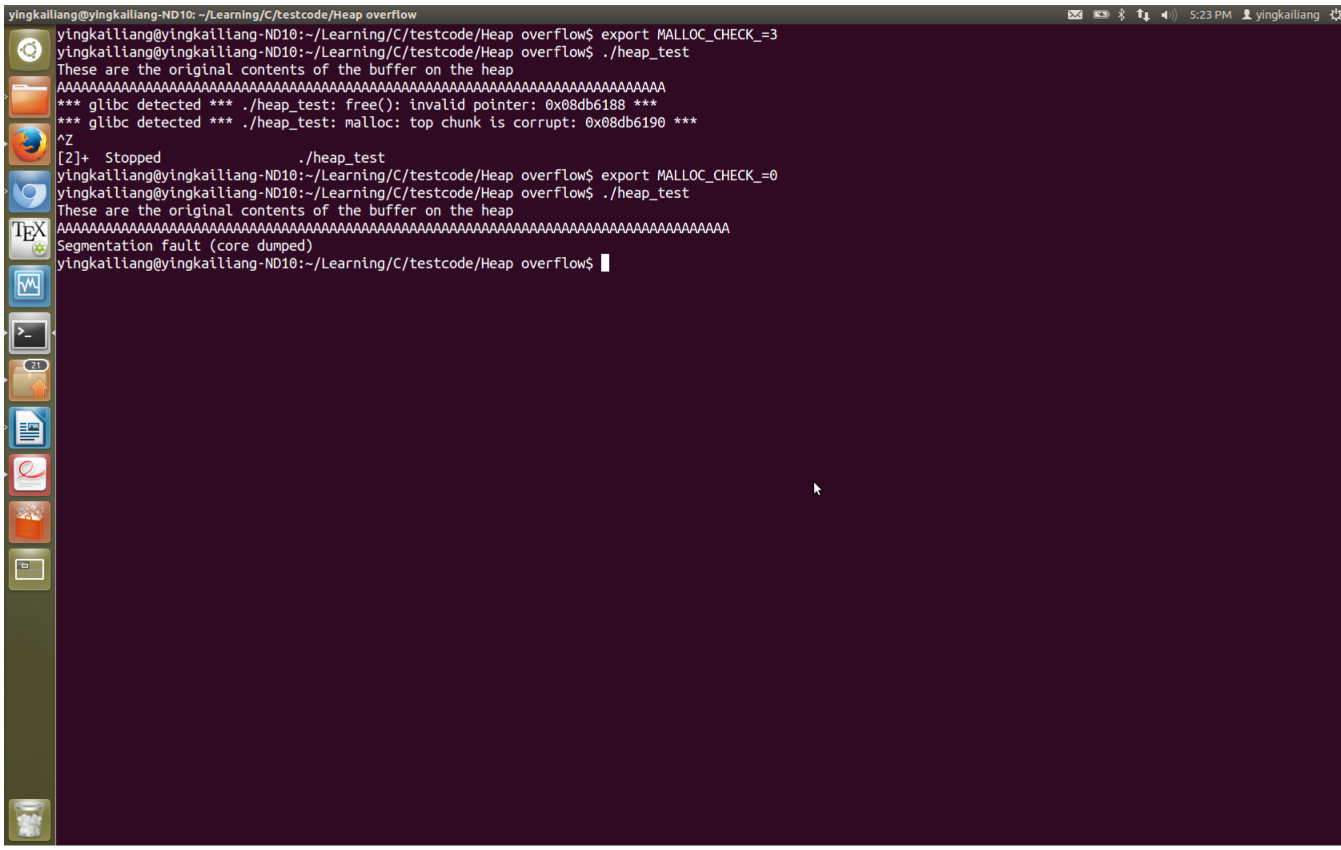
# 1 Heap Buffer overflow

## 1.1 Protection

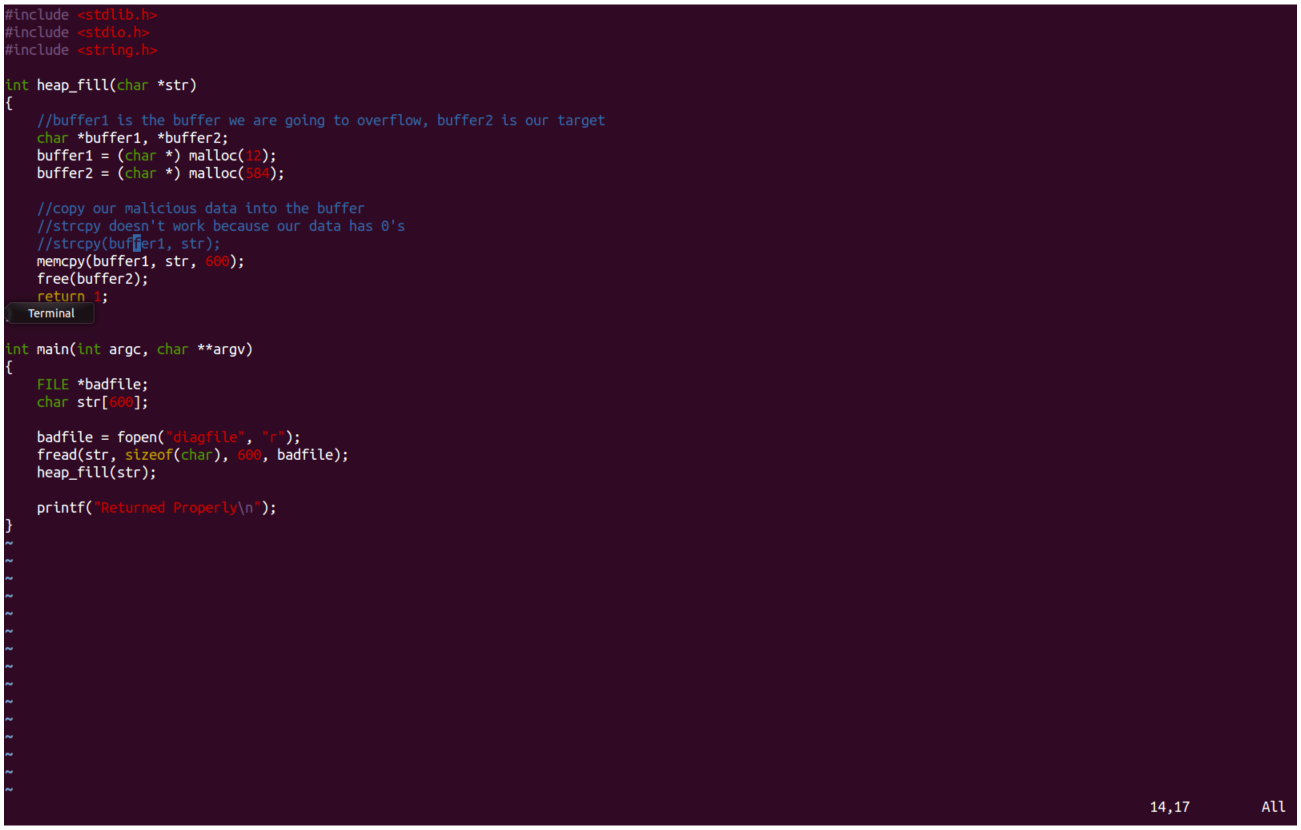
- memory randomize
- stack guard
- no excutable stack (optional)
- malloc check



```
yingkailiang@yingkailiang-ND10: ~/Learning/C/testcode/Heap overflow
```

```
yingkailiang@yingkailiang-ND10:~/Learning/C/testcode/Heap overflow$ export MALLOC_CHECK_=3
yingkailiang@yingkailiang-ND10:~/Learning/C/testcode/Heap overflow$ ./heap_test
These are the original contents of the buffer on the heap
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
*** glibc detected *** ./heap_test: free(): invalid pointer: 0x08db6188 ***
*** glibc detected *** ./heap_test: malloc: top chunk is corrupt: 0x08db6190 ***
^Z
[2]+  Stopped                 ./heap_test
yingkailiang@yingkailiang-ND10:~/Learning/C/testcode/Heap overflow$ export MALLOC_CHECK_=0
yingkailiang@yingkailiang-ND10:~/Learning/C/testcode/Heap overflow$ ./heap_test
These are the original contents of the buffer on the heap
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault (core dumped)
yingkailiang@yingkailiang-ND10:~/Learning/C/testcode/Heap overflow$ 
```

## 1.2 vulnerable program

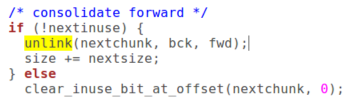


```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int heap_fill(char *str)
{
    //buffer1 is the buffer we are going to overflow, buffer2 is our target
    char *buffer1, *buffer2;
    buffer1 = (char *) malloc(12);
    buffer2 = (char *) malloc(584);

    //copy our malicious data into the buffer
    //strcpy doesn't work because our data has 0's
    //strcpy(buffer1, str);
    memcpy(buffer1, str, 600);
    free(buffer2);
    return 1;

int main(int argc, char **argv)
{
    FILE *badfile;
    char str[600];

    badfile = fopen("diagfile", "r");
    fread(str, sizeof(char), 600, badfile);
    heap_fill(str);

    printf("Returned Properly\n");
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                                14,17         All
```

## 1.3 hypothetical attack

Create a fake free chunk,after the chunk we which we need to free. I have fully control of what ”fake free chunk” look at.

The idea is based on glibc malloc.c source code free() function.



```
/* consolidate forward */
if (!nextinuse) {
  unlink(nextchunk, bck, fwd);
  size += nextsize;
} else
  clear_inuse_bit_at_offset(nextchunk, 0);
```

How to create this fake chunk: