FUJITSU

shaping tomorrow with you

Applying Clang Static Analyzer to Linux Kernel

2012/6/7

FUJITSU COMPUTER TECHNOLOGIES LIMITED
Hiroo MATSUMOTO

管理番号 1154ka1

Abstract

FUJITSU

- Now there are many great static analyzers, we can find bugs with them automatically. But in terms of the accuracy and range of bug detection, there are room for improvement.

- Clang Static Analyzer is Open Source static analyzer. We can control the accuracy and range of bug detection with it. What is more, we can enhance the ability by adding our analysis codes.

- Applying Clang Static Analyzer to Linux Kernel will **reduce review time** of codes in making driver or merging patches.

- This presentation will describe a way of applying Clang Static Analyzer to Linux Kernel and what problem happens.

Introduction

FUJITSU

- Company Profile
  - Software/Hardware/Testing developer of Embedded systems.
  - Destributor of linux/RTOS for embedded systems .
- My Profile
  - Research the method of static code-analysis.
  - Program linux device drivers.
  - Test with LTP and LSB.
- Topics
  - Trial of Clang Static Analyzer for tracing jiffies in 2011.
  - Challenge applying Clang Static Analyzer on linux in 2012.

FUJITSU

Agenda

- **Clang**
- Clang Static Analyzer
- Applying to Linux Kernel
- Demo

What is Clang?

FUJITSU

- LLVM front-end for c/c++/objective-c compiler.
  - Source code to Intermediate representation.
  - Intermediate representation to object.
- Building Mac OS, iOS, Android and Free BSD.
- Clang can be also used for Linux.

```
$ cat Makefile
CC  = clang --analyze # or ccc-analyzer
OBJ = $(patsubst %.c, %, $(wildcard *.c))

.PHONY: all clean
all: ${OBJ}
clean:
        -rm -f ${OBJ}
```

Superiority of Clang

FUJITSU

- Less compile time and memory resources than GCC.
- Compatibility with GCC.
  - Clang has GCC options and GCC extensions (not fully).
- **Static Analysis framework.**
  - About 60 checkers.
  - Adding specific checker.

Example of finding error automatically with Clang

FUJITSU

- An error around malloc memory allocation.
  - GCC cannot find it.
  - Clang can find it.

```
$ cat memleak.c
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  unsigned int *mem;
  mem = malloc(sizeof(*mem));
  if (mem) /** This check should be !mem */
    return 1;
  *mem = 0xdeadbeaf;
  free(mem);
  return 0;
}
```

Analyzed code that has an error.

```
$ gcc -Wall memleak.c
$
```

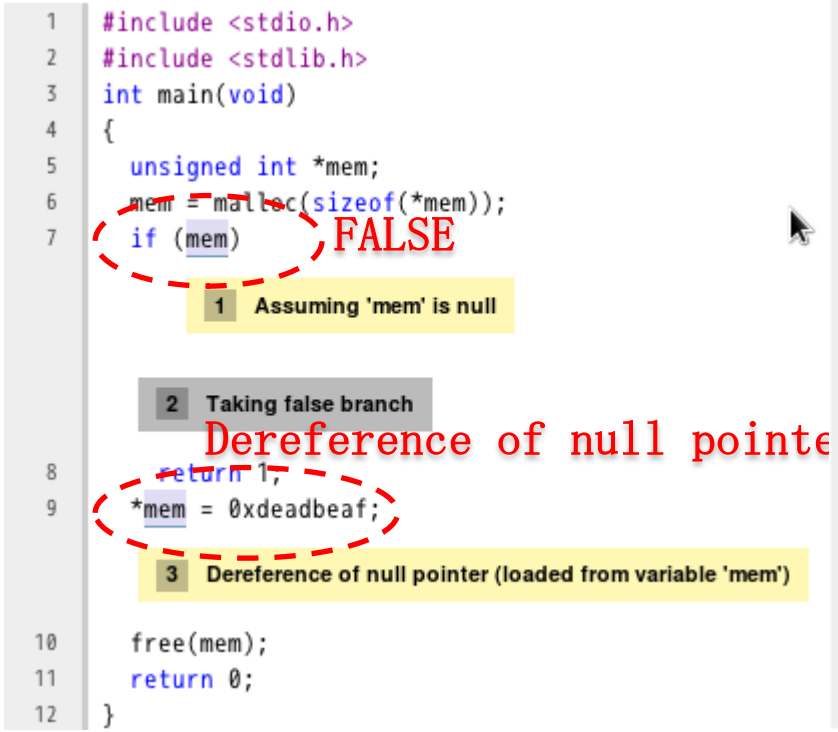
Compile with GCC

```
$ ccc-analyzer memleak.c
memleak.c:8:12: warning: Memory is never
 released; potential leak of memory pointed
 to by 'mem'
    return 1;
           ^

memleak.c:9:8: warning: Dereference of null
 pointer (loaded from variable 'mem')
  *mem = 0xdeadbeaf;
   ~~~ ^

2 warnings generated.
$
```

Compile with Clang

HTML Report

- Path sensitive with assuming
  - DereferenceChecker
  - MallocChecker



Status of building Linux Kernel with Clang

FUJITSU

- LLVM Linux Project
  - Building x86_64 binary and succeed to boot in 2010.
  - This needs patches for Linux Kernel and Clang.
  - https://github.com/lll-project

- Now need patches for building Linux Kernel too.
  - C Language compatibility.
  - Assembler compatibility.

Cross Compile with Clang

FUJITSU

- Support x86, x86_64, ARM, PowerPC and more.

```
$ ls llvm/lib/Target/
ARM               Makefile      Target.cpp                    TargetMachine.cpp
CMakeLists.txt    Mangler.cpp   TargetData.cpp                TargetMachineC.cpp
CellSPU           Mips          TargetELFWriterInfo.cpp       TargetRegisterInfo.cpp
CppBackend        NVPTX         TargetInstrInfo.cpp           TargetSubtargetInfo.cpp
Hexagon           PTX           TargetIntrinsicInfo.cpp       X86
LLVMBuild.txt     PowerPC       TargetJITInfo.cpp             XCore
MBlaze            README.txt    TargetLibraryInfo.cpp
MSP430            Sparc         TargetLoweringObjectFile.cpp
```

- Need host triple specification
  - GCC's host triple specification is fixed when compiling GCC.
  - Clang's host triple specification is fixed when running Clang.

```
$ clang -march=armv7-a -mcpu=cortex-a9 -ccc-host-triple arm-none-linux
 -ccc-gcc-name arm-none-linux-gnueabi-gcc -I/opt/arm/usr/include hello.c -S
```

FUJITSU

Agenda

- Clang
- **Clang Static Analyzer**
- Applying to Linux Kernel
- Demo

What is Clang Static Analyzer?

FUJITSU

- Static Analyzer with AST (Abstract Syntax Tree)
  - Clang generates AST when compiling.
  - Checkers uses AST.
    - About 60 Checkers

```
$ ls llvm/tools/clang/lib/StaticAnalyzer/Checkers/
AdjustedReturnValueChecker.cpp   CommonBugCategories.cpp              ObjCContainersChecker.cpp
AnalyzerStatsChecker.cpp         DeadStoresChecker.cpp                ObjCSelfInitChecker.cpp
ArrayBoundChecker.cpp            DebugCheckers.cpp                    ObjCUnusedIVarsChecker.cpp
ArrayBoundCheckerV2.cpp          DereferenceChecker.cpp               PointerArithChecker.cpp
AttrNonNullChecker.cpp           DivZeroChecker.cpp                   PointerSubChecker.cpp
BasicObjCFoundationChecks.cpp    FixedAddressChecker.cpp              PthreadLockChecker.cpp
BoolAssignmentChecker.cpp        GenericTaintChecker.cpp              RetainCountChecker.cpp
BuiltinFunctionChecker.cpp       IdempotentOperationChecker.cpp       ReturnPointerRangeChecker.cpp
CMakeLists.txt                   InterCheckerAPI.h                    ReturnUndefChecker.cpp
CStringChecker.cpp               IteratorsChecker.cpp                 StackAddrEscapeChecker.cpp
CStringSyntaxChecker.cpp         LLVMConventionsChecker.cpp           StreamChecker.cpp
CallAndMessageChecker.cpp        MacOSKeychainAPIChecker.cpp          TaintTesterChecker.cpp
CastSizeChecker.cpp              MacOSXAPIChecker.cpp                 UndefBranchChecker.cpp
CastToStructChecker.cpp          Makefile                             UndefCapturedBlockVarChecker.cpp
CheckObjCDealloc.cpp             MallocChecker.cpp                    UndefResultChecker.cpp
CheckObjCInstMethSignature.cpp   MallocOverflowSecurityChecker.cpp    UndefinedArraySubscriptChecker.cpp
CheckSecuritySyntaxOnly.cpp      MallocSizeofChecker.cpp              UndefinedAssignmentChecker.cpp
CheckSizeofPointer.cpp           NSAutoreleasePoolChecker.cpp         UnixAPIChecker.cpp
CheckerDocumentation.cpp         NSErrorChecker.cpp                   UnreachableCodeChecker.cpp
Checkers.td                      NoReturnFunctionChecker.cpp          VLASizeChecker.cpp
ChrootChecker.cpp                OSAtomicChecker.cpp                  VirtualCallChecker.cpp
ClangCheckers.cpp                ObjCAtSyncChecker.cpp
ClangSACheckers.h                ObjCContainersASTChecker.cpp
```

Package of Checkers

FUJITSU

- Checkers belong to package
  - Core package
    - Null pointer checker
    - Undefined reference checker
  - DeadCode package
    - Dead stored to variable checker
    - Unreachable code checker
  - Security package
    - Array bound checker
    - Malloc overflow checker
  - Unix package
    - Pthread lock checker
    - Malloc checker

Adding Checker

FUJITSU

- Checkers.td
  - Package and checker.
  - Select analysis package with --analyzer-checker option.
- CMakeLists.txt
- <Checker>.cpp
  - A code of checker with AST.
  - Logic for detecting bugs

Clang Static Analyzer Framework

FUJITSU

- Framework calls checker for each conditions.

Method	Condition for being called
check::PreStmt<xxx>	This is called before xxx statement. xxx statement is like variable declaration.
check::PostStmt<xxx>	This is called after xxx statement.
check::BranchCondition	This is called when branch is occurred.
check::Location	This is called when storing value.
check::Bind	This is called at equal statement.
check::EndPath	This is called when traversing path is ended.
check::EndAnalysis	This is called when all traversing path is ended.
eval::Call	This is called at callee statement.
eval::Assume	This is called when assumption is occurred like branch.
check::ASTDecl<FunctionDecl>	This is called at function declaration which is top of AST.
and more	

Example of checker using eval::Call

FUJITSU

Checker

- Output function name for each callee statement.

```
class SampleEvalCall : public Checker<eval::Call> {
public:
  bool evalCall(const CallExpr *CE, CheckerContext &C) const {
    /** Output function name */
    llvm::errs() << C.getCalleeName(CE) << " is called¥n ";
    return false;
  }
};
```

Analyzed code (hello.c)

```
#include <stdio.h>

int main(void)
{
  printf("hello, ");
  puts("world");
  return 0;
}
```

Running

```
$ clang --analyze hello.c # or ccc-analyzer hello.c
printf is called
puts is called
```

FUJITSU

Agenda

- Clang
- Clang Static Analyzer
- **Applying to Linux Kernel**
- Demo

This presentation's approach

FUJITSU

- Errors related assembler for building Linux Kernel.
  - This needs to modify code related with assembler in Clang.

```
xxx/arch/x86/include/asm/dwarf2.h:55:20: error: unexpected token in '.macro' directive
.macro cfi_ignore a=0, b=0, c=0, d=0
                   ^

xxx/arch/x86/include/asm/dwarf2.h:56:6: error: unexpected '.endm' in file, no current macro
definition
.endm
     ^

xxx/arch/x86/include/asm/dwarf2.h:102:22: error: unexpected token in '.macro' directive
 .macro movq_cfi reg offset=0
                     ^

xxx/arch/x86/include/asm/dwarf2.h:103:7: error: invalid register name
 movq %¥reg, ¥offset(%rsp)
```

- This presentation approach will not treat assembler.
- Not modify Clang and modifies Linux Kernel only.
- Treating compatibilities of C language.
- Using clang with -fsyntax-only option.

Using -fsyntax-only

FUJITSU

- Toolchain without CC can be empty.
- Kbuild needs some compiled file.
  - bounds.s and asm-offsets.s for header file.
- Some Makefile target needs compiled file.
  - empty.o and some objects.
- Some CONFIG needs compiled file.
  - CONFIG_MODVERSION
  - CONFIG_DYNAMIC_FTRACE
  - CONFIG_DEBUG_SECTION_MISMATCH
- **First time is building with GCC.**
  Second time is analyzing with clang -fsyntax-only after copying GCC's compiled files.

BUILD_BUG_ON is always error

FUJITSU

- Assertion for condition be fixed when building Linux Kernel.
  - If condition is TRUE, array size will be negative.

```
#define BUILD_BUG_ON(condition) ((void)sizeof(char[1 - 2*!!(condition)]))
```

- Clang will assume TRUE and FALSE for condition.

```
CC      init/main.o - due to target missing
In file included from init/main.c:13:
In file included from include/linux/module.h:13:
In file included from include/linux/kmod.h:27:
In file included from include/linux/sysctl.h:933:
include/linux/rcupdate.h:822:2: error: array size is negative
        BUILD_BUG_ON(!__builtin_constant_p(offset));
        ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

include/linux/kernel.h:719:52: note: expanded from macro 'BUILD_BUG_ON'
#define BUILD_BUG_ON(condition) ((void)sizeof(char[1 - 2*!!(condition)]))
                                                   ^~~~~~~~~~~~~~~~~~~~
```

Assigned value is garbage or undefined

FUJITSU

- Statement like “int a = a”
  - Evaluating right ‘a’ firstly ?
  - Evaluating left ‘int a’ firstly ?
- This trick suppresses GCC’s “uninitialized value” warning.
- Clang warns not only “unused value” but also “Assigned value is garbage or undefined”.

```
  480 |#define PVOP_VCALL_ARGS                                 ¥
  481 |        unsigned long __eax = __eax, __edx = __edx, __ecx = __ecx
...
  499 |#define PVOP_VCALL_ARGS                                    ¥
  500 |        unsigned long __edi = __edi, __esi = __esi,        ¥
  501 |                __edx = __edx, __ecx = __ecx, __eax = __eax
```

```
  104 |#define uninitialized_var(x) x = x
```

- This will make many noise.

Not support some alignment extension

FUJITSU

- Issues related with alignment.
  - Not support __alignof__(type var)

```
crypto/shash.c:68:56: error: 'aligned' attribute ignored when parsing type
        return len + (mask & ~(__alignof__(u8 __attribute__ ((aligned))) - 1));
```

  - Not support (type __attribute__((aligned(x)))) cast
    - http://llvm.org/bugs/show_bug.cgi?id=11071

```
drivers/staging/sep/sep_driver.c:2102:9: error: 'aligned' attribute ignored when parsing type
                                                (aligned_u64)app_out_address;
                                                ^~~~~~~~~~~~

include/linux/types.h:125:42: note: expanded from macro 'aligned_u64'
#define aligned_u64 __u64 __attribute__((aligned(8)))
                                         ^~~~~~~~~~
```

Not support variable length array in struct field

FUJITSU

- Support variable length array but not support variable length array in struct field.

```
644 static int i2400m_download_chunk(struct i2400m *i2400m, const void *chunk,
645                                   size_t __chunk_len, unsigned long addr,
646                                   unsigned int direct, unsigned int do_csum)
647 {
648         int ret;
649         size_t chunk_len = ALIGN(__chunk_len, I2400M_PL_ALIGN);
650         struct device *dev = i2400m_dev(i2400m);
651         struct {
652                 struct i2400m_bootrom_header cmd;
653                 u8 cmd_payload[chunk_len];
654         } __packed *buf;
```

variable length from function's argument

variable length array in struct field

```
drivers/net/wimax/i2400m/fw.c:653:6: error: fields must have a constant size:
'variable length array in structure' extension will never be supported
                u8 cmd_payload[chunk_len];
                   ^
```

Not support declaration of function in function

FUJITSU

- Declaration of function in function is usefull for namespace issue.

declaration of function

```
2328 |static void hotkey_compare_and_issue_event(struct tp_nvram_state *oldn,
2329 |                                            struct tp_nvram_state *newn,
2330 |                                            const u32 event_mask)
2331 |{
<snip>
2346 |        void issue_volchange(const unsigned int oldvol,
2347 |                             const unsigned int newvol)
2348 |        {
<snip>
2359 |        }
```

declaration of function in function

```
drivers/platform/x86/thinkpad_acpi.c:2347:35: error: expected ';' at end of
declaration
                             const unsigned int newvol)
                                                       ^
                                                       ;
```

Not support pragma pack in struct field

FUJITSU

- pragma pack used for alignment of struct field.

definition of struct

```
 682 |struct s_TPL {  /* Transmit Parameter List (align on even word boundaries) */
<snip>
 709 |        Fragment FragList[TX_FRAG_NUM]; /* Maximum: nine frame fragments in one
 710 |                                         * TPL actual version of firmware: 9
 711 |                                         * fragments possible.
 712 |                                         */
 713 |#pragma pack()
<snip>
 717 |        TPL *NextTPLPtr;                 /* Pointer to next TPL in chain. */
```

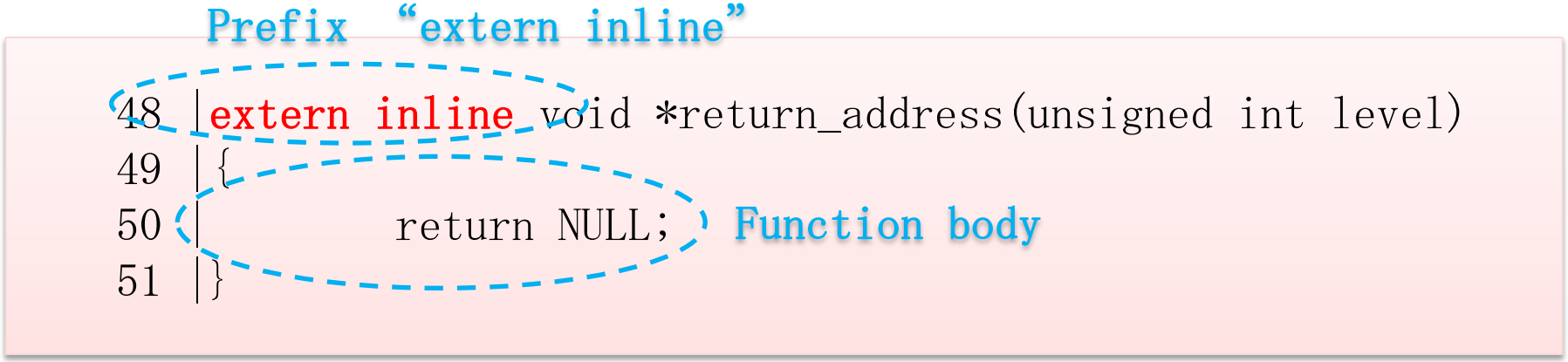
pragma pack in struct field

```
drivers/net/tokenring/tms380tr.h:713:9: error: type name requires a specifier
or qualifier
#pragma pack()
        ^
```

Not support extern inline definition?

FUJITSU

- Ftrace on ARM platform uses return_address function
  - arch/arm/kernel/return_address.c
  - arch/arm/include/asm/ftrace.h
- Function body with prefix “extern inline”




```
arch/arm/kernel/return_address.c:65:7: error:
void *return_address(unsigned int level) redefinition of 'return_address'
      ^
```

For applying to Linux Kernel

FUJITSU

- Clang needs to support GCC assembler extensions.
  - issue of ".macro" directive
  - may be more issue
- Clang needs to support GCC C language extensions.
  - issue of alignment extension
  - pragma pack in struct field
  - variable length in struct field
- Kbuild needs to take care of Clang
  - BUILD_BUG_ON
  - Trick of "int a = a"
  - declaration of function in function
- Need to resolve about return_address

FUJITSU

Agenda

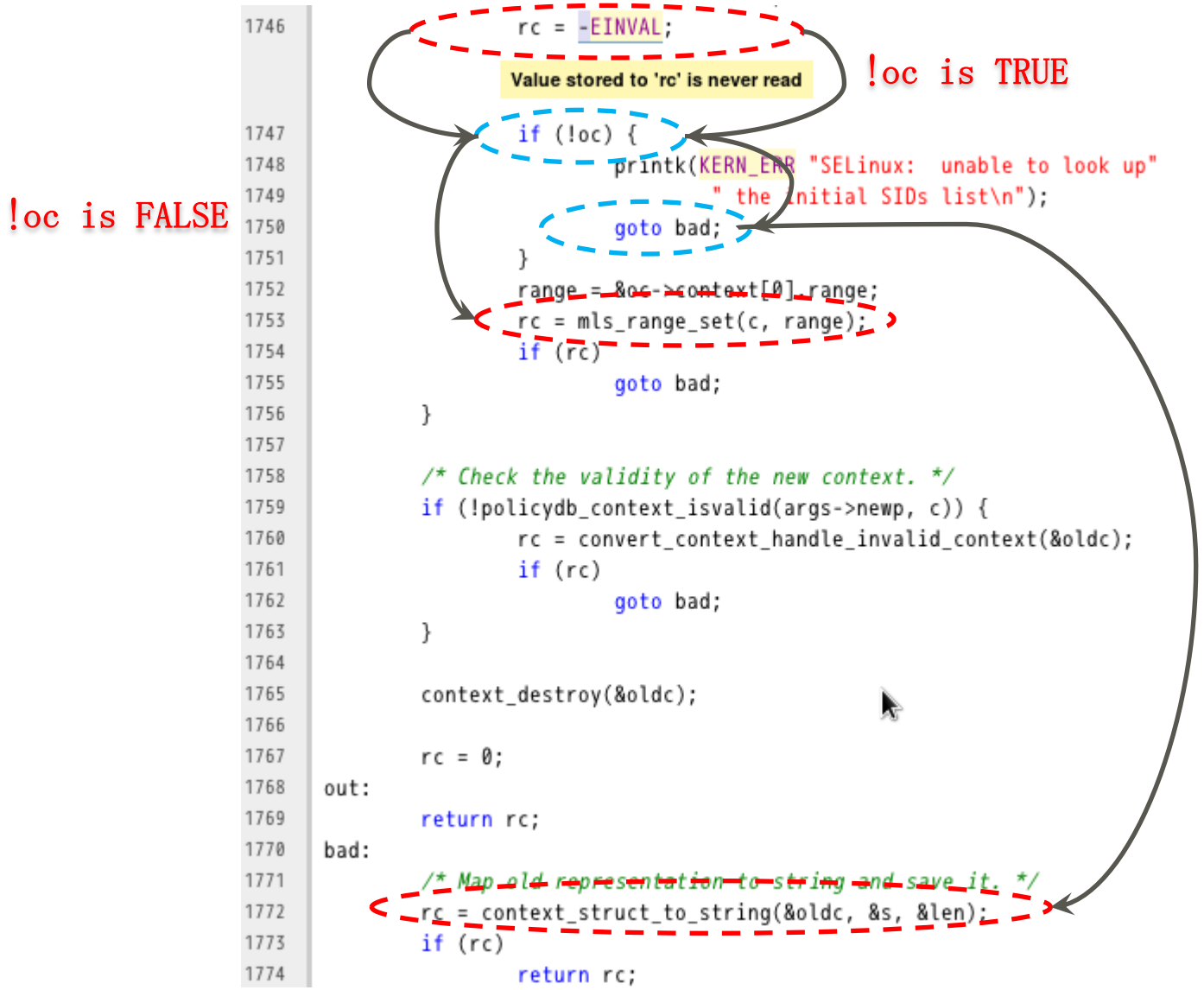
- Clang
- Clang Static Analyzer
- Applying to Linux Kernel
- **Demo**

Checkers result

Bug Type	Quantity	Display?
All Bugs	**8091**	☑
Dead store		
Dead assignment	4690	☑
Dead increment	354	☑
Dead initialization	203	☑
Logic error		
Array subscript is undefined	23	☑
Assigned value is garbage or undefined	311	☑
Branch condition evaluates to a garbage value	71	☑
Called function pointer is an uninitalized pointer value	3	☑
Called function pointer is null (null dereference)	31	☑
Dangerous variable-length array (VLA) declaration	3	☑
Dereference of null pointer	1504	☑
Dereference of undefined pointer value	24	☑
Division by zero	49	☑
Function call argument is an uninitialized value	288	☑
Garbage return value	32	☑
Out-of-bound array access	7	☑
Result of operation is garbage or undefined	422	☑
Return of address to stack-allocated memory	2	☑
Stack address stored into global variable	9	☑
Unix API	65	☑

- Bug summery
  - make x86_64_defconfig allyesconfig
  - using a way of previous slides.
- Dead assignment
  - Good result but may be optimized
- Dereference of null pointer
  - May be false positive
- Unix API
  - Issue about memcpy's argument
  - May be false positive

Dead assignment



- Good result but compiler optimization will work.

Dereference of null pointer

```
2790  static int sd_resume(struct device *dev)
2791  {
2792          struct scsi_disk *sdkp = scsi_disk_get_from_dev(dev);
2793          int ret = 0;
2794
2795          if (!sdkp->device->manage_start_stop)
                  Access to field 'device' results in a dereference of a null pointer (loaded from variable 'sdkp')
2796                  goto done;
2797
2798          sd_printk(KERN_NOTICE, sdkp, "Starting disk\n");
2799          ret = sd_start_stop_device(sdkp, 1);
2800
2801  done:
2802          scsi_disk_put(sdkp);
2803          return ret;
2804  }
```

- Dereference of null pointer if sdkp->device is NULL.
- Caller of this function may take care of it.
- This may be false positive.

Unix API

be set to .instantiate operation in struct

FUJITSU

```
789 static int rxrpc_instantiate_s(struct key *key, const void *data,
790                                 size_t datalen)
791 {
792         struct crypto_blkcipher *ci;
793
794         _enter("{%x},,%zu", key_serial(key), datalen);
795
796         if (datalen != 8)
```

1 Taking false branch

```
797                 return -EINVAL;
798
799         memcpy(&key->type_data, data, 8);
```

arguments are unknown

2 Within the expansion of the macro 'memcpy':

a Null pointer argument in call to memory copy function

- Caller of this function via .instantiate operation is out of file.
- Clang worried about argument.

Checkers for Linux Kernel

FUJITSU

- Checkers are not fit to Linux Kernel.
  - MallocChecker supports malloc, free…

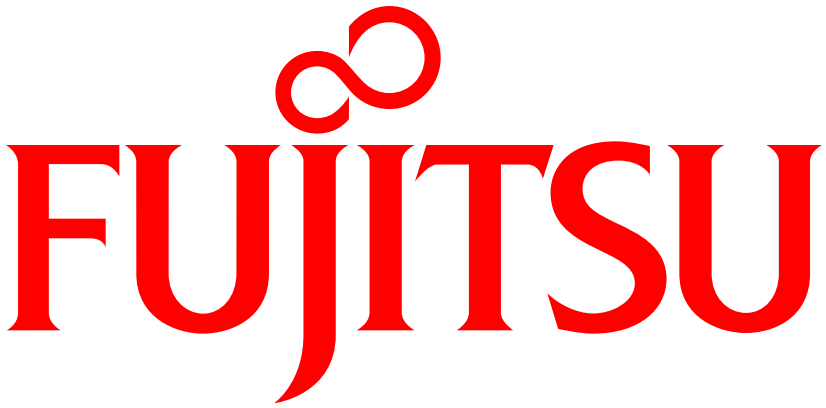
```
336 |void MallocChecker::initIdentifierInfo(ASTContext &Ctx) const {
337 |  if (!II_malloc)
338 |    II_malloc = &Ctx.Idents.get("malloc");
339 |  if (!II_free)
340 |    II_free = &Ctx.Idents.get("free");
```

    - Trial of “malloc” to “kmalloc” and “free” to “kfree” made not good result.
- Checkers framework may be not fit to Linux Kernel.
  - Clang traverse a path of only one function.
  - Need to traverse a path of one-to-one functions for detecting memory leak and other bugs. This can be done soon.
    - .probe operation and .remove operation
- Checkers for Linux Kernel specific
  - Detecting sleep statement during preemption disable.

Conclusion

FUJITSU

- It is need to expand Kbuild and Clang.
  - Clang will support GCC options and extensions day by day.
  - Will Kbuild support Clang in the future ?
- Checkers are needed to be customized for Linux Kernel.
  - Customize is not difficult so much.
  - Linux Kernel needs many Linux Kernel specialized checkers.
  - That will reduce our review time.



shaping tomorrow with you