DragonFlyBSD - Bug #3114

Using malloc(size_max) gives strange results

12/18/2017 09:31 AM - ddegroot

Status:	Closed	**Start date:**	12/18/2017
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Userland	**Estimated time:**	0.00 hour
Target version:	5.0.0		

Description

While porting d-lang dmd/druntime/phobos to DragonFlyBSD, it became apparent that using 'malloc(size_t)' to deduce malloc and alignment rules, gave some unexpected results.

Example:
```
malloc size:9223372036854775807, malloc failed, ptr == NULL, errno:12                    // expected result (INTPTR_MAX)
malloc size:72036854775808,  ptr == 0x800800000                                // this is fine
malloc size:18446744073709551613,  ptr == 0x800455000                           // unexpected UINTPTR_MAX / SIZE_MAX
```

Related dlang:druntime PR: https://github.com/dlang/druntime/pull/1999

History

#1 - 12/20/2017 03:32 PM - ddegroot

I have had a look at the libc/stdlib/malloc.c code, and came to the conclusion that i would not be able to create a patch that would definitively fix this type of malloc/calloc/realloc issue.

An interesting link, regarding other allocators and this exact same issue: http://kqueue.org/blog/2012/03/05/memory-allocator-security-revisited/. This might provide some inspiration and potential test cases.

#2 - 12/30/2017 04:22 PM - ddegroot

- File nmalloc.c.diff added

This seems to resolve the issue. But i am not quite sure if this is the right way.

Notes:
- i am doing the check twice at this moment, only the slaballoc one should be required.
- in _slaballoc size is rewritten and it might be better to use a separate local variable instead. This would help with bug tracing. Changing the size that was requested makes it harder to track what is happening, and prevents later checks if so required. (BTW: there are multiple (other) locations where the requested size is rewritten as well).
- in _slaballoc size is not checked for overflow (i think), before it is rewritten (line 922).

With this patch in place the results resemble the Linux Results exactly.

```
diff --git a/lib/libc/stdlib/nmalloc.c b/lib/libc/stdlib/nmalloc.c
index b39aaf301..d9bc90fb8 100644
--- a/lib/libc/stdlib/nmalloc.c
+++ b/lib/libc/stdlib/nmalloc.c
@@ -753,6 +753,8 @@ zoneindex(size_t *bytes, size_t *chunking)
return(0);
}

+#define MUL_NO_OVERFLOW       (1UL << (sizeof(size_t) * 4))
+
/*
* malloc() - call internal slab allocator
*/
@@ -761,6 +763,11 @@ __malloc(size_t size)
{
```

~~~
void *ptr;

+       if ((size >= MUL_NO_OVERFLOW ) || (SIZE_MAX < size)) {
+               errno = ENOMEM;
+               return(NULL);
+       }
+
ptr = _slaballoc(size, 0);
if (ptr == NULL)
errno = ENOMEM;
@@ -769,8 +776,6 @@ __malloc(size_t size)
return(ptr);
}

-#define MUL_NO_OVERFLOW        (1UL << (sizeof(size_t) * 4))
-
/*
* calloc() - call internal slab allocator
*/
@@ -982,6 +987,9 @@ _slaballoc(size_t size, int flags)
bigalloc_t big;
bigalloc_t *bigp;

+               if ((size >= MUL_NO_OVERFLOW ) || (SIZE_MAX < size) ) {
+                       return(NULL);
+               }
/*
* Page-align and cache-color in case of virtually indexed
* physically tagged L1 caches (aka SandyBridge).  No sweat
@@ -989,7 +997,8 @@ _slaballoc(size_t size, int flags)
*
* (don't count as excess).
*/
-              size = (size + PAGE_MASK) & ~(size_t)PAGE_MASK;
+               size = (size + PAGE_MASK) & ~(size_t)PAGE_MASK;         /* Note: Changing size, without checking overflow.
+                                                                 also might be better to use a different variable instead of the original request size */

/*
* If we have overflown above when rounding to the page
```
~~~

#3 - 12/31/2017 04:14 AM - ddegroot

- File test_malloc.c added

Update test_malloc.c

#4 - 12/31/2017 04:27 AM - ddegroot

- Status changed from New to Closed

Overflow issue was already fixed by zrj in master.
Results do not match 100% with what linux does, but that should be ok.

Files

test_malloc.c	1.01 KB	12/18/2017	ddegroot
test_malloc_results.txt	1.97 KB	12/18/2017	ddegroot
nmalloc.c.diff	1.81 KB	12/31/2017	ddegroot
test_malloc.c	1.11 KB	12/31/2017	ddegroot