Nume și grupă:

Sisteme de Operare

30 august 2016

Timp de lucru: 90 de minute

Notă: Toate răspunsurile trebuie justificate

1. (**7 puncte**) În urma rulării comenzii `stat` pe un fișier obținem că dimensiunea acestuia este `1GB`, dar numărul de blocuri ocupate este `0`. Cum explicați?

2. (**7 puncte**) Prioritățile statice nu pot fi schimbate pe parcursul rulării unui proces. Care este problema cu un planificator care folosește **doar** priorități statice?

3. (**7 puncte**) *Thrashing*-ul este un fenomen negativ al sistemului de memorie virtuală în care se fac schimburi continue între memoria principală și disc (*swap in* și *swap out*). Descrieți un scenariu care duce la apariția *thrashing*-ului.

4. (**7 puncte**) De ce un proces într-un sistem de operare modern nu va dispune niciodată de mai multe pagini fizice decât pagini virtuale?

5. (**7 puncte**) De ce putem afirma că paginarea ierarhică este un compromis spațiu-timp (*space time tradeoff*)?

6. (**10 puncte**) De ce este indicat să folosim capabilități în loc de *setuid* pentru executabile care au nevoie de privilegii?

7. (**10 puncte**) De ce unele sisteme pe 32 de biți care folosesc *memory mapped I/O* pot avea maxim `3GB` de memorie RAM (numit și *3GB barrier*) în loc de maxim `4GB`?

8. (**10 puncte**) De ce mecanismele de sincronizare sunt mai complexe și mult mai frecvent folosite în codul care rulează în *kernel mode* față de codul care rulează în *user mode*?

9. (**10 puncte**) În ce situație apelul `send(sockfd, buffer, 1000)` întoarce o valoare cuprinsă între `0` și `1000` (excluzând `0` și `1000`)?

10. (**10 puncte**) Pe un sistem dat un proces poate dispune de un număr maxim de thread-uri. Cum putem crește această limitare?

11. (**25 puncte**) Vi se cere să proiectați și să implementați o aplicație de *reverse engineering* și *binary analysis* pentru executabile, care să fie capabilă atât de analiză statică cât și de analiză dinamică. Se presupune că nu aveți acces la codul sursă și că doriți să aflați cât mai rapid informații legate de executabil și de modul său de funcționare. Urmăriți acțiuni precum dezasamblare, construire de graf de flux de control (*control flow graph*), investigarea apelurilor de sistem, de bibliotecă, acoperirea codului (*coverage*), zone de cod *hot* și zone *cold* (apelate des, respectiv rar).

a. Propuneți o arhitectură de principiu a aplicației (ce componente va avea și cum se vor conecta între ele). (**5 puncte**)
b. Ce funcționalități trebuie să ofere sistemul de operare pentru a putea implementa cerințele aplicației? (**7 puncte**)
c. Cum ați implementa în cadrul aplicației descoperirea zonelor de cod *hot* și *cold*? (**6 puncte**)
d. Cum ați folosi aplicația pentru descoperirea vulnerabilităților de memorie care ar putea conduce la execuția de cod arbitrar? (**7 puncte**)

În conformitate cu ghidul de etică al Departamentului de Calculatoare, declar că nu am copiat și nu voi copia la această lucrare. De asemenea, nu am ajutat și nu voi ajuta pe nimeni să copieze la această lucrare.

Nume și grupă:

Semnătură:.....