WG 14 Document: N1559

Submitter: Martin Sebor
Submission Date: 2011-03-11
Source:
Reference Document:
Version:
Date: 2011-03-06
Subject: Behavior of realloc() with zero size

Problem Description

The C99 standard specifies two distinct implementation behaviors for a call to the realloc(p, n) function when (p != NULL) and (n == 0) hold:

1) the call fails without attempting to allocate any storage and returns NULL

2) the function attempts to allocate storage and if the allocation is successful returns a pointer to the allocated space; otherwise, it returns NULL

In either case, when the function returns NULL it is required not to free the space pointed to by p. To avoid leaks, programs must test the return value and free the space by a call to free when the return value is NULL.

This is then the sequence of calls a conforming C99 program must make in order to avoid leaking memory:

```
void f(void *p, size_t n)
{
    void *q = realloc(p, n);

    if (q == NULL) {
        free(p);
        return;
    }

    // use q
    free(q);
}
```

The POSIX standard, which is intended to be aligned with C99 in this respect, specifies the same behavior as outlined in (1) and (2) above. However, POSIX inadvertently deviates from C99 in case (1) in that it requires implementations to free the space pointed to by p.

The following is the sequence of calls a conforming POSIX program must make in order to avoid leaking memory and in order to avoid calling free() twice on the same pointer:

```
void f(void *p, size_t n)
{
    errno = 0;
    void *q = realloc(p, n);

    if (q == NULL) {
        if (errno == ENOMEM)
            free(p);
        else {
            // p already freed!
        }

        return;
    }

    // use q
    free(q);
}
```

Existing Practice

Several otherwise conforming C99 implementations of (1) exist that behave according to the POSIX specification but fail to conform to C99 in this respect (i.e., they free the space and return NULL). This non-conformance make writing portable programs difficult. Worse, existing conforming C99 programs that currently run correctly on implementations of (2) are subject to the exploits of the double free vulnerability when ported to the implementations of (1) that follow the POSIX specification.

At least one implementer has acknowledged this non-conformance to C99 but declined a request to change the behavior of the implementation on the grounds that the non-conformance is deliberate and that changing the implementation to conform isn't possible for compatibility reasons (since doing so would introduce memory leaks into programs that rely in the existing behavior). See:
http://sourceware.org/bugzilla/show_bug.cgi?id=12547

Suggested Technical Corrigendum

The following three possible courses of action present themselves:

1. Do nothing and accept the fact that some C99 implementations may not be able to conform to the requirements of the C99 standard in this regard for compatibility reasons.

2. Change C to more closely align with the POSIX requirements so as to make the existing implementatins conforming. This resolution is being sought by the POSIX group and is also

preferred by the implementers of the affected implementations.

3. Change both the C and POSIX standards to require realloc() to always attempt to allocate space even for zero-size requests. This would simplify the specification and, at least in theory, also make it easier to write portable programs. This option would require the support of the implementers of the affected implementations. Without such support this option becomes equivalent to option 1 above.

We suggest adopting option 2.