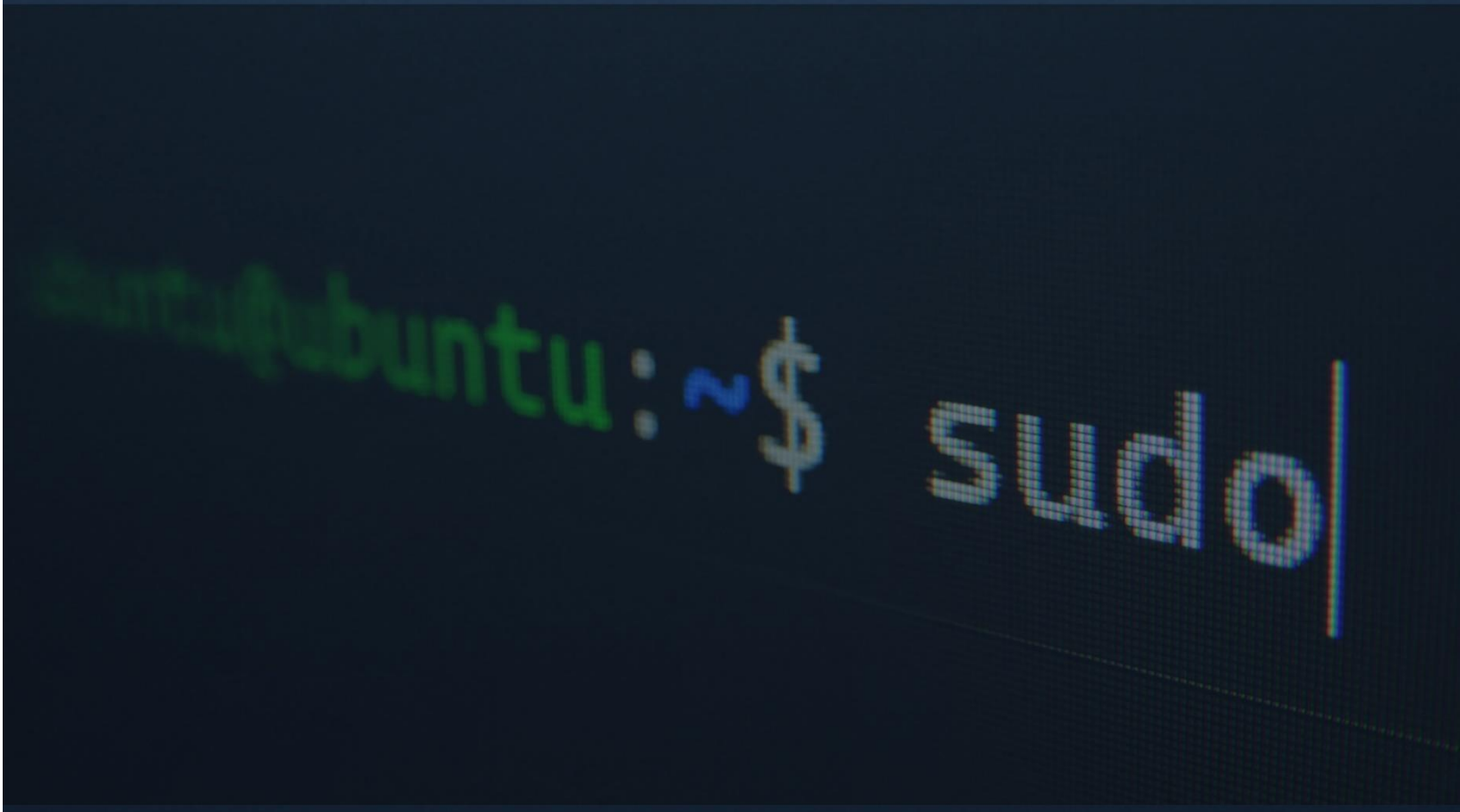
Administracja systemem Linux II



Hands-On Labs

by Marcin Kujawski

Administracja systemem Linux II

Ćwiczenia laboratoryjne

opracowane przez

Marcin Kujawski

Spis Treści

Lab 1: Strumienie i przekierowania I/O

Zadania:

- Napisz skrypt, który będzie tworzyć katalog backup w katalogu głównym użytkownika i przekieruj błędy (jeśli wystąpią) do pliku copy.err.
- Skopiuj zawartość całego /usr/bin/ do wyżej stworzonego katalogu. Standardowe wyjście oraz błędy podczas wykonania przekieruj do pliku copy.log.

Rozwiązanie:

```
#!/bin/bash
# Backup dla plików binarnych z /usr/bin
# Lab 1

echo "Tworzenie katalogu backup"
mkdir ~/backup 2> copy.err
echo "Kopiowanie plików... "
cp -rp /usr/bin/* ~/backup > copy.log 2>&1
```

Lab 2: Wyrażenia regularne

Zadania:

Napisz komendę, która:

- wyświetli na ekran tylko wina francuskie oraz pokaże tylko jego nazwę, rodzaj i cenę,
- zliczy wszystkie inne wina pochodzące z innych krajów,
- zamieni słowo wytrawne na półwytrawne w linii 2 i 3,
- dopisze w piątej linijce nowe wino wg wzorca w pliku.

Plik wejściowy:

```
Nazwa wina,Kraj pochodzenia,Rodzaj,Cena
Chardonnay Lyngrove,RPA,białe wytrawne,95.00
Cabernet Sauvignon Merlot Lyngrove,RPA,czerwone
wytrawne,95.00
Baron De France,Fra,białe wytrawne,25.00
Anjou Blanc Chenin,Fra,białe wytrawne,33.00
Anjou D'rose,Fra,różowe półwytrawne,33.00
Anjou Rouge Cabernet,Fra,czerwone półwytrawne,33.00
Bordeaux Graveschateau Saint Galier,Fra,białe wytrawne,89.00
Don Kichot Tinto,Spa,czerwone półwytrawne,25.00
Rioja Miralcampo,Spa,czerwone wytrawne,62.00
Rioja Miralcampo,Spa,białe wytrawne,62.00
Sherry Rich Cream,Spa,czerwone słodkie,109.00
Sole D'italia,Ita,czerwone wytrawne,24.00
Valpolicella,Ita,czerwone wytrawne,45.00
Chianti Villa Bellafonte,Ita,czerwone wytrawne,94.00
```

Rozwiązanie:

- wyświetli na ekran tylko wina francuskie oraz pokaże tylko jego nazwę, rodzaj i cenę:

```
$ grep Fra wina | awk -F"," '{ print $1 "," $3 "," $4 }'
```

- zliczy wszystkie inne wina pochodzące z innych krajów”

```
$ echo -n "Inny win jest: " ; grep -v Fra wina | wc -l
```

- zamieni słowo wytrawne na półwytrawne w linii 2 i 3:

```
$ sed '2,3s/wytrawne/półwytrawne/g' wina
```

- dopisze w piątej linijce nowe wino wg wzorca w pliku:

```
$ sed '5i\Nowe Wino Włoskie,Ita,czerwony słodkie,44.00' wina
```

Lab 3: Narzędzia archiwizujące

Zadania:

- Utwórz katalog o nazwie backup, a w nim umieść skompresowane archiwum tar wszystkich plików w katalogu /usr/include, przy czym katalogiem najwyższego poziomu jest include. Możesz użyć dowolnej metody kompresji (gzip, bzip2 lub xzip).
- Wylistuj pliki znajdujące się w archiwum.
- Utwórz katalog o nazwie restore oraz rozpakuj i zdekompresuj archiwum.
- Porównaj zawartość z oryginalnym katalogiem, z którego powstało archiwum.

Rozwiązanie:

```
$ mkdir /tmp/backup
$ cd /usr ; tar zcvf /tmp/backup/include.tar.gz include
$ cd /usr ; tar jcvf /tmp/backup/include.tar.bz2 include
$ cd /usr ; tar Jcvf /tmp/backup/include.tar.xz include
```

lub

```
$ tar -C /usr -zcf include.tar.gz include
$ tar -C /usr -jcf include.tar.bz2 include
$ tar -C /usr -Jcf include.tar.xz include
```

Sprawdzenie efektowności kompresji:

```
$ du -sh /usr/include
```

Listowanie archiwum:

```
$ ls -lh include.tar.*
$ tar tvf include.tar.xz
```

Dekompresja:

```
$ cd .. ; mkdir restore ; cd restore
$ tar xvf ../backup/include.tar.bz2
```

Zauważ, że nie jest potrzebne podawanie opcji j, J, lub z kiedy wykonujemy dekompresję narzędziem tar. Jest on na tyle sprytny że sam rozpozna typ archiwum.

Lab 4: Monitoring i performance

Zadania:

- Obciąż testowo server na 10 minut.
- Obserwuj performance przez czas testu używając różnych narzędzi: top, htop, atop, nmon.
- Porównaj które narzędzie jest najbardziej przydatne aby wychwycić jak najwięcej informacji.

Rozwiązanie:

```
$ dnf install stress-ng -y
$ stress-ng -c 8 -i 4 -m 6 -t 10m
```

Powyższe polecenie odpala:

- 8 procesów intensywnie korzystających z CPU, z których każdy wykonuje obliczenia sqrt(),
- 4 procesów intensywnie korzystających z wejścia/wyjścia, z których każdy wykonuje operację sync(),
- 6 procesów intensywnie wykorzystujących pamięć, z których każdy wykonuje funkcję malloc(), domyślnie przydzielając 256 MB,
- czas obciążenia = 10 minut

```
$ top
$ htop
$ atop
$ nmon
```

Lab 5: Systemd oraz systemctl

Zadania:

- Napisz prosty skrypt w bashu, która będzie co 10 sekund zapisywał do pliku /tmp/my-service.log datę. Skrypt ma obsługiwać parametr -o z argumentami start/stop/status aby zarządzać działaniem skryptu.
- Utwórz plik serwisu, który będzie zarządzać tą aplikacją, tak aby była możliwość zatrzymania i uruchomienia skryptu poprzez systemctl.
- Sprawdź domyślny target systemu i zmień go na inny.

Utworzenie skryptu:

```
$ vi /usr/bin/script.sh
```

```
#!/bin/bash

while getopts "o:" flaga
do
        case "${flaga}" in
                o) OPERATION=${OPTARG};;
        esac
done

if [ -z "$OPERATION" ] ; then
        echo
        echo "Użycie skryptu: $0 {start|stop|status}"
        exit 0
fi
```

```
if [ "$OPERATION" = "start" ] ; then
        /usr/bin/skrypt.sh &
        echo $!>/var/run/skrypt.pid
        echo "Script $0 started"
elif [ "$OPERATION" = "stop" ] ; then
        kill `cat /var/run/hit.pid`
        rm /var/run/skrypt.pid
        echo "Script $0 stopped"
elif [ "$OPERATION" = "status" ] ; then
        if [ -e /var/run/skrypt.pid ]; then
                echo   skrypt.sh   is   running,   pid=`cat
/var/run/skrypt.pid`
        else
                echo skrypt.sh is NOT running
                exit 1
        fi
fi

while true
do
 echo `date` >> /tmp/my-service.log
 sleep 10
done
```

Nadanie praw do uruchomienia:

```
$ chmod +x /usr/bin/script.sh
```

Jeśli SELinux jest w trybie „Enforcing” zadbaj o konteksty. Powyższe polecenie nadaje taki sam kontekst jak serwis SSHD, tak więc Linux nie będzie mieć problemu, aby i zezwolić na dostęp do pliku naszego serwisu i uruchomić go.

```
chcon -t sshd_exec_t /usr/bin/skrypt.sh
```

Utworzenie pliku serwisu:

```
$ vi /lib/systemd/system/shellscript.service
```

```
[Unit]
Description=Mój własny Skrypt Shell
After=network.target

[Service]
ExecStart=/usr/bin/skrypt.sh -o start
ExecStop=/usr/bin/skrypt.sh -o stop

[Install]
WantedBy=default.target
```

Restart systemd oraz zarządzanie stworzonym serwisem:

```
$ systemctl daemon-reload
$ systemctl enable shellscript.service
$ systemctl start shellscript.service
$ systemctl status shellscript.service
```

Zmiana domyślnego targetu systemu:

```
$ systemctl get-default
$ systemctl set-default multi-user.target
```

Lab 6: Planowanie zadań

Zadania:

- Dodaj do crona wykonanie skryptu pisanego w Lab 5 co dziennie co pół godziny w godzinach 20:00 – 4:00 od poniedziałku do piątku.
- Dodaj jednorazowe zadanie do wykonania w sobotę i w niedzielę o 15:00, które restartuje serwis sshd.

Rozwiązanie:

```
$ crontab -e


*/30 20,21,22,23,0,1,2,3,4 * * 1-5 /usr/bin/skrypt.sh
```

```
$ systemctl restart sshd | at 3:00 PM 04/29/2023
$ systemctl restart sshd | at 3:00 PM 04/30/2023
```

Lab 7: Tworzenie partycji

Zadania:

- Stwórz partycję kompatybilną ze standardem UEFI o wielkości 500M i formacie ‘Linux’.
- Dodaj kolejną partycję typu ‘SWAP’ wielkości 200M na tym samym fizycznym urządzeniu.
- Zapisz i sprawdź na systemie za pomocą komendy: ‘cat /proc/partitions’ czy zmiany są widoczne.

Tworzenie nowych partycji programem gdisk (UEFI):

```
$ gdisk /dev/sdb

Command (? for help): n
Partition number (1-128, default 1): <Enter>
First sector (34-2097118, default = 2048) or {+-}size{KMGTP}:
<Enter>
Last sector (2048-2097118, default = 2097118) or {+-
}size{KMGTP}: +500M
Last sector (2048-2097118, default = 2097118) or {+-
}size{KMGTP}: <Enter>
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300): <Enter>
Changed type of partition to 'Linux filesystem'

Command (? for help): n
Partition number (2-128, default 2): <Enter>
```

```
First sector (34-2097118, default = 1026048) or {+-
}size{KMGTP}: <Enter>
Last sector (1026048-2097118, default = 2097118) or {+-
}size{KMGTP}: +200M
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300): 8200
Changed type of partition to 'Linux swap'

Command (? for help): p
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         1026047   500.0 MiB   8300  Linux filesystem
   2         1026048         1435647   200.0 MiB   8200  Linux swap

Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL
OVERWRITE EXISTING PARTITIONS!!

Do you want to proceed? (Y/N): Y
```

Sprawdzenie czy partycja jest widoczna z poziomu systemu:

```
$ cat /proc/partitions
major minor  #blocks  name

   8        0   31457280 sda
   8        1    1048576 sda1
   8        2   30407680 sda2
  11        0    1048575 sr0
 253        0   28282880 dm-0
 253        1    2121728 dm-1
   8       16    1048576 sdb
```

```
   8       17     512000 sdb1
   8       18     204800 sdb2
$ lsblk /dev/sdb
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sdb      8:16   0    1G  0 disk
├─sdb1   8:17   0  500M  0 part
└─sdb2   8:18   0  200M  0 part
```

Lab 8: Tworzenie filesystemów i montowanie zasobów dyskowych

Zadania:

- Stwórz filesystem ext4 oraz zamontuj wcześniej stworzoną partycje typu Linux permanentnie w systemie bez praw do zapisu. Użyj folderu /part1 do montowania partycji. Partycja ma być zamontowana po UUID w trybie Read-Only zawsze po starcie systemu.
- Na swapowej partycji stwórz przestrzeń wymiany oraz także ustaw, aby montowała się automatycznie przy starcie systemu. Do montowania partycji użyj labela ‘swap-extra’.
- Zmień tryb zapisu dla filesystemu /part1 na Read-Write i sprawdź czy po zmianie pod katalogiem /part1 da się cokolwiek zapisać.
- Wylistuj wszystkie partycje swapowe skonfigurowane na systemie i sprawdź ich użycie.
- Usuń wszystkie partycje.

Tworzenie filesystemu oraz pobranie wartości UUID:

```
$ mkfs.ext4 /dev/sdb1
$ mkdir /part1
$ blkid
$ vi /etc/fstab
UUID=<uuid> /part1    ext4    defaults,ro        0 0
```

Tworzenie przestrzeni wymiany z odpowiednim labelem:

```
$ mkswap -L swap-extra /dev/sdb2
$ vi /etc/fstab
LABEL=swap-extra        none    swap    defaults        0 0
```

Zmiana trybu Read-Only na Read-Write:

```
$ mount -o remount,rw /part1
$ cd /part1
$ touch plik1
```

Włączenie wszystkich przestrzeni wymiany i wylistowanie ich obecnego użycia:

```
$ swapon -a
$ swapon -s

Nazwa pliku                             Typ             Rozmiar Użyte   Priorytet
/dev/dm-1                               partition       2121724         0               -2
/dev/sdb2                               partition       204796          0               -3

$ free -m

              total        used        free      shared  buff/cache   available
Mem:           1750         824         565          11         519         926
Swap:          2271           0        2271
```

Odmontowanie i usunięcie obu partycji:

```
$ umount /part1
$ swapoff /dev/sdb2
$ gdisk /dev/sdb
```

```
Command (? for help): d
Partition number (1-2): 2

Command (? for help): d
Using 1

Command (? for help): p

Number  Start (sector)    End (sector)  Size       Code  Name

Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL
OVERWRITE EXISTING PARTITIONS!!

Do you want to proceed? (Y/N): Y
```

Lab 9: Logical Volume Management

Zadania:

- Utwórz na nowo dodanym dysku fizycznym dwa wolumeny fizyczne (/dev/sdb1, /dev/sdb2).
- Stwórz grupę wolumenową o nazwie ‘vg1’ zawierająca oba wolumeny fizyczne.
- Stwórz logiczny wolumen o rozmiarze 100 MB działający jako mirror o nazwie ‘lv-mirror’.
- Utwórz logiczny wolumen o rozmiarze 50 MB o nazwie ‘lv-data’.
- Utwórz filesystem typu ext4 na wolumenie ‘lv-mirror’.
- Utwórz filesystem typu xfs na wolumenie ‘lv-data’.
- Zamontuj oba zasoby pod innymi mount pointami.
- Stwórz przykładowe dane na obu filesystemach.
- Zrób snapshot wolumenu ‘lv-data’.
- Zasymuluj awarię i odtwórz wolumen ‘lv-data’ ze snapshotu.

Wyczyszczenie tablicy partycji (wymagane jeśli były już na urządzeniu blokowym inne partycje) oraz stworzenie dwóch partycji Linux LVM pod dwa fizyczne wolumeny:

```
$ wipefs -a /dev/sdb
$ gdisk /dev/sdb

Command (? for help): n
Partition number (1-128, default 1): 1
```

```
First sector (34-2097118, default = 821248) or {+-
}size{KMGTP}: <Enter>
Last sector (821248-2097118, default = 2097118) or {+-
}size{KMGTP}: +200M
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300): 8e00
Changed type of partition to 'Linux LVM'

Command (? for help): n
Partition number (1-128, default 1): 2
First sector (34-2097118, default = 821248) or {+-
}size{KMGTP}: <Enter>
Last sector (821248-2097118, default = 2097118) or {+-
}size{KMGTP}: +200M
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300): 8e00
Changed type of partition to 'Linux LVM'

Command (? for help): p

Number  Start (sector)    End (sector)  Size       Code  Name
   1          821248         1230847   200.0 MiB   8E00  Linux LVM
   2          411648          821247   200.0 MiB   8E00  Linux LVM
```

Utworzenie fizycznego wolumenu:

```
$ pvcreate /dev/sdb1
$ pvcreate /dev/sdb2
```

Utworzenie grupy wolumenowej:

```
$ vgcreate vg1 /dev/sdb[1-2]
```

Utworzenie logicznego wolumenu lustrzanego (mirror).

```
$ lvcreate -L 100MB -m1 -n lv-mirror vg1
```

Po utworzeniu woluminu lustrzanego metadane zostaną automatycznie skopiowane do obu woluminów. Możemy to obserwować w kolumnie 'Cpy%Sync'.

Wylistowanie i analiza wolumenu lustrzanego (mirror):

```
$ lvs -a -o +devices
```

Powyższe polecenie wyświetla szczegółową konfigurację mirrora:

- SubLV przechowujące bloki danych lub bloki parzystości, mają przyrostek '_rimage_#'. Nazywane też czasem DataLV.
- SubLV przechowujące metadane RAID mają sufiks '_rmeta_#'. Metadane RAID zawierają informacje o superbloku, typie RAID, bitmapie i informacje o stanie urządzenia. Nazywane też czasem MetaLV.

Utworzenie logicznego wolumenu lustrzanego (mirror).

```
$ lvcreate -L 50MB -n lv-data vg1
```

Stworzenie filesystemu ext4 na wolumenie ‘lv-mirror’:

```
$ mkfs.ext4 /dev/vg1/lv-mirror
```

Stworzenie filesystemu xfs na wolumenie ‘lv-data’:

```
$ mkfs.xfs /dev/vg1/lv-data
```

Zamontowanie lv-mirror pod mount pointę /mirror oraz lv-data pod /data:

```
$ mkdir /data /mirror
$ mount /dev/vg1/lv-mirror /mirror
$ mount /dev/vg1/lv-data /data/
```

Oczywiście należy też dodać wpisy do /etc/fstab aby montowanie odbywało się automatycznie przy każdym starcie systemu.

Stworzenie przykładowych dwóch plików na obu filesystemach:

```
$ dd if=/dev/urandom of=/mirror/testfile1 bs=20MB count=1
$ dd if=/dev/urandom of=/mirror/testfile2 bs=50MB count=1
$ dd if=/dev/urandom of=/data/testfile1 bs=10MB count=1
$ dd if=/dev/urandom of=/data/testfile2 bs=20MB count=1
```

Wykonanie snapshotu na wolumenie ‘lv-data’:

```
lvcreate -L 50MB -s -n lv-data-snap1 /dev/vg1/lv-data
```

Dodanie pewnych zmian na /data:

```
$ touch /data/newfile1 /data/newfile2
```

Odtworzenie wolumenu ‘lv-data’ ze snapshotu zrobionego wcześniej. Aby to wykonać musimy najpierw odmontować filesystem:

```
$ umount /data
$ lvconvert --merge /dev/vg1/lv-data-snap1
```

UWAGA!

Jeśli wykonasz komendę „lvconvert --merge” bez odmontowania zasobu, wówczas trzeba użyć poniższej komendy, aby odświeżyć logiczny wolumen i zainicjować revert snapshota.

```
$ lvchange –refresh /dev/vg1/lv-data
```

Zauważ, że po zmergowaniu snapshot zniknął. Aby sprawdzić czy snapshot został przywrócony poprawnie zamontujmy z powrotem filesystem na mount point:

```
$ mount /dev/vg1/lv-data /data
$ cd /data
$ ls -la
```

Pliki, które zostały utworzone po snapshocie zniknęły – co oznacza, że przywrócenie ze snapshota zostało zrobione pomyślnie.

Lab 10: Szyfrowanie dysków

Zadania:

- Utwórz nową partycję dla zaszyfrowanego urządzenia blokowego za pomocą fdisk. Upewnij się, że system jest świadomy nowej tablicy partycji
- Sformatuj partycję za pomocą cryptsetup używając LUKS
- Utwórz i otwórz zaszyfrowane urządzenie blokowe o nazwie ‘secret-disk’.
- Dodaj wpis do /etc/crypttab, aby system pytał o hasło przy ponownym uruchomieniu.
- Sformatuj system plików jako system plików ext4.
- Utwórz punkt montowania dla nowego systemu plików, tj. /secret.
- Dodaj wpis do /etc/fstab, aby system plików był montowany przy starcie systemu.
- Spróbuj zamontować zaszyfrowany system plików.
- Przetestuj całą konfigurację poprzez ponowne uruchomienie komputera.

Stworzenie nowej partycji:

```
$ fdisk /dev/sdb
```

Odświeżenie tablicy partycji:

```
$ partprobe -s
```

Stworzenie partycji LUKS i otworzenie jej pod nazwą secret-disk:

```
$ cryptsetup luksFormat /dev/sdb1
$ cryptsetup luksOpen /dev/sdb1 secret-disk
```

Dodanie wpisu do /etc/crypttab:

```
$ vi /etc/crypttab

secret-disk /dev/sdb1 none
```

Stworzenie filesystemu i mount pointu:

```
$ mkfs -t ext4 /dev/mapper/secret-disk
$ mkdir -p /secret
```

Dodanie wpisu do /etc/fstab:

```
$ vi /etc/fstab

/dev/mapper/secret-disk /secret ext4 defaults 0 0
```

Zamontowanie filesystemu na mount point /secret:

```
$ mount /secret
```

Test poprzez reboot:

```
$ reboot
```

Lab 11: RAID (opcjonalnie)

Zadania:

- Utwórz dwie partycje 200 MB typu raid (fd) za pomocą fdisk, albo za pomocą LVM.
- Utwórz urządzenie RAID 0 o nazwie /dev/md0 używając tych dwóch partycji.
- Sformatuj urządzenie RAID jako system plików ext4. Następnie zamontuj je w /myraid.
- Zapisz konfigurację macierzy do pliku /etc/mdadm.conf.
- Zbadaj /proc/mdstat, aby zobaczyć stan macierzy.
- Zatrzymaj macierz i ponownie ją wystartuj.

Stworzenie nowej partycji. Utwórz dwie partycje typu RAID (fd):

```
$ fdisk /dev/sdb
```

Odświeżenie tablicy partycji:

```
$ partprobe -s
```

Sprawdzenie czy możemy użyć dyski pod macierz:

```
$ mdadm --examine /dev/sdb[1-2]
```

Stworzenie macierzy:

```
$ mdadm --create /dev/md0 --level=stripe --raid-devices=2
/dev/sdb[1-2]
```

Stworzenie mount pointu oraz utworzenie filesystemu:

```
$ mkdir /myriad
$ mkfs.ext4 /dev/md0
$ vi /etc/fstab

/dev/md0 /myraid ext4 defaults 0 0
```

Utworzenie pliku konfiguracyjnego:

```
$ mdadm --detail --scan --verbose >> /etc/mdadm.conf
```

Wylistowanie wszystkich macierzy:

```
$ cat /proc/mdstat
```

Wyświetlenie szczegółów macierzy i ich dysków:

```
$ mdadm --detail /dev/md0
$ mdadm -E /dev/sdb[1-2]
```

Zatrzymanie macierzy i ponowne wystartowanie macierzy:

```
$ mdadm --assemble /dev/md0
```

Lab 12: Zarządzanie hasłami użytkownika

Zadania:

- Utwórz konto użytkownika testuser1, które będzie używało powłoki Korna (ksh) jako domyślnej powłoki. (jeśli nie ma /bin/ksh zainstaluj ją lub użyj powłoki C w /bin/csh).
- Ustaw hasło dla ‘testuser1’ na ‘student’.
- Zajrzyj do /etc/shadow. Jaka jest aktualna data wygaśnięcia konta dla ‘testuser1’?
- Ustaw datę wygaśnięcia konta ‘testuser1’ na 1 maja 2023 roku oraz wymuś zmianę hasła przy logowaniu.

Stworzenie nowego użytkownika i ustawienie hasła:

```
$ useradd -m -c /bin/ksh -c "Test User1" testuser1
$ passwd testuser1
```

Weryfikacja konta w pliku /etc/shadow:

```
$ cat /etc/shadow | grep -i testuser1

testuser1:$6$nCtD9DhvaFcOabSz$JD.mP95LGQdLx/NSOdK09ABwtXsZQ0q
lLahAHYghn3TEnW7JWZBDORKIouY1JRs5FGZ1midPcirkj/zJfxHKv.:19459
:0:99999:7:::
```

Data wygaśnięcia konta to: „nigdy”

Można też podejrzeć to za pomocą komendy chage:

```
$ chage -l testuser1
```

Ustawienie wygasanie konta na 1 maja 2023:

```
$ chage -E "01/05/2023" testuser1
```

Wymuszenie zmiany hasła przy logowaniu:

```
$ chage -d 0 testuser1
```

Lab 13: Access Control List

Zadania:

- Zaloguj się jako użytkownik student i utwórz katalog ‘projekty’. Sprawdź domyślne uprawnienia dla tego katalogu.
- Przyznaj uprawnienia do odczytu i zapisu użytkownikowi ‘testuser1’. Uruchom to polecenie jako student i korzystaj z notacji oktalnej.
- W katalogu ‘projekty’ utwórz podkatalog ‘proj1’ i sprawdź, czy odziedziczył ustawienia ACL z katalogu nadrzędnego.
- Utwórz plik ‘plik1’ w katalogu projekty i sprawdź, czy odziedziczył ACL z katalogu nadrzędnego.
- Skasuj wszystkie domyślne ustawienia ACL.

Stworzenie katalogu ‘projekty’:

```
$ mkdir projekty
```

Przyznanie uprawnień do odczytu i zapisu użytkownikowi ‘testuser1’:

```
$ setfacl -m d:u:testuser1:6 projekty
$ getfacl -c projekty
```

Stworzenie podkatalogu ‘proj1’ i sprawdzenie uprawnień:

```
$ cd projekty/
$ mkdir proj1
$ getfacl -c proj1/
```

Stworzenie pliku ‘plik1’ i sprawdzenie uprawnień:

```
$ touch plik1
$ getfacl -c plik1

user::rw-
user:testuser1:rw-
group::r-x                      #effective:r--
mask::rw-
other::r--
```

Wynika stąd, że maksymalne uprawnienia dla członków grupy to odczyt. Prawo do zapisu i wykonywania jest nieefektywne ze względu na ustawienia maski.

Skasowanie wszystkich domyślnych ustawień ACL:

```
$ setfacl –k projekt
$ getfacl –c projekt
```

Lab 14: PAM

Zadania:

- Zmodyfikuj PAM tak, aby użytkownik, który zaloguje się błędnie 3 razy (poda 3 razy złe hasło podczas logowania) zostanie zablokowany na okres 5 minut.
- Skonfiguruj PAM w ten sposób, aby użytkownik root też podlegał tej zasadzie.
- Po okresie 5 minut użytkownik ma zostać odblokowany i ma mieć możliwość dokonania ponownej próby zalogowania się.

Zainstaluj pakiet authselect-compat aby ułatwić sobie edycję plików PAM:

```
$ dnf install authselect-compat -y
```

Zaktualizuj konfigurację uwierzytelnienia PAM:

```
$ authconfig  --enablefaillock --faillockargs="deny=3
unlock_time=60 even_deny_root" --update
```

Edycja plików /etc/pam.d/system-auth oraz /etc/pam.d/password-auth i dodanie opcji do linijki zawierającej nazwę modułu PAM pam_faillock:

```
$ vi /etc/pam.d/system-auth
$ vi /etc/pam.d/password-auth
```

Dopisanie argumentów wytłuszczonych na czerwono poniżej:

```
pam_faillock.so preauth audit deny=3 even_deny_root
unlock_time=60
```

Wykonanie testu I błędne podanie hasła 3 razy dla dowolnego użytkownika.

Sprawdzenie czy konto zostało zablokowane:

```
$ faillock --user <test-user>
```

Opjonalnie można też sprawdzić pliki logów:

- /var/log/secure
- /var/log/audit
- /var/log/messages

Odblokowanie konta:

```
$ faillock --user <test-user> --reset
```

Ponowny test logowania podając już poprawne hasło. Logowanie powinno działać.

Lab 15: Procesy i sygnały

Zadania:

- Utwórz skrypt w Bashu o nazwie ‘skrypt.sh’, który będzie pracować w nieskończonej pętli (użyj komendy sleep z 10 sekundowym czasem).
- Dodaj funkcję przechwycenia sygnału SIGINT oraz zaloguj takie zdarzenie do pliku /tmp/skrypt.signals.
- Odpal skrypt w tle (lub na jednej konsoli) a następnie zasymuluj wystąpienie sygnału SIGINT.
- Sprawdź, czy skrypt przechwycił sygnał i czy wpis w logu istnieje.

Stworzenie skryptu:

```
$ vi skrypt.sh
```

```
#!/bin/bash
# Przechwycenie sygnału przerwania programu SIGINT

LOGFILE=/tmp/skrypt.signals
CTRLC=0

function przechwyc_sigint {
  CTRLC=$(( $CTRLC + 1 ))
  echo
  if [[ $CTRLC > 0 ]]; then
    echo "Wystąpił SIGINT po raz $CTRLC" >> $LOGFILE
  fi
}

trap przechwyc_sigint SIGINT
```

```
while true
do
  echo "Spimy..."
  sleep 10
done
```

Odpalenie skryptu w tle:

```
$ chmod +x skrypt.sh
$ ./skrypt
```

Symulacja wystąpienia sygnału SIGINT poprzez wciśnięcie Ctrl+C lub poprzez komendę kill:

```
$ kill -2 3006
```

Sprawdzenie pliku logu:

```
$ tail -f /tmp/skrypt.signals
# lub
$ cat /tmp/skrypt.signals
```

Lab 16: Konfiguracja jądra i moduły

Zadania:

- Sprawdź, czy możesz spingować swój system.
- Sprawdź aktualną wartość parametru ‘net.ipv4.icmp_echo_ignore_all’, który służy do włączania i wyłączania odpowiedzi na ping (domyślnie wartość 0 pozwala odpowiadać na pingi).
- Ustaw wartość na 1 za pomocą narzędzia ‘sysctl’, a następnie sprawdź, czy pingi odpowiadają.
- Ustaw wartość z powrotem na 0 i pokaż oryginalne zachowanie po przywróceniu ustawień.
- Zmień wartość modyfikując ‘/etc/sysctl.conf’ i zaaplikuj ustawienia z tego pliku bez ponownego uruchomienia systemu.

Sprawdzenie odpowiedzi na ping:

```
$ ping 192.168.1.90
```

Odpowiedź na ping jest prawidłowa.

Sprawdzenie wartości dla parametru ‘net.ipv4.icmp_echo_ignore_all’:

```
$ sysctl -a | grep -i net.ipv4.icmp_echo_ignore_all
```

Zwrócona wartość jest równa 0.

Ustawienie wartości na 1:

```
$ sysctl net.ipv4.icmp_echo_ignore_all=1
```

Sprawdzenie odpowiedzi na ping:

```
$ ping 192.168.1.90
```

Brak odpowiedzi na ping.

Dodanie linijki do pliku /etc/sysctl.conf:

```
net.ipv4.icmp_echo_ignore_all=1
```

Wymuszenie przeładowania ustawień jądra:

```
$ sysctl -p
```

Ponowne sprawdzenie odpowiedzi na ping:

```
$ ping 192.168.1.90
```

Brak odpowiedzi na ping – oznacza to że implementacja jest poprawna i za każdym razem jak system będzie ponownie uruchamiany konfiguracja będzie zaczytywana z pliku.

Lab 17: Konfiguracja sieci

Zadania:

- Dodaj kolejny adres IP do aktualnego interfejsu sieciowego za pomocą ‘nmcli’.
- Adres IP ma być o 100 wyższy niż aktualnie przypisany, czyli jeśli Twój adres to 192.168.1.46, to nowy IP będzie równy 192.168.1.146.
- Dodaj statyczną trasę (route). Wszystko co leży w sieci 172.16.0.0/24 ma iść przez IP 192.168.1.250.
- Spraw, aby konfiguracja była konsystentna po restarcie systemu.

Sprawdzenie aktualnych połączeń:

```
$ nmcli con
```

Sprawdzenie aktualnego adresu IP:

```
$ nmcli con show "ens33" | grep IP4.ADDRESS
```

Dodanie nowego adresu IP:

```
$ nmcli con modify "ens33" +ipv4.addresses 192.168.1.190/24
```

Aktywuj ponownie połączenie (już z dwoma adresami IP):

```
$ nmcli con up ens33
```

Sprawdzenie aktualnych adresów IP:

```
$ nmcli con show "ens33" | grep IP4.ADDRESS
```

Widoczne są teraz dwa adresy IP – dobrze.

Dodanie statycznej routy dla sieci 172.16.0.0/24:

```
$ nmcli conn mod "ens33" +ipv4.routes "172.16.0.0/24
192.168.1.250"
```

Ponowna aktywacja połączenie w celu dokonania zmian:

```
$ nmcli con up ens33
```

Sprawdzenie czy nowa trasa jest widoczna:

```
$ route -n
```

WAŻNE !!!

Oczywiście alternatywą do używania nmcli jest ip, jednak zmiany po reboocie systemu nie byłyby widoczne.

```
$ ip address add 192.168.1.190/24 dev ens33
$ ip route add 172.16.0.0/24 via 192.168.1.250
```

Lab 18: Firewall

Zadania:

- Stwórz nową zonę o nazwie ‘development’.
- Stwórz nowy serwis ‘myWebApp’ dodając do niego następujące usług: https, mysql, nfs, ssh oraz dns.
- Dodatkowo serwis ma zezwalać na komunikację na niestandardowym porcie 9090/tcp oraz na pingi.
- Jako docelowe adresy zezwól tylko 192.168.1.100.
- Dodaj ten jeden serwis do zony.
- Nie aktywuj zony ani nie przypisuj jej interfejsu sieciowego.
- Na końcu wylistuj konfigurację dla zony i serwisu.

Stworzenie nowej zony:

```
$ firewall-cmd --new-zone development –permanent
$ firewall-cmd --reload
```

Stworzenie nowego serwisu:

```
$ firewall-cmd --permanent --new-service=myWebApp --set-
description="My Web App service"
$ firewall-cmd --reload
```

Dodanie usług do serwisu:

```
$ firewall-cmd --permanent --service=myWebApp --add-port=443/tcp
$ firewall-cmd --permanent --service=myWebApp --add-port=3306/tcp
$ firewall-cmd --permanent --service=myWebApp --add-port=2049/tcp
```

```
$ firewall-cmd --permanent --service=myWebApp --add-port=2049/udp
$ firewall-cmd --permanent --service=myWebApp --add-port=22/tcp
$ firewall-cmd --permanent --service=myWebApp --add-port=53/tcp
$ firewall-cmd --permanent --service=myWebApp --add-port=53/udp
$ firewall-cmd --permanent --service=myWebApp --add-port=9090/tcp
$ firewall-cmd --permanent --service=myWebApp --add-protocol=icmp
```

Dodanie docelowego adresu IP:

```
$ firewall-cmd --permanent --service=myWebApp --set-
destination=ipv4:192.168.1.100/32
```

Dodanie serwisu do zony:

```
$ firewall-cmd --zone=development --add-service=myWebApp –
permanent
```

Ostateczne przeładowanie firewalld:

```
$ firewall-cmd --reload
```

Wyświetlenie informacji o zonie:

```
$ firewall-cmd --zone=development --list-all
```

Wyświetlenie informacji o serwisie:

```
$ firewall-cmd --info-service=myWebApp
```

Lab 19: NFS

Zadania:

- Stwórz NFS share o nazwie '/devops', który przynależy do grupy o nazwie 'devops' (gid=2000).
- Zadbaj o to, aby każdy plik utworzony w tym folderze miał dziedziczone prawa grupy z katalogu nadrzędnego. Grupa ma prawa read/write. Właścicielem katalogu ma być root, a inni nie mają mieć żadnych praw do niego.
- Wyeksportuj zasób NFS dla wszystkich, z prawami read/write.
- Przetestuj uprawnienia zasobu montując go z poziomu użytkownika będącego członkiem grupy 'devops'.

Stworzenie zasobu NFS:

```
$ mkdir /devops
```

Stworzenie grupy 'devops' z gid 2000:

```
$ groupadd -g 2000 devops
```

Nadanie odpowiednich uprawnień:

```
$ chgrp devops /devops
$ chmod 2770 /devops
```

Zdefiniowanie zasobu NFS: vi /etc/exports

```
/devops      *(rw)
```

Wyeksportowanie zasobu NFS:

```
$ exportfs -rav
```

Po stronie klienta stworzenie grupy ‘devops’ z gid 2000 oraz przykładowego użytkownika będącego członkiem grupy:

```
$ groupadd -g 2000 devops
$ useradd -g devlops devuser
```

Stworzenie przykładowego mountpointu /dev-nfs i zamontowanie zasobu:

```
$ mkdir /dev-nfs
$ mount centos9:/devops /dev-nfs
```

Testowanie:

```
$ su - devuser
$ cd /dev-nfs
$ touch plik
```

Wykonanie powyższego jako user ‘student’ powoduje błąd.

Test pomyślny.

Lab 20: Samba

Zadania:

- Stwórz zasób sieciowy ‘archiwumX’ i udostępnij go jako zasób Samby dla grupy ‘fbi’ jako read only oraz dla użytkowników ‘mulder’ i ‘scully’ jako read/write.
- Stwórz wymaganych użytkowników oraz grupy.
- Przetestuj działanie zasobu.

Stworzenie nowej grupy i użytkowników:

```
$ groupadd fbi
$ useradd -g fbi mulder
$ useradd -g fbi scully
$ useradd -g fbi fbiuser
```

Stworzenie katalogu z odpowiednimi uprawnieniami:

```
$ mkdir /archiwumX
$ chgrp fbi /archiwumX
$ chmod 2770 /archiwumX
```

Stworzenie zasobu Samba: vi /etc/samba/smb.conf:

```
[archiwumX]
        comment = Archiwum X folder
        path = /archiwumX
        valid users = mulder scully @fbi
        write list = mulder scully
        browsable = yes
```

Dodanie użytkowników do samby:

```
$ smbpasswd -a mulder
$ smbpasswd -a scully
$ smbpasswd -a fbiuser
```

Mountowanie CIFSa jako mulder i sprawdzenie czy prawa zapisu działają:

```
$ mount -o username=mulder //192.168.1.70/archiwumX
/archiwumX
$ cd /archiwumX
$ touch mulderfile
```

Zapis ok.

Mountowanie CIFSa jako mulder i sprawdzenie czy prawa zapisu działają:

```
$ cd
$ umount /archiwumX
$ mount -o username=fbiuser //192.168.1.70/archiwumX
/archiwumX
$ cd /archiwumX
$ touch fbiuserfile
```

Zapis kończy się błędem – Permission denied.

Lab 21: SeLinux

Zadania:

- Zainstaluj serwer mariadb.
- Skonfiguruj lokalizację bazy na katalog ‘/mariadb’.
- Uruchom serwis i upewnij się że działa.
- Wstępnie skonfiguruj bazę uruchamiając skrypt ‘mysql_secure_installation’

Instalacja bazy:

```
$ dnf install mariadb-server
```

Utworzenie folderu pod bazę:

```
$ mkdir /mariadb
```

Edycja pliku konfiguracyjnego i zmiana folderu bazy: vi /etc/my.cnf.d/mariadb-server.cnf

```
[mysqld]
#datadir=/var/lib/mysql
datadir=/mariadb
```

Uruchomienie serwisu:

```
$ systemctl enable mariadb --now
```

Pierwszy błąd: złe uprawnienia do katalogu:

```
mariadb-prepare-db-dir[34509]: chown: changing ownership of '/mariadb': Operation not permitted
```

```
mariadb-prepare-db-dir[34484]: Cannot change ownership of the database directories to the 'mysql'
mariadb-prepare-db-dir[34484]: user.  Check that you have the necessary permissions and try again.
mariadb-prepare-db-dir[34445]: Initialization of MariaDB database failed.
mariadb-prepare-db-dir[34445]: Perhaps /etc/my.cnf is misconfigured or there is some problem with
permissions of /mariadb.
mariadb-prepare-db-dir[34445]: Initialization of MariaDB database was not finished successfully.
mariadb-prepare-db-dir[34445]: Files created so far will be removed.
systemd[1]: mariadb.service: Control process exited, code=exited, status=1/FAILURE
systemd[1]: mariadb.service: Failed with result 'exit-code'.
systemd[1]: Failed to start MariaDB 10.5 database server.
```

Nadanie poprawnych uprawnień:

```
$ chown mysql:mysql /mariadb/
```

Ponowna próba uruchomienia serwisu:

```
$ systemctl enable mariadb --now
```

Drugi błąd: selinux:

```
mariadb-prepare-db-dir[34616]: See the MariaDB Knowledgebase at https://mariadb.com/kb
mariadb-prepare-db-dir[34616]: Please report any problems at https://mariadb.org/jira
mariadb-prepare-db-dir[34616]: The latest information about MariaDB is available at
https://mariadb.org/.
mariadb-prepare-db-dir[34616]: Consider joining MariaDB's strong and vibrant community:
mariadb-prepare-db-dir[34616]: https://mariadb.org/get-involved/
mariadbd[34663]: 2023-04-16 11:49:30 0 [Note] /usr/libexec/mariadbd (mysqld 10.5.16-MariaDB) starting as
process 34663 ...
mariadbd[34663]: 2023-04-16 11:49:30 0 [Warning] Can't create test file /mariadb/centos9.lower-test
systemd[1]: mariadb.service: Main process exited, code=exited, status=1/FAILURE
systemd[1]: mariadb.service: Failed with result 'exit-code'.
systemd[1]: Failed to start MariaDB 10.5 database server.
```

Patrzymy w logi:

```
$ less /var/log/messages
```

Widzimy wpisy SELinuksa – coś jest blokowane. Szukamy ‘sealert’:

```
$ sealert -l b3aa0fb6-be1f-4ccc-a3bb-250e24e05ea7

SELinux powstrzymuje /usr/libexec/mariadbd przed dostępem
read w plik plugin.frm.
```

Złe konteksty nowego katalogu – sprawdzamy i poprawiamy:

```
$ ls -laZ /var/lib/mysql/
$ semanage fcontext -a -t mysqld_db_t "/mariadb(/.*)?"
$ restorecon -Rv /mariadb/
```

Ponowna próba uruchomienia serwisu:

```
$ systemctl enable mariadb --now
$ systemctl status mariadb
```

Serwisu uruchomił się. Odpalamy skrypt konfiguracyjny bazę:

```
$ mysql_secure_installation
```