Dve kategoriji:

- 1. TOU/TOC:
  - TOC - preverimo, da imamo dostop do nekega vira
  - vmes je delta časa
  - sledi TOU - uporabi se isti sklop v memory (isti podatek), ki je od TOC lahko bil spremenjen
  - primer z DNS:
    - v vmesnem času lahko zamenjamo IP, na katerega domena kaže in se bo request naredil na npr. interen IP (tisto kar smo želeli preprečiti)
  - problematično v kernelih, npr. pri sudo je bil podoben primer
- 2. side channel:
  - to je malo prekompleksno za vaje

time_0:

- `docker build -t test .`
- `docker run -d --rm -p 1337:1337 test`
- `resolve` iz inputtanega niza dobi dejanski končni file
- če damo noter `../../../../../../../bin/bash` nam bo resolvalo to pot in ugotovilo, da gre za `/bin` folder => ne moremo kar tako traversati
- lahko naredimo symlink do `/tmp/test`
- če pred začetkom download.py naredimo symlink, se bo resolvalo in bo videlo, da želimo pisati v prepovedan folder
- zato moramo najprej vpisati `test.txt`, potem naredimo datoteko `test.txt` in symlink `ln -s /tmp/test test.txt` => ko bomo vpisali URL (`http://google.com`), se bo vsebina v bistvu pisala v `/tmp/test`; ko pridemo do TOC se pot resolva v `home/tim/.../test.txt`, kar ni v deny seznamu
- rešitev: najprej odpremo file handler in nad njim delamo preverjanja - datoteka se ne more spreminjati dokler ne releasamo file handlerja in ne bomo morali vmes delati symlinkov

time_1:

- poslušamo, da user naredi connection
- vsak connection teče v svojem threadu
- če receive faila, se connection zapre in thread izbriše
- deli programa, ki so shared med nitmi:
  - BSS (statične spremenljivke/globalne spremenljivke)

    - glaven del heapa je sharan, en del heapa je thread local, da bolje dela memory management
- za vsak thread je svoj stack (stack ni shared), ker ima vsaka nit svoj stack trace, ki ga mora držati
- `nc localhost 1337` da naredimo connection v terminalu
- funkcija `vuln` bo vrnila dolžino sporočila, ki ga dobimo nazaj - če je ta dolžina enaka 0 ali negativna (prišlo je do napake), se bo serve funkcija končala (skoči ven iz do..while zanke) in thread se bo izbrisal
- primer privilege escalation zaradi dirty bita - lahko dostopamo do nekih datotek, do katerih nimamo zares dostopa
- rešitev:
    - `buf` ne bi bil globalen, ampak ga pošiljamo kot parameter, da ga imamo na stacku; lahko bi ga dali tudi na heap; user input (oz. stvari, ki jih rabiš) si skopiraš nekam k sebi, kjer se ne more spremeniti (skopiraš na stack ali heap)
    - lahko bi rešili z mutex zaklepanjem (locking)