What’s new in GCC -fanalyzer?

David Malcolm <dmalcolm@redhat.com>

GNU Tools Cauldron 2022
September 2022
Overview

• What is -falyzer?
• Diagnostic Serialization
• 15 new -Wanalyzer-* warnings (so far)
• Internal Improvements
• -falyzer on Linux kernel
What is -fanalyzer?
What is -fanalyzer?

- GCC option: a static analyzer
- Explores “interesting” interprocedural paths through the code via “symbolic execution” looking for bugs to warn about
  (for some definitions of “interesting” and of “bugs”)
- Can have false positives and false negatives
- https://gcc.gnu.org/wiki/StaticAnalyzer
What is -fanalyzer?

- GCC 10: 15 new warnings
- GCC 11: 7 new warnings (22 total)
- GCC 12: 5 new warnings (27 total)
- GCC 13: 15 new warnings so far (42 total)

https://gcc.gnu.org/wiki/StaticAnalyzer
Diagnostic Serialization
Diagnostic Serialization: recording

- GCC 9: `-fdiagnostics-format=json`
- New in GCC 13:
  - `-fdiagnostics-format=json-stderr`
  - `-fdiagnostics-format=json-file`
  - `-fdiagnostics-format=sarif-stderr`
  - `-fdiagnostics-format=sarif-file`
```c
#include <stdlib.h>

void test (void *ptr, int flag)
{
    if (flag) /* Step 1: following 'true' branch (when 'flag != 0')... */
        free(ptr); /* Step 2: ...to here */
    free(ptr); /* Step 3: first 'free' here */
    /* Step 4: second 'free' here; first 'free' was at (3) */
}
```
Diagnostic Serialization: playback using GCC

• Not yet in trunk:
  • [PATCH 00/12] RFC: Replay of serialized diagnostics
  • e.g. the results of a javascript linter:

$ ./xgcc -B. -S ..../..../sarif-tutorials/samples/1-Introduction/simple-example.sarif
file:///C:/dev/sarif/sarif-tutorials/samples/Introduction/simple-example.js:1:5: error: 'x' is assigned a value but never used. [no-unused-vars]
New -fanalyzer warnings in trunk
Warnings relating to misuses of `<stdarg.h>`

- `-Wanalyzer-va-arg-type-mismatch`
- `-Wanalyzer-va-list-exhausted`
- `-Wanalyzer-va-list-leak`
- `-Wanalyzer-va-list-use-after-va-end`
-Wanalyzer-va-arg-type-mismatch

<source>: In function 'void consume_long(const char*, ...)':
<source>:9:5: warning: 'va_arg' expected 'long int' but received 'int' for variadic argument 1 of 'ap' [CWE-686] [-Wanalyzer-va-arg-type-mismatch]
  9 |   v = va_arg (ap, long);
  ^
  'void test_int_to_long()': events 1-2
       13 | void test_int_to_long (void)
       | ^~~~~~~~~~~~~
       | |
       | (1) entry to 'test_int_to_long'
       14 |
       15 | consume_long("fmt", 1066);
       | ^~~~~~~~~~~~~~~~~~~~~~~~~~~
       | |
       | (2) calling 'consume_long' from 'test_int_to_long' with 1 variadic argument

+-+ 'void consume_long(const char*, ...)': event 3

       4 | consume_long (const char *fmt, ...)
       | ^~~~~~~~~~~~~
       | |
       | (3) entry to 'consume_long'

'void consume_long(const char*, ...)': event 4

       9 | v = va_arg (ap, long);
       | ^
       | |
       | (4) 'va_arg' expected 'long int' but received 'int' for variadic argument 1 of 'ap'
In function 'void consume_n_ints(int, ...)':

10 | void test (void)
   | ^~~~
   | |
   | (1) entry to 'test'

15 |
16 | consume_n_ints (2, 1066);
   | ~~~~~~~~~~~~~~~~~~~~~~~
   | |
   | (2) calling 'consume_n_ints' from 'test' with 1 variadic argument

+---> 'void consume_n_ints(int, ...)': events 3-8

4 | consume_n_ints (int n, ...)
   | ~~~~~~~~~~~
   | |
   | (3) entry to 'consume_n_ints'

......

9 | for (int i = 0; i < n; i++)
   | ~~~~
   | |
   | (4) following 'true' branch (when 'i < n')...
   |
   | (6) following 'true' branch (when 'i < n')...

10 | v = va_arg (ap, int);
   | ~~~~~
   | |
   | (5) ...to here
   | (7) ...to here
   | (8) 'ap' has no more arguments (1 consumed)
In function `void test_leak_of_va_copy(int, ...)':

```
9 | }    
    ^
'veoid test_leak_of_va_copy(int, ...)': events 1-2

7 | va_copy (ap2, ap1);
   ^~~~~~~

(1) 'va_copy' called here

8 | va_end (ap1);

9 | }
   ^

(2) missing call to 'va_end' to match 'va_copy' at (1)
```
<source>: In function 'void test_va_arg_after_va_end(int, ...)':
<source>:9:5: warning: 'va_arg' after 'va_end' [-Wanalyzer-va-list-use-after-va-end]
  9 |   i = va_arg (ap, int);
  |   ^
'veoid test_va_arg_after_va_end(int, ...)': events 1-2
  7 |   va_start (ap, placeholder);
  |   ^~~~~~~~~
  |   |
  |   (1) 'va_start' called here
  8 |   va_end (ap);
  ~~~~~~
  |
  |   (2) 'va_end' called here
'veoid test_va_arg_after_va_end(int, ...)': event 3
  9 |   i = va_arg (ap, int);
  |   ^
  |   |
  |   (3) 'va_arg' after 'va_end' at (2)
Other new warnings in trunk (by me)

• -Wanalyzer-jump-through-null
• -Wanalyzer-putenv-of-auto-var
In function `void test_callbacks()`:  

```
15 |   cb.on_redraw ();
   |   ~~~~~~~~~~~~
'void test_callbacks()': event 1

15 |   cb.on_redraw ();
   |   ~~~~~~~~~~~~

(1) jump through null pointer here
In function `void test_auto_buf_name_and_value(const char*, const char*)':

8 | putenv (buf);
  | ~~~~~~~~~~~~~

'void test_auto_buf_name_and_value(const char*, const char*)': event 1

8 | putenv (buf);
  | ~~~~~~~~~~~~~

(1) 'putenv' on a pointer to automatic variable 'buf'

6 | char buf[100];
  | ^~~

8:10: note: 'buf' declared on stack here

8:10: note: perhaps use 'setenv' rather than 'putenv'

8 | putenv (buf);
  | ~~~~~~~~~~~~~
New warnings in trunk by Tim Lange (GSoC)

- Wanalyzer-allocation-size
- Wanalyzer-imprecise-fp-arithmetic
- Wanalyzer-out-of-bounds
- Tested on coreutils, curl, httpd and openssh.
In function 'void test(int32_t)':

7 | _ptr = (int32_t *)malloc (n * sizeof (int16_t));
   | ^  ^
6 | ~~~~~~~~~~
5 | 'void test(int32_t)': event 1
4 | 7 | _ptr = (int32_t *)malloc (n * sizeof (int16_t));
   | ^  ^
3 | ~~~~~~~~~~
2 | |
1 | |
   | (1) allocated '((long unsigned int)n * 2)' bytes and assigned to 'int32_t*' (aka 'int*') here; 'sizeof (int32_t {aka int})' is '4'
In function 'test':

```
6 |   int *ptr = malloc (sizeof (int) * f);
   | ^~~~~~~~~~~~~~~~~~~
  'test': event 1
```

```
6 |   int *ptr = malloc (sizeof (int) * f);
   | ^~~~~~~~~~~~~~~~~~~
   |   (1) operand 'f' is of type 'float'
```

<source>:5:14: note: only use operands of an integer type inside the size argument
In function 'test_symbolic_size_with_terminator_oob':

```
17 |   str->data[len] = '\0';
    | ~~~~~~~~~~~~~~~~~~~~~~^~~~
'test_symbolic_size_with_terminator_oob': events 1-5

13 |   struct str *str = malloc(sizeof(str) + len);
    ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    | (1) capacity is 'len + 8' bytes
    | (2) capacity is 'len + 8' bytes
14 |   if (str) {
    ~
    | (3) following 'true' branch (when 'str' is non-NULL)...
15 |     str->len = len;
        ~~~~~~~~~~~~~~
        | (4) ...to here
16 |     memset(str->data, 'x', len);
17 |     str->data[len] = '\0';
        ~~~~~~~~~~~~~~~~~~~~~~
        | (5) write of 1 byte at offset 'len + 8' exceeds the buffer
```
File descriptor support

- GSoC project by Immad Mir
  - State machine for tracking state of file descriptors
- In trunk: five new warnings, three new attributes
  - -Wanalyzer-fd-use-without-check
  - -Wanalyzer-fd-access-mode-mismatch
  - -Wanalyzer-fd-double-close
  - -Wanalyzer-fd-leak
  - -Wanalyzer-fd-use-after-close
  - __attribute__((fd_arg(N)))
  - __attribute__((fd_arg_read(N)))
  - __attribute__((fd_arg_write(N)))
- Special-casing of: open, creat, dup, dup2, dup3, read, write, close
In function 'void test(int, void*)':

`read` on possibly invalid file descriptor 'fd'

```
7 |    read (fd, buf, 1);
   |    ~~~~~~~~~~
```

'void test(int, void*)': events 1-2

```
6 |    int fd = dup2 (old_fd, 3);
   |          ~~~~~~~~~~~~~~
   |         (1) opened here
7 |    read (fd, buf, 1);
    ~~~~~~~~~~~~~~~
    |    (2) 'fd' could be invalid: unchecked value from (1)
```
-Wanalyzer-fd-access-mode-mismatch

```c
void test(char*, void*) {
  int f = open(path, O_RDONLY);
  if (f != -1) {
    write(f, buf, 1);
  }
}
```

1. opened here as read-only
2. assuming 'f' is a valid file descriptor (>= 0)
3. following 'true' branch (when 'f != -1')...
4. ...to here
5. 'write' on read-only file descriptor 'f'
In function `void test(char*, void*)':

```
8 |     close (fd);
 |     ^~~~

'void test(char*, void*)': events 1-3

6 |     int fd = open (path, O_RDWR);
 |     ^~~~~~~~~~~~~~~~~~~
 |     (1) opened here

7 |     close (fd);
 |     ~~~~~~~
 |     (2) first 'close' here

8 |     close (fd);
 |     ~~~~~~~~~
 |     (3) second 'close' here; first 'close' was at (2)
```
In function `void test(char*, void*)`:

```c
7 | }
7 | ^

'void test(char*, void*)': events 1-2

6 | int fd = open (path, O_RDWR);
6 | ^~~~~~~~~~
7 | (1) opened here
7 | }

| (2) 'fd' leaks here; was opened at (1)
```
In function 'void test(int)'

warning: 'f' on closed file descriptor 'fd' [-Wanalyzer-fd-use-after-close]

9 |   f (fd);
   |   ~~~^~~~

'void test(int)': events 1-2

8 |   close (fd);
 |   ~~~~~^~~~
 |   (1) closed here
9 |   f (fd);
 |   ~~~~~
 |   (2) 'f' on closed file descriptor 'fd'; 'close' was at (1)

note: argument 1 of 'void f(int)' must be an open file descriptor, due to '__attribute__((fd_arg(1)))'

4 |   void f (int fd) __attribute__((fd_arg(1)));
New warnings (in development)
New warnings I’m working on

- -Wanalyzer-deref-before-check
- -Wanalyzer-infinite-recursion
- -Wanalyzer-infinite-loop

- Requires some internal reworking of analyzer since there might not be a gimple stmt associated with an infinite loop
New warning from Tim: -Wanalyzer-restrict

1) void h(int n, int * restrict p, int * restrict q, int * restrict r)
2) {
3)   int i;
4)   for (i = 0; i < n; i++)
5)     p[i] = q[i] + r[i];
6) }

/path/to/main.c:70:13: warning: passing argument 3 to ‘restrict’-qualified parameter aliases with argument 4 [-Wrestrict]

70 |   h(100, a, b, b);
 |     ^  ~

• But https://www.open-std.org/jtc1/sc22/wg14/www/docs/n2912 seems to suggest that the above code is correct
Internal Improvements
Internal improvements

• Reimplemented call_string class (done)
• Fixups to how -fanalyzer emits execution paths in the face of inlined functions (done)
• Ability for GCC plugins to specify the behavior of a specific function to the analyzer (done)
• Use of std::unique_ptr (in progress: need a make_unique somewhere)
Trying *-fanalyzer on the Linux kernel
106358: [meta-bug] tracker bug for building the Linux kernel with -fanalyzer
106218: Analyzer false positives with Linux kernel's err.h
106229: False positives from -Wanalyzer-tainted-array-index with unsigned char index
104954: Analyzer takes a very long time on Linux kernel-drivers/gpu/drm/amd/display/de/eeales/deo_eales.e
104955: Analyzer slowdown with many diagnostics
104943: Analyzer fails to purge state for local structs
106204: False positive from -Wanalyzer use of uninitialized value with ftrivial auto var init-zero
106225: False positives from -Wanalyzer tainted divisor
106284: False positives from -Wanalyzer tainted array index with optimized conditionals
106319: False positives from -Wanalyzer va_arg type mismatch on int promotion
106321: False positives from -Wanalyzer tainted array index with switch with ranged cases
106359: -fanalyzer takes a very long time on Linux kernel: sound/see/seeedes/es47(85,90).e
106373: False positives from -Wanalyzer tainted array index on comparison with non-const
106374: [13 Regression] -fanalyzer ICE with certain const static vars
106383: False positives from -Wanalyzer va list exhausted
106394: False positive from -Wanalyzer allocation size with empty array
Trust boundaries in kernel

• New warning:
taint-demo.c:47:16: warning: use of attacker-controlled value ‘cmd.idx’ in array look up without checking for negative [CWE-129] [-Wanalyzer-tainted-array-index]
47 |     arr[cmd.idx] = cmd.val;
     ~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~
‘taint_signed_array_access’: events 1-5

42 |     if (copy_from_user(&cmd, src, sizeof(cmd)))
     ^
     |   (1) following ‘false’ branch...
43 |     return -EFAULT;
44 |     if (cmd.idx >= 16)
     ~~~~~~~~~~~
     |   |   (2) ...to here
     |   (3) following ‘false’ branch...
......
47 |     arr[cmd.idx] = cmd.val;
     ~~~~~~~~~~~~~~~~~~~~~~~~~~
     |   |   (5) use of attacker-controlled value ‘cmd.idx’ in array lookup without checking for negative
     |   (4) ...to here
In function ‘infoleak_stack_padding’:

```
103 |   if (copy_to_user(dst, &st, sizeof(st)))
   |     ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
103 |     ‘infoleak_stack_padding’: events 1-3

98 |   struct infoleak_3 st;
   |     ^~
   |     (1) region created on stack here
   |     (2) capacity: 8 bytes

........

103 |   if (copy_to_user(dst, &st, sizeof(st)))
   |     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   |     (3) uninitialized data copied from stack here
```

```
infoleak-demo.c:103:7: note: 3 bytes are uninitialized
103 |   if (copy_to_user(dst, &st, sizeof(st)))
   |     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

infoleak-demo.c:92:6: note: padding after field ‘a’ is uninitialized (3 bytes)
92 |   u8 a;
   |     ^

infoleak-demo.c:98:21: note: suggest forcing zero-initialization by providing a ‘{0}’ initializer
98 |   struct infoleak_3 st;
   |     ^
   |     = {0}
```
How to specify trust boundaries in kernel?

• Have got part of the way there via a GCC plugin (240 lines)
• Have a v2 of
  
  #pragma GCC custom_address_space

  • (not yet posted to list)
• ...but doing it as an attribute may be preferable from the kernel point-of-view
• May also want an __attribute__((noderef))
Summary

• General diagnostic improvements in GCC 13
  • Serialization
  • Metadata
• Lots of new warnings in -fanalyzer in GCC 13
• -fanalyzer on Linux kernel
• Want to implement a new warning?
  • Lots of ideas in Bugzilla...
  • ...or choose your own
QUESTIONS
THANKS