Nume și grupă:

Sisteme de Operare

3 iunie 2017

Timp de lucru: 90 de minute

Notă: Toate răspunsurile trebuie justificate

1. (**7 puncte**) Ce conține tabela de descriptori de fișiere a unui proces? (adică ce se găsește în fiecare intrare din tabela de descriptori de fișiere)

2. (**7 puncte**) Pe un sistem cu un singur core rulează la un moment dat procesul idle (*system idle process*). Câte procese se găsesc în coada/cozile `READY` și `WAITING` în acel moment?

3. (**7 puncte**) Un sistem de fișiere Unix conține următoarele zone: `superblock`, `imap` (bitmap cu ocuparea *inode*-urilor), `dmap` (bitmap cu ocuparea blocurilor de date), `izone` (*inode*-urile), `dzone` (blocurile de date). Unde se găsesc numele intrărilor în sistemul de fișiere (*dentry*-urile)?

4. (**7 puncte**) Fie secvența de mai jos în care ambele apeluri se realizează cu succes:

```
execve(args[0], args, NULL);
fork();
```

De ce există un **singur** proces care rulează această secvență?

5. (**7 puncte**) În ce situație se blochează apelul `recv(sock, buffer, 1000, 0)`?

6. (**10 puncte**) O structură de tip buffer circular este definită de structura de mai jos:

```
struct ring_buffer {
        elem_t buffer[BUFSIZ];
        size_t get_idx;
        size_t put_idx;
        size_t num_elems;
        /* ... */
}
```

Operațiile pe buffer-ul circular sunt `put()` și `get()` sau `produce()` și `consume()`. Pentru sincronizarea accesului se folosesc variabile condiție și mutex-uri, definite în zona marcată cu `/* ... */` în cadrul structurii. Câte variabile condiție și mutex-uri sunt necesare pentru folosirea corectă, sincronizată a buffer-ului circular?

7. (**10 puncte**) Un programator vrea să măsoare cât mai precis timpul de rulare a unei funcții. Pentru aceasta configurează sistemul astfel încât programul/procesul în cauză să nu folosească spațiul de swap și, în timpul rulării funcției, să fie dezactivată preemptivitatea. De ce alege programatorul să facă aceste două acțiuni?

8. (**10 puncte**) Fie următoarea funcție de thread, rulată la crearea thread-ului:

```
void *th_func(void *arg)
{
        char *p;
        char v;

        p = &v;
```

```
        * (p + 8 * 1024 * 1024) = 'a';
        * (p - 8 * 1024 * 1024) = 'b';
        ...
}
```

În secvența de mai sus nici una dintre operațiile de dereferențiere nu cauzează eroare de acces la memorie (*segmentation fault*). Cum explicați?

9. (**10 puncte**) Un atacator creează payload-ul de mai jos (un input pentru un atac) care exploatează o vulnerabilitate de tipul buffer overflow:

```
payload = 32 * "A" + byte_string_address_of_system + byte_string_address_of_exit +
               + byte_string_address_of_bin_sh
```

Explicați conținutul payload-ului și succesiunea părților în conținut.

10. (**10 puncte**) vDSO (*virtual dynamic shared object*) este o mini bibliotecă mapată în Linux în spațiul de adresă *user space* al tuturor proceselor. Este folosită pentru a reduce overhead-ul de apel de sistem, kernel-ul plasând informații direct în spațiul de adresă al procesului fără a mai fi nevoie de tranziția *user space* - *kernel space*. vDSO este folosit de apelul de sistem `gettimeofday()` pentru a obține timpul curent. De ce nu poate fi folosit vDSO și pentru apelul de sistem `getpid()`, apel care obține PID-ul procesului curent?

11. (**25 puncte**) Vi se propune să implementați un framework de generare de *challenge*-uri de tip *exploit* pentru învățare în crearea de atacuri. Astfel de *challenge*-uri sunt executabile care conțin o vulnerabilitate (sau mai multe) de lucru cu memoria care poate fi exploatată pentru a obține un shell. Cazul clasic este o vulnerabilitate de tipul *buffer overflow* care poate fi exploatată pentru a conduce la rularea de cod arbitrar și astfel la obținerea unui shell.
a. Fie o structură de tipul `struct exploit_challenge_config` prin care definiți criterii pentru crearea unui *challenge*. Ce câmpuri va conține această structură? (**6 puncte**)
b. De ce componente software existente aveți nevoie pentru implementarea framework-ului? (utilitare, framework-uri, biblioteci) (**5 puncte**)
c. Indicați, schematic, secvența de pași pentru generarea unui *challenge*. (**8 puncte**)
d. Presupunând că furnizați un parametru pentru nivel de dificultate, cum veți particulariza implementarea pentru un nivel de dificultate dat? (**6 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:.....