Questions

1. Which capabilities API (seccomp-bpf, AppArmor, or SELinux) did you choose? Why did you make that choice?
   - I chose seccomp-bpf for my assignment as it is quick and easy to understand. I used ‘strace’ to trace the system calls used by the programs. The output of ‘strace’ helps in quick coding with seccomp-bpf

2. What was the process you used to ascertain the list of system calls required by each program?
   - I used ‘strace’ to trace system calls and signals.

3. What system calls are needed by each?
   - **server.c** - rt_sigreturn, exit, read, write, access, fstat, mmap, close, arch_prctl, mprotect, munmap, socket, connect, sendto, getpid, getuid, setsockopt, bind, listen, clone, wait4, openat, accept, execve, setuid, chroot, mkdir, chdir, getcwd, dup2, sleek, fcntl, exit_group, brk, stat, dup
   - **client.c** - rt_sigreturn, exit, read, write, access, fstat, mmap, close, arch_prctl, mprotect, munmap, socket, connect, sendto, openat, prctl, exit_group, brk

4. What happens when your application calls the prohibited system call? What is the application behavior that results from the call?
   - We get message - “ killed by SIGSYS (core dumped)” which indicates that a prohibited system call function has been called and the application terminates immediately. When there is a prohibited system call inside child, if we just use ‘strace’ for server, we can see that the server.c is completed without going to child.c. Inorder to understand error in child.c, we have to use ‘strace -f’.