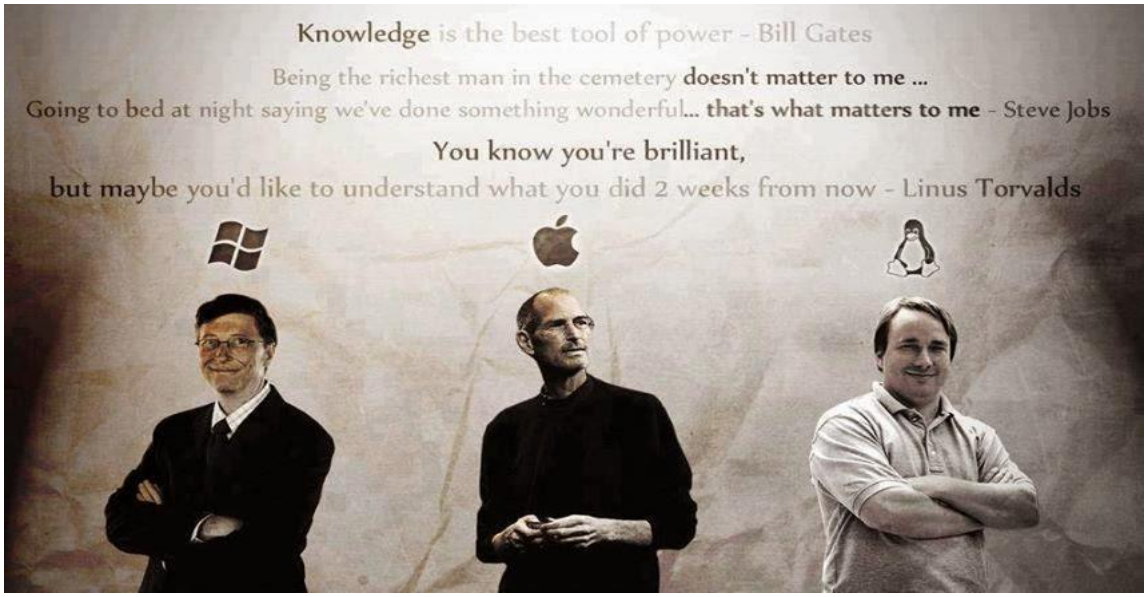
Lab 5. CONTROLUL ACCESULUI



CAPABILITATI LINUX

Resurse necesare laboratorului

O masina virtuala ce ruleaza o distributie de Linux (Ubuntu 14.04, 16.04, Lubuntu 14.04, etc).
Cateva surse:
https://www.despre-linux.eu/permisiuni-ale-folderelor-si-fisierelor-in-linux/
https://www.despre-linux.eu/category/comenzi-linux/
https://linuxize.com/tags/terminal/
https://profs.info.uaic.ro/~fltiplea/IS/AccessControl_01.pdf
https://profs.info.uaic.ro/~eonica/isec/lab04.html
https://profs.info.uaic.ro/~eonica/isec/lab05.html

Pachete necesare **`libattr1-dev`** si **`libcap2.21`** Pentru instalare se folosesc comenzile:

```
sudo apt-get install libattr1-dev
wget
http://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.
21.tar.gz
```

si respectiv:

```
tar xvf libcap-2.21.tar.gz
cd libcap-2.21
sudo make
sudo make install
```

Se verifica daca SELinux este instalat si se dezactiveaza in caz afirmativ, folosind comenzile:

```
ls/ etc/selinux/config      //verifica existenta fisierului de configurare a SELinux
sudo setenforce 0       // dezactivare temporara
```

Observatie : Security Enhanced Linux (SELinux) este o versiune a kernel-ului de Linux si a utilitarelor associate. Isi propune sa demonstreze utilitatea controlului accesului obligatoriu

(mandatory access controls). Caracteristica esentiala este ca utilizatorul nu are control absolut asupra resurselor pe care le creeaza, si atunci se doreste ca fiecare program/server sa utilizeze cantitatea minima de resurse necesara pentru a rula. Astfel dispare conceptul de utilizator privilegiat (root.(

Drepturile de acces asupra unui fisier sunt vizibile folosind comanda: **`ls –l`**

Modificarea drepturilor se face prin intermediul comenzii: **`chmod -l`**

`chmod [categorie utilizatori] [+-=] [drepturi] [cale]`

`Categorie utilizatori` arata caror utilizatori (subcampul din campul drepturilor) li se aplica modificarea si poate fi '**`u`**' - proprietar, '**`g`**' - grup, '**`o`**' - ceilalti sau '**`a`**' - toate categoriile. '**`+`**' , '**`–`**' sau '=' indica adaugarea, eliminarea, sau setarea directa a unor drepturi fata de, sau respectiv in locul celor existente deja.

Permisiuni ale fisierelor Linux: '**`r`**' (read) '**`w`**' (write) '**`x`**' (execute) pe entitati: proprietar, grup, alti utiizatori.

Exemplu **-rwxr-x--x** este un set de drepturi de acces de citire, scriere si executie pentru proprietarul unui fisier, citire si executie pentru grup si doar executie pentru altii.

Sticky bit: **t** in loc de **x** inseamna stergerea/modificarea fisierelor se poate realiza doar de catre proprietar. Schimbarea proprietarului unui fisier se face folosind comanda chown:

```
sudo chown [numeuser]:[numegrup] [numefisier]
```

Contul de root :ofera un acces privilegiat pe masina virtuala creata, detine controlul absolut al masinii si trebuie folosit numai atunci cand este nevoie. Folosirea necorespunzatoare poate duce la pierderea de informatii in mod involuntar.

Daca proprietarul unui program este chiar root-ul atunci programul ruleaza cu privilegiile root-ului. Root-ul are intotdeauna acces la toate fisierele si directoarele.

Comanda **sudo** permite unui alt utilizator (de incredere) rularea unui set restrans de comenzi cu privilegii de root. Nu trebuie folosit pentru un program care permite lucrul cu un shell, deoarece utilizatorul va putea face orice ca si root .

Orice process are doua ID-uri: RUID-ul si EUID-ul. RUID-ul) Real UID (identifica proprietarul real al procesului .EUID-ul (Effective UID) identifica privilegiile unui process. Controlul accesului se bazeaza pe EUID. Detinatorul real este utilizatorul care a introdus comanda, deţinătorul efectiv fiind cel care determină accesul la resursele sistemului. RUID şi EUID sunt, de regula, aceleasi, iar procesul are drepturile de acces pe care le are utilizatorul care l-a pornit .

De exemplul daca un *User1* porneste un program, procesul respectiv şi toate procesele care sunt pornite de acest proces vor fi deţinute de utilizatorul *User1* şi nu de administratorul sistemului. Daca un alt utilizator *User2* solicită accesul la anumite fisiere, el va fi determinat pe baza permisiunilor pe care le are *User1* şi nu pe cele ale administratorului de sistem, *root*. Cand un SUID (Set User ID) este executat, RUID ≠ EUID, RUID-ul devine egal cu ID-ul userului *User2,* iar EUID-ul ramane egal cu ID-ul userului initial (initiatorul programului *User1*). Toate procesele executate de la consola (shell) vor prelua, ca UID Real, ID-ul lui *User2* (dupa principiul RUID-ul nu se schimba).

Set User ID (SUID (In Linux exista o serie de operatii care pot fi realizate doar de catre utilizatorul root (configurare interfata rețea, modificare parola, back-up fisiere, etc.). Acestea corespund programelor SUID (fisiere ce au ca proprietar utilizatorul root si au bitul SUID setat). Astfel, pe parcursul executarii acestor operatii, utilizatorul normal devine temporar root, ceea ce reprezintă o vulnerabilitate de securitate a sistemului. Unele aplicatii (precum ping) necesita prezenta unui bit special SUID sau SGID (Set Group ID). Prezenta acestui bit pe un executabil permite executia acestuia cu drepturile celui ce detine executabilul (de obicei root – (unul din tipurile de atacuri preferate de atacatori. Numai aplicatiile sigure trebuie sa aiba setat bit-ul respectiv .

Exemplu **`-rwsr-x--x`**

Fisierele executabile care au bitul SUID setat au dreptul de a realiza operatii ce necesita drept de root (daca proprietarul fisierului este root). Setarea bitului SUID poate deschide brese de securitate. Daca un executabil care are bitul SUID este corupt, atunci un atacator poate obtine drepturi de root pe sistem. Din aceasta cauza a fost introdusa posibilitatea de a seta **capabilitati** asupra unui executabil. Acestea permit un set particular strict de operatii privilegiate asociat fiecarei capabilitati. In momentul in care un program este rulat, sistemul de operare verifica capabilitatile asociate cu respectivul executabil. Prin cresterea nivelului de granularitate a nivelului drepturilor acordate este redus astfel riscul ca un utilizator neautorizat sa obtina drepturi extinse in sistem.

Intr-un sistem bazat pe capabilitati, la executarea unui program, procesul corespunzator este initializat cu o listă de **capabilitati**. In momentul in care procesul respectiv incearca sa acceseze o resursa, sistemul de operare verifica daca procesul are capabilitatile necesare si decide acordarea sau interzicerea accesului la resursa solicitată (citire, scriere, modificare, stergere).

Probabil ca dintre toate sistemele de operare **Windows** detine supremația (accesibilitate si vulnerabilitate). Este usor de inteles, instalabil pe (aproape) orice platforma, accesibil tututor atat din punct de vedere tehnic cat si financiar. Dar pe cat este de accesibil, pe atat este de vulnerabil atacurilor. Cat despre **Mac** on an **Apple**, este puternic, performant, incoruptibil, rapid, dar si excesiv de scump. **Linux** vine ca un intermediar. Cu un pic de vointa, este usor de inteles si accesibil tuturor, inclusiv serviciilor secrete americane sau germane, comerţului online sau sistemelor bancare, si aceasta datorita invulnerabilitatii si stabilitatii, aproape perfecte, a sistemului. In plus, e gratuit.

Exista totusi capabilitati Linux vulnerabile. Iata cateva dintre ele.

Chsh-Change Shell este un program SUID care are capabilitatea de a schimba programele default shell (login-ul shell), stocate in fisierul /etc/passwd. Pentru a schimba default shell-ul se folosesc comenzile

chsh -s [shell-name]
chsh -s [shell-name] [user-name]
chsh -s /bin/bash
chsh -s /bin/bash user-name

(Eventual cu folosirea, in prealabil, a comenzii which bash pentru a afla locatia pe disc a comenzi)

Obseratie: Un user cu drepturi de root poate schimba login-ul shell pentru orice user normal folosind comanda:

$ sudo chsh -s /bin/bash user-name

Vulnerabilitate: Atacatorii pot crea un nou cont de root.

Un program mai poate fi atacat prin "inputuri" care nu sunt vizibile in program, asa zisele variabile de mediu. Acestea pot fi stabilite de un utilizator inainte de a rula un program.

Variabila de mediu de tip `PATH:` este utilizata de programe shell pentru a localiza o comanda, daca userul nu furnizeaza locatia completa unde se afla comanda. Exemplu:

system(): `call /bin/sh`
system("ls")
`/bin/sh` foloseste variabila de mediu PATH pentru a localiza comanda "ls"

Vulnerabilitate:
Atacatorul poate manipula variabila PATH si controla modul in care "ls" este gasita.

Anumite vulnerabilitati se datoreaza si privilegiilor programelor in timpul executiei acestora. Cateodata, programele care se bucura de anumite privilegii pot furniza scurgeri de informatie.

Exemplu programul `su` : este un program SUID cu privilegii, care da posibilitatea userilor sa se schimbe intre ei (exemplu user1 cu user2), asadar, programul incepe cu EUID ca root si RUID ca user1. Dupa verificarea password, atat EUID – ul cat si RUID – ul devin user2. Astfel de programe conduc la slabirea (spargerea) capabilitatilor.

Comanda ping

Pentru a verifica daca un calculator este disponibil in retea, se foloseste `ping`. Aceasta comanda trimite cereri de raspuns catre alte calculatoare.

Crearea unui nou grup se poate face in Linux folosind comanda `groupadd:`

```
sudo groupadd [numegrup]
```

Noul grup va fi vizibil in fisierul /etc/groups. Crearea unui nou utilizator in cadrul unui grup se face utilizand comanda `useradd:`

```
sudo useradd -g [numegrup] -d [director home] [numeuser]
```

Alte comenzi specifice grupurilor se pot gasi pe site-ul:

https://www.despre-linux.eu/category/comenzi-linux/

Acordarea si testarea capabilitatilor in Linux

Pentru acordarea/testarea capabilitatilor in Linux (Ubuntu) sunt necesare pachete **`libattr1-dev`** si **`libcap2.21`**

Asignarea de capabilitati unui fisier (executabil) se face prin comanda **`setcap`**:

```
setcap [nume_capabilitati]=[flaguri capabilitate] [program]
```

Daca doriti sa asignati mai multe capabilitati enumerarea acestora se face cu virgula. Flagurile ce se pot asocia la setul de capabilitati asignat sunt urmatoarele:

- **p** - permitted: capabilitatile permise pentru programul respectiv, care nu sunt neaparat si active (in functie)
- **e** - effective: capabilitatile efectiv in functie pentru programul respectiv (totdeauna un subset al celor cu flagul p)
- **i** - inheritable: capabilitatile ce pot fi transmise de programul respectiv catre un alt proces lansat de acesta

Afisarea capabilitatilor asociate unui fisier se poate face prin comanda **getcap**.

```
getcap program
```

Eliminarea completa capabilitatilor asociate se realizeaza prin comanda **setcap** folosind optiunea **–r** , si trebuie rulata cu drepturi de **root**.

```
setcap –r program
```

Afisarea capabilitatilor asociate unui proces se poate face prin comanda **getpcaps**.

Setul de capabilitati existente in Linux se poate consulta folosind comenzile:

```
man capabilities
```

```
less /usr/include/linux/capability.h
```

O parte din capabilitatile existente si o serie (partiala) din privilegiile asociate sunt urmatoarele:

- **cap_net_raw** - capabilitatea de a deschide un socket raw
- **cap_dac_read_search** - capabilitatea de a ignora drepturile primare de citire asupra fisierelor, si de citire si executie asupra directoarelor
- **cap_dac_override** - capabilitatea de a ignora drepturile primare de citire, scriere, executie
- **cap_chown** - capabilitatea de a face schimbari asupra userului sau grupului proprietar pentru un fisier
- **cap_fowner** - capabilitatea de a ignora verificarile de permisiuni pentru anumite operatii ce necesita ca utilizatorul asociat procesului sa fie acelasi cu utilizatorul proprietar al unui fisier afectat de proces (ex. chmod; operatii precum copiere, redenumire, etc in anumite situatii a unui fisier cu alt proprietar decat comanda respectiva)
- **cap_setuid** - capabilitatea de a manipula arbitrar identificatorul utilizatorului pentru un proces
- **cap_kill** - capabilitatea de a ignora verificarile de permisiuni pentru trimiterea de semnale

Aplicatii laborator

1. Explicati de ce comenzile '**passwd**', '**chsh**', 'su', and '**sudo**' trebuie sa aiba setat bitul SUID. Ce se intampla in caz contrar.

Eliminati bitul SUID din comanda "**passwd**" si rulati comanda folosind capabilitati. Observati ce se intampla.

Exemplul 1.

```
$ ls –l /usr/bin/passwd
-rwsr-xr-x 1 root root 41284 Sep 12  2019 /usr/bin/passwd
```

- Cand un *User2* vrea sa schimbe parola (unui grup), executa /usr/bin/passwd.

- Atunci RUID-ul va fi *User2* dar EUID-ul va fi root.

– *User2* poate folosi passwd pentru a schimba numai propria parola, pentru ca passwd verifica RUID-ul si, daca acesta nu este root, doar parola proprietarului initial (ex. *User1*) va fi schimbata.

- Este necesar ca EUID-ul sa devina root pentru ca procesul necesita permisiuni de scriere asupra /etc/passwd si/sau /etc/shadow.

Exemplul 2.

```
user@ubuntu:~$ which passwd
/usr/bin/passwd
user@ubuntu:~$ ls -al /usr/bin/passwd
-rwsr-xr-x 1 root root 41284 Sep 12  2019 /usr/bin/passwd
user@ubuntu:~$ cp /usr/bin/passwd /tmp/
user@ubuntu:~$ ls -al /tmp/passwd
-rwxr-xr-x 1 user user 41284 Aug  7 20:04 /tmp/passwd
```

Observatia 1. Cand se copie passwd to /tmp/ se pierd capabilitatile de `root`. Analog se in tampla pentru chsh, su and sudo.

Observatia 2. Pentru a afla locatia pe disc a unei comenzi se poate utiliza **`which`** '**`comanda`**'.

2. Creati un fisier "shadow" si afisati/modificati, etc capabilitatile acestuia**.**

3. Creati un program `mycat` care simuleaza `cat` datii capabilitati `de root` Aplicati `mycat` pentru /etc/shadow file

Exemplul 3.

```
user@ubuntu:~$ cp /bin/cat ./mycat
user@ubuntu:~$ sudo chown root mycat
user@ubuntu:~$ ls -l mycat
-rwxr-xr-x 1 root user 41284 Oct 1 14:09 mycat

user@ubuntu:~$ ./mycat /etc/shadow: Permission denied /nu e program cu drepturi (privilegiat)

user@ubuntu:~$ sudo chmod 4755 mycat  //devine program cu drepturi
user@ubuntu:~$ ./mycat /etc/shadow
```

4. Creati un nou utilizator care să fie inclus in grupul sudo (utilizand comenzile useradd şi usermod).

Hint: https://linuxize.com/post/how-to-create-users-in-linux-using-the-useradd-command/

5. Fiind logati ca utilizatorul nou creat, eliminati bitul SUID pentru comanda mount

6. Care sunt capabilitatile necesare pentru a rula comanda mount fara drepturi de root?

7. Puteti afla care este exact capabilitatea necesara pentru a rula comanda ping fara drepturi de root? Daca da, cum?

8. Care sunt capabilitatile necesare pentru a rula comanda more/etc/shadow (fara drepturi de root). Cum le-ati aflat?