System Call Vulnerabilities in Linux Storage Stack

Nima Mohammadi
Sepand Haghighi

Fall - 2016

Outline

- Introduction
- Static Code Analysis
- Static Analyzer List
- Kernels
- Static Analysis Report
- Conclusion

Source Code Analysis

- As a developer, source code is the malleable object
- Many techniques can be shared between source and binary analysis
- Flip side: Improve RE by looking for blind spots in source analysis

Source and Binary Analysis

- Binary
  - Hard to map back to source
  - CPU dependent
  - Language indep ?
  - Environment independent?
  - 3rd party utility

- Source
  - Easy to identify location of flaw
  - CPU independent
  - Language depen.
  - Environment independent?
  - 1st party only

Tools to find bugs

- Static Code Analyzer
- Dynamic Runtime Checker
- Fuzzer/Test Suits
- Tracers to understand code
- Tools to understand source

Static and Dynamic Analysis

- Static
  - Complete coverage
  - false positives
  - Can analyze anytime, anywhere
  - Precise description of problem, unknown impact

- Dynamic
  - Very rare to get close to 100%
  - No false positives
  - Requires ability to run program
  - Precise understanding of impact, possibly unknown cause

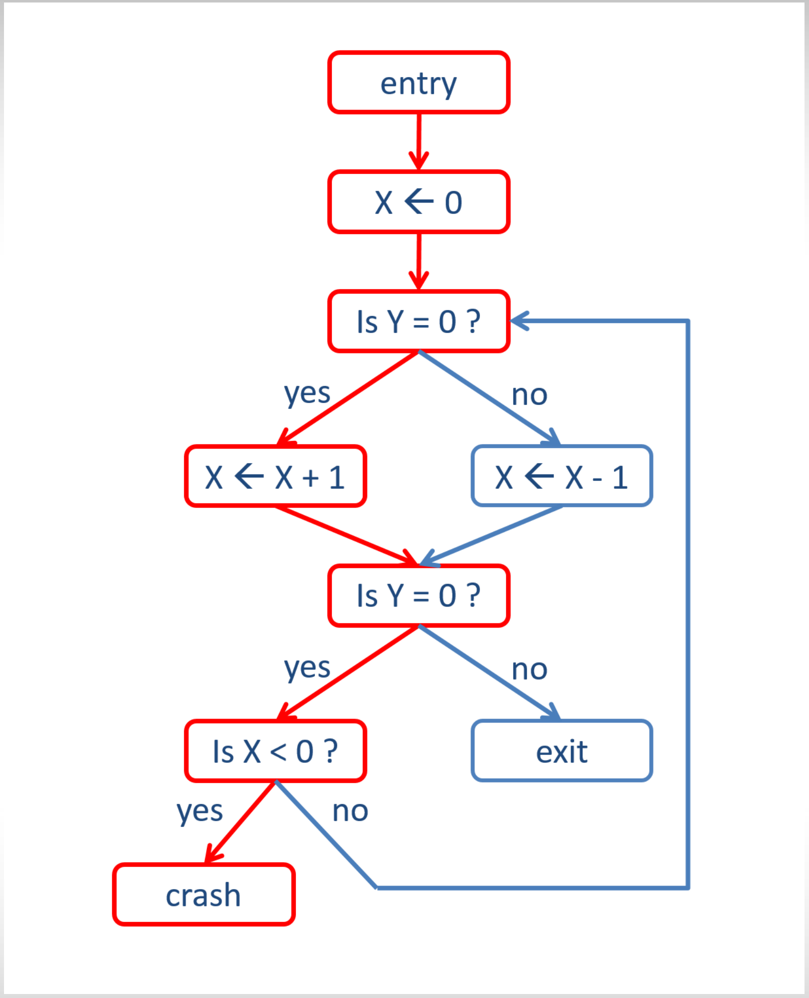
Static Code Analysis

- The source has a lot of knowledge embedded
  - Extract the good parts, ignore the remainder
- Many types of analysis
  - Different ways to approach the problem
  - Different goals
    - Find Defects
    - Enhance Run time analysis
    - Gain insight into code

Static Code Analysis

	Complete	Incomplete
Sound	Reports all errors Reports no false alarms **Undecidable**	Reports all errors May report false alarms **Decidable**
Unsound	May not report all errors Reports no false alarms **Decidable**	May not report all errors May report false alarms **Decidable**

Static Code Analysis



Simulated Execution

- Flow sensitive

```
bar() { free(g_p); }
baz() { free(g_p); }
foo() { if (x) bar(); else baz(); }
```

- Context sensitive

```
if (use_malloc)
    p = malloc();
/* … */
if (use_malloc)
    free(p);
```

- How much state to track?
  - Exponential number of paths
  - Loops
  - Heap is unbounded

Static Analyzer List

- BLAST
- FRAMA-C
- Flaw-Finder
- Sparse
- CppCheck
- Smatch

BLAST

- **B**erkeley **L**azy **A**bstraction **S**oftware Verification **T**ool (**BLAST**)
- Written in Ocaml
- Stable Version : 2.7.3 , 2014
- automatic abstraction refinement to construct an abstract model that is then model-checked for safety properties

Frama-C

- Framework or Modular Analysis of C Programs.
- Written in Ocaml
- Stable Version : Silicon , 2 Dec 2016
- Run Different Analysis :
  - Value Analysis
  - Jessie
  - Impact Analysis
  - Slicing
  - Spare Code

Flaw-Finder

- Examine C/C++ Codes
- Written in Python
- Stable Version : 2014-08-03
- It's very useful for quickly finding and removing at least some potential security problems before a program is widely released to the public
- Flawfinder is specifically designed to be easy to install and use. After installing it, at a command line just type:

```
flawfinder directory_with_source_code
```

Sparse

- Examine Linux Kernels
- Written in C
- Stable Version : 0.5.0 , 29 Jan 2014
- Open Source , MIT License
- Sparse defines the following list of attributes:
  - Address_space
  - Bitwise
  - Force
  - Context

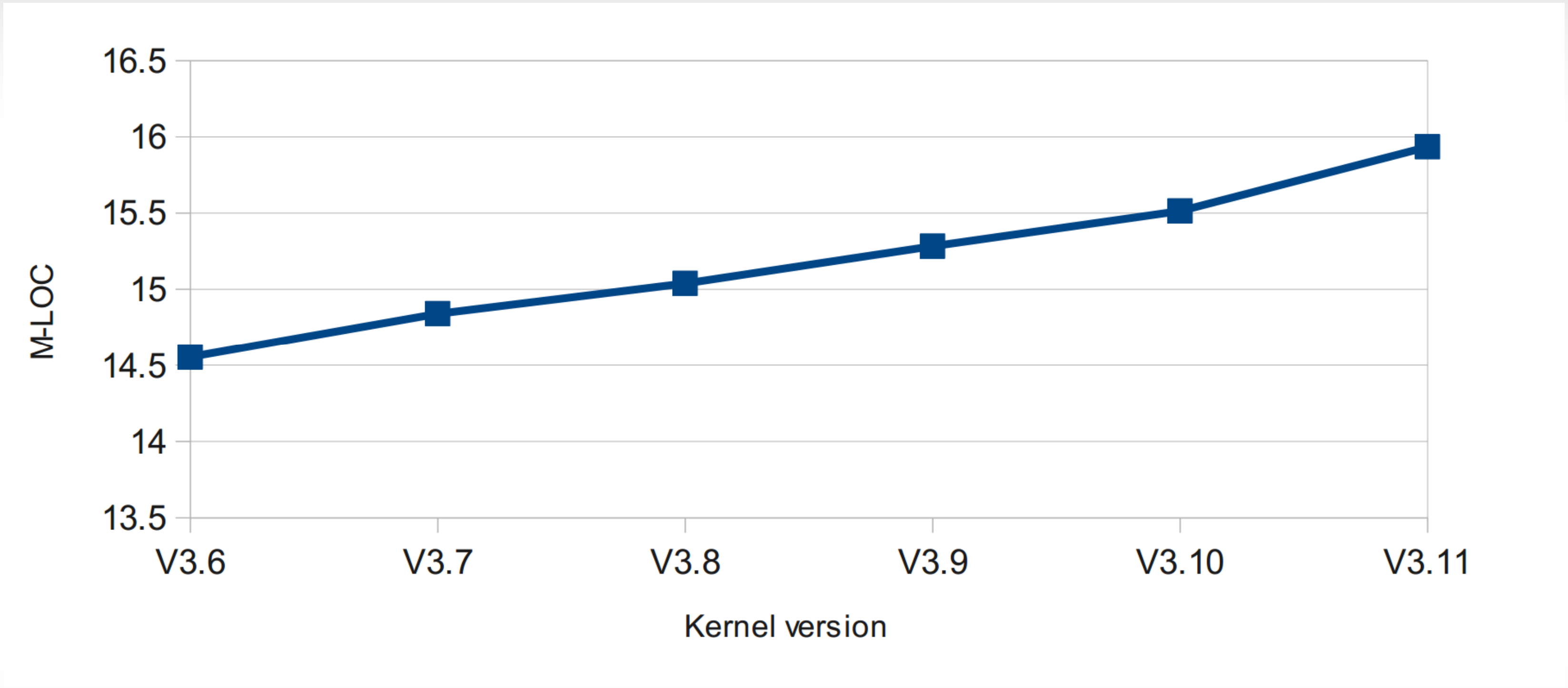
CPP Check

- Static Analysis of C/C++ Codes
- Written in C++
- Stable Version : 8 Oct 2016
- General Public GNU Licesne
- Some of checks that are supported :
  - Automatic Variable
  - Memory Leaks
  - Resources Leaks
  - Bounds Checking
  - Invalid usage of standard library

Smatch

- Static Analysis of C Source Code
- Written in C Stable Version : 8 Oct 2016
- General Public GNU Licesne
- C static analysis tool which developed, and which uses to test the mainline Linux kernel code for security bugs.

Linux Complexity Growing



Filesystems List

ext2	Second Extended FileSystem
ext4	Fourth Extended Filesystem with extents
btfrs	B-Tree filesystem
hfs	Macintosh HFS Filesystem
jffs2	The Journalling Flash File System, v2
reiserfs	ReiserFS journaled filesystem
ubifs	UBIFS - UBI File System
udf	Universal Disk Format Filesystem

Filesystem Analysis Report

FS	Line of Code	Smatch	CppChek	Sparse	FlawFindr	Blast	Frama-C
ext2	6840	3	--	0	29	--	--
ext4	32754	30	--	0	103	--	--
btrfs	82450	84	--	69	191	--	--
hfs	4618	22	--	0	20	--	--
jffs2	13771	40	--	6	45	--	--
reiserfs	21742	40	--	6	159	--	--
ubifs	21988	6	--	0	63	--	--
udf	8980	26	--	0	63	--	--

Conclusion

- Linux has a lot of great tools for making kernel development easier
- We need them to keep up with the growing complexity
- But Still many improvement possible

Thank you!