Using Clang Static Analyzer With the Linux Kernel Code

Presented by:

Behan Webster

(LLVMLinux project lead)

Presentation Date: 2015.08.19

LLVMLinux project

Static Analyzer (Wikipedia)

- Static program analysis is the analysis of computer software that is performed without actually executing programs
- In most cases the analysis is performed on some version of the source code, and in the other cases, some form of the object code

Static Analyzer (Original)

- Originally tools which looked for common problems using pattern matching on source code
- Typically written outside of the compiler, not really understanding the code, its meaning nor intent

Static Analyzer (Semantic)

- Semantic analyzers understand the meaning of the code
- They employ compiler technology to look at what the code is doing and what it means

Clang Static Analyzer

- The clang static analyzer uses clang/LLVM
- Analyzes paths through the code within a compilation unit (a file)
- Looks for deeper, potentially cross-function, issues
- (issues that are often only found at runtime)
- http://clang-analyzer.llvm.org/

How does it work?

- The clang static analyzer is run at compile time
- Applies checkers to compiled code
- Checkers work at the AST/LLIR level
- They can be used to look for common issues
- Each checker looks for a specific kind of issue

Example Generic Checkers

- Branch condition evaluates to a garbage value
- Dangerous variable-length array (VLA) declaration
- Dereference of null pointer
- Dereference of undefined pointer value
- Division by zero
- Garbage return value
- Stack address stored into global variable
- Unix API

Issues with Static Analysis

- Analysis is performed at the compilation unit level
- Not all inputs or context are known
- Assumptions are made
- Not all assumptions are valid
- The result is false positives

Other Issues with Static Analysis

- It makes your compile take a lot longer
- Checkers are run during/after compilation
- Some checkers can take $O(n^2)$ time (worst case)
- Typically overall compile times are 2-4x longer

Further Issues with Static Analysis

- Most checkers were written for user space
- We turn off most of these since we are looking at system level code

How does it work?

- Run your build under scan-build
- Perl script which generates html output from analysis
- Uses ccc-analyzer which messes with CC/CFLAGS

```
$ scan-build make foo
```

- (A bit more complicated than that for the kernel)

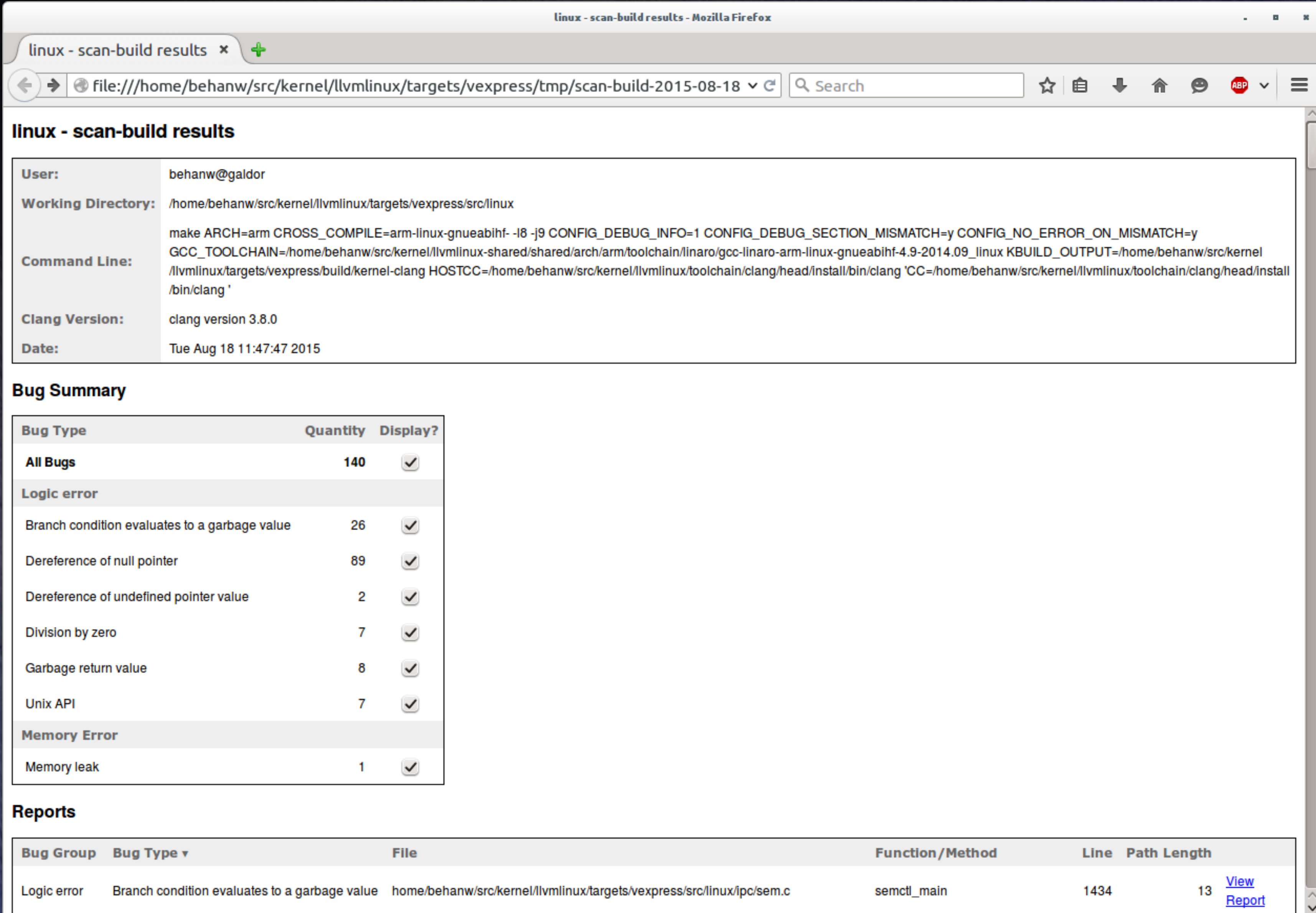
Analyzing the kernel

- Mainline kernel still needs patches to compile with clang
- Only works with the latest version of clang
- Requires a patch to ccc-analyzer to work

Just show me how to run it...

- git clone http://git.linuxfoundation.org/llvmlinux.git
- cd llvmlinux/target/vexpress
- make kernel-scan-build
- firefox scan-build-2015-08-18-114747-30457-1/index.html

Html Output


linux - scan-build results - Mozilla Firefox
linux - scan-build results
file:///home/behanw/src/kernel/llvmlinux/targets/vexpress/tmp/scan-build-2015-08-18
Search

linux - scan-build results

User:	behanw@galdor
Working Directory:	/home/behanw/src/kernel/llvmlinux/targets/vexpress/src/linux
Command Line:	make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -l8 -j9 CONFIG_DEBUG_INFO=1 CONFIG_DEBUG_SECTION_MISMATCH=y CONFIG_NO_ERROR_ON_MISMATCH=y GCC_TOOLCHAIN=/home/behanw/src/kernel/llvmlinux-shared/shared/arch/arm/toolchain/linaro/gcc-linaro-arm-linux-gnueabihf-4.9-2014.09_linux KBUILD_OUTPUT=/home/behanw/src/kernel/llvmlinux/targets/vexpress/build/kernel-clang HOSTCC=/home/behanw/src/kernel/llvmlinux/toolchain/clang/head/install/bin/clang 'CC=/home/behanw/src/kernel/llvmlinux/toolchain/clang/head/install/bin/clang '
Clang Version:	clang version 3.8.0
Date:	Tue Aug 18 11:47:47 2015

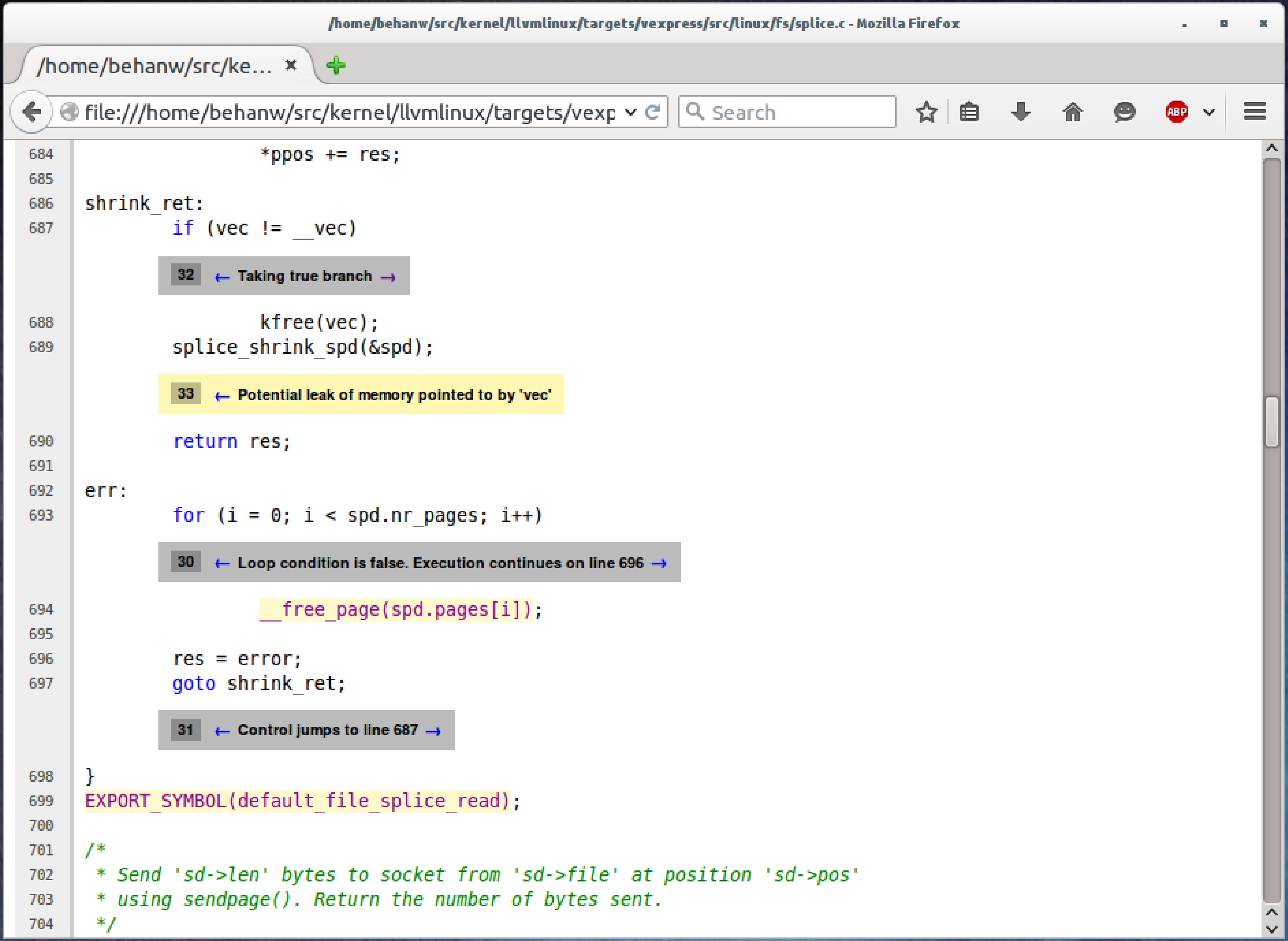
Bug Summary

Bug Type	Quantity	Display?
All Bugs	**140**	✓
Logic error		
Branch condition evaluates to a garbage value	26	✓
Dereference of null pointer	89	✓
Dereference of undefined pointer value	2	✓
Division by zero	7	✓
Garbage return value	8	✓
Unix API	7	✓
Memory Error		
Memory leak	1	✓

Reports

Bug Group	Bug Type ▾	File	Function/Method	Line	Path Length	
Logic error	Branch condition evaluates to a garbage value	home/behanw/src/kernel/llvmlinux/targets/vexpress/src/linux/ipc/sem.c	semctl_main	1434	13	View Report

Potential Memory Leak (not confirmed)


/home/behanw/src/kernel/llvmlinux/targets/vexpress/src/linux/fs/splice.c - Mozilla Firefox
/home/behanw/src/ke...
file:///home/behanw/src/kernel/llvmlinux/targets/vexp
Search

```
684                 *ppos += res;
685 
686 shrink_ret:
687         if (vec != __vec)
        32  ← Taking true branch →
688                 kfree(vec);
689         splice_shrink_spd(&spd);
        33  ← Potential leak of memory pointed to by 'vec'
690         return res;
691 
692 err:
693         for (i = 0; i < spd.nr_pages; i++)
        30  ← Loop condition is false. Execution continues on line 696 →
694                 __free_page(spd.pages[i]);
695 
696         res = error;
697         goto shrink_ret;
        31  ← Control jumps to line 687 →
698 }
699 EXPORT_SYMBOL(default_file_splice_read);
700 
701 /*
702  * Send 'sd->len' bytes to socket from 'sd->file' at position 'sd->pos'
703  * using sendpage(). Return the number of bytes sent.
704  */
```

Can Linux Specific Checkers be Added?

- Yes.
- There is a whole mechanism for adding your own checkers
- http://clang-analyzer.llvm.org/checker_dev_manual.html

- Linux kernel specific checkers are the ultimate goal

Contribute to the LLVMLinux Project

- Project wiki page
  - http://llvm.linuxfoundation.org
- Project Mailing List
  - http://lists.linuxfoundation.org/mailman/listinfo/llvmlinux
  - http://lists.linuxfoundation.org/pipermail/llvmlinux/
- IRC Channel
  - #llvmlinux on OFTC
  - http://buildbot.llvm.linuxfoundation.org/irclogs/OFTC/%23llvmlinux/
- LLVMLinux Community on Google Plus