Laboratorium sieci komputerowych
ćw.1. Praca w linii poleceń

Dawid Chmielewski

22 marca 2022

Temat ćwiczenia: Praca w linii poleceń

1 Ogólny cel ćwiczenia

Celem ćwiczenia było utrwalenie elementarnej wiedzy z zakresu pracy w linii poleceń oraz współpracy z serwerem volt. Ćwiczenie realizowałem za pomocą Terminala systemu ArchLinux oraz interpretera poleceń PowerShell systemu MS Windows.

2 Praca z protokołem SSH

Protokół SSH ma architekturę typu klient-serwer: użytkownik (klient) łączy się z serwerem, na którym działa proces obsługujący połączenia. Ćwiczenie polegające na wykorzystaniu tego protokołu w praktyce rozpocząłem od konfiguracji, czyli zainstalowania wszystkiego co potrzebne. Dla systemu MS Windows dodanie niezbędnych rozszerzeń jest możliwe po wybraniu ścieżki prezentowanej przeze mnie poniżej.

```
Aplikacje i funkcje > Funkcje opcjomalne > Dodaj funkcję
```

Konieczne rozszerzenia do zainstalowania nazywają się tutaj Klient OpenSSH oraz Serwer OpenSSH. Dla systemu Linux konfiguracja polega na zastosowaniu w oknie terminala poleceń prezentowanych przeze mnie niżej.

```
sudo apt update
sudo apt install openssh-server
sudo systemctl start sshd
sudo systemctl status sshd
```

W tych poleceniach realizuje się kolejno: pobieranie informacji o pakietach, instalowanie klienta Open SSH, stworzenie serwera oraz sprawdzenie statusu po jego uruchomieniu. Polecenie sudo podnosi jednorazowo uprawnienia użytkownika do poziomu administratora- jest to o wiele bezpieczniejsze od ciągłej pracy jako administrator.

Sprawdzenie statusu serwera sshd dało następujący wynik:

```
status sshd
sshd.service - OpenSSH Daemon
 Loaded: loaded (/usr/lib/systemd/system/sshd.service; disabled; vendor preset: disabled)
```

```
 Active: active (running) since Tue 2022-03-15 14:16:05 UTC; 1h 54min ago
    Main PID: 2487 (sshd)
  Tasks: 1 (limit: 4599)
 Memory: 1.5M
    CPU: 8ms
 CGroup: /system.slice/sshd.service
         2487 "sshd: /usr/bin/sshd -D [listener] 0 of 10-100 startups"

Mar 15 14:16:05 archiso systemd[1]: Started OpenSSH Daemon.
Mar 15 14:16:05 archiso sshd[2487]: Server listening on 0.0.0.0 port 22.
Mar 15 14:16:05 archiso sshd[2487]: Server listening on :: port 22.
live@archiso ~ %
```

Połączenie z serwerem volt osiągnąłem poprzez zastosowanie właściwej komendy, którą zamieściłem poniżej.

```
ssh chmield2@volt.zet.pw.edu.pl
```

Koniczne jest w takim przypadku wpisywanie hasła. Po zalogowaniu można je zmienić poleceniem passwd, w takim przypadku należy podać stare hasło oraz dwukrotnie wpisać nowe:

```
volt% passwd
Changing local password for chmield2
Old Password:
New Password:
Retype New Password:
volt%
```

W ramach ćwiczenia zaimplementowałem też logowanie się bez konieczności wpisywania hasła. By to osiągnąć, wygenerowałem parę kluczy ssh za pomocą algorytmu Ed25519:

```
ssh-keygen -t ed25519
```

Po czym wysłałem klucz publiczny z pary na volta:

```
 ssh-copy-id -i /home/live/.ssh/id_ed25519.pub chmield2@volt.zet.pw.edu.pl
```

Po tym wszystkim możliwe było logowanie się bez hasła:

```
PS C:\Users\Dawid> ssh chmield2@volt.zet.pw.edu.pl
Last login: Mon Mar 14 18:18:00 2022 from 10.12.5.8

(...)
```

3 Komendy linii poleceń

Realizacja ćwiczenia pozwoliła mi na przypomnienie komend używanych podczas pracy w linii poleceń. Za najważniejsze z tych, które dostępne są w środowisku Linux, uważam:

- sudo - podniesienie uprawnień do poziomu administratora w ramach jednego polecenia,

- ls - wyświetlenie listy plików w katalogu, do którego użytkownik podaje ścieżkę, domyślnie- tego, w którym użytkownik się znajduje,
- chmod- nadawanie lub usuwanie praw do pliku: odczytu (r), modyfikacji (w) oraz wykonywaia (x) dla danych klas: użytkownik (u), grupa (g), inni (o) lub wszyscy (a). np. chmod u+x a <- nadanie prawa wykonwania pliku o nazwie a użytkownikowi,
- cd - przejście do innego katalogu w konsoli; ważną zasadą jest trzymanie się własnego katalogu domowego podczas pracy,
- ping - sprawdzenie stanu połączenia między komputerem a daną witryną lub innym komputerem,
- man - wyświetlanie pomocy do polecenia podanego jako argument wywołania. W sposób obszerny opisuje zastosowanie danego polecenia wraz z użyciem flag,
- touch, rm - tworzenie i usuwanie pliku,
- mkdir, rmdir - tworzenie i usuwanie katalogów (Uwaga, polecenie rmdir służy do usuwania wyłącznie pustych katalogów - dla niepustych konieczne jest użycie rm -r),
- passwd - zmiana hasła, tak jak pokazywałem w ramach punktu numer 2 niniejszego sprawozdania,
- pwd - wyświetlenie pełnej ścieżki aktualnego katalogu,
- scp - bezpieczne kopiowanie plików pomiędzy dwoma maszynami w sieci.
- alias - tworzenie lub wyświetlanie aliasów- komend, które zastępują wiele słów (komend) jednym. Poniżej pokazuję przykład dla aliasu `ll`, który zastępuje wyrażenie `ls -l`, czyli wypisania plików w katalogu w formie listy.

```
volt% alias ll
ll='ls -l'
volt% ll
total 1
drwxr-xr-x  3 chmield2  stud   512 13 mar 18:28 Sprawozdanie
drwxr-xr-x  2 chmield2  stud   512 15 mar 22:55 c1
-rwxr--r--  1 chmield2  stud   210 15 mar 18:31 dodawanie
-rw-r--r--  1 chmield2  stud  4892 16 mar 13:39 dziennik1.txt
-r--r--r--  1 chmield2  stud   770  9 mar 14:00 kartkowka-sroda.txt
lrwxr-xr-x  1 chmield2  stud    22 13 mar 19:54 labsk -> /usr/local/zetis/labsk
-rwxr--r--  1 chmield2  stud   415 15 mar 23:37 skrypt
volt%
```

4 Skrypty

Skrypty to inaczej wykonywalne pliki z szeregiem poleceń do wykonania przez interpreter. Są stosowane między innymi do usprawnienia i przyspieszenia pracy użytkownika. Aby użytkownik mógł użyć skryptu, konieczne jest przyznanie mu prawa do wykonania (x). Domyślnie, dostęp do skryptu jest dostępny wyłącznie z poziomu katalogu, w którym się on znajduje. Możliwe jest jednak wyeksportowanie ścieżki ze skryptem do zmiennej PATH, dzięki czemu będzie on dostępny globalnie. Rozważmy prosty skrypt, którego jedynym zadaniem jest wypisanie argumentu wywołania. Znajduje się on w moim katalogu domowym:

```
volt% pwd
/home/stud/chmield2
volt% cat skrypt
#!/bin/sh
echo $1
```

Wykonanie go z poziomu nie będzie stanowiło problemu, jednak po przejściu do innego katalogu nie będzie to już możliwe:

```
volt% ./skrypt "argument pierwszy"
argument pierwszy
volt% cd ../
volt% ./skrypt "argument pierwszy"
zsh: no such file or directory: ./skrypt
(127)
```

Kluczem do rozwiązania tego problemu jest dodanie ścieżki skryptu do zmiennej PATH. Zmienna PATH przechowuje wszystkie katalogi, które są przeszukiwane w celu znalezienia zewnętrznego polecenia (w odróżnieniu od poleceń wewnętrznych, które są wbudowane we wnętrzu interpretera poleceń- np. echo). Dodanie więc ścieżki z moim skryptem sprawiło, iż może być on przeze mnie uruchamiany z dowolnego miejsca. Właściwą ścieżkę dodałem za pomocą komendy export:

```
volt% export PATH=$PATH:/home/stud/chmield2:
```

To rozwiązanie sprawiło, iż byłem w stanie uruchomić swój skrypt z dowolnego miejsca, np. z katalogu nadrzędnego.

```
volt% pwd
/home/stud
volt% skrypt "argument pierwszy"
argument pierwszy
```