Binary Exploitation

- Buffer Overflow
  - Crash the application
  - Set variable values, to alter program behavior, maybe bypassing controls (licence checks)
  - Jump to any instruction of the program, maybe bypassing controls
  - Execute injected shellcode to open a shell on the remote host (crash the server, create user and pwd, read /etc/passwd, gain root privilege setuid(0), privilege escalate setreuid(0), flush iptables db to turn off firewall..). First put on EAX (first parameter) the syscall to execute, then can INT80 (\xcd\x80)
- Bypassing countermeasures
  - Return Oriented Programming: don't inject code, only jump to instructions of the program that we want to execute (gadgets). Using pivoting technique we can use gadgets to change the value of the stack pointer to use different parts of the stack
    - Possible pitfalls
      - Needed gadgets not available
      - Number of needed gadgets may be too high and may not fit in the space available for the buffer overflow
      - There is no guarantee that any code can be written combining the gadgets available
  - Ret2libc: If we can find the addresses of the libc functions that we want to use, we can use a simple ROP to load parameters and prepare return space, and then use libc functions to our favor
- Countermeasures
  - Data Execution Prevention DEP: can't run code from Writable segments. Bypassable using ROP or Ret2libc
    - Data segments RW: stack, heap, .bss, .ro, .data
    - Code segments RX: .text, .plt
  - Stack canary: random value, different for each execution, same for all the functions called by the application, that are added on the stack between function data and return address by the OS after each call, and are checked at every function exit. Overwriting the return address will kill the canary if we didn't use another vulnerability to find the canary location and substitute the right value
  - Address Space Layout Randomization ASLR: randomize the starting memory location where the segments are loaded so that attackers cannot use fixed addresses found with gdb but must use offset+finding the base addresses using another vulnerability