

Forum:[Hack Forums](#)

Thread: [\[C\] heap block while string manipulation with realloc \(noobie\)](#)

lockhead - December 27, 2018, 6:36 am

Hi, I was doing some basic testing regarding string manipulation/run-time resizing and ran into this weird issue.

The program runs fine via the console but I get this weird message when stepping through it with gdb and just can't figure out what the hell is going on:

```
Code #include "stdlib.h"
#include "stdio.h"
#include "string.h"
char *foo(){
    char *t = malloc(sizeof(char));
    int i = 0;
    while(i<5){
        t[i] = 'a';
        t = (char *)realloc(t, sizeof(char));
        i++;
    }
    t[i] = 0;
    return t;
}

int main(int argc, char const *argv[])
{
    char *c;
    c = foo();
    printf("%s\n", c);
    //free(c);
    return 0;
}
```

Here's the gdb output:

As you can see, I encounter this issue while calling realloc for the second time, so t="aa"

```
Code Breakpoint 1, foo () at testing.c:5
5      char *t = malloc(sizeof(char));
(gdb) display t
1: t = 0x61ffcc "äÿa"
(gdb) display i
2: i = 6422296
(gdb) display t+i
3: t+i = 0xc3fee4 <Address 0xc3fee4 out of bounds>
(gdb) display t[i]
4: t[i] = <error: Cannot access memory at address 0xc3fee4>
(gdb) next
6      int i = 0;
4: t[i] = <error: Cannot access memory at address 0x1042998>
```

```

3: t+i = 0x1042998 <Address 0x1042998 out of bounds>
2: i = 6422296
1: t = 0xa22a80 "\230««««««««««pîpîpîp"
(gdb) next
7      while(i<5){
4: t[i] = -104 '\230'
3: t+i = 0xa22a80 "\230««««««««««pîpîpîp"
2: i = 0
1: t = 0xa22a80 "\230««««««««««pîpîpîp"
(gdb) next
8      t[i] = 'a';
4: t[i] = -104 '\230'
3: t+i = 0xa22a80 "\230««««««««««pîpîpîp"
2: i = 0
1: t = 0xa22a80 "\230««««««««««pîpîpîp"
(gdb) next
9      t = (char *)realloc(t, sizeof(char));
4: t[i] = 97 'a'
3: t+i = 0xa22a80 "a««««««««««pîpîpîp"
2: i = 0
1: t = 0xa22a80 "a««««««««««pîpîpîp"
(gdb) next
10     i++;
4: t[i] = 97 'a'
3: t+i = 0xa22a80 "a««««««««««pîpîpîp"
2: i = 0
1: t = 0xa22a80 "a««««««««««pîpîpîp"
(gdb) next
7      while(i<5){
4: t[i] = -85 '«'
3: t+i = 0xa22a81 "««««««««««pîpîpîp"
2: i = 1
1: t = 0xa22a80 "a««««««««««pîpîpîp"
(gdb) next
8      t[i] = 'a';
4: t[i] = -85 '«'
3: t+i = 0xa22a81 "««««««««««pîpîpîp"
2: i = 1
1: t = 0xa22a80 "a««««««««««pîpîpîp"
(gdb) next
9      t = (char *)realloc(t, sizeof(char));
4: t[i] = 97 'a'
3: t+i = 0xa22a81 "a««««««««««pîpîpîp"
2: i = 1
1: t = 0xa22a80 "aa««««««««««pîpîpîp"
(gdb) next
warning: HEAP[testing.exe]:
warning: Heap block at 00A22A78 modified at 00A22A81 past requested size of 1

```

Program received signal SIGTRAP, Trace/breakpoint trap.

0x775c1c40 in ?? ()

I'm running this on a 64bit Windows machine. Any help for a helpless idiot like me would be greatly

appreciated! ?

incognito04 - December 27, 2018, 7:36 am

You allocate a byte of memory and attempt to write outside of it's boundaries.

Dem?n - December 27, 2018, 9:00 am

You've probably already solved the issue but hint, it's this line ->

```
t = (char *)realloc(t, sizeof(char));
```

lockhead - December 27, 2018, 10:20 am

(Dec 27, 2018 08:36 AM)incognito04 Wrote: [\[CITE\]](#) You allocate a byte of memory and attempt to write outside of it's boundaries.

No I'm not (at least i think). I'm writing to then, then realloc another byte to write to later.

This shouldn't go out of bounds imho.

(Dec 27, 2018 10:00 AM)Dem?n Wrote: [\[CITE\]](#) You've probably already solved the issue but hint, it's this line ->

```
t = (char *)realloc(t, sizeof(char));
```

Nope, I didn't ?

I figured it was this line, but I can't find what's wrong with it. I'm just allocating another byte to t, to write to later.

I don't see how this is causing issues or going out of bounds...

incognito04 - December 27, 2018, 11:04 am

(Dec 27, 2018 11:20 AM)lockhead Wrote: [\[CITE\]](#) No I'm not (at least i think). I'm writing to then, then realloc another byte to write to later.

This shouldn't go out of bounds imho.

Nope, I didn't ?

I figured it was this line, but I can't find what's wrong with it. I'm just allocating another byte to t, to write to later.

I don't see how this is causing issues or going out of bounds...

Yes you are, when you're calling realloc, your reallocating a byte of memory at the same address malloc was called

lockhead - December 27, 2018, 11:40 am

(Dec 27, 2018 12:04 PM)incognito04 Wrote: [\[CITE\]](#) Yes you are, when you're calling realloc, your reallocating a byte of memory at the same address malloc was called

Isn't that how it's suppose to be? Also done seen on tutorialspoint. Or am I going crazy...

Code `#include <stdio.h>`

`#include <stdlib.h>`

```
int main () {  
    char *str;
```

```
    /* Initial memory allocation */
```

```
    str = (char *) malloc(15);
```

```
    strcpy(str, "tutorialspoint");
```

```
    printf("String = %s, Address = %u\n", str, str);
```

```

/* Reallocating memory */
str = (char *) realloc(str, 25);
strcat(str, ".com");
printf("String = %s, Address = %u\n", str, str);

free(str);

return(0);
}

```

Would you mind posting the correct code?

0xDEAD10CC - December 27, 2018, 12:52 pm

Why would you realloc at all when you know there's 5 iterations? I see no point in reducing the performance of this code to allocate 1 byte on the heap and attempt to increase the allocation byte by byte (even though that's not what this code is doing). What's wrong with the stack here? Allocate 6 bytes and iterate through index 0-4 for the values and 5 for the null terminator if you didn't zero-initialize it or memset to begin with.

"I don't see how this is causing issues or going out of bounds..." - maybe you need to read the docs for realloc() more carefully.

You're changing the allocation size to 1/sizeof(char) every single time, not increasing it by one...

lockhead - December 27, 2018, 1:23 pm

(Dec 27, 2018 01:52 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Why would you realloc at all when you know there's 5 iterations? I see no point in reducing the performance of this code to allocate 1 byte on the heap and attempt to increase the allocation byte by byte (even though that's not what this code is doing). What's wrong with the stack here? Allocate 6 bytes and iterate through index 0-4 for the values and 5 for the null terminator if you didn't zero-initialize it or memset to begin with.

"I don't see how this is causing issues or going out of bounds..." - maybe you need to read the docs for realloc() more carefully.

You're changing the allocation size to 1/sizeof(char) every single time, not increasing it by one...

It's not about performance, this is test code. It's suppose to read a string from file of an unspecified length.

Nah it wasn't a docs issue, it was me not being able to think. I finally realized the issue was that I wasn't increasing the size and kept reallocating 1 byte, not +1.

Here's the result

```

Code #include "stdlib.h"
#include "stdio.h"
#include "string.h"
char *foo(){
    char *t = malloc(sizeof(char));
    int i = 0;
    int size = 2;
    while(i<5){
        t[i] = 'a';
        t = (char *)realloc(t, sizeof(char) * size);
        i++;
        size++;
    }
}

```

```
t[i] = 0;
return t;
}
```

```
int main(int argc, char const *argv[])
{
    char *c;
    c = foo();
    printf("%s\n", c);
    //free(c);
    return 0;
}
```

0xDEAD10CC - December 27, 2018, 2:13 pm

(Dec 27, 2018 02:23 PM)lockhead Wrote: [\[CITE\]](#) It's not about performance, this is test code. It's suppose to read a string from file of an unspecified length.

Nah it wasn't a docs issue, it was me not being able to think. I finally realized the issue was that I wasn't increasing the size and kept reallocating 1 byte, not +1.

Here's the result

```
Code #include "stdlib.h"
#include "stdio.h"
#include "string.h"
char *foo(){
    char *t = malloc(sizeof(char));
    int i = 0;
    int size = 2;
    while(i<5){
        t[i] = 'a';
        t = (char *)realloc(t, sizeof(char) * size);
        i++;
        size++;
    }
    t[i] = 0;
    return t;
}
```

```
int main(int argc, char const *argv[])
{
    char *c;
    c = foo();
    printf("%s\n", c);
    //free(c);
    return 0;
}
```

It will be a performance issue if you read a full file like this; reallocations are expensive.... Also there is a function `ftell()`, and you should be reading the file in buffered chunks most of the time.

And why are you using quotes over standard angle brackets for standard headers?

Also `sizeof(char) == 1` always.

lockhead - December 28, 2018, 6:22 am

(Dec 27, 2018 03:13 PM)0xDEAD10CC Wrote: [\[CITE\]](#) It will be a performance issue if you read a full file like this; reallocations are expensive.... Also there is a function `ftell()`, and you should be reading the file in buffered chunks most of the time.

And why are you using quotes over standard angle brackets for standard headers?

Also `sizeof(char) == 1` always.

Thank you for the help! I'll note and fix what you pointed out