



## How to prevent memcpy buffer overflow?

[Ask Question](#)


7



3

There are some binary buffer with fixed size in a program that are used to store data. And memcpy is used to copy the buffer from one to another one. Since the source buffer may be larger than the destination buffer. How can I detect if there is buffer overflow?

c

memcpy

buffer-overflow

fortify-source

edited Sep 12 '14 at 16:06



jww


52.7k ● 39 ● 222 ● 483

asked Aug 31 '12 at 7:06



Michael D

674 ● 2 ● 10 ● 27

- 1 Detect? You do know destination buffer size? Then write code like this `memcpy(src, dst, sizeof(dst))` – [BSen](#) Aug 31 '12 at 7:07 

Compare the size of the source buffer and the destination buffer and see which is larger? – [SingerOfTheFall](#) Aug 31 '12 at 7:07

- 1 @BSen that `sizeof` will just give the size of a pointer. – [juanchopanza](#) Aug 31 '12 at 7:11

Treat it as pseudocode. It can give size of a buffer depending on declaration style. I do realize, that `sizeof(dst *)` gives pointer size and `sizeof(dst [123])` gives buffer size, at least when 1 element = 1 byte. – [BSen](#) Aug 31 '12 at 7:14

## 4 Answers



8



You have to know how much data is in the source buffer and how much space is available in the target buffer.

Do not call `memcpy()` if there is not enough space in the target buffer for all the data you want to copy from the source buffer. (You have to decide whether it is OK to truncate the data if the source is bigger than the target.)

If you don't know, rewrite the code so that you do know how much space there is; otherwise, it is not safe.

Note that if there is a chance of the source and target buffers overlapping, you should use `memmove()` rather than `memcpy()`.

In C++, look askance at using `memcpy()` in the first place; that is a C-style operation rather than C++.

answered Aug 31 '12 at 7:09



[Jonathan Leffler](#)

560k ● 89 ● 665 ● 1017

Thanks. what is the proper way to do the memory copy in C++? –

[Michael D](#) Aug 31 '12 at 7:51



- 1 [@MichaelD](#): Store your data in a `std::vector<>`, and just use `vector2 = vector1`. – [MSalters](#)  
Aug 31 '12 at 7:56

How can I insert data into vector? use `push_back` to insert date byte by byte? – [Michael D](#) Aug 31 '12 at 8:04

If the binary data is from network  
e.g. tcp, how to insert to vector  
without sacrificing much  
performance? – [Michael D](#) Aug  
31 '12 at 8:13

Make your next move  
with a career site that's by developers

5

How can I detect if there is  
buffer overflow?

I think you have three or four  
choices (give or take).

The first choice is to provide a  
"safe" function for `memcpy`. This is  
what I require in code under my  
purview, and I regularly audit for  
it. I also require all parameters are  
validated, and all parameters are  
asserted.

The assertions create self  
debugging code. I want  
developers to write code; and I  
don't want them to waste time  
debugging. So I require them to  
write code that debugs itself.  
ASSERTs also documents things  
rather well, so they can skip on  
the documentation. In release  
builds, the ASSERTs are removed  
by preprocessor macros.

```
errno_t safe_memcpy(void* dest, si
{
    ASSERT(dest != NULL);
    ASSERT(src != NULL);
    ASSERT(dsize != 0);
    ASSERT(ssize != 0);
    ASSERT(cnt != 0);

    // What was the point of this
    if(cnt == 0)
        retrn 0;

    if(dest == NULL || src == NULL
        return EINVAL;
```

```

ASSERT(ssize <= RSIZE_MAX);
ASSERT(cnt <= RSIZE_MAX);

if(dsize > RSIZE_MAX || ssize
    return EINVAL;

size_t cc = min(min(dsize, ssi
memmove(dest, src, cc);

if(cc != cnt)
    return ETRUNCATE;

return 0;
}

```

If your `safe_memcpy` returns non-0, then there was an error like a bad parameter or potential buffer overflow.

The second choice is to use "safer" functions provided by the C Standard. C has "safer" functions via [ISO/IEC TR 24731-1, Bounds Checking Interfaces](#). On conforming platforms, you can simply call `gets_s` and `sprintf_s`. They offer consistent behavior (like always ensuring a string is `NULL` terminated) and consistent return values (like 0 on success or an `errno_t`).

```

errno_t err = memcpy_s(dest, dsiz
...

```

Unfortunately, gcc and glibc does not conform to the C Standard. Ulrich Drepper (one of the glibc maintainers) called bounds checking interfaces "[horribly inefficient BSD crap](#)", and they were never added.

The third choice is to use the platform's "safer" interfaces, if present. On Windows, that happens to be the same as those in [ISO/IEC TR 24731-1, Bounds Checking Interfaces](#). You also have the String Safe library.

On Apple and BSD, you have don't have a "safer" function for

On Linux, your fourth choice is to use FORTIFY\_SOURCE. FORTIFY\_SOURCE uses "safer" variants of high risk functions like `memcpy`, `strcpy` and `gets`. The compiler uses the safer variants when it can deduce the destination buffer size. If the copy would exceed the destination buffer size, then the program calls `abort()`. If the compiler cannot deduce the destination buffer size, then the "safer" variants are not used.

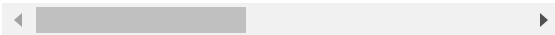
To disable FORTIFY\_SOURCE for testing, you should compile the program with `-U_FORTIFY_SOURCE` or `-D_FORTIFY_SOURCE=0`.

answered Sep 12 '14 at 16:12



jww

52.7k ● 39 ● 222 ● 483



You should always know and check the src and dest buffers size !



```
void *memcpy(void *dest, const voi
```

```
n should never be greater than  
src OR dest size.
```

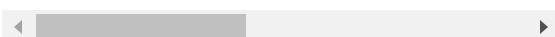
edited Dec 19 '18 at 14:06

answered Aug 31 '12 at 7:08



phsym

1,154 ● 9 ● 17



If for example you have:



destination 4 bytes size

source 5 bytes size

buffer:

```
size_t getCopySize(size_t sourceSi
{
    return (destSize <= sourceSize
}
memcpy(destination, source, getCop
```

Basing on your application you could also make sure that the remaining data will be copied at a later time, or you can skip it if some data can be ignored.

answered Aug 31 '12 at 7:27



Giordano

261 ● 2 ● 5

