**More on Memory**

**Writeable Memory**

* When an OS or an application starts, there are sections of the memory that you are freely able to write too
  + Often the stack or the heap
  + Normally identified per application when its compiled
* When creating exploits, understanding wta is marked as writeable space and non-wrteble space is import
  + E.g., doing a buffer overflow into a region of memory that isn’t writeable, then you wont be able to do much with it

**Non-writeable memory**

* Read only regions
* Normally identified once the app is compiled
  + Parts of the memory that the program now wants to keep safe
* This is to keep core functions running and to help protect against exploits
* If you attempt to write to a non-writeable memory, you will receive this error
  + ATTEMPTED TO WRITE TO READONLY MEMORY

**Executable Memory**

* Being able to run instructions stored at a memory addr makes it executable
  + When EIP jumps through memory addr and executes what’s in there, those memory addrs have to be executable
* Most common in the .txt and .code segment of binary
  + As code has to be executable
* Useful for exploitation as it makes building an exploit easier

**Non-Executable Memory**

* Normally done by something called Executabe Space Protextion (ESP)
  + DEP or Data Execution Protection on Windows
* This marks a specific region of memory addrs as non-executable
  + So, if we were to overflow into this region due to vulnerability, we could not execute code – this is called depth
  + There are ways to bypass this