CAN WE RUN C CODE AND BE SAFE?

RUNNING GENTOO LINUX WITH ADDRESS SANITIZER.

Hanno Böck

WHO AM I?

Freelance journalist, mostly IT security topics.

Fuzzing Project: Improve the security of free software, supported by Linux Foundation's Core Infrastructure Initiative.

C / C++ AND MEMORY

Memory corruption, buffer overflows, double free, use after free, out of bounds reads, ...

Summary: Software reads or writes memory it shouldn't.

SAFER C?

Accessing invalid memory is "undefined behavior".

How about a C variant that prevents invalid memory access?

VALGRIND

SOFTBOUND+CETS

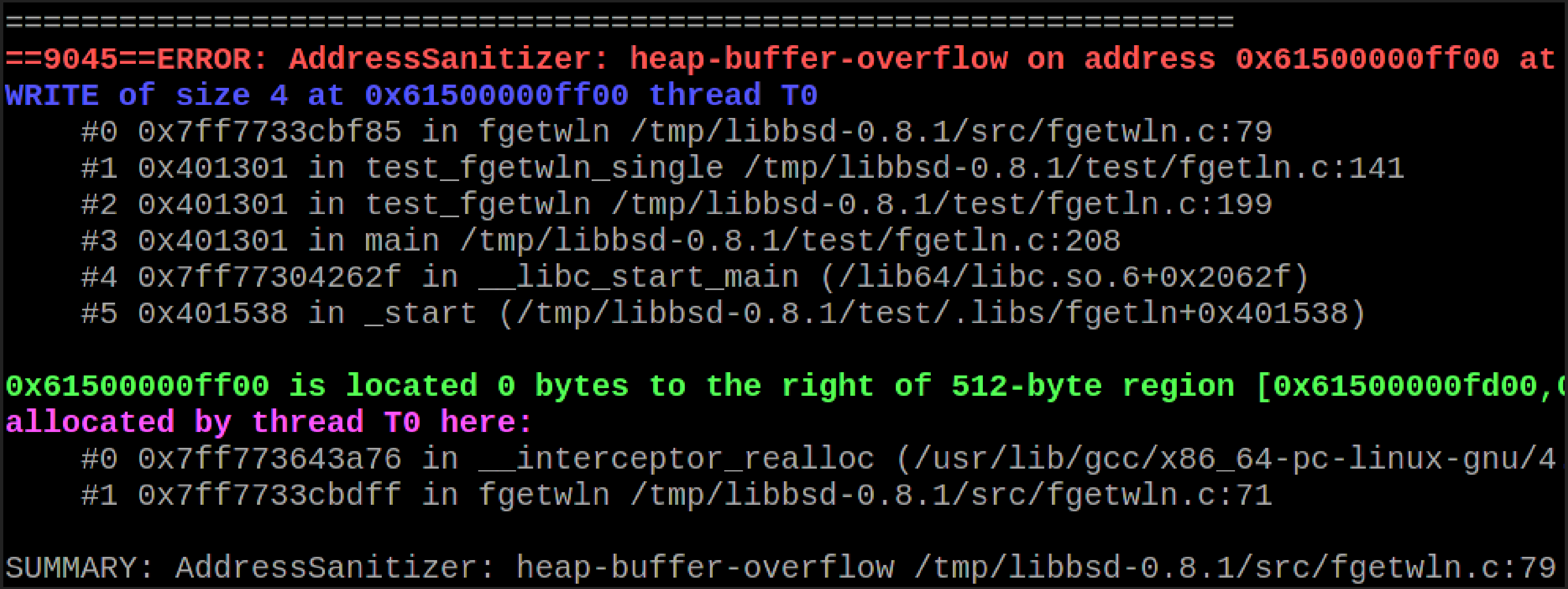
ADDRESS SANITIZER

ADDRESS SANITIZER (ASAN)

CFLAGS="-fsanitize=address"

Acceptable overhead (50-100% performance, lots of memory)

Practical - works usually out of the box



```
=================================================================
==9045==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x61500000ff00 at
WRITE of size 4 at 0x61500000ff00 thread T0
    #0 0x7ff7733cbf85 in fgetwln /tmp/libbsd-0.8.1/src/fgetwln.c:79
    #1 0x401301 in test_fgetwln_single /tmp/libbsd-0.8.1/test/fgetln.c:141
    #2 0x401301 in test_fgetwln /tmp/libbsd-0.8.1/test/fgetln.c:199
    #3 0x401301 in main /tmp/libbsd-0.8.1/test/fgetln.c:208
    #4 0x7ff77304262f in __libc_start_main (/lib64/libc.so.6+0x2062f)
    #5 0x401538 in _start (/tmp/libbsd-0.8.1/test/.libs/fgetln+0x401538)

0x61500000ff00 is located 0 bytes to the right of 512-byte region [0x61500000fd00,
allocated by thread T0 here:
    #0 0x7ff773643a76 in __interceptor_realloc (/usr/lib/gcc/x86_64-pc-linux-gnu/4
    #1 0x7ff7733cbdff in fgetwln /tmp/libbsd-0.8.1/src/fgetwln.c:71

SUMMARY: AddressSanitizer: heap-buffer-overflow /tmp/libbsd-0.8.1/src/fgetwln.c:79
```

Can we have a Linux full system built with Address Sanitizer?

GENTOO WITH ASAN

Just add -fsanitize=address to the CFLAGS and recompile everything.

If only it were that easy...

EXCLUDING SOME CORE PACKAGES

gcc, glibc - difficult, recursion problems, let's exclude them

DEPENDENCIES

ASAN executable, non-ASAN library: fine

ASAN library, non-ASAN executable: breaks

Consider compilation order and dependencies.

BUGS

ASAN terminates software if it reads invalid memory.

So we can't run software that always reads invalid memory...

Bugs fixed in Bash, Coreutils, man-db, syslog-ng, screen, nano, ...

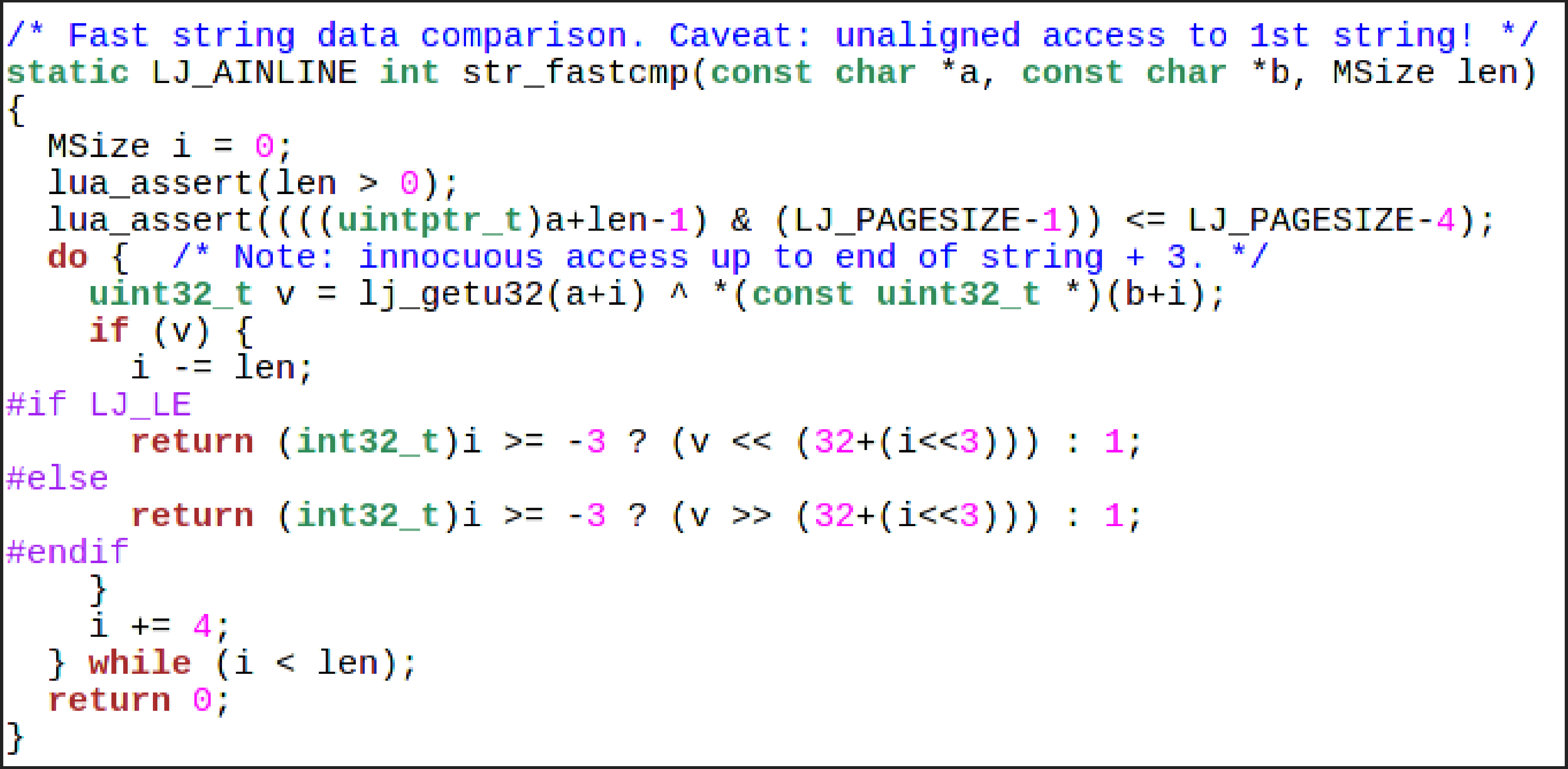
DOCUMENTED BUGS

BUG DENIAL

"This is a false positive, it must be a bug in Address Sanitizer"

```
  /* Scan the input one machine word at a time. */
#ifndef SVN_UTF_NO_UNINITIALISED_ACCESS
  /* This may read allocated but uninitialised bytes beyond the
     terminating null.  Any such bytes are always readable and this
     code operates correctly whatever the uninitialised values happen
     to be.  However memory checking tools such as valgrind and GCC
     4.8's address santitizer will object so this bit of code can be
     disabled at compile time. */
  for (; ; data += sizeof(apr_uintptr_t))
    {
      /* Check for non-ASCII chars: */
      apr_uintptr_t chunk = *(const apr_uintptr_t *)data;
      if (chunk & SVN__BIT_7_SET)
        break;

      /* This is the well-known strlen test: */
      chunk |= (chunk & SVN__LOWER_7BITS_SET) + SVN__LOWER_7BITS_SET;
      if ((chunk & SVN__BIT_7_SET) != SVN__BIT_7_SET)
        break;
    }
#endif
```



```c
/* Fast string data comparison. Caveat: unaligned access to 1st string! */
static LJ_AINLINE int str_fastcmp(const char *a, const char *b, MSize len)
{
  MSize i = 0;
  lua_assert(len > 0);
  lua_assert((((uintptr_t)a+len-1) & (LJ_PAGESIZE-1)) <= LJ_PAGESIZE-4);
  do {  /* Note: innocuous access up to end of string + 3. */
    uint32_t v = lj_getu32(a+i) ^ *(const uint32_t *)(b+i);
    if (v) {
      i -= len;
#if LJ_LE
      return (int32_t)i >= -3 ? (v << (32+(i<<3))) : 1;
#else
      return (int32_t)i >= -3 ? (v >> (32+(i<<3))) : 1;
#endif
    }
    i += 4;
  } while (i < len);
  return 0;
}
```

Reading invalid memory is not correct, even if you don't use what you read.

Such code is "undefined behavior": Can break under different compiler / OS / architecture.

Bugs and false positives with ASAN are extremely rare.

LIBTOOL

When linking shared libraries libtool filters unknown flags from LDFLAGS: Breaks ASAN builds.

Fix upstream (not released), but scripts bundled (ltmain.sh).

Workaround via portage hook.

PTHREAD

libasan provides pthread_create(), but not full pthread API.

configure scripts check for pthread_create(), assume -lpthread not needed.

Breaks...

MORE BUILD SYSTEM ISSUES

Perl: Uses LD_PRELOAD for libperl, uses miniperl to run compilation perl script.

If you LD_PRELOAD an ASAN library you can't run non-ASAN executables: GCC segfaults.

(Still looking for a good workaround the Perl devs might accept)

HOW USEFUL IS ALL THIS?

It finds bugs, that's good.

Usefulness as an exploit mitigation system unclear.

ASAN FOR EXPLOIT MITIGATION

Tor hardened browser already using it.

Prevents all linear buffer overflows and out of bounds reads.

Non-linear out of bounds access might still be exploitable.

Use after free - limited protection.

ASAN might introduce new attack vectors.

ALTERNATIVES

Maybe in the future there will be something like ASAN, but better.

Fixing the bugs ASAN finds now is a good preparation.

Exploit mitigation old: DEP, ASLR, Stack Canaries (old).

New: LLVM Code-Flow Integrity and Safe-Stack.

BONUS SLIDES (IF THERE IS TIME)

ASAN LOGGING

Sometimes applications disable or redirect stderr - you can't see the ASAN error.

ASAN can log errors into files.

```
ASAN_OPTIONS="log_path=/var/log/asan/asan-error"
```

MORE SANITIZERS

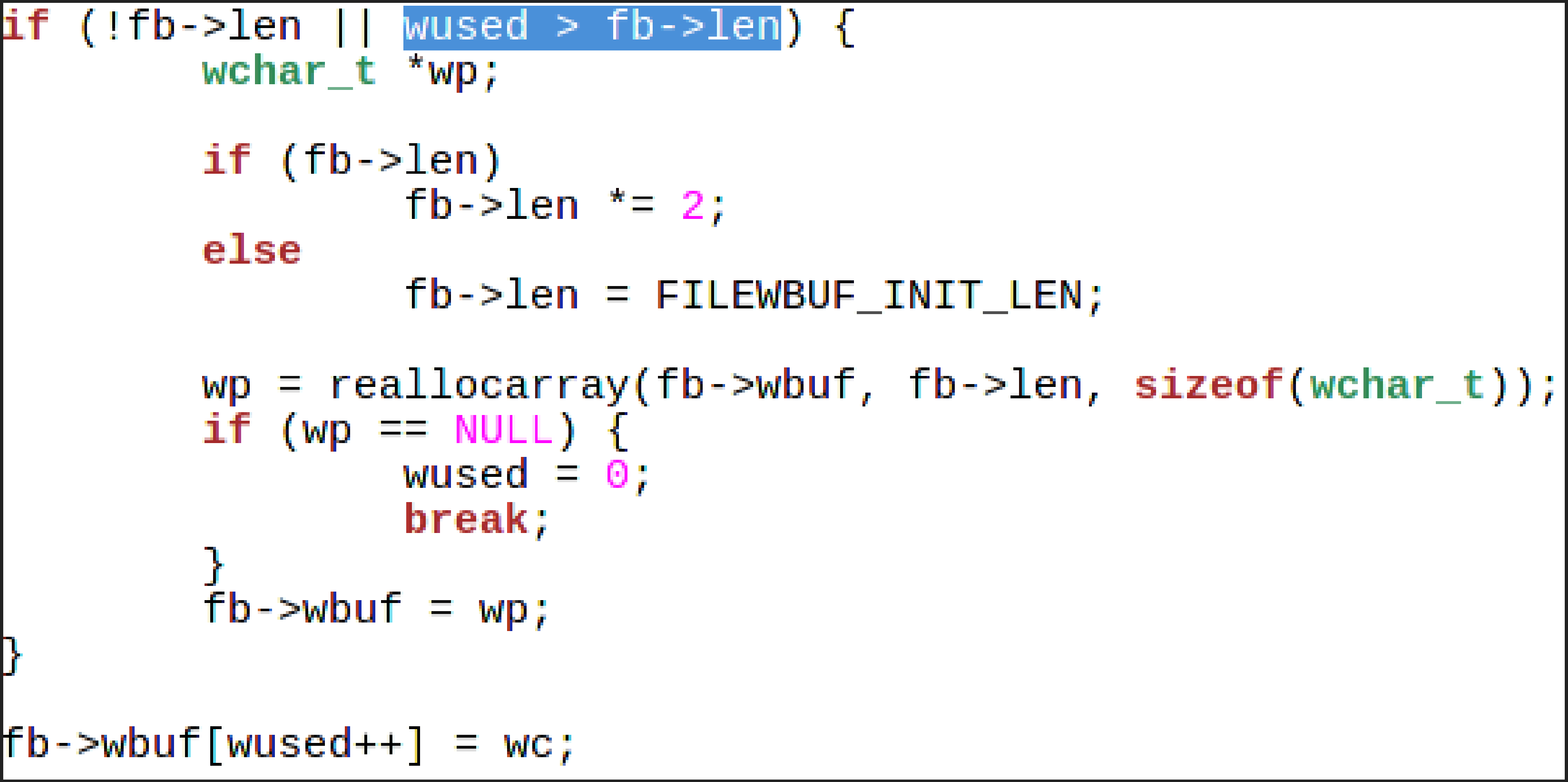
UBSAN: Undefined behavior (things like invalid shifts, signed overflows, unaligned access)

"Problem:" It finds so many things...

TSAN: Thread Sanitizer (race conditions)

MSAM: Memory Sanitizer (uninitialized memory)

BUG EXAMPLE (LIBBSD, CVE-2016-2090)



```
if (!fb->len || wused > fb->len) {
        wchar_t *wp;

        if (fb->len)
                fb->len *= 2;
        else
                fb->len = FILEWBUF_INIT_LEN;

        wp = reallocarray(fb->wbuf, fb->len, sizeof(wchar_t));
        if (wp == NULL) {
                wused = 0;
                break;
        }
        fb->wbuf = wp;
}

fb->wbuf[wused++] = wc;
```

QUESTIONS?

https://wiki.gentoo.org/wiki/AddressSanitizer

https://fuzzing-project.org/