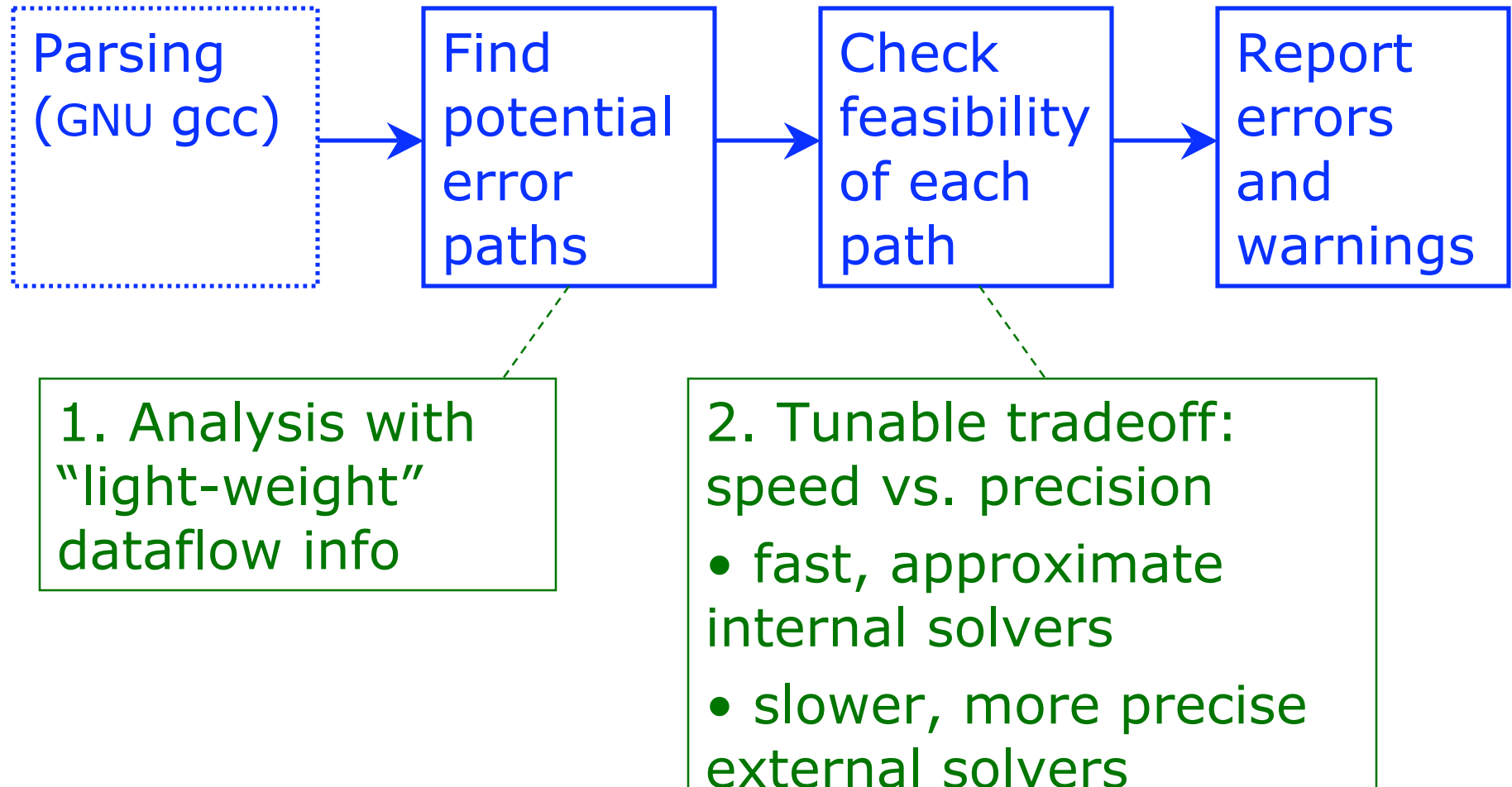
UNO

- It is a simple tool for source code analysis
- It is designed to intercept primarily the three most common types of software defects
 - Use of uninitialized variables
 - Null-pointer references
 - Out-of-bounds array indexing

Orion

- A static analyzer from Bell Labs offering
 - analysis of C, C++
 - tunable to increase speed/precision
 - incremental interprocedural analysis
 - built-in and user-defined checks
 - aim at “semantic” errors (*use-before def* rather than *type mismatch*)
 - concentrate on UNO errors first

Orion's Approach: 2-Phase Analysis



Pros and Cons of evaluated tools

- Splint was a helpful static tool
 - Can be used during development and debugging
 - It produces more warnings leading to confusion
- No guarantee that all messages indicate real bugs or all bugs will be found
- Splint some times misses the significant errors
- Splint hides error report in a list of explanations
- Orion analyzes C and C++ source code

Conclusion

No tool will eliminate all security risks

References

- 1)<http://www.splint.org>
- 2)<http://www.cs.virginia.edu/~evans/pubs/ieeesoftware.pdf>
- 3)<http://lclint.cs.virginia.edu/usenix01.pdf>
- 4)<http://cosi.clarkson.edu/docs/staticanalysistools/UsingStaticAnalysisTools.html#Annotations>
- 5)<http://www.cs.virginia.edu/pipermail/splint-discuss>
- 6)<http://testingfaqs.org/t-static.html>
- 7)http://samate.nist.gov/index.php/Source_Code_Analyzers
- 8)http://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis
- 9)http://www.dse.nl/~thelosen/artikelen/static_analysis.pdf
- 10)http://www.soft32.com/download_206036.html
- 11)<http://lclint.cs.virginia.edu/faq.html#quest1>
- 12)<http://www.die.net/linux/man/man/efence.3.html>